Université de Genève Faculté des Lettres

Traitement Automatique des Langues et Apprentissage des Langues Assisté par Ordinateur : bilan, résultats et perspectives

THÈSE N^o 721

présentée à la Faculté des Lettres de l'Université de Genève pour obtenir le grade de docteur ès lettres par

Sébastien L'HAIRE

Jury de thèse

Directeur Prof. Eric WEHRLI (Université de Genève)
Président Prof. Jacques MOESCHLER (Université de Genève)
Membres Prof. Georges ANTONIADIS (Université Stendhal, Grenoble)
Prof. Thierry CHANIER (Université B. Pascal, Clermont-Ferrand)
Dr. Paola MERLO (Université de Genève)

Juin 2011

La Faculté des lettres, sur le préavis d'une commission composée de Messieurs les professeurs Jacques MOESCHLER, président du jury ; Eric WEHRLI, directeur de thèse ; Georges ANTONIADIS (Université Stendhal, Grenoble) ; Thierry CHANIER (Université Blaise Pascal, Clermont-Ferrand) ; Dr. Paola MERLO (Université de Genève), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 15 juin 2011

Le Doyen: Eric WEHRLI

Thèse N^o 721

Table des matières

Liste des tableaux

Table des figures

Remerciements

Cette thèse a nécessité un nombre inhabituel d'années pour aboutir. J'évoquerai tout d'abord un plan trop ambitieux, comprenant des états de l'art très fouillés dans lesquels je me suis lancé tête baissée au début de ma rédaction et qui ont pris des années de rédaction. Puis est venu s'ajouter au plan la partie de la correction orthographique, qui a nécessité un certain temps de développement et de nombreuses heures pour coder le corpus. Les trois dernières années de rédaction, je ne disposais que de quelques heures par jour le soir et les week-ends, la journée étant consacrée à mon emploi dans le privé. Enfin, on évoquera des activités non scientifiques et d'autres circonstances personnelles qui ont considérablement ralenti le processus d'écriture.

Je tiens à remercier tout particulièrement mon directeur de thèse Eric Wehrli, qui a patiemment attendu que je daigne lui présenter mes travaux. Eric a toujours répondu présent quand j'avais besoin de ses lumières et de ses services. C'est grâce à lui que la Faculté m'a accordé des dérogations et que j'ai pu disposer de financements. Le président du jury Jacques Moeschler m'a également engagé dans diverses circonstances et soutenu dans mes démarches.

Les membres du Jury, Paola Merlo, Thierry Chanier et Georges Antoniadis méritent également mes remerciements pour avoir accepté de lire cet énorme pavé et pour leurs commentaires constructifs.

Ma gratitude va aussi à toutes les personnes qui ont collaboré au projet FreeText dont nous pouvons toutes et tous être fiers. Malgré quelques tensions, les trois années de développement de notre prototype restent un souvenir inoubliable. Je dois naturellement mentionner spécialement l'équipe d'UGEN – l'équipe genevoise, sans jargon administratif européen – au complet. Mentionnons la responsable Catherine Walther Green, qui a patiemment corrigé de nombreux travaux, Anne Vandeventer Faltin, avec qui j'ai

eu de nombreuses discussions et qui m'a laissé reprendre le développement de son correcteur orthographique, Hugues Péters et Keith Potter.

Mille mercis également aux enseignants qui m'ont envoyé les textes bruts de leurs apprenants, qui constituent la grande partie de mon corpus d'erreurs orthographiques.

Thierry 'Titi' Etchegoyhen m'a aidé lors de mes premiers travaux sur la comparaison sémantique de phrases, lorsque j'ai dû me plonger dans un code particulièrement ardu, malgré mes réflexions sur les Basques. Jean-Philippe 'Jipé' Goldman m'a conseillé en particulier sur le système expert de phonétisation. Enfin, Arnaud Gaudinat et Luka 'les bons tuyaux' Nerima m'ont passablement soutenu pour des questions techniques.

Pour sa gentillesse et sa disponibilité, merci à Eva Capitao, secrétaire du département, qui n'est jamais avare d'encouragements et agrémente régulièrement les pauses cafés de ses pâtisseries. Mention spéciale pour le gâteau au pavot ! Les autres collaborateurs du LATL et du département méritent également mes remerciements pour contribuer à un cadre de travail stimulant et agréable. Merci également à toutes et à tous pour avoir supporté mon caractère, notamment les diatribes contre le merveilleux système Window$ et le néanmoins extraordinaire environnement de programmation Blackbox...

Je n'oublierai pas la patience de mon entourage pendant ces années de rédaction. Il n'est pas évident de supporter de nombreuses absences pour cause de rédaction de thèse, ainsi que les sautes d'humeur dues à la fatigue et aux difficultés techniques. Merci aussi à ceux qui prenaient régulièrement des nouvelles de l'avancement de mes travaux, même si parfois je sentais une pointe d'ironie face à cet interminable serpent de mer. J'y suis finalement arrivé et je crois que le jeu en valait la chandelle !

Chapitre 1

Introduction

Apprendre une langue étrangère est aujourd'hui devenu courant. Pour un nombre croissant de professions, la connaissance plus ou moins approfondie d'une ou plusieurs langues étrangères est carrément indispensable. S'il existe plusieurs milliers de langues parlées dans le monde, certaines langues dominent comme l'anglais, l'arabe, l'espagnol, le russe, le portugais, le mandarin, le français, etc., que ce soit comme langue maternelle, comme langue officielle ou comme langue véhiculaire. Dans le monde de la communication d'aujourd'hui, apprendre une ou plusieurs langues facilite les échanges.

Autrefois, dans les pays occidentaux, l'apprentissage des langues se limitait souvent aux langues anciennes (latin et, dans une moindre mesure, grec et hébreu), dans le but de lire les textes religieux, scientifiques et philosophiques dans le texte. L'apprentissage d'une langue était fortement axé sur la grammaire et les exercices de traduction, principalement la version, de la langue ancienne vers la langue vivante [1]. Les langues étrangères vivantes étaient essentiellement apprises dans deux buts : tout d'abord, les élites européennes maîtrisaient la langue française, langue diplomatique, ou le latin, langue de la religion et de la science ; d'autre part, pour le commerce, il était indispensable de connaître une langue commune dans les comptoirs, lieux de contact et d'échanges commerciaux entre différentes cultures.

Aujourd'hui, apprendre une langue étrangère est tantôt un plaisir, tantôt une nécessité, pour les migrants, les métiers commerciaux ou les loisirs. Les méthodes d'apprentissage sont maintenant axées vers la capacité à commu-

1. Pour ces langues dites mortes, l'enseignement reste encore de nos jours très traditionnel. Tout au plus a-t-on renoncé aux exercices de thème, qui consistaient à traduire des textes littéraires ou oratoires d'une langue vivante vers la langue morte.

niquer et moins vers la grammaire. L'axe est davantage porté vers le fond plutôt que vers la forme, même s'il est nécessaire de maîtriser l'essentiel des règles afin d'être compris par son interlocuteur. La forme reste toutefois très importante pour l'écrit (Loritz, 1987; Kempen, 1999; Chapelle, 2001; Defays et Deltour, 2003).

Cette thèse a pour sujet les outils de traitement automatique des langues appliqués à l'apprentissage des langues assisté par ordinateur, leurs acquis et les perspectives d'avenir. Elle est centrée sur les travaux que nous avons eu l'occasion de réaliser au cours de notre parcours académique. Dans cette introduction, nous évoquons d'abord la problématique et les enjeux de cette thèse (§1.1). Puis nous passons aux objectifs et au plan du reste de la thèse (§1.2).

1.1 Problématique et enjeux

Utiliser un ordinateur pour apprendre est devenu aujourd'hui banal. Dans la plupart des cas, loin d'être un remplaçant des enseignants, l'ordinateur doit être considéré comme un complément utile et une aide à l'apprentissage. On trouve des programmes pour enseigner des compétences variées, de qualité et de prix très variables. L'essor de ce type de logiciel a été facilité par la démocratisation de l'informatique, alors qu'il est courant de nos jours de trouver un ordinateur ou davantage dans les ménages. Un ordinateur a l'avantage de pouvoir présenter les notions à acquérir à travers diverses formes, notamment du texte, des animations et même des jeux ; par ailleurs, l'ordinateur est plus disponible qu'un enseignant et peut corriger de manière fiable certaines erreurs commises lors d'évaluations. Les élèves peuvent progresser à leur propre rythme et répéter les exercices à l'envi ; ils sont moins inhibés face à une machine que face à une classe et un enseignant chargé de les évaluer (Garrett, 1995; Levy, 1997; Nerbonne, 2003; Hubbard et Levy, 2006).

La convivialité et la simplicité d'utilisation des ordinateurs facilite l'emploi de didacticiels dans de multiples circonstances, que ce soit dans un cadre institutionnel (école, université, ...), professionnel ou privé. Les élèves travaillent tantôt seuls, tantôt à plusieurs, parfois en classe, parfois dans d'autres lieux (centres de documentation, salles d'informatique, en déplacement, à domicile etc.), tantôt en complément de l'enseignement classique en classe (*blended learning*), tantôt comme unique outil d'apprentissage. Les concepts enseignés sont très variés, que ce soit la géographie, la mécanique, la finance, la biologie, etc. Les niveaux sont aussi variés, selon les publics

cible, de l'introduction au contenu extrêmement spécialisé et pointu.

Les institutions tirent également profit de l'utilisation de tels logiciels, tant des didacticiels commerciaux prévus pour une utilisation autonome privée que des produits spécialement conçus pour une salle d'ordinateurs ou pour l'enseignement à distance. Par ailleurs, les enseignants utilisent les produits tels quels ou les adaptent à leurs propres besoins. Les enseignants disposent parfois d'outils très utiles pour suivre le travail des élèves : résultats des tests, traces, parcours dans le logiciel etc. Ainsi, les enseignants peuvent se consacrer à d'autres aspects de leur travail en se débarrassant de tâches répétitives (Bailin et Levin, 1989).

La suite de cette section est organisée comme suit. Dans un premier temps, nous abordons l'apprentissage des langues assisté par ordinateur (§1.1.1). Puis nous nous penchons sur le traitement des langues par ordinateur (§1.1.2).

1.1.1 Apprentissage des langues assisté par ordinateur

L'Apprentissage des Langues Assisté par Ordinateur (ALAO) [2] est un domaine très prolifique de l'utilisation de l'ordinateur pour la didactique. Pour souligner le rôle plus actif des élèves et le rôle central qu'ils ont dans le processus d'apprentissage, on utilisera dorénavant le terme d'*apprenant* [3].

La didactique des langues est un domaine en constante évolution, à l'intersection de la linguistique, de la psychopédagogie et de la sociologie ; l'apprentissage des langues, lui, se situe au carrefour de la langue, de la culture et de la communication (Defays et Deltour, 2003). Les apprenants doivent apprendre à utiliser la langue, à travers de structures de plus en plus complexes et variées et à l'aide d'un vocabulaire de plus en plus riche, dès le niveau débutant, à l'oral comme à l'écrit. La majeure partie des logiciels traitent des langues dominantes et peu sont consacrés à des langues minoritaires.

Il y a trois types d'utilisation de l'ALAO, selon Nerbonne (2003): (i) aca-

2. Autrefois, le sigle d'EAO (Enseignement Assisté par Ordinateur) était très populaire. Nous choisissons le terme d'*apprentissage* plutôt que d'*enseignement*, pour souligner le rôle plus actif des élèves. En anglais, le terme le plus répandu actuellement est *Computer-Assisted Language Learning* (CALL). On trouve aussi TELL (*Technology-Enhanced Language Learning*), qui inclut les autres technologies que l'ordinateur, CBI (*Computer Based Instruction*), CAI (*Computer-Assisted* ou *Computer-Aided Instruction*), ITS (*Intelligent Tutoring System*), CALLLT (*CALL using Language Technologies*) et NBLT (*Network-based language teaching*).

3. En anglais, *learner*.

démique et scolaire, avec manque de moyens en matériel et logiciels et enseignants parfois réticents ; (ii) industrie, avec beaucoup de moyens, pour un enseignement spécialisé ; (iii) les autodidactes qui apprennent sans aide, un marché en expansion parfois qualifié d'*edutainment*. Sur le plan du niveau de la langue, plus le public-cible est avancé, plus la question de l'évaluation automatique se pose, comme nous aurons largement l'occasion de l'aborder dans cet ouvrage.

Apprendre une langue est surtout apprendre à communiquer, c'est-à-dire transmettre un contenu porteur de sens à un destinataire. Le but de ces actes de langage est d'agir sur l'environnement du locuteur à travers des mots ; ces buts peuvent être d'informer, d'affirmer, d'inciter, de convaincre, de promettre, etc. (Austin, 1962; Searle, 1969).

La communication est toutefois sujette à des erreurs. Comme tout signal, le langage peut être perturbé par du bruit (bruit sonore, illisibilité de l'écriture, etc.), mais surtout, les langues sont extraordinairement ambigües : les mots ont souvent plusieurs significations, que nous distinguons généralement sans problème grâce au contexte de l'énoncé et à notre connaissance générale du monde, très souvent de manière immédiate, inconsciente et sans difficulté. En général, lorsqu'ils communiquent, les êtres humains négocient sans cesse le sens, par exemple grâce à des précisions durant la conversation. Les apprenants d'une langue étrangère sont souvent handicapés par des lacunes en vocabulaire et en syntaxe, qui nécessitent l'emploi de stratégies de substitution comme l'emploi de périphrases. De plus, ils commettent de nombreuses fautes lexicales, orthographiques, morphologiques, syntaxiques, etc.

Pour apprendre, il est indispensable que les apprenants manipulent la langue. Il existe des exercices traditionnels comme les textes à trous, les questionnaires à choix multiples, les exercices d'appariement de mots, de remise en ordre de mots etc. Ces exercices sont facilement corrigés par l'ordinateur. Ces exercices sont également utiles pour vérifier la bonne compréhension d'un texte par les apprenants. Toutefois, très tôt, il est nécessaire que les apprenants produisent des phrases complètes, dont la complexité va croissant au fil de leur progression. Des techniques basiques peuvent corriger des fautes simples, comme la comparaison de chaînes et la reconnaissance de patrons, mais la complexité et la variété des erreurs commises nécessitent un traitement plus intelligent.

1.1.2 Traitement automatique des langues

Le Traitement Automatique des Langues (TAL) [4] est un domaine de l'Intelligence Artificielle (IA) qui traite divers aspects de la langue et du discours, à tous les niveaux, lexical, morphologique, syntaxique, sémantique ou pragmatique. Parmi les applications du TAL, le grand public connaît essentiellement la correction orthographique et syntaxique, la reconnaissance de la parole et la synthèse vocale. Pour identifier les erreurs des apprenants, en poser le diagnostic et proposer une correction ou une stratégie de remédiation, le TAL fournit une aide précieuse et indispensable.

Par contre, le TAL est lui-même très sensible aux erreurs, du fait qu'une application attend en principe une entrée sans erreurs pour lui appliquer un traitement. Une application *robuste* est capable de fournir un résultat au moins partiel, même en cas d'entrées incorrectes ou inconnues [5] (Vandeventer Faltin, 2003). Le second problème rencontré avec les applications de TAL est la *surgénération* des propositions – avant tout, des structures syntaxiques – qu'il faut trier pour en sélectionner les meilleures. Enfin, le troisième problème est posé par la *surdétection des erreurs* et les *erreurs de diagnostic*, comme les apprenants n'ont pas un jugement aussi affuté que les locuteurs natifs et se méfient moins des erreurs. Tout au long de cet ouvrage, nous décrirons abondamment les possibilités d'application du TAL à l'ALAO. Pour souligner l'emploi des techniques intelligentes du TAL, on parle alors d'Apprentissage *Intelligemment* Assisté par Ordinateur (ALIAO).

Levin et Evans (1995, p. 90) constatent que grâce aux outils de TAL, les concepteurs d'exercices peuvent se concentrer sur le contenu, sans avoir à anticiper chaque erreur potentielle des apprenants. Les techniques de TAL sont pourtant loin d'êtres parfaites. A tous les niveaux, les ambiguïtés de la phrase sont nombreuses (Bouillon *et al.*, 1998). Souvent, un mot peut avoir plusieurs catégories syntaxiques, ce que l'on nomme *ambiguïté lexicale* : en anglais, un mot peut souvent être un verbe (*man*, homme ; *to man* équiper [un navire]) ; en français, *belle* est un nom et un adjectif, *voile* est un verbe conjugué ou un nom, etc. L'*ambiguïté sémantique* réside dans l'homonymie

4. On trouvera aussi le terme de TALN (Traitement Automatique du Langage Naturel). En anglais, le terme le plus répandu est NLP (*Natural Language Processing*).

5. Il existe de nombreux mots inconnus, qui sont souvent correctement orthographiés et employés dans une phrase, mais ne sont pas listés dans le lexique de l'application informatique. Une application robuste cherchera alors à pallier ces lacunes, comme nous le verrons au chapitre 3. Dans le domaine syntaxique, un analyseur robuste devinera la catégorie lexicale d'un mot et cherchera à fournir des morceaux d'analyses quand il ne peut donner une analyse complète. Pour la synthèse vocale, les mots dont on n'a pas la prononciation devront être phonétisés par le système grâce à des règles.

et la polysémie : *lentille* peut signifier un objet d'optique, une espèce de plante ou la partie comestible de celle-ci. On rencontre également les *ambiguïtés syntaxiques* : certains éléments de la phrase peuvent être liés à plusieurs autres éléments, ce qui implique des changements d'interprétation [6]. Examinons les phrases suivantes :

(1) a. Jean regarde l'homme sur la colline avec ses jumelles.
b. La soupe de légumes du restaurant.

En (1a), nous avons deux syntagmes prépositionnels *sur la colline* et *avec ses jumelles*. Ces syntagmes peuvent tous deux être rattachés soit au syntagme nominal *l'homme*, soit au syntagme verbal *regarde l'homme* et donc se rapporter au sujet *Jean*. Le syntagme *avec ses jumelles* peut même être attaché à *colline* ; cet attachement est difficilement interprétable au niveau sémantique, mais tout à fait valable du point de vue syntaxique. En outre, *jumelles* dénote soit deux individus de sexe féminin, soit un instrument d'optique. Le pronom possessif *ses* peut se rapporter à *Jean*, à *l'homme* ou à un tiers mentionné auparavant dans le discours. Ainsi, avec dix mots seulement, on atteint déjà un nombre important d'interprétations. De même, en (1b), les deux groupes prépositionnels *de légumes* et *du restaurant* doivent être rattachés à *soupe* comme complément du nom. Cependant, avec la même préposition *de*, *légumes* désigne le type de soupe et est un complément du nom de matière tandis que *restaurant* est un complément du nom possessif [7].

Les ordinateurs ne sont capables de comprendre qu'une fraction du langage et les résultats de leur traitement ne sont pas parfaits. Traiter la langue automatiquement par ordinateur est un processus difficile qui, à vrai dire, est loin d'être résolu pour une couverture large de la langue. La difficulté de la tâche d'analyse est considérable. Au niveau syntaxique, une phrase peut avoir un grand nombre de structures correctes [8].

Lorsque le but d'un logiciel d'ALAO est un apprentissage en autonomie – c'est-à-dire sans supervisation d'un enseignant – de nombreuses erreurs

6. On trouve également des ambiguïtés sémantiques et pragmatiques (Sabah, 2000), qui varient fortement selon les langues et les cultures (Warga, 2007).

7. On peut pronominaliser ce complément par *sa soupe de légumes*, quand *leur soupe du restaurant* ne permet pas de relier *leur* à *légumes*. L'autre indice est qu'on ne peut dire *soupe du restaurant de légumes*. Ce test fonctionne aussi pour *étudiant en linguistique*, *verre à vin*, etc.

8. Plus une grammaire prend en compte des structures complexes, plus l'analyseur produit de structures possibles pour une même phrase. Cependant, de nombreuses structures correctes n'ont aucune plausibilité linguistique ni sémantique et leurs interprétations seraient rejetées par des humains. Il existe diverses méthodes de filtrage des analyses, notamment au moyens d'heuristiques ou de méthodes statistiques.

doivent être détectées, correctement marquées et, si possible, des explications de remédiation – voire une correction – fournies. Les attentes vis-à-vis des ordinateurs sont immenses et la déception face à des résultats mitigés – voire médiocres – est à la mesure de la tâche qui reste à accomplir. Les apprenants commettent de nombreuses erreurs, qu'il est parfois difficile à anticiper, d'autant plus lorsque les productions de l'apprenant sont libres, sans restrictions de structures ni de vocabulaire.

L'intégration du TAL dans l'ALAO n'est pas encore très répandue. Kraif *et al.* (2004) voient trois raisons pour expliquer ce phénomène : (i) ces techniques manquent encore de fiabilité ; (ii) ces ressources sont difficiles et chères à développer ; (iii) les utilisateurs (éditeurs, apprenants, enseignants, concepteurs...) sont peu au courant des possibilités offertes par ces techniques. Jager (2001, p. 103) fait en revanche preuve d'un optimisme prudent, en considérant les technologies du TAL comme suffisamment avancées, du moins pour les apprenants qui ont un niveau suffisamment élevé pour juger de la qualité des diagnostics. Borin (2002), quant à lui, suggère que les enseignants utilisent les applications de TAL pour corriger les textes et réponses à des exercices. Les erreurs seraient alors mises en évidences et les enseignants pourraient formuler leur propre rétroaction. Cette manière de faire éviterait les problèmes dus aux feedbacks erronés. Les locuteurs natifs bénéficient du soulignement d'erreurs potentielles mais restent libres de prendre en compte ou non les suggestions de correction, alors que les apprenants n'ont pas encore cette intuition de la langue.

1.2 Structure et objectifs

Cette thèse vise trois objectifs, par ordre d'importance croissant :

i. effectuer un survol du domaine de l'ALAO, de ses domaines d'application et de ses enjeux ;
ii. décrire les applications du traitement automatique des langues dans le domaine de l'ALAO ;
iii. présenter trois applications particulières que nous avons réalisées au cours de notre parcours académique, dans le domaine de la correction orthographique et du diagnostic d'erreurs syntaxiques et "sémantiques" de bas niveau.

Dans la plupart des thèses, un état de l'art plus ou moins étendu offre un historique général du domaine et survole les techniques appliquées dans

le domaine précis des travaux présentés, dans le but d'en souligner l'apport scientifique. Nous avons également cette préoccupation mais nos deux premiers objectifs vont plus loin.

Au chapitre 2, notre état de l'art de l'ALAO survole l'histoire de ce domaine. Il traite des enjeux pédagogiques et didactiques et des questions de conception d'un logiciel. Nous consacrons quelques pages à la question de l'évaluation qualitative des logiciels. Nous décrivons également diverses techniques d'évaluation des connaissances des apprenants. Nous terminons par une description des divers types d'ALAO. Par ailleurs, l'annexe B contient des descriptions de logiciels qui utilisent des techniques de TAL.

Le second état de l'art, au chapitre 3, traite en détail des divers domaines du TAL et de leur application à l'ALAO. Une première section survole les différents domaines du TAL et en décrit les applications. Puis nous consacrons des sections spécifiques à la correction orthographique, à l'analyse syntaxique et détection d'erreurs grammaticales et à divers formalismes sémantiques et lexicaux. Tout au long de ce chapitre, nous décrivons l'état des applications et leur évolution prévisible. Par ailleurs, une seconde annexe (§C) traite de logiciels de correction orthographique ou grammaticale ou d'aide à la rédaction qui, sans être des logiciels d'ALAO, offrent une aide pédagogique appréciable et sont conçus pour des apprenants.

Venons-en maintenant aux applications que nous avons réalisées ou contribué à réaliser. Au chapitre 4, nous décrivons le projet de recherche européen FreeText qui visait à développer un logiciel d'ALIAO pour apprenants du français de niveau moyen, auquel nous avons participé. Nous décrivons le contenu général et l'architecture du logiciel et nous nous livrons à un bilan du projet.

Au chapitre 5, nous décrivons l'outil de TAL principal de FreeText, le système d'analyse syntaxique et de diagnostic d'erreurs. Comme le diagnostic d'erreurs a déjà fait l'objet de la thèse de notre ancienne collègue Anne Vandeventer Faltin (2003), nous ne faisons qu'en survoler les techniques. Par contre, nous nous penchons sur les outils d'aide à l'apprentissage qui ont constitué l'essentiel de notre contribution à ce projet : la grammaire en couleurs, l'affichage du diagnostic d'erreurs et l'arbre syntaxique. Une discussion de bilan clôt ce chapitre.

Ensuite, le chapitre 6 présente le correcteur orthographique *FipsOrtho*. Au sein du projet FreeText, le correcteur orthographique était à la charge d'un autre partenaire, mais deux collègues, Anne Vandeventer Faltin et Mar Ndiaye, ont commencé à développer un prototype de correcteur. Quelques

temps après la fin du projet, nous en avons repris le code en en améliorant le fonctionnement et en ajoutant des méthodes de correction. Dans un premier temps, nous avons soumis notre correcteur à un premier test en lui soumettant une liste de mots erronés. Ce premier test, que nous décrivons en détail, a permis d'améliorer les performances de notre système en nous permettant d'ajuster certains paramètres. Un second test a consisté à soumettre un grand nombre de phrases – la plupart provenant d'apprenants – à notre correcteur et à les annoter dans un corpus de 14 494 mots, afin d'étudier les erreurs commises par les apprenants. Nous terminons ce chapitre par une discussion de bilan.

Enfin, le chapitre 7 porte sur deux outils "sémantiques" plus expérimentaux et moins aboutis. Le premier outil est la comparaison *sémantique* de phrases. Pour évaluer une phrase, la syntaxe ne suffit pas toujours. Une phrase syntaxiquement correcte peut ne pas convenir comme réponse à un exercice. Le projet FreeText incluait un outil de comparaison sémantique de phrases. Cette technique consiste à comparer la réponse de l'apprenant à une réponse stockée dans le système par le concepteur de l'exercice, au moyen de *structures pseudo-sémantiques*. Il s'agit de structures représentant le sens profond d'une phrase, qui mêlent informations lexicales et caractéristiques abstraites. Au moment du projet, nous avons ébauché un prototype d'outil, mais, pour des raisons que nous exposerons plus loin, le consortium du projet a décidé d'en arrêter le développement. Bien après la fin du projet, nous avons largement modifié les algorithmes de cet outil et sommes arrivés à des résultats encourageants. Nous illustrons simplement le potentiel de l'outil en montrant son fonctionnement dans divers cas de figure. Le second outil "sémantique" est la reformulation de phrases, où un apprenant pourrait manipuler les phrases à l'aide de leurs représentations sémantiques simplifiées. Nous nous bornons à esquisser théoriquement ce que pourrait être un tel système et son apport potentiel pour les apprenants.

Nous concluons cette thèse (§8) en reprenant les principales constatations de cet ouvrage, en faisant le bilan des apports du TAL à l'ALAO et en esquissant quelques perspectives de recherche.

Chapitre 2

État de l'art de l'apprentissage des langues par ordinateur

Dans ce chapitre, nous exposons l'état de l'art de l'Apprentissage des Langues Assisté par Ordinateur (ALAO). Dans un premier temps, nous esquissons une brève histoire de l'ALAO (§2.1). Nous poursuivons par un rapide survol des étapes de construction d'un logiciel, des compétences nécessaires (§2.2), des théories pédagogiques (§2.3) et de la notion de feedback (§2.4). Puis nous parlons des méthodologies d'évaluation des logiciels (§2.5), des méthodes d'évaluation automatique des connaissances des apprenants (§2.6) et des différents types d'ALAO (§2.7). Enfin, nous terminons le chapitre par une discussion (§2.8). Tout au long de cet ouvrage, nous ferons référence à des logiciels d'ALAO ainsi qu'à des logiciels de correction orthographique et grammaticale. Pour un grand nombre d'entre eux, le lecteur trouvera une description plus complète dans les annexes B et C.

2.1 Histoire de l'ALAO

L'histoire de l'ALAO est souvent décrite en partant d'un découpage historique lié aux théories pédagogiques, comme on le trouve chez Warschauer (1996), qui distingue trois phases dans l'histoire du développement de logi-

ciels d'ALAO :

- L'*approche béhavioriste* (années 1970-80) : l'ordinateur était un moyen d'apporter du contenu à l'apprenant et d'exercer ses connaissances par des exercices de drill répétitif ;
- L'*approche communicative* (de 1980 à nos jours) : l'ordinateur aide les apprenants à utiliser leurs compétences communicatives dans des tâches authentiques [1]. L'accent est mis sur l'utilisation des formes et sur le sens plutôt que sur la forme elle-même. Il est incité à produire du langage plutôt que de le manipuler. La grammaire est enseignée de manière implicite. L'apprenant a un grand degré de contrôle sur le rythme d'apprentissage et sur l'interaction. Il peut former des hypothèses sur la langue et les confirmer ou les infirmer par la suite ;
- L'*approche intégrative* : grâce au multimédia et à *Internet*, l'écriture, la lecture, la compréhension et la production peuvent être combinée dans une seule activité. Les apprenants doivent accomplir une tâche communicative et disposent d'aides variées à l'apprentissage, comme des glossaires, des aides grammaticales, etc. Les ressources sont liées entre elles et l'apprenant a une grande autonomie dans son chemin d'apprentissage. Des outils de communication synchrone et asynchrone permettent aux apprenants d'échanger avec leurs pairs ou avec des locuteurs natifs de la langue qu'ils apprennent.

On trouvera de nombreux ouvrages présentant une histoire de l'ALAO comme Levy (1997) ou Davies (2003, 2006). Pour notre part, nous avons choisi une approche différente liée à l'histoire de l'informatique.

2.1.1 Les débuts

L'informatique est née dans les années 1930 avec les premiers calculateurs électromécaniques (Zanella et Ligier, 1989). Cependant, l'informatique n'a pris son essor que grâce aux transistors (1947) et aux circuits intégrés (1963). Grâce à cette dernière invention, la taille des machines s'est sensiblement réduite et les coûts ont également baissé, même si les ordinateurs restaient rares et coûteux. C'est pourquoi dans le monde académique, ces outils n'étaient accessibles qu'aux universités disposant de moyens conséquents. L'accès aux ordinateurs était plutôt réservé aux applications de cal-

1. Garrett (1995) définit un document authentique comme du matériel conçu à l'intention de locuteurs natifs dans un pays où la langue cible est parlée. Cette définition contraste avec un document produit pour un but pédagogique.

cul scientifique. Cependant, des projets pédagogiques sont nés relativement tôt dans l'histoire de l'informatique.

PLATO (*Programmed Logic for Automatic Teaching Operations*, Clancy, 1973; Hart, 1995, §B.2.27) est un projet qui s'est étendu du début des années 1960 à la fin des années 1970 à l'Université de l'Illinois. Tournant sur un gros ordinateur et des terminaux, il était destiné à fournir un apprentissage individualisé aux apprenants dans divers domaines, dont l'enseignement des langues. *PLATO* permettait de gérer les parcours des apprenants et bénéficiait des avancées technologiques de l'époque, dont la synthèse vocale (§3.1.4), mais la correction des exercices était rudimentaire, par reconnaissance de patrons (§3.1.1.1).

TICCIT (*Time-Shared, Interactive, Computer-Controlled Information Television*, Hendricks *et al.*, 1983; Jones, 1995, §B.2.36) a été développé dès 1971 à l'Université Brigham Young. Le projet était surtout basé sur de la vidéo interactive. Enfin *CALIS* (*Computer Assisted Language Instruction System*, Borchardt, 1995, §B.1.1) est né à la fin des années 1970 à l'Université Duke. C'est le premier logiciel doté d'un langage auteur (§2.7.2), qui permettait de créer facilement des exercices.

On peut constater que cette période a été peu prolifique, du fait que les logiciels nécessitaient des ordinateurs coûteux. De plus, les enseignants et pédagogues n'avaient que peu de marge de manœuvre dans la conception de type d'exercices et pour la création de nouveaux exercices. Les interfaces étaient limitées par une résolution faible des écrans et des possibilités limitées – voire inexistantes – de présenter les informations de manière graphique. Enfin, les apprenants n'étaient pas autant à l'aise dans l'utilisation des ordinateurs.

2.1.2 La révolution microinformatique

En 1971, la firme *Intel* lance le premier microprocesseur, qui regroupe les circuits sur une seule puce et permet donc un gain de place et un coût de production moins élevé. Dorénavant, de nombreux constructeurs se sont mis à produire des machines à faible coût, branchables sur un écran de télévision. En 1981, *IBM* sort le premier *Personal Computer* (PC) qui tournait sous le système d'exploitation en ligne de commande *MsDos*. En 1984, *Apple* produit l'ordinateur *MacIntosh*, dont la facilité d'utilisation et l'interface graphique ont connu un franc succès. D'autres systèmes concurrents existent mais disparaissent progressivement. Parmi les avancées technologiques notables, mentionnons encore le *modem*, qui permet de faire communiquer des

ordinateurs à travers une ligne téléphonique. Enfin, en 1991, *Microsoft* sort *Windows 3.1*, la première véritable interface graphique dans le monde PC.

C'est dans cette période qu'ont véritablement eu lieu les débuts de l'ALAO. En effet, les langages-auteur (§2.7.2) ou des langages de programmation tels que *BASIC* (§2.7.1) permettaient le développement de nombreux tutoriels et autres logiciels (Levy, 1997, p. 1). Les enseignants disposaient aussi d'un nombreux choix d'ordinateurs, dont certains, comme le *Commodore Amiga*, étaient dotés d'une synthèse vocale (§3.1.4). Ce grand choix s'est aussi mué en handicap, par manque de standards : un logiciel développé pour un ordinateur particulier n'était plus utilisable après la fin de production de l'appareil.

Les appareils de télématique de loisir (*Minitel* ou *VideoText*) ou les serveurs à distance de type *Compuserve* (§2.7.5) ont permis de développer des systèmes d'ALAO plus interactifs, malgré les coûts relativement élevés de communication et d'utilisation et les obstacles techniques à une utilisation en classe.

Comme exemples de logiciels de l'époque, mentionnons Helm et McIver (1974, §B.1.2), *BASIC Parser* (Cook, 1988, §B.2.2), *Herr Komissar* (DeSmedt, 1995, §B.3.5) ou les systèmes de Kempen (1992, §B.2.15). Enfin, signalons encore que, témoins du dynamisme et de l'essor de la branche de l'ALAO, les associations professionnelles CALICO (1983) en Amérique du nord, et EUROCALL (1987) en Europe, naissent à cette période.

2.1.3 La révolution du multimédia, de l'hypertexte et de l'*Internet*

Vers la fin des années 1980, les ordinateurs personnels haut-de-gamme commencent à être équipés de lecteurs de CD-ROM, qui permettent d'installer des programmes plus gourmands en espace disque bien plus facilement qu'avec des disquettes. Les premières cartes son et les normes de compression audio et vidéo font également leur apparition. La qualité des cartes graphiques et des écrans augmente. Enfin, la vitesse des processeurs et la capacité de mémoire des ordinateurs augmente fortement en parallèle avec une baisse des coûts. Les ordinateurs peuvent désormais combiner facilement audio, texte et vidéos. Ils deviennent alors *multimédias* [2]. En parallèle, en 1989, naît le *World Wide Web* sur le réseau *Internet* (§2.7.5) qui va révolu-

2. En 1995 sort le système d'exploitation *Windows* 95, qui est le premier système d'exploitation multimédia pour PC.

tionner le monde de l'enseignement, de l'information et de la communication. Divers progrès techniques diminuent fortement les coûts de connexion au réseau et augmentent la vitesse de transmission entre ordinateurs, permettant désormais un visionnement confortable de vidéos.

Pour l'ALAO, le multimédia offre la possibilité de disposer de documents variés, permettant d'exercer tous les aspects de la communication (compréhension, écriture, lecture, production). Quant à *Internet*, il offre une immense variété de documentation et permet aux apprenants de communiquer ou d'accéder beaucoup plus facilement à la langue qu'ils étudient. De plus, les logiciels d'ALAO peuvent être diffusés très facilement à travers *Internet*. Enfin, dans les pays développés, l'immense majorité des apprenants dispose d'ordinateurs personnels, souvent portables, et d'une connexion *Internet* permanente et performante à domicile.

2.1.4 Perspectives

Dans le domaine de l'informatique – et à plus forte raison de l'ALAO – il est hasardeux, voire risqué, de se lancer dans des prévisions. Nous voyons tout de même trois champs de développement.

Tout d'abord, l'accès au réseau *Internet* est appelé à se développer encore plus. Il est à prévoir que le débit des connexions *Internet* augmente fortement, avec l'arrivée de la technologie de la fibre optique jusque dans les domiciles privés. Ainsi, la diffusion de vidéos volumineuses sera grandement facilitée. Certains obstacles à la formation à distance seront alors abolis.

En second lieu, les téléphones mobiles deviennent des ordinateurs de plus en plus puissants. Il existe déjà de nombreuses applications pour ces appareils (Daniels, 2006; Chinnery, 2006), notamment en utilisant le *podcasting*. Nous reviendrons sur ce sujet au paragraphe 2.7.5.

Enfin, les ordinateurs deviennent de plus en plus puissants et la capacité des disques, des mémoires et des processeurs sont phénoménales. Ceci lève désormais de nombreux obstacles pour une utilisation plus intensive de l'intelligence artificielle (§2.7.4) et en particulier du traitement automatique des langues (§3).

2.2 Principes de conception d'un logiciel

Dans cette section, nous traitons des principes de conception d'un logiciel d'ALAO. Un programme d'ALAO devra être conçu en tenant compte de l'utilisation qui doit en être faite. On peut en effet concevoir un programme qui propose des documents et des activités autonomes, ou qui sert comme support à des activités en classe. Dans ce cas, l'enseignant aura un rôle à jouer. L'ordinateur peut aussi soulager l'enseignant de tâches répétitives, comme pour des exercices de drill qu'un ordinateur est capable de corriger de manière fiable. Un logiciel d'ALAO peut être vu de points de vue multiples :

i. celui de l'*apprenant* : il est au centre de l'attention du logiciel, avec son style d'apprentissage et ses besoins (Garrett, 1995; Labour, 2001; Defays et Deltour, 2003) ;
ii. celui de l'*enseignant* : le concepteur du logiciel ou l'enseignant qui choisit de l'utiliser en classe ou comme complément à domicile a également des besoins, des attentes et une expérience de l'enseignement et des erreurs fréquentes des apprenants ;
iii. celui de la *pédagogie* : le logiciel doit favoriser l'acquisition de connaissances, d'une part en présentant les connaissances de manière optimale et d'autre part en mettant à disposition des tâches et divers scénarios qui permettent d'atteindre les objectifs de manière efficace, quel que soit le style d'apprentissage de l'apprenant ;
iv. celui du *contenu* : le matériel pédagogique à disposition des apprenants ;
v. celui de la *technique* : le logiciel peut être disponible sur une ou plusieurs plateformes informatiques, sur *Internet*, disposer de diverses interfaces, de divers outils d'aide à l'apprentissage ou de diagnostic d'erreur, etc. ;
vi. celui du *contexte d'utilisation* : destiné à l'apprentissage individuel, en classe, en autonomie (Blin, 1999) ou collaboratif [3], avec ou sans supervision d'un enseignant ou d'un tuteur [4].

L'élaboration d'un bon logiciel d'ALAO repose sur une équipe pluridisciplinaire : spécialistes en pédagogie, spécialistes du domaine du logiciel, graphistes, informaticiens, spécialistes du multimédia et de l'acquisition des langues (Demaizière et Dubuisson, 1992; Tschichold, 2006). Souvent, les

3. Certains enseignants détournent un logiciel de l'utilisation prévue par son concepteur ou son éditeur pour d'autres scénarios d'utilisation.
4. Dans le cadre de l'apprentissage à distance, un tuteur est un enseignant de référence qui prend en charge un petit nombre d'apprenants.

concepteurs de logiciels manquent de compétences dans l'un ou l'autre domaine et tentent d'appliquer les nouveautés technologiques sans grande réflexion. D'autre part, par manque de connaissances globales du domaine, ils réinventent la roue ou tombent dans des pièges que d'autres ont déjà rencontrés auparavant. A l'inverse, les enseignants expérimentés font des logiciels très pointus et adaptés aux besoins de leurs propres apprenants mais sur des points très spécifiques et sans faire appel aux technologies les plus récentes.

Le processus de conception d'un logiciel est très complexe et souvent cyclique (Demaizière et Dubuisson, 1992; Levy, 1997, 1999). Les grandes étapes sont :

i. analyse des besoins, définition d'un public cible, étude de marché, identification des prérequis, spécification des objectifs pédagogiques, choix de la plateforme informatique, fixation du temps imparti pour l'accomplissement des tâches par l'apprenant, etc. Cette étape est parfois réalisée grâce à des questionnaires ;
ii. scénario pédagogique détaillé ;
iii. maquette complète de l'interface et des détails ;
iv. réalisation ;
v. test complet du logiciel et corrections ;
vi. documentation et maintenance du logiciel.

Dillenbourg (1993) liste trois erreurs souvent commises par des concepteurs inexpérimentés de logiciels. La première est de concevoir un didacticiel comme un livre, sans que l'apprenant ait à résoudre une tâche. La seconde est de croire que des présentations animées sont suffisantes pour faciliter l'apprentissage ; l'apprenant doit au contraire résoudre des problèmes, acquérir des connaissances, mais aussi des capacités et des automatismes. La dernière erreur est de ne pas varier les tâches et les contextes.

Les tâches effectuées dans un logiciel doivent être aussi réalistes que possible et présenter des situations variées auxquelles l'apprenant sera confronté. Le réalisme des tâches présente une motivation intrinsèque supplémentaire pour les apprenants.

La structure d'un logiciel joue un grand rôle. Certains ont un seul point d'entrée et d'autres en proposent plusieurs. Il existe aussi des parcours plutôt linéaires, avec une progression des difficultés, ou des parcours plus libres. Certains scénarios sont directifs ou laissent au contraire plus de liberté aux apprenants (Wenger, 1987; Hambuger *et al.*, 1999). Swartz (1992) souligne le besoin de progression dans le processus d'apprentissage et de consolidation

des connaissances par les exercices variés qui font ressortir les caractéristiques principales des formes enseignées.

Du point de vue des matériaux pédagogiques, il est nécessaire de se poser la question de la correspondance et de l'adéquation des matériaux, contenus et outils techniques d'apprentissage avec les théories pédagogiques (Delcloque, 1998, p. 2). Le choix des matériaux est déterminant pour la motivation des apprenants (Duchiron, 2003). Il est important de montrer la langue telle qu'elle est utilisée et de l'enseigner en contexte, à l'aide de documents authentiques, oraux (sons et vidéos) ou écrits (Laurier, 2000) [5]. D'un autre côté, on soulignera l'importance des matériaux d'explications pédagogiques et des aides à l'apprentissage.

2.3 Théories pédagogiques

Dans cette section, nous décrivons rapidement les théories du béhaviorisme, du socio-constructivisme et de la collaboration. Les théories de l'apprentissage ont été considérablement influencées par le *béhaviorisme* (Skinner, 1954). Cette théorie se base sur le renforcement des connaissances (*renforcement*). L'acquisition des connaissances est vue comme un procédé mécanique. L'apprenant est exposé à des stimuli dont la répétition renforce les connaissances. Celles-ci sont vues comme des habitudes. Skinner prône l'individualisation de l'enseignement à travers des machines à enseigner. L'enseignement doit alors déclencher une réaction chez l'apprenant, de sorte à modeler son comportement futur. Le contenu de la connaissance est fragmenté en pas discrets, de manière très directive et les apprenants reçoivent un *feedback* (rétroaction, §2.4) immédiat, pour le renforcement des connaissances (Levy, 1997).

Vygotsky (1962) a élaboré la théorie du *socio-constructivisme*, basé sur le *constructivisme* (Piaget, 1969) : l'individu est inséparable de son milieu social ; par conséquent la pensée et la connaissance se développent en fonction de conditions sociales et historiques particulières. Ainsi, la langue et la culture influencent le développement cognitif, car elles ont un effet sur la manière qu'ont les humains d'appréhender, d'expérimenter, de communiquer et de comprendre la réalité et de se comporter socialement (Oxford, 1995). La connaissance est donc un processus de co-construction de savoir au cours d'une activité sociale.

La langue s'apprend par une régulation sociale puis par l'auto-régulation.

5. Pour récolter des matériaux, on se heurte souvent au problème du copyright.

Vygotsky postule l'existence d'une *zone de développement proximal* :

> "[The Zone of Proximal Development is] the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance, or in collaboration with more capable peers." (Vygotsky, 1962)

Appelée également *zone de croissance* (Oxford, 1995), cette zone peut être définie comme un ensemble de connaissances non encore acquises par l'apprenant mais qu'il a la capacité d'assimiler, avec une aide externe et des tâches de renforcement. Ces tâches doivent être intéressantes et d'une difficulté progressive. Elles doivent attirer l'attention vers les points critiques et cibler vers la forme correcte. Le but est de permettre à l'apprenant de les appliquer de manière autonome.

L'*apprentissage collaboratif* part de l'observation que les apprenants en tandem (ou en plus grands groupes) obtiennent des résultats supérieurs à ceux des apprenants qui résolvent un problème tout seuls (Renié et Chanier, 1995; Dillenbourg *et al.*, 1996; Mendelsohn et Jermann, 1997; Chapelle, 2003). Chaque étape de la résolution d'un problème doit être discutée et le raisonnement doit être précisé, révisé ou réfuté. Le fait de verbaliser ses connaissances amène l'apprenant à les renforcer en les appliquant à un contexte précis en expliquant son raisonnement à son/ses partenaires (Dillenbourg, 1994). Cette interaction entre participants peut être permanente, ou alors les tâches sont réparties entre les apprenants et les résultats de chacun sont ensuite assemblés. Au niveau cognitif, les apprenants se partagent la charge : l'apprenant au clavier ou à la souris doit se concentrer sur ses gestes et sur la frappe au clavier, tandis que l'autre ou les autres peuvent se concentrer sur le résultat. C'est pourquoi, spécialement dans le cadre de l'étude des langues, il est important de s'assurer de l'alternance des rôles, par exemple lors de la rédaction d'un texte. Il est également nécessaire de s'assurer de l'homogénéité des groupes (Dillenbourg et Schneider, 1995), au niveau des connaissances, de l'âge et d'autres facteurs distinguant les participants.

2.4 Feedback

La rétroaction (ou *feedback*) est un mécanisme qui traite une entrée et effectue une action ou apporte une réponse en sortie (Heift et Schulze, 2007). Dans le domaine de l'ALAO, il s'agit de traiter une réponse de l'apprenant

et de l'évaluer de manière adéquate afin de faciliter la remédiation et le processus d'apprentissage. Selon les théories socio-constructivistes, l'apprenant doit être actif dans le processus de remédiation (Chapelle, 1998) et il ne faut pas immédiatement donner la solution correcte [6]. Le feedback doit être adapté au niveau de l'apprenant et ne doit pas l'induire en erreur [7].

Le feedback le plus basique consiste à signaler simplement si le résultat est juste ou faux. On distingue généralement une rétroaction *intrinsèque*, inhérente à la tâche, et la rétroaction *extrinsèque*, qui apporte des informations additionnelles. On distingue de nombreuses variantes de feedback :

- reformulation : le système énonce la phrase correcte, par exemple dans un dialogue, en accentuant le point où l'erreur a été commise ;
- feedback direct ou indirect, où l'apprenant est invité à vérifier certains points ou certaines règles, ou doit faire des exercices de remédiation ;
- feedback écrit, oral ou visuel ;
- feedback progressif : partant du général, on précise de plus en plus le feedback si l'apprenant ne trouve pas ;
- feedback immédiat ou retardé ;
- feedback contrôlé par le système ou déclenché par l'apprenant ;
- toutes les erreurs en même temps ou une erreur à la fois ;
- toutes les erreurs sont signalées ou seulement la ou les plus importantes d'après le niveau de l'apprenant.

Lister et Ranta (1997) proposent les types de feedback suivants, dans le cadre de l'enseignement oral par des êtres humains :

i. correction explicite : l'erreur est signalée et corrigée ;
ii. reformulation : la phrase est reformulée sans erreur ;
iii. demandes de clarification : un signe de mécompréhension (*pardon ?*) ou une question (*que voulez-vous dire par ... ?*) ;

6. Cependant, Allen (1997) prône que les erreurs sans importance soient corrigées automatiquement et ne donnent pas lieu à des pénalités. Les signes de ponctuation manquants ou superflus ou les majuscules manquantes non significatives entrent dans cette catégorie.
7. Demaizière et Dubuisson (1992) soulignent qu'il peut être perturbant pour un apprenant de voir un message du type "je ne comprends pas votre réponse" car il peut induire l'illusion que la machine est intelligente, alors que la plupart du temps, elle signifie que le concepteur du programme ou de l'exercice n'a pas anticipé ou pris en compte la réponse de l'apprenant. Cette illusion peut être accentuée par l'emploi de personnages animés dans les logiciels qui semblent dialoguer avec l'apprenant.

iv. feedback métalinguistique : commentaires, questions ou indication d'une erreur sans donner de correction (*il y a une erreur*, *pouvez-vous découvrir votre erreur?*, *X est au masculin*, etc.) ;
v. incitation (*elicitation*) : l'apprenant doit compléter la phrase (*c'est un...*), ou répondre à une question (*comment ça s'appelle?*, *how do we say X in French?*, etc.) ;
vi. répétition : la phrase est répétée en accentuant le lieu de l'erreur ;

Parfois les enseignants combinent ces différentes stratégies. La reformulation n'est pas efficace car elle n'incite pas l'apprenant à s'autocorriger, ni la correction explicite. Les autres stratégies sont bien plus efficaces et favorisent l'assimilation (Lister et Ranta, 1997). Heift et Schulze (2007) soulignent qu'il est important de prendre en compte le facteur temps. Une erreur déjà commise dans une session bien antérieure pourrait avoir moins de poids qu'une erreur commise dans la même séance.

2.5 Méthodologies d'évaluation des logiciels

Il est essentiel d'évaluer un logiciel de plusieurs points de vue. Tout d'abord, lors de son élaboration, un logiciel subit un processus de révision pour corriger son contenu. Dans l'idéal, le concept pédagogique devrait être évalué – par exemple sur une première section complète du logiciel – par des apprenants et des enseignants. Lorsqu'un logiciel est terminé, on l'évalue du point de vue de l'efficacité pédagogique et du contenu.

Jamieson *et al.* (2004) énumèrent des principes d'évaluations pour un logiciel. Le premier principe est le *potentiel d'apprentissage*. Il est évalué selon trois critères :

- *mise en valeur du contenu*. Ceci peut être fait selon trois modes, qui peuvent être évalués : (a) la mise en évidence du contenu, par la forme ou par la répétition ; (b) la modification du contenu par l'entremise d'aides à l'apprentissage (images, traductions, dictionnaires, simplifications...) ; (c) l'élaboration du contenu, par l'ajout d'appositions, de propositions relatives etc.
- *interaction*. L'interaction se passe tant avec la machine qu'avec d'autres personnes.
- *production*. L'apprenant doit produire des phrases ou des mots. Ce critère englobe aussi les activités de planification avant la production

proprement dite (recherche d'information), la correction de la production et les aides pendant la production.

Le second principe porte sur la *qualité de l'évaluation* des apprenants. Les activités évaluées doivent être authentiques et correspondre au matériel présenté dans les tutoriels. De plus, les évaluations doivent aider à l'apprentissage et les scores doivent refléter correctement les résultats d'apprentissage (le nombre d'éléments évalués doit notamment être suffisant pour être représentatif de l'apprentissage).

Chapelle (2001) propose quant à elle un cadre d'évaluation des logiciels selon les critères suivants :

- potentiel d'apprentissage d'une langue, c'est-à-dire la plus-value pédagogique qui distingue un simple outil permettant de pratiquer une langue d'un véritable logiciel qui enseigne des notions ;
- adaptation à l'apprenant : le logiciel doit être adapté au niveau visé des apprenants ;
- focalisation sur le sens : l'attention de l'apprenant doit être attirée vers le sens de la langue qui est requise pour accomplir une tâche ;
- impact : les activités doivent permettre de développer les stratégies métacognitives des apprenants, de sorte à ce qu'ils puissent les appliquer également en dehors des activités d'apprentissage ;
- authenticité : les contenus et les activités doivent être proches d'une situation authentique ;
- praticité : les logiciels doivent être simples d'utilisation afin que l'apprenant puissent se concentrer sur les tâches d'apprentissage.

Selon Rüschoff (1998), les évaluations portent tantôt sur des fonctionnalités (débogage), tantôt sur l'aspect formatif. Les données peuvent être recueillies automatiquement par le logiciel, ou au moyen de pré- et post-tests. Parfois, les évaluateurs demandent aux utilisateurs de verbaliser les actions qu'ils effectuent, les réflexions qu'ils mènent etc. (*think aloud*) ; ces commentaires peuvent aussi être recueillis par écrit au moyen d'un journal de bord. Les données sont parfois qualitatives, parfois quantitatives (*student tracking*, enregistrement des interactions et réponses de l'apprenant). L'évaluation qualitative est beaucoup plus difficile à réaliser.

2.6 Méthodes d'évaluation automatique des connaissances

Dans cette section, nous commençons par les Questionnaires à Choix Multiples (§2.6.1). Nous poursuivons avec d'autres méthodes courantes d'évaluation automatique (§2.6.2) et terminons par une discussion de conclusion (§2.6.3).

2.6.1 Questionnaires à Choix Multiples

Les *Questionnaires à Choix Multiples* (QCM) sont la méthode la plus courante pour tester automatiquement les connaissances des apprenants par ordinateur. Pour chaque question, plusieurs réponses possibles sont listées. Une ou plusieurs réponses sont correctes. En général, pour obtenir le maximum de points, toutes les réponses correctes doivent être données et aucune réponse incorrecte ne doit être sélectionnée.

Cette méthode est facilement implémentable, d'où sa popularité. Les candidats sont évalués pour leurs connaissances et non pas pour leur capacité de s'exprimer par écrit. Mais d'un autre côté, les QCM ont une valeur pédagogique limitée car les apprenants ne doivent pas justifier leur réponse, sauf rares exceptions. Ils peuvent donc cocher une réponse (ou plusieurs) au hasard. S'il est permis de faire plusieurs tentatives et/ou de revenir en arrière, ils peuvent facilement améliorer leur score. De plus, si les apprenants ont une connaissance partielle du sujet, ils n'auront aucun point, alors qu'une réponse en texte libre peut être corrigée de manière nuancée et certains éléments de réponse peuvent être trouvés et compter comme une réponse partielle.

Plusieurs solutions existent pour éviter ces pièges :

- le retour en arrière est bloqué, le résultat de la réponse n'est disponible qu'à la fin du questionnaire, voire après la validation des résultats par l'enseignant ;
- plusieurs questions abordent le même sujet ou un sujet similaire. Cela diminue le facteur chance car celui qui répond au hasard aura moins de chances de cocher deux fois la bonne réponse ;
- on introduit des réponses inexactes mais proches de la solution, appelées *distracteurs* ;

– le nombre de réponses par question est varié, les réponses correctes n'apparaissent pas toujours dans la même position.

Dans le domaine de l'enseignement des langues, l'intérêt des QCM est relatif. Ils permettent de tester les connaissances de vocabulaire et la capacité à distinguer les formes conjuguées ou déclinées. Des questions de compréhension d'un texte ou d'une séquence audio ou vidéo sont une autre application. Mais ils ne permettent pas de tester les capacités productives, qui sont essentielles dans ce domaine.

Keller (2000, ch. 5) liste quelques problèmes que l'on peut rencontrer avec un test sur le *web*. Les conditions de l'expérimentation peuvent varier au niveau de l'environnement (bruit et autres types de distraction pouvant déconcentrer le sujet) et au niveau de la saisie des données. Il est important d'empêcher l'apprenant de revenir en arrière, de relire les données une fois la réponse donnée. Il est aussi important que les réponses ne puissent être données qu'une seule fois. Une mesure du temps de réponse est important afin d'éliminer les réponses trop rapides qui indiquent alors un manque de soin.

2.6.2 Autres méthodes d'évaluation automatique

Pour l'enseignement des langues, le texte à trous est un outil très répandu. Des mots sont retirés d'un texte et remplacés par une ligne ; l'apprenant doit les restituer. Plusieurs variantes sont possibles :

– les mots manquants sont listés et l'apprenant doit les mettre au bon endroit ;
– les mots manquants sont listés et mis dans une forme canonique. L'apprenant doit alors placer au bon endroit le verbe, nom etc. et le conjuguer ou l'accorder correctement ;
– l'apprenant doit taper la phrase au clavier ou glisser-déplacer des boîtes étiquetées avec les mots ou segments manquants.

On peut aussi utiliser des exercices d'appariement entre une liste de phrases ou segments de phrases et une autre liste de phrases ou de segments de phrases. Par exemple, une des listes peut contenir des propositions subordonnées et l'autre des phrases principales. Il est possible aussi de créer des exercices de compréhension, avec un texte d'un côté et des images correspondantes de l'autre. On peut corser la difficulté de l'exercice en n'ayant

pas le même nombre d'éléments dans les deux listes, afin d'éviter les effets du hasard.

2.6.3 Discussion

La plupart des logiciels d'apprentissage de langues se basent uniquement sur des techniques d'évaluation de base, bien que leurs fonctionnalités soient limitées. Les réponses fermées ont une application restreinte aux exercices de vocabulaire, de compréhension ou aux exercices de morphologie (déclinaison, conjugaison etc.). Les réponses ouvertes ne peuvent être évaluées que grossièrement par reconnaissance de patrons (§3.1.1.1) et la rétroaction ne permettra pas à l'apprenant de déterminer la cause de son erreur. Il est donc préférable d'utiliser des méthodes de traitement automatique des langues (TAL), qui examinent les productions des apprenants avec plus de finesse, ce qui permet un diagnostic plus détaillé et plus fiable. Nous évoquerons au chapitre 3 des techniques permettant d'évaluer la réponse de l'apprenant sur le plan syntaxique et même sémantique.

2.7 Types d'ALAO et outils

Mendelsohn et Jermann (1997) proposent la typologie suivante pour l'Enseignement par Ordinateur.

- *didacticiels* : l'apprenant doit résoudre un problème de manière plus ou moins active, à travers des activités plus ou moins fermées ;
- *progiciels* : logiciels d'utilisation standard comme les traitements de textes, tableurs, palettes graphiques, etc., réutilisés pour des activités pédagogiques ;
- *tutoriels* : présentation des matières à enseigner et guidage de l'apprenant afin de l'aider à résoudre les problèmes ;
- *environnements d'apprentissage intelligents* : tutoriels dotés d'intelligence artificielle (§2.7.4) afin de guider le processus d'apprentissage, aider l'apprenant à résoudre les problèmes et poser un diagnostic sur ses erreurs ;
- *micromondes* : environnement généralement graphique simulant une situation réelle où l'apprenant interagit avec des agents artificiels pour accomplir une tâche.

Marquet (2004) propose la typologie suivante :

i. tutoriels : apprentissage cognitiviste, connaissances présentées de manière ordonnées, l'apprenant doit lire ;
ii. exerciseurs : approche béhavioriste, dispense les exercices ;
iii. jeux éducatifs : même principe, mais on capte l'attention par le jeu ;
iv. tuteurs intelligents : représentation des connaissances, l'apprenant doit assimiler la matière ;
v. hypermédias : cognitivistes et constructivistes, la connaissance est en accès libre et l'apprenant explore ;
vi. micromondes et simulateurs : constructivistes, la connaissance est modélisée ou matérialisées, l'apprenant construit, manipule ou observe ;
vii. logiciels d'apprentissage collaboratif : la connaissance est construite entre apprenants.

Dans la suite de cette section, nous évoquons successivement les techniques de base (§2.7.1), les logiciels-auteur (§2.7.2), les micromondes (§2.7.3), les systèmes intelligents (§2.7.4) et l'utilisation des réseaux d'ordinateurs comme *Internet* (§2.7.5).

2.7.1 Techniques de bases

Aux débuts de la microinformatique et de l'ALAO, les applications étaient toutes relativement limitée par les interfaces rudimentaires et le manque de mémoire et de puissance. De nombreux constructeurs de micro-ordinateurs (Commodore, Apple, Atari, Sinclair, BBC, etc.) intégraient directement un interpréteur du langage *BASIC* (*Beginners All-purposes Symbolic Instruction Code*) qui avait l'avantage de la simplicité. Il n'est donc pas étonnant que de nombreux logiciels d'ALAO datant de cette époque aient été programmés en *BASIC*. On peut citer *CALIS* (Borchardt, 1995, §B.1.1), *BASIC Parser* (Cook, 1988, §B.2.2), *Generate* (Hackenberg, 1985, §B.2.9), etc. Farrington (1982) utilise le langage *Fortran* pour des exercices de français.

D'autres préfèrent réutiliser des programmes courants comme les logiciels de traitement de texte, les tableurs et les outils de base de données pour créer leurs propres logiciels (Demaizière et Dubuisson, 1992; Mendelsohn et Jermann, 1997), notamment en utilisant le langage de macrocommandes intégré dans ces produits pour créer des exercices simples. Ariew (1982) utilise un logiciel de traitement de textes associé à un système de fichiers pour générer des tests de langue et en recueillir les résultats imprimés. Desmarais et Bisaillon (1998) décrivent l'utilisation du correcteur orthographique d'un

traitement de textes à des fins didactiques. Certains outils de traitement de texte ont été spécialement dotés d'outils d'aide à l'écriture et à la révision dans un but pédagogique. C'est par exemple le cas de *HBJ Writer* (D'Aoust, 1990), aussi connu sous son ancien nom de *WANDAH*. Burston (1993) ou Phinney (1996) décrivent l'utilisation de programmes de base à des fins pédagogiques. Les logiciels de traitement de texte peuvent être utilisés pour des exercices de vocabulaire, de transformation de phrases, d'exercices à trou etc. Pour les apprenants de niveau moyen, les traitements de textes peuvent fournir des exercices de post-édition de textes (tâche plus facile que la création de textes) ou la rédaction de petits textes. L'utilisation des dictionnaires de synonymes et des vérificateurs orthographiques est bénéfique. D'autres fonctions sont utiles, comme l'insertion de commentaire ou de marque de révision, dont sont dotées la plupart des logiciels de traitement de textes, que ce soit pour les apprenants dans une tâche collaborative (§2.3) ou pour les enseignants.

Enfin Glencross (1995) décrit les utilisations possibles du dictionnaire *Robert Electronique* sur CD-ROM. Outre les exercices d'orthographe, on peut créer des exercices sémantiques sur les relations des mots (synonymes, antonymes, etc.). Les citations permettent également de repérer les différents contextes d'emploi des mots. L'utilisation de caractères de remplacement permet de faire des recherches morphologiques sur les suffixes comme *-ment*, *-tion* etc. Les relations de synonymie permettent de traiter des phénomènes de registre, puisque *argent* mène aux mots argotiques *grisbi, fric, pèze* etc.

2.7.2 Logiciels-auteur

Murray (1999) définit un logiciel-auteur comme un outil qui doit permettre à des non-programmeurs de réaliser un logiciel sans connaissances en programmation, ni pour la présentation du contenu et des exercices, ni pour les connaissances du système expert ; en outre, le système doit aider l'enseignant à construire le logiciel selon des principes de bonne conception. Parmi les logiciels commerciaux, citons *Auhtorware*, *Director* et *Storyboard*[8].

Par ailleurs, des projets scientifiques ont développé leur propre logiciel-auteur. *Fun With Texts* (Glencross, 1993) est un logiciel qui utilise un texte annoté manuellement avec des fonctions grammaticales. Les apprenants doivent reconstruire le texte en retrouvant les mots manquants, ou replacer des mots d'une liste dans un texte à trous. Demaizière (1982) présente *Ordinateurs Pour Etudiants* (*OPE*), un système d'enseignement développé dès

8. Nous parlerons du logiciel *HotPotatoes* au §2.7.5.

1969, doté notamment d'un module de grammaire anglaise, où les apprenants doivent traduire des phrases simples en français. *OPE* fonctionne grâce à deux langages-auteur, l'un pour l'analyse des réponse des apprenants, l'autre pour gérer les exercices. *MALTED* [9] (Bangs et Shield, 1999, *Multimedia Authoring programme for Language Tutors and Educational Development*,) est une série d'outils de création d'exercices à l'aide de ressources multimédia réutilisables stockés dans une base de données. Toole et Heift (2002) ont développé *Tutor Assistant*, un logiciel-auteur qui permet d'adapter les contenus et les exercices du logiciel intelligent *ESL Tutor* (§B.4.23) ou de créer de nouveaux contenus. *IDIOMA-TIC* (Desmet *et al.*, 2004; Desmet et Héroguel, 2005, §B.1.3) est un logiciel-auteur qui sert au développement d'exercices de langue sur le *web*, principalement axé sur des questions semi-ouvertes, tels que des exercices de traduction, de correction, de reformulation, de dictée pour étudiants avancés. Grâce à une technique de reconnaissance de patrons (§3.1.1.1), les exercices sont corrigés et reçoivent une rétroaction appropriée. Les rétroactions possibles sont de type général (renvoi à la règle), ciblé (focalisé sur des éléments de l'exercice) ou focalisé sur l'erreur commise. Le système est doté d'un système de traçage des performances des apprenants.

On trouvera quelques descriptions de logiciels-auteurs au §B.1.

2.7.3 Micromondes

Les micromondes sont des environnements, généralement graphiques, dans lesquels les apprenants doivent effectuer des actions et résoudre des problèmes. Le logiciel de micromonde le plus connu est *LOGO* (Papert, 1970; Harvey, 1986; Wenger, 1987, p. 124 s.) et est destiné d'une part à faire découvrir les notions de géométrie et d'orientation sur un plan, et d'autre part à apprendre aux apprenants à subdiviser des tâches pour atteindre un but final. A l'aide d'un langage de commande très simple, l'apprenant faisait se déplacer un curseur en forme de tortue. En se déplaçant, la tortue pouvait tracer des traits ou ne pas laisser de trace. Des procédures pouvaient être écrites pour les tâches répétitives. Les apprenants pouvaient utiliser des boucles d'itération et des instructions conditionnelles [10].

Selon Danna (1997, p. 10), les micromondes transmettent les notions à

9. `http://www.malted.com/`, consulté le 27 août 2006.
10. Sous *Windows*, le logiciel *MSWLogo* est un émulateur *LOGO* qui permet non seulement la réalisation de petits programmes graphiques mais aussi de programmes complexes. Le logiciel d'apprentissage d'orthographe et de grammaire *L'Orthophile* (§B.5.17) fonctionne sous cet environnement.

acquérir par des actions et facilitent la transmission des connaissances par "*l'immersion de l'utilisateur dans un contexte qui facilite la compréhension et la mémorisation de données relatives à ces actions.*"

Les micromondes permettent parfois à l'apprenant d'interagir avec des agents virtuels, humains ou non, en deux ou trois dimensions. Morton et Jack (2005) décrivent les interactions possibles avec l'apprenant :

- montrer à l'apprenant comment réaliser une tâche ;
- guides de navigation dans un environnement complet ;
- attirer l'attention de l'apprenant sur des éléments de la scène qui donnent des informations ou indices, par le regard ou par des gestes ;
- donner un *feedback*, verbal ou non verbal, par rapport aux actions ou productions de l'apprenant ;
- émettre les signaux conversationnels habituels dans une interaction d'humain à humain, comme la prise de parole, l'expression d'une opinion personnelle ou la réaction à l'avis de l'apprenant ;
- expression de l'émotion et suscitation de l'émotion de l'apprenant, comme moyen d'accentuer la motivation à apprendre ;
- simulation d'une situation d'apprentissage en groupe, soit en tant que membre du groupe, soit comme substitut de l'enseignant.

On trouvera des descriptions de logiciels à l'annexe B.3.

2.7.4 Systèmes intelligents

L'*Intelligence Artificielle* (IA) désigne les techniques qui simulent le comportement humain pour résoudre des problèmes et/ou acquérir de nouvelles connaissances à partir de nouvelles données ou de nouvelles situations. On appelle *systèmes intelligents* les logiciels dotés d'intelligence artificielle. Il est nécessaire d'élaborer une représentation adéquate des connaissances nécessaires pour permettre à l'ordinateur de les manipuler.

Dans le domaine de l'ALAO, les systèmes intelligents sont capables de guider un apprenant dans un parcours pédagogique approprié, d'identifier ses erreurs et d'y apporter une remédiation afin d'améliorer ses connaissances [11]. En somme, ils regroupent les connaissances d'un expert tant sur le plan du

11. On parle alors d'ALIAO (*Apprentissage des Langues Intelligemment Assisté par Ordinateur*), en anglais ICALL (*Intellligent Computer-Assisted Language Learning*).

domaine lui-même que sur le plan pédagogique (Wenger, 1987; Matthews, 1992).

Ces systèmes intelligents d'apprentissage des langues sont dotés de plusieurs modules qui interagissent et communiquent entre eux, comme pour tout logiciel doté d'intelligence artificielle, qui sont dans la plupart des cas les modules suivants :

- Module expert ;
- Module de l'apprenant ;
- Module pédagogique ;
- Module de communication.

Nous décrivons à présent les différents modules dont sont en général dotés les systèmes intelligents.

2.7.4.1 Module expert

Le module expert comporte la connaissance du domaine à enseigner et doit être capable de résoudre les problèmes posés à l'apprenant, de manière psychologiquement et cognitivement plausible. Il peut être à la fois la *source de la connaissance à enseigner* et une *base de règles pour évaluer les connaissances* des apprenants et pour *résoudre les problèmes posés* (Wenger, 1987, p. 14). La tâche de modélisation de l'intelligence humaine demande des efforts conséquents, même pour un domaine limité (Bailin, 1995, p. 327).

Il existe plusieurs manières de représenter des connaissances pour un système expert. D'après Demaizière et Dubuisson (1992), le système comprend généralement des connaissances *descriptives* ou *conceptuelles* (bases de *faits*), des connaissances *opératoires* (bases de *règles*) et de connaissances *stratégiques* (*heuristiques* ou *métarègles* qui dirigent la résolution du problème en orientant le choix des règles prioritaires). Les connaissances peuvent être représentées de manière *procédurale* ou *déclarative* (Danna, 1997, pp. 20-21) : les connaissances *déclaratives* sont un ensemble de déclarations, qui peuvent être ensuite soumises à un mécanisme de raisonnement, appelé généralement *moteur d'inférences*, qui manipule les déclarations pour inférer de nouvelles connaissances. Par contre, si l'on veut plutôt s'attacher à l'ordre d'application des déclarations, alors une modélisation procédurale est plus adéquate ; elle consiste à décrire les procédures et l'enchaînement de celles-ci pour arriver à un résultat.

Le module expert doit être capable d'identifier les erreurs commises et de montrer le ou les raisonnements qui l'amènent à la solution (Wenger, 1987; Dillenbourg, 1994; Dillenbourg et Schneider, 1995). Certains systèmes ne montrent que la meilleure solution possible, d'autres listent toutes les solutions. Certains affichent toutes les étapes de raisonnement alors que d'autres ne donnent que le résultat. Dans l'idéal, les stratégies utilisées devraient uniquement utiliser des règles et stratégies déjà connues par l'apprenant, qui sont un sous-ensemble des connaissances du système expert (Danna, 1997).

Dans le cas de l'ALIAO, les connaissances du système expert sont par exemple les règles de grammaire, un lexique et une liste des erreurs potentielles des apprenants. Le chapitre 3 traitera en détail des techniques utilisées pour analyser la langue, en particulier dans un contexte d'apprentissage.

2.7.4.2 Module de l'apprenant

Le module de l'apprenant contient des informations sur l'apprenant, son parcours à travers le logiciel, son niveau de connaissance et les stratégies d'apprentissage et de remédiation qu'il met en œuvre (Wenger, 1987; Dillenbourg, 1989; Chanier *et al.*, 1992; Danna, 1997; Hambuger *et al.*, 1999; Kang et Maciejewski, 2000; Heift et Schulze, 2007). La représentation des connaissances et des erreurs des apprenants est généralement similaire à celles des règles du système expert. Elle peut être constituée de règle d'erreurs, qui représentent les erreurs fréquemment observées chez les apprenants. Le modèle de l'apprenant est souvent constitué grâce à un prétest, passé avant l'utilisation du logiciel proprement dit. Dans l'idéal, ce modèle est actualisé en permanence.

2.7.4.3 Module pédagogique

Le module pédagogique est aussi appelé module d'enseignement, module de diagnostic ou module tuteur. Il gère le parcours et les objectifs pédagogiques en proposant les exercices en fonction des difficultés, des points traités et des besoins de l'apprenant, de sorte à lui faire acquérir les connaissances nécessaires pour résoudre un problème. "*Il doit décider du prochain exercice à soumettre à l'élève (contenu, difficulté, etc.), des conseils à lui fournir, du moment où ses interventions seront les plus pertinentes pour l'apprenant, etc. Pour prendre de telles décisions, il s'appuie entre autres sur des connaissances décrivant le comportement d'un tuteur humain en terme de tactiques pédagogiques.*" (Danna, 1997, p. 16). Outre la séquence des expli-

cations et des exercices, le module pédagogique gère parfois le diagnostic des erreurs de l'apprenant.

Dans le cas de l'ALIAO, les capacités à acquérir sont de communiquer, oralement ou par écrit, et de comprendre un énoncé oral ou écrit. Selon le concept pédagogique sous-jacent, la réaction du système en cas de problème sera immédiate ou différée ; le parcours pédagogique peut être opportuniste (selon les difficultés rencontrées ou les besoins) et ne pas prévoir d'action à long terme (Danna, 1997). Au contraire, le parcours peut être planifié ou dirigé et prévoir un plan d'interactions successives, qui peuvent être révisées en fonction des difficultés rencontrées par l'apprenant. Enfin, on trouve aussi des solutions intermédiaires entre les deux.

2.7.4.4 Module d'interface

Le module d'interface ou de communication gère le dialogue entre l'apprenant et le logiciel et la manière dont les informations sont présentées. L'acceptation du logiciel par les apprenants dépend aussi de la qualité du module d'interface (Wenger, 1987).

Ce module est souvent intégré dans le module pédagogique. Il est essentiel de représenter les connaissances de l'apprenant ou du système de manière compréhensible par l'autre partie (Danna, 1997, p. 17). L'ergonomie du logiciel et la qualité des dialogues et de l'interaction doivent être par conséquent soignés. Le style peut être sous forme de dialogue ou plus sous forme de guide plus ou moins dirigiste.

Le module d'interface doit parfois être dotée de capacités conversationnelles (Wenger, 1987). Dans ce cas, des techniques de traitement du langage sont requises, tant pour les messages produits par le système (génération, §3.1.5) que pour la compréhension des phrases produites par l'apprenant (analyse syntaxique, §3.3, et sémantique, §3.4).

2.7.4.5 Interaction des modules

Ces quatre modules doivent interagir, par exemple à l'aide de la technique du *tableau noir*, où les différents modules mettent leurs informations à disposition des autres modules, qui peuvent les infirmer ou les confirmer. L'évaluation des connaissances par le module expert sert au module pédagogique pour déterminer le parcours. Le module de l'apprenant indique au

module pédagogique et au module expert quelles sont les connaissances déjà acquises par l'apprenant, de sorte que de nouveaux contenus soient progressivement introduits et que la résolution des problèmes n'utilise que des stratégies déjà connues. Cette modélisation du comportement de l'enseignant humain permet d'individualiser et d'optimiser le parcours pédagogique.

D'après Garrett (1995), les limites techniques des logiciels intelligents restreignent parfois un peu trop la variété des activités. Les outils sophistiqués ne sont donc pas la panacée.

On trouvera une série de descriptions de systèmes intelligents à l'annexe B.4.

2.7.5 Internet et ALAO

Internet est le réseau informatique qui lie les ordinateurs du monde entier et permet d'échanger des données, notamment à travers le courrier électronique et le *World Wide Web* dont nous parlerons dans la suite de ce paragraphe. C'est un réseau qui relie des millions de réseaux informatiques, de taille variable, dans le domaine privé, public, académique, gouvernemental ou des affaires.

En parallèle, né en 1969 mais connaissant son apogée dans les années 1980, le service *Compuserve* est le premier fournisseur d'accès à des services en ligne (courrier électronique, forums, etc.) à large échelle aux Etats-Unis, à travers une ligne téléphonique et un modem [12]. En 1982, la France lance le système télématique *Minitel*. Rézeau (1994) décrit une application utilisant des resources de *Compuserve* et du *Minitel* pour construire un concordancier (§3.1.7).

Entre 1989 et 1991 naît le *World Wide Web* (*WWW*, *web* ou *toile* en français), un système multimédia d'échange d'informations proposé par Tim Berners Lee et Robert Caillau. Ce système basé sur des serveurs décentralisés à travers des documents hypertextes [13] est basé sur un langage de balises

12. MODulateur-DEModulateur, appareil servant à deux ordinateurs d'échanger des données à distance à travers une ligne téléphonique, en transformant les données en un signal sonore et en le convertissant à nouveau en données informatiques à l'autre bout de la ligne.
13. Inventée par Bush (1945) et perfectionné par Nelson (1965), les hypertextes sont un système non séquentiel et non linéaire de textes reliés entre eux par un réseau conceptuel de nœuds et de liens. Lorsque les documents sont également non-textuels (vidéos, sons, animations, images), on parle d'*hypermédias*. Le logiciel *Hypercard* sous *MacIntosh*, basé sur les hypertextes, a permis la naissance de nombreux didacticiels (Svenconis et Kerst, 1995).

simple, nommé *HTML* (*HyperText Markup Langage*) [14], qui permet de formater les documents du web [15].

Le *web* constitue en soi une ressource éducative (Allodi *et al.*, 1998; Debski et Levy, 1999). On y trouve des informations sur des sujets infiniment variés et de qualité ou de degré d'approfondissement variables, pour tous les niveaux d'enseignement. Grâce à la facilité de création de documents, les enseignants se servent du *web* comme d'un outil : ils exposent les travaux de leurs élèves, ils publient des journaux de classe etc. Ils ont également accès à des documents authentiques pour leurs propres enseignements, dans leur propre langue ou dans la langue qu'ils enseignent. Ils peuvent également échanger leurs outils et supports pédagogiques. Les moteurs de recherche sont également un formidable concordancier (§3.1.7, Chinnery, 2008). Quant aux apprenants, ils trouvent aussi une ressource d'informations indispensables. Malheureusement, le *web* fourmille de documents non vérifiés et non validés ; il n'est donc pas rare que les documents contiennent des erreurs, tant au niveau du fond que de la forme (orthographe, syntaxe...).

Depuis sa naissance, le *web* est devenu de plus en plus riche en contenu et de plus en plus accessible. Les écoles de tous niveaux sont de plus en plus reliées à Internet, ainsi que les ménages privés. Les pages *web* deviennent de plus en plus complexes et interactives. Du côté des serveurs, de nombreux langages de programmation (*Perl, PHP, ASP, JSP, Cold Fusion,* etc.) permettent notamment de gérer l'interaction avec des bases de données et de créer des applications informatiques complexes [16].

Du côté du navigateur [17], les pages utilisent le langage *Javascript*, notamment pour manipuler le contenu des documents [18] et contrôler les in-

14. `http://www.w3.org/standards/webdesign/htmlcss`, dernier accès le 6.12.10.
15. *XML* (*eXtensible Markup Langage*) est un langage de balise semblable au *HTML* mais qui est ouvert et extensible. Il permet une structuration plus ou moins stricte de documents à travers une définition des balises et des imbrications licites d'éléments nommée *DTD* (*Document Type Definition*). De nombreux standards basés sur XML existent : *MathML* pour les mathématiques, la *TEI* (*Text Encoding Initiative*) dans le domaine de l'édition, *SVG* dans le domaine des graphiques vectoriels, etc. Dans cet ouvrage, nous utilisons *XML* pour coder des analyses syntaxiques (§5.3.1) et pour le codage de la sortie d'un correcteur orthographique et un corpus d'erreurs (§6.2).
16. La technologie qui permet de transmettre des données nécessaires pour l'exécution d'un programme sur le *web* est appelée *Common Gateway Interface* (CGI), qui gère l'encodage des paramètres. A côté des langages de scripts, il est aussi possible d'appeler des programmes compilés.
17. Un navigateur (client, butineur ou *browser*) est un programme qui tourne sur l'ordinateur de l'utilisateur, gère la communication avec les serveurs web et affiche les pages *web*.
18. Une page *web* chargée dans un navigateur est traduite en mémoire par une représentation hiérarchique appelée *Document Object Model* (DOM). Le *DOM* est manipulable par

formations envoyées aux serveurs. Afin d'alléger le chargement des pages en ne mettant à jour qu'une petite part du contenu lorsqu'une action est déclenchée ou un formulaire est envoyé est apparu dès 2003 un ensemble de technologies nommé *Ajax* (*asynchronous Javascript and XML*, *Javascript et XML asynchrones*) qui envoie une requête en tâche de fonds sans empêcher la lecture d'une page ni le déclenchement d'autres actions, d'où le terme d'*asynchrone*, et met à jour des conteneurs, à savoir le contenu de balises qu'il faut remplacer, grâce à *Javascript*.

HotPotatoes [19] (Arneil et Holmes, 1999) est une suite de programmes sous *Windows* qui permettent à des enseignants de réaliser facilement des exercices interactifs en *Javascript* sur la toile sans connaissances préalables de programmation. Les exercices disponibles sont des questionnaires à choix multiples, exercices à réponse courte, textes à trous etc. Les possibilités de rétroaction sont nombreuses. Les interfaces d'exercices sont disponibles en de nombreuses langues. Dans un sens, *HotPotatoes* est assimilable à un langage-auteur (§2.7.2).

Flash est une plateforme multimédia destinée à la production d'animations et à l'intégration de vidéos et de sons dans une page *web*. Les animations et applications sont programmées dans le langage *ActionScript*. Pour que les animations fonctionnent correctement, les navigateurs *web* doivent être dotés d'un programme complémentaire (*plug-in*) gratuit. Par contre, les logiciels permettant de développer les applications sont payants. Brett (2004, 2006) présente l'outil de création d'exercices *Conductor*, qui consiste en un seul programme en *Flash* qui permet de créer une variété d'exercices multimédias. *Conductor* se base sur un fichier externe en *XML* qui peut être facilement édité. Le contenu peut être sauvé dans une base de données et/ou être envoyé par courriel aux enseignants au moyen de scripts *PHP*. *ESPADA* (B.4.19, Koller, 2003, 2004) est un système d'apprentissage multiligue pour les langues romanes, qui utilise la technologie *Flash* et *ActionScript* pour des animations montrant les règles de grammaire.

Pour bâtir un site *web* d'une certaine complexité, il existe de nombreux outils de gestion de contenu (*Web Content Management Systems*, *CMS*),

une interface standard de programmation (*Application Programming Interface*, API) implémentée par de nombreux langages de programmation, dont *Javascript* et *PHP*. Vers la fin des années 1990, les principaux navigateurs de l'époque, *Internet Explorer* et *Netscape*, se livraient à une guerre commerciale et implémentaient chacun des balises propriétaire non standard et des langages de script *JScript* et *Javascript*. A l'époque, on parlait d'*HTML* dynamique (*Dynamic HTML, DHTML*) plutôt que de *DOM*. Certaines applications *web* ne fonctionnaient que pour un navigateur. En l'an 2000, le consortium W3C a publié la version 2.0 du *DOM*, ce qui a mis fin à une situation absurde. Depuis, les incompatibilités entre navigateurs sont moindres.

19. Disponible sous `http://hotpot.uvic.ca/`, dernier accès le 20 août 2006.

dont certains logiciels libres. On citera notamment *Spip*, *Typo3* et *Joomla*. L'intérêt principal des *CMS* est de pouvoir mettre à jour rapidement et facilement un site *web*, sans besoin de beaucoup de connaissances. Par ailleurs, l'indexation des articles est automatique. En 1995 est apparu un type de *CMS* particulier appelé *Wiki*, d'après un mot hawaïen signifiant *rapide* ou *vite*. Il s'agit d'un site *web* dont le contenu est très facilement modifiable par les utilisateurs du site, avec ou sans authentification préalable. Le but est de faciliter la rédaction collaborative d'un contenu. De nombreux enseignants utilisent les *Wikis* pour leurs projets de classe (Godwin-Jones, 2003). L'encyclopédie en ligne collaborative *Wikipedia* est basée sur ce principe.

De nombreuses personnes dans le monde écrivent des *blogs* [20] (abrégé de *Web Log*, littéralement calepin ou carnet *web*), qui consistent en des articles généralement courts, sur un sujet précis (art, informatique, politique, etc.) ou sur la vie de son auteur (sorte de journal intime). Généralement les articles sont classés par ordre chronologique inverse et peuvent être commentés par les lecteurs. On trouve de nombreux sites permettant d'ouvrir son *blog* gratuitement et également des logiciels à installer sur son propre site *web*. Les *blogs* sont beaucoup utilisés par les enseignants de langues, tant pour rechercher du contenu authentique que pour faire écrire les apprenants (Godwin-Jones, 2003).

Par ailleurs, comme le principe du *web* est de relier des documents entre eux, il est parfois utile de pouvoir reprendre le contenu – ou une liste des derniers articles – d'un site sur un autre qui traite du même sujet ou d'intérêts analogues. Ce principe est appelé *syndication* : il s'agit de mettre à la disposition une liste d'articles (avec parfois leur contenu) dans des formats *XML* simples appelés *RSS* (*Really Simple Syndication*) ou *Atom*. Ces listes sont appelées *flux* et sont généralement créées automatiquement à partir des bases de données des *CMS*. Les flux peuvent être une liste des derniers articles dans l'ordre chronologique, ou une liste regroupant les derniers articles d'un même auteur ou d'une même rubrique. Ils permettent aussi de détecter la mise à jour d'un site. Les *CMS* sont également généralement dotés de lecteurs de flux qui permettent de reprendre le contenu d'autres sites. Par ailleurs, il existe des programmes spécialisés appelés *agrégateurs de flux* – parfois ce sont des pages *Internet* – qui permettent à un utilisateur de s'abonner aux flux de son choix et de les organiser à sa guise, afin de composer par exemple un journal personnalisé. Les enseignants tirent également grand parti des flux *RSS*, qui leur permettent de créer des activités pédagogiques et des exercices en lien avec l'actualité, avec des documents authentiques.

20. On trouve aussi le terme de *blogue*.

Terminons par mentionner qu'il existe des outils de gestion de contenu spécialisés dans l'enseignement. On parle alors de plateformes d'enseignement (*Learning Management System, LMS*) ou de plate-formes de téléapprentissage (Chanier et Vetter, 2006). Citons *Moodle*, *Dokeos*, *Manhattan Virtual Classroom*, *Plone*, *WebCT*, *TopClass*, *Lenya* etc. Ces plateformes permettent de gérer l'inscription des apprenants, le calendrier des cours, l'évaluation automatique des connaissances etc. Il existe différents standards qui permettent aux enseignants de formater un parcours pédagogique défini et de partager des ressources pédagogiques avec d'autres enseignants. Les *Learning Objects Metadata* ou *Learning Objects Models* (*LOM*, Godwin-Jones, 2004b; Armitage et Bowerman, 2005) sont des unités indépendantes qui peuvent être utilisées pour l'éducation ou l'enseignement. Les *LOM* peuvent être assemblées pour créer un cours. Le contenu de chaque *LOM* est décrit à l'aide de métadonnées qui permettent de retrouver dans une banque de données le contenu adéquat pour un besoin précis (titre, description, mots-clés, auteur, coût d'utilisation, spécifications techniques et objectifs d'enseignement). Les contenus peuvent être des animations, des textes, d'autres objets multimédias, ou des combinaisons de ces éléments. Les *LOM* permettent de réutiliser des ressources et peuvent faciliter la mise à jour automatique de contenus de cours si un *LOM* est révisé ou de nouveaux *LOM* sont ajoutés à la base de données. *SCORM* (*Sharable Content Object Reference Model*) est une implémentation du standard *LOM* (Godwin-Jones, 2004b).

Poursuivons ce tour d'horizon par d'autres outils disponibles sur *Internet*. Le courrier électronique (que l'on nomme aussi *email*, courriel ou mél) est un moyen de communication extrêmement répandu qui permet un échange asynchrone entre personnes. Donaldson et Kötter (1999) soulignent que les échanges de mails entre apprenants est utile car ils doivent apprendre à manier la langue ou à utiliser des stratégies de substitution ou de paraphrases pour combler leurs lacunes, quitte à utiliser un mot de leur propre langue. *International Email Tandem Network* (Levy, 1997, p. 32) est un réseau de correspondance entre apprenants qui correspondent dans la langue de l'autre. Les discussions ont lieu par liste de discussion ou par un forum avec des modérateurs. Le projet a été initié en 1993. Un échange est plus bénéfique si l'apprenant dialogue dans la langue de son interlocuteur (Woodin, 1997). Par contre, en utilisant sa propre langue, un apprenant peut aussi bénéficier de l'échange, en soignant sa propre expression et en jouant un rôle de tuteur en corrigeant les erreurs de son partenaire (Ware et O'Dowd, 2008).

S'apparentant au courrier électronique, les forums sont des outils où les utilisateurs participent à une discussion en publiant des messages lisibles

par tous, ou du moins par les personnes autorisées à participer à la discussion. Les discussions sont généralement organisées en thèmes généraux. Un utilisateur publie un message et les autres utilisateurs y répondent et commencent un échange. Un message s'inscrit dans un historique de discussion si il répond à un message ; ce message "parent" est peut-être lui-même une réponse à un autre message et ainsi de suite ; il est alors possible de retracer la *généalogie* d'un message. Cette généalogie des messages est plus connue sous le nom de *fil de discussion* ou *thread* en anglais. Les forums permettent des échanges entre apprenants, notamment la négociation de sens (Lamy et Goodfellow, 1998). Caws (2005) utilise les forums pour faciliter la rédaction de textes par petits groupes d'apprenants.

Les outils de conférence écrite, orale et vidéo présent les avantages et les inconvénients d'une communication synchrone. Pour l'écrit, il existe les *Internet Relay Client* (IRC) ou *chat* [21], où les utilisateurs se connectent à un serveur et s'identifient par un pseudonyme. Un serveur donne accès généralement à de nombreux salons ou salles de discussion et l'utilisateur doit se connecter à l'un ou plusieurs d'entre eux pour dialoguer. L'utilisateur peut envoyer des messages vus par tout le salon ou dialoguer individuellement en privé avec d'autres utilisateurs. Aujourd'hui, de nombreux logiciels permettent de se connecter à des salons de discussion.

Le rythme d'un tel échange doit être rapide. Les messages défilent et l'utilisateur a peu de temps pour les lire (ou doit revenir en arrière avec les barres de défilement). Les phrases sont courtes et la syntaxe, l'orthographe et la ponctuation sont peu soignés (Véronis et Guimier de Neef, 2006; Chapelle, 2003; Lotherington et Xu, 2004). Du point de vue de l'enseignant, la méthode présente l'inconvénient du manque de contrôle : l'échange est trop rapide pour espérer une régulation (Paramskas, 1999). Certains apprenants connaissant bien les abréviations et ayant une bonne vitesse de frappe peuvent accaparer la conversation. Mangenot (1998a,b) propose l'utilisation d'outils de conférence pour l'écriture en commun d'une histoire. L'enseignant doit alors animer le débat pour relancer le dialogue et proposer de trancher entre plusieurs scénarios.

La technologie émergente de téléphonie sur *Internet*, *Voice over IP*, permet de se servir de son ordinateur personnel pour téléphoner partout dans le monde. Il suffit de s'inscrire auprès d'un fournisseur de logiciel (généralement gratuit) pour pouvoir communiquer oralement avec les utilisateurs d'*Internet* utilisant le même logiciel qui sont connecté au même moment à *In-*

21. Anglicisme qui signifie bavarder. Les francophones connaissent aussi le terme de *clavardage*, composé de *clavier* et *bavardage*.

ternet et au serveur de communication. Le logiciel le plus répandu, *Skype* [22], permet de connecter cinq ordinateurs pour la même conférence téléphonique. Une version récente permet aussi de faire une conversation vidéo à l'aide de caméras *web* (*webcams*). Ce logiciel est utile pour des échanges entre classes et peut être intégré à des plateformes d'enseignement (Godwin-Jones, 2005).

Enfin, les téléphones mobiles permettent de plus en plus souvent de se connecter à *Internet*, à des vitesses qui vont considérablement s'améliorer les prochaines années. Des enseignants commencent à développer des applications d'ALAO sur téléphone mobile, comme un programme d'apprentissage de vocabulaire par *SMS* [23] (Chinnery, 2006) ou du contenu didactique (Houser, 2006; Houser et Thornton, 2006). Daniels (2006) prône l'utilisation du téléphone mobile qui permet à l'apprenant de trouver des matériaux en dehors de la classe. Le projet *SigmaStar* (§B.2.31) vise à développer des jeux sur téléphones mobiles en *Java* pour l'apprentissage des langues. On mentionnera encore les assistants personnels (*Personal Data Assistant, PDA*), les tablettes tactiles ou les livres électroniques qui peuvent avoir une utilisation pédagogique (Chen et Chen, 2004; Schulze, 2004; Chinnery, 2006).

La technique du *Podcasting* [24] commence à être utilisée pour l'enseignement des langues (Godwin-Jones, 2005; Scinicariello, 2006). Elle consiste à télécharger automatiquement des fichiers audio *MP3* sur un baladeur numérique ou un ordinateur personnel, pour une écoute immédiate ou ultérieure, grâce à un abonnement à un service de diffusion qui permet de personnaliser les choix. Pour l'enseignement des langues, le *Podcasting* permet aux apprenants d'écouter divers documents, comme un journal radiodiffusé ou télévisé ou une émission de radio, créée par et pour des locuteurs natifs. Cette technologie permet d'écouter et de répéter les documents à l'envi, parfois en faisant d'autres choses.

Actuellement encore, deux obstacles importants empêchent l'essor de ces appareils : le coût de connexion et de transfert est encore prohibitif pour transférer un volume important de données ; de plus, les fabricants de ces appareils se livrent à une guerre commerciale et aucun standard n'apparait. L'apparition de nouvelles normes pour les réseaux sans fils promet un bel avenir aux applications utilisant des appareils mobiles (Godwin-Jones, 2004a).

22. `http://www.skype.com/`, consulté le 2 août 2006.
23. *Short Message Service*, message de 160 caractères maximum transmis sur les téléphones mobiles.
24. Chanier et Vetter (2006) utilisent le terme de baladodiffusion.

2.8 Discussion

Nous avons vu que l'ALAO est un domaine encore jeune mais qui est appelé à se développer énormément, tant grâce aux avancées technologiques qu'à travers les besoins énormes d'outils d'apprentissage. Les coûts de production de ces logiciels sont très variables. On trouve des logiciels produits par des enseignants pour leurs propres besoins, souvent à l'aide de logiciels-auteurs, qui utilisent quelque types d'exercices et n'offrent que quelques heures d'activités. D'autre part, il existe des logiciels produits par des éditeurs, avec des moyens considérables, qui représentent de nombreuses heures d'utilisation. Parfois même, ces logiciels sont dotés d'outils technologiques pointus, notamment la reconnaissance vocale (§3.1.3). Néanmoins, ces logiciels manquent quelques fois de pertinence pédagogique.

Au niveau intermédiaire, on trouvera des prototypes de logiciels, construits par des équipes de recherche. Ces produits offrent souvent une technologie avancée, comme nous le verrons au chapitre 3. Toutefois, les matériaux pédagogiques et les exercices ne sont parfois pas très nombreux. La couverture de la langue est souvent limitée et le taux de détection des erreurs d'apprenants, ainsi que le taux de diagnostic erroné sont trop élevé pour être utilisables à grande échelle. Nous aurons amplement l'occasion d'y revenir au chapitre suivant.

Chapitre 3

Traitement Automatique des Langues et Apprentissage des Langues Assisté par Ordinateur

Au chapitre précédent, nous avons évoqué les différents champs de l'Apprentissage des Langues Assisté par Ordinateur (ALAO). Dans ce chapitre, nous étudions l'application du Traitement Automatique des Langues (TAL) à l'ALAO. Nous débutons par un survol des différents domaines du TAL et de leur application possible en ALAO à la section 3.1. Ensuite, nous consacrons une section à la correction orthographique (§3.2). Nous poursuivons avec l'analyse syntaxique et la correction d'erreurs (§3.3). Puis nous décrivons quelques formalismes sémantiques en discutant leur utilisation pour la détection d'erreurs à la section 3.4. Ensuite, nous évoquons une série formalismes lexicaux et leur utilisation possible pour l'ALAO (§3.5). Enfin, nous apportons quelques remarques de conclusion (§3.6).

3.1 Survol des différents domaines du TAL et de leur application à l'ALAO

Dans cette section, nous décrivons tour à tour les outils de base (§3.1.1), les étiqueteurs (§3.1.2), la reconnaissance de la parole (§3.1.3), la synthèse vocale (§3.1.4), la génération syntaxique (§3.1.5), la traduction automa-

tique (§3.1.6), les outils de traitement de corpus comme les concordanciers (§3.1.7), les outils de résumé et d'évaluation automatique (§3.1.8), et enfin les systèmes de dialogue homme-machine (§3.1.9). Nous traitons dans des sections spécifiques la vérification orthographique (§3.2), l'analyse syntaxique et correction grammaticale (§3.3), les outils sémantiques (§3.4) et lexicaux (§3.5).

3.1.1 Outils de base

Les outils de bas niveau procèdent généralement à un pré-traitement des phrases qui est utile pour l'analyse syntaxique et d'autres techniques de TAL ; ils peuvent s'avérer utiles pour l'ALAO. Nous décrivons la technique de reconnaissance de patrons (§3.1.1.1). Ensuite, nous parlons des analyseurs lexicaux (§3.1.1.2). Puis nous évoquons les détecteurs de langue (§3.1.1.3). Nous poursuivons par les analyseurs morphologiques, les lemmatiseurs, les conjugueurs et les déclineurs (§3.1.1.4) et les phonétiseurs (§3.1.1.5). Enfin, nous terminons par une discussion (§3.1.1.6).

3.1.1.1 Reconnaissance de patrons ou *pattern matching*

Cette technique peu évoluée est notamment destinée à évaluer les réponses à des questions ouvertes ou semi-ouvertes, où l'apprenant doit rédiger une phrase complète ou un segment de phrase. Selon Winograd (1983), un patron linguistique peut être défini comme une description d'une forme possible d'une langue, par analogie avec un patron au sens propre, qui est un objet dont la forme est identique à la pièce qu'on veut découper dans du tissu ou d'autres matières [1]. Un patron servira à retrouver dans un texte des formes ayant une même construction, comme dans l'exemple suivant :

(2) a. "Il y a __ sur __"
    b. "Il y a une guêpe sur ta tête."
    c. "Il y a beaucoup de misère sur terre."

Le patron (2a) peut servir à repérer les phrases (2b) et (2c). On peut définir des patrons bien plus complexes, qui contiennent des variables ou

1. Vu son caractère extrêmement basique et peu souple, le classement de la reconnaissance de patrons parmi le TAL peut être sujet à controverse. Néanmoins, cette technique est extrêmement répandue dans les logiciels d'ALAO.

des caractères de remplacement. Par exemple, les *expressions régulières* (ou *expressions rationnelles*) sont une technique de notation compacte qui permet de définir un ensemble de chaîne de caractères. Elles sont utilisées dans de nombreux éditeurs de textes, dans des langages de script (§2.7.5) ou dans des systèmes d'exploitation comme *Unix* ou *Linux*, pour rechercher des fichiers ou des éléments dans un fichier ; on utilise également les expressions régulières pour remplacer des chaînes de caractères par d'autres. Voici quelques exemples d'expressions régulières, tirés de l'encyclopédie en ligne *Wikipedia* [2] :

- "chat|chien" : reconnait uniquement les mots "chat" et "chien".
- "([cC]hat|[cC]hien)" : "chat", "Chat", "chien" et "Chien".
- "ch+t" : "cht", "chht", "chhht", etc.
- "a[ou]+" : "aou", "ao", "auuu", "aououuuoou" etc.
- "peu[xt]?" : "peu", "peux" et "peut".

Dans le domaine de l'ALAO, la reconnaissance de patrons est utilisée pour repérer la présence d'éléments dans une réponse, ce qui sert de base à une évaluation. Généralement, cette technique signale les éléments manquants et les éléments superflus. Le concepteur d'exercices doit fournir au système la ou les réponses possibles, en utilisant un patron dans un formalisme compréhensible par l'ordinateur et le plus simple possible. Plus un patron est complexe, plus la possibilité est élevée d'accepter des phrases incorrectes, sur le plan syntaxique ou sémantique. Il est aussi courant de fournir des patrons pour les réponses incorrectes que le concepteur de l'exercice anticipe, soit par son expérience personnelle, soit à l'aide d'un corpus de tests récolté lors d'une phase d'évaluation de logiciels (§§2.5, 3.1.7).

La technique de reconnaissance de patrons est très répandue à cause de sa simplicité de définition et d'implémentation. Le diagnostic est toutefois limité, car il est sensible aux erreurs d'orthographe et d'accord, si aucun correcteur orthographique ne repère les mots inconnus ou si une alternative ne prévoit pas l'erreur d'accord. Les concepteurs d'exercices doivent écrire des formules très complexes et anticiper les erreurs communes des apprenants. Une réponse correcte pourra être rejetée par le système à cause de lacunes dans le patron. À l'inverse, des réponses incorrectes pourront être acceptées à cause d'un patron insuffisamment spécifié ou, au contraire, trop complexe. Au niveau grammatical, les structures syntaxiques complexes comme les phrases interrogatives utilisent des dépendances à longue distance qui sont

2. `http://fr.wikipedia.org/`, dernier accès le 28 juillet 2006. v. §2.7.5. Le caractère | représente une alternative ("ou") ; les parenthèses carrées [] représentent des variantes ; le + représente un caractère présent une ou plusieurs fois ; enfin le ? indique un / des caractères optionnels (zéro ou une fois).

difficilement traitables par des patrons. De plus, l'apprenant n'aura souvent qu'une évaluation sommaire de sa production (juste ou faux), sans explications sur les erreurs commises et sur la manière d'y remédier.

Pour contourner le problème des petites erreurs orthographiques (ponctuation, accentuation, casse), Burston (1989, 1990) propose de normaliser les chaînes en ôtant les espaces, les signes de ponctuation et les accents et autres diacritiques et en mettant les chaînes entièrement en majuscules ou minuscules. Les variations orthographiques peuvent être surmontées en acceptant la disparition ou l'insertion d'un ou deux caractères. Cette technique est nommée *reconnaissance relâchée de formes*.

Passons maintenant à quelques descriptions de logiciels. Dans *PLATO* (§B.2.27), un des ancêtres de l'ALAO, une technique de reconnaissance de patrons permettait de corriger des phrases complètes. Le correcteur grammatical *Le Patron* (§C.22) utilise des expressions régulières pour traiter les erreurs d'apprenants du français. Le correcteur en logiciel libre *Graviax* [3] fait de même pour l'anglais. Le logiciel auteur *IDIOMA-TIC* (Desmet et Héroguel, 2005, §B.1.3) utilise une technique souple nommée *approximate string matching*. Quant au *McGill Language Learning Environment* (§B.2.17), il utilise des patrons basés sur la notation BNF (*Backus-Naur Form*, Wirth, 1987). Citons encore *ELSE* (*Elementary Language Study Exerciser*, Allen, 1997) et *ALLES* (*Advanced Long-distance Language Education System*, §B.4.3, Schmidt *et al.*, 2004). Enfin, Menzel (1988) utilise un système de reconnaissance de patrons associé à un réseau de contraintes (§3.3.2.2). L'auteur prône de réduire au minimum le nombre des erreurs.

3.1.1.2 Segmenteurs de chaînes et analyseurs lexicaux

Les segmenteurs de chaînes (*tokenizers*) ou analyseurs lexicaux divisent une chaîne de caractères en unités, généralement des unités lexicales. Ils procèdent à un pré-traitement d'un texte pour les analyseurs morphologiques (§3.1.1.4), syntaxiques (§3.3) et pour les autres outils de traitement du langage, en s'appuyant généralement sur un lexique et en isolant certaines abréviations ou expressions numériques. Cette tâche est essentielle mais loin d'être triviale (Nazarenko, 2006). Les signes de ponctuation et les séparations des mots sont des indications précieuses sur la structure des phrases [4]. Le point détermine généralement une fin de phrase mais fonctionne aussi comme marqueur d'abréviation comme dans *M. Wehrli*. De plus, on consi-

3. `http://sourceforge.net/projects/graviax/`, consulté le 10 mars 2007.

4. Cependant, le manque de systématique dans leur utilisation dans les textes rend ces indications peu fiables (Manning et Schütze, 2000, sct. 4.2).

dère généralement *d'abord* et *aujourd'hui* comme une seule unité, alors que d'ordinaire l'apostrophe est un délimiteur. Le tiret est un autre délimiteur problématique : *porte-monnaie* et *c'est-à-dire* sont une seule unité, mais pas *voulez-vous*, *cet homme-là* etc. Les nombres écrits en toutes lettres (*dix-sept*) et les mots composés comme *pomme de terre*, *chemin de fer*, *encore que* ou *chou-fleur* posent d'autres types de difficultés.

Grover *et al.* (2000) présentent l'analyseur lexical *LT TTT* (*Language Technology Group Text Tokenisation Toolkit*), qui fonctionne à l'aide de transducteurs (§3.3.4.1). De son côté, le logiciel *Mirto* (§B.2.21, Antoniadis *et al.*, 2004b) utilise un analyseur lexical pour un prétraitement des textes utilisés dans ces exercices, notamment pour générer des exercices à trou. Mikheev (2003) et Schwartz *et al.* (2004) montre d'autres exemples d'utilisation d'analyseurs lexicaux pour l'ALAO.

Pour conclure, nous proposons de mettre à profit le résultat des analyseurs lexicaux pour montrer aux apprenants quelles sont les différentes phrases ou segments de phrases trouvés par l'analyseur lexical et les alternatives possibles, ce qui peut constituer une indication d'erreur.

3.1.1.3 Détecteurs de langue

Les détecteurs de langue permettent de déterminer automatiquement la langue dans laquelle est écrite un texte. Ces outils sont généralement basées sur une méthode statistique, par exemple par fréquence de trigrammes. On appelle trigramme une séquence de trois lettres présentes consécutivement dans un mot. Ainsi, le mot *arbre* est composé des trigrammes *arb*, *rbr* et *bre*. Pour certaines applications, on ajoute aussi les espaces, et donc les trigrammes *#ar* et *re#* (où le signe # représente l'espace). La langue française compte environ 1600 trigrammes différents. Il est possible de déterminer les fréquences de trigrammes de chaque langue en analysant un gros corpus de textes, ce qui constitue en quelque sorte la signature d'une langue. En comptant les fréquences de trigrammes d'un texte, on peut déterminer de quel modèle de langue il est le plus proche.

Mirto (§B.2.21) et *Exills* (§B.2.8) se servent de détecteurs de langues pour s'assurer que l'apprenant utilise bien la bonne langue dans les outils et exercices.

3.1.1.4 Analyseurs morphologiques, lemmatiseurs, conjugueurs et déclineurs

Les analyseurs morphologiques permettent de dériver la construction morphologique d'un mot connu ou de faire des hypothèses sur la nature d'un mot inconnu. Un mot est constitué d'une racine et d'affixes, qui sont de petites unités destinées à marquer un nombre, un temps, une personne, etc. Un analyseur morphologique tentera de restituer la ou les racines possibles d'un mot et d'analyser sa valeur. Le mot *suis* sera soit la première personne de l'indicatif présent du verbe *être*, soit les première et deuxième personnes de l'indicatif présent du verbe *suivre* ; *va* sera la troisième personne de l'indicatif et deuxième personne de l'impératif du verbe *aller*, etc.

La morphologie est traditionnellement divisée entre formation des mots (morphologie dérivationnelle et compositionnelle) et morphologie flexionnelle (ten Hacken et Tschichold, 2001; Bouillon *et al.*, 1998; Nazarenko, 2006). La morphologie flexionnelle consiste à décliner un substantif ou un adjectif ou conjuguer un verbe à partir d'un lexème, tandis que la morphologie dérivationnelle décrit la composition d'un mot à partir de racines et d'affixes ou la création d'un mot à partir d'un autre (*centre* → *centrer*, *licencier* → *licenciement*, etc.).

Suivant les besoins, un outil peut soit fournir une forme unique qu'il détermine comme correcte, soit toutes les analyses possibles en laissant le choix à l'utilisateur. Un analyseur morphologique peut utiliser deux techniques différentes : l'approche par règles peut laisser des formes non résolues si aucune règle ne permet de déterminer un choix, mais commet peu d'erreurs ; en revanche, l'approche stochastique fournit généralement une analyse pour toutes les formes, tout en donnant un taux d'erreur plus élevé. L'approche stochastique nécessite un corpus d'apprentissage, étiqueté ou non par des humains.

Cas particulier d'analyseur morphologique, un lemmatiseur (*stemmer*, Antoniadis *et al.*, 2004a) est un outil qui détermine le lemme d'un mot, c'est-à-dire sa forme de base comme on la trouve dans un dictionnaire (ainsi que son éventuelle dérivation morphologique à partir de son radical). Un conjugueur permet de donner les tableaux de conjugaison d'un verbe. Un déclineur fait de même pour les noms et adjectifs d'une langue à déclinaisons. Parfois, les conjugueurs et déclineurs sont couplés à un lemmatiseur pour retrouver la forme de base du verbe et afficher toutes les formes fléchies.

L'analyse morphologique d'un mot peut être aussi utile pour analyser des mots inconnus ou difficiles, par exemple pour la recherche dans un diction-

naire. En français, il est courant de pouvoir former des mots nouveaux par l'utilisation d'affixes et de suffixes. Les adjectifs peuvent être modifiés par l'adjonction de préfixes comme *hyper-*, *multi-*, *archi-*, etc. Dans cet ouvrage, nous utilisons abondamment le substantif *apprenant*, venant du vocabulaire spécialisé de la pédagogie et formé sur le participe présent du verbe *apprendre*, bien que la plupart des dictionnaires ne connaissent pas ce terme comme substantif. Il faut toutefois prendre garde de ne pas accepter n'importe quelle formation de nouveaux mots.

Passons maintenant aux descriptions de quelques logiciels utilisant des outils morphologiques. Le logiciel FreeText (§4) fournit un conjugueur capable de donner la conjugaison complète d'un verbe français à partir de son infinitif ou d'une forme fléchie. Les temps composés sont générés à partir de la conjugaison des auxiliaires et du participe passé. Le conjugueur gère aussi les verbes essentiellement pronominaux, les verbes défectifs comme *pleuvoir*, *choir*, etc. Il existe de nombreux exemples pour de nombreuses langues : anglais (*VERBCON*, §B.2.37, *Compounds* §B.4.13, Boucher *et al.*, 1993; Boucher et Sébillot, 1993; Danna, 1997), allemand (Greene *et al.*, 2004), basque (*XUXEN*, §C.37, Agirre *et al.*, 1992, Díaz de Ilarraza *et al.*, 1998, 1999), japonais (*CoCoA*, Feng *et al.*, 2000), tchèque (Smrž, 2004), gaélique (Keogh *et al.*, 2004), espagnol (*The Spanish Verb*, Soria, 1997, §B.2.32), etc.

Pour conclure, nous suggérerons d'analyser les mots pour tenter de retrouver la racine (ou plusieurs racines potentielles) puis, dans un dialogue avec l'apprenant, de reconstituer la forme correcte.

3.1.1.5 Phonétiseurs

Les phonétiseurs sont des outils qui calculent une ou plusieurs représentations phonétiques pour une chaîne de caractères. Ainsi, pour la chaîne *portions*, un phonétiseur devrait fournir les représentations [pɔʁsjõ], qui correspond au substantif *portion*, et [pɔʁtjõ], qui correspond à l'indicatif imparfait première personne du singulier du verbe *porter*. Un phonétiseur doit pouvoir fournir des résultats pour des mots connus comme pour des noms propres ou des noms inconnus. Nous reviendrons plus en détail sur le problème de la phonétisation des mots inconnus dans la section 3.1.4 consacrée à la synthèse vocale.

Gaudinat et Goldman (1998) et Ndiaye et Vandeventer Faltin (2004) décrivent un phonétiseur destiné à un outil de synthèse vocale (§3.1.4), un correcteur orthographique (§§3.2, 6) et un diagnostic d'erreurs grammaticales (§§3.3.3.2, 5.2.2). Cet outil fonctionne à l'aide d'un système expert qui

utilise des règles déterministes pour traiter des mots inconnus. Il ne fournit par conséquent que la chaîne phonétique la plus plausible. En général, les phonétiseurs utilisent la technique des transducteurs (§3.3.4.1).

Les phonétiseurs peuvent être utiles pour des apprenants dans deux cas de figure. Tout d'abord, il est utile de disposer de la ou des transcriptions phonétiques d'un mot. Dans le cas de l'exemple de *portions*, il sera utile de disposer d'un étiqueteur (§3.1.2) pour déterminer la catégorie lexicale du mot (partie du discours). Ensuite, un phonétiseur permettra à l'apprenant de corriger un mot mal orthographié en cherchant dans le lexique avec une clé de recherche phonétique. Nous reviendrons sur cette utilisation dans la partie sur la correction orthographique (§3.2.3.5).

3.1.1.6 Discussion

Bien que les outils de traitement des langues décrits ci-dessus soient sommaires, ils peuvent s'avérer fort utiles à plusieurs titres. Tout d'abord, ils sont une aide à l'apprentissage d'une langue, spécialement pour améliorer l'expression écrite. Les apprenants peuvent gagner du temps en accédant à des outils d'aide qui les aident à combler leurs lacunes en morphologie et en orthographe. Ensuite, ils peuvent également servir à l'élaboration d'exercices, en épargnant aux enseignants des tâches fastidieuses. Même si les concepteurs d'exercices doivent souvent réviser ou superviser les résultats des outils de TAL, le gain de temps est appréciable par rapport à un travail manuel. Le logiciel MIRTO (§B.2.21) illustre particulièrement bien tout le parti qu'on peut tirer de tels logiciels.

Ces outils sont assez simples et ont un taux d'erreurs assez bas, comme le relèvent Antoniadis *et al.* (2004b). Ces technologies sont donc suffisamment mûres pour permettre leur utilisation pour des logiciels d'ALAO. Enfin, la reconnaissance de patrons permet de pallier le manque de ressources linguistiques ou le manque de capacités techniques.

3.1.2 Étiqueteurs

Dans cette section, nous commençons par un survol de la problématique de l'étiquetage. Puis nous décrivons quelques techniques d'étiquetage probabiliste. Nous examinons ensuite quelques exemples d'utilisation d'étiqueteurs pour des logiciels d'ALAO. Nous terminons par quelques remarques de conclusion.

L'étiquetage consiste à attribuer à chaque mot une étiquette indiquant sa catégorie ainsi que diverses propriétés (genre, nombre, cas, temps, etc.) [5]. L'étiquetage passe par trois phases : (i) segmentation des mots (§3.1.1.2) ; (ii) recherche des mots dans le lexique pour retrouver les catégories possibles ; (iii) désambiguïsation (Paroubek et Rajman, 2000). Ainsi, le mot *dorment* peut recevoir une étiquette comme VER-IND-SUBJ-PRE-3-PLU, qui indique sa valeur de verbe à la troisième personne du pluriel du présent de l'indicatif ou du subjonctif. La taille du jeu d'étiquettes dépend de la finesse des informations représentées, allant de quelques dizaines à plusieurs centaines d'étiquettes différentes. Dans certains cas, il est difficile de lever totalement l'ambiguïté (Paroubek et Rajman, 2000).

Dans le cas des mots inconnus, des algorithmes permettent de deviner la catégorie du mot (Paroubek et Rajman, 2000). Il est parfois possible de généraliser des règles : un mot finissant par *-ment* peut être un adverbe ; un mot commençant par une majuscule au milieu d'une phrase est un nom propre, etc.

Passons maintenant aux descriptions de techniques d'étiquetage. La plupart des étiqueteurs utilisent des informations statistiques. Selon Manning et Schütze (2000), il y a deux sources possibles d'information pour l'étiquetage. La première source est de regarder les catégories des mots environnants. Même si ces mots sont ambigus, certaines séquences de catégories, comme *déterminant + adjectif + nom*, ont une probabilité d'occurrence élevée tandis que d'autres sont hautement improbables. La seconde source d'information est la probabilité d'occurrence d'une catégorie lexicale. Une des catégories possibles sera plus employée qu'une autre. Manning et Schütze (2000) proposent différents modèles d'étiqueteurs probabilistes. Tout d'abord, les mots de la phrase sont recherchés dans un lexique, qui retourne les parties du discours (ou catégories lexicales) possibles, ainsi que d'autres informations pertinentes comme les temps verbaux, les personnes etc. On peut calculer des probabilités d'après des bigrammes ou des trigrammes, c'est-à-dire sur les catégories et valeurs de deux ou trois mots. Les trigrammes sont une technique plus efficace car ils permettent de tenir compte davantage du contexte. Les probabilités sont calculées d'après un corpus d'entraînement où les catégories des mots sont annotées par des experts (apprentissage supervisé, Paroubek et Rajman, 2000), ou d'après un corpus non annoté.

Les *Modèles de Markov Cachés* (*Hidden Markov Models, HMM*) [6] sont une technique très répandue pour l'étiquetage probabiliste. Cette technique

5. La notion de phrase et de mots est fort variable d'un système à un autre (Habert, 2006).

6. Voir aussi la section 3.1.3 pour l'application des HMM à la reconnaissance vocale.

consiste à calculer la probabilité qu'une chaîne appartienne à une certaine catégorie lexicale compte tenu de la catégorie d'un ou plusieurs éléments précédents (probabilités de transition). Ainsi, un modèle sur les bigrammes calculera les probabilités sur un seul élément précédent et un modèle sur des trigrammes sur les deux éléments précédents. Les HMM doivent être entraînés sur un gros corpus de textes, annoté ou non. Le taux d'erreurs rapporté dans la littérature est de 1 à 5%. Pour l'utilisation de HMM pour l'étiquetage, citons notamment les étiqueteurs de Abney (1997), de Brun *et al.* (2002), ainsi que *LOCOLEX* de Xerox (Cutting *et al.*, 1992).

L'algorithme de Viterbi (1967) mesure la probabilité qu'une information a été modifiée par une autre en se basant sur des arbres de n-grammes. L'étiqueteur *IMS Tree Tagger* (Schmidt, 1994) est basé sur cet algorithme avec un arbre de décisions de trigrammes, pour pallier les problèmes causés par des données trop peu nombreuses pour en tirer des probabilités vraiment fiables.

CLAWS (*Constituent-Likelihood Automatic Word-Tagging System*, Garside, 1987; Marshall, 1987) est un étiqueteur probabiliste robuste, qui identifie les unités de mesure, les ordinaux etc. et recherche les expressions idiomatiques (v. p. 66) pour restreindre l'ensemble d'étiquettes. Atwell et Elliott (1987) décrivent l'utilisation de *CLAWS* pour détecter les erreurs de locuteurs non natifs, en ajoutant des étiquettes d'erreurs à leur modèle.

L'étiqueteur *Net-Tagger* (Schmid, 1994) utilise la technique des réseaux de neurones artificiels. Celle-ci est basée sur une simplification du fonctionnement du cerveau humain, où les neurones combinent les informations provenant de nombreux autres neurones (10 000, selon certaines estimations) pour produire une nouvelle information en sortie. Dans les réseaux de neurones artificiels, une couche de neurones prend les entrées de la couche précédente. Chaque entrée (ou synapse) est multipliée par un poids synaptique différent, qui est peu à peu défini lors de la phase d'apprentissage. Ces entrées sont combinées et le résultat de cette combinaison passe ensuite par une fonction d'activation. Le résultat de la fonction d'activation est transmis à la couche suivante s'il dépasse un certain seuil. Les réseaux de neurones sont entraînés sur des corpus et sont ensuite capables, comme le cerveau humain, d'appliquer des règles rencontrées dans des cas similaires à des cas qu'ils n'ont pas encore rencontrés. *Net-Tagger* tente de déterminer l'étiquette correcte en tenant compte des trois mots précédents et des deux mots suivants.

Pour terminer, certains étiqueteurs n'utilisent pas d'informations statistiques. C'est le cas de *FipsTag*, basé sur l'analyseur *Fips* (Wehrli, 1997; Goldman *et al.*, 2000, §5), dont l'analyse syntaxique sert à déterminer des

étiquettes dans un jeu d'environ cinquante étiquettes différentes. Des étiqueteurs utilisent des jeux de règles de désambiguïsation (Paroubek et Rajman, 2000), qui comptent un ou plusieurs milliers de règles. D'autres outils étiquettent en deux phases, avec un premier dégrossissage des catégories qui ignore certaines informations comme le genre et le nombre, puis une seconde phase de désambiguïsation à l'aide de règles. D'autres encore apprennent des règles sur la base de corpus étiquetés, ce qui permet de déterminer itérativement un contexte de plus en plus contraint pour déterminer la bonne étiquette (Brill, 1995). Enfin, citons encore la méthodes des automates (§3.3.4.1).

Décrivons maintenant quelques logiciels utilisant des étiqueteurs. Metcalf et Meurers (2006) utilisent l'étiqueteur *TnT* (Brants, 2000), qui est basé sur les HMM, pour générer des exercices (textes à trou, glisser-déplacer, §2.6.2) à partir de textes authentiques. Par contre, *Glosser-RuG* (§B.2.11, Dokter et Nerbonne, 1998) utilise *Locolex* de Xerox Grenoble. Ce logiciel est basé sur des automates (§3.3.4.1) et reconnaît, pour la version française, environ 300 000 formes de 50 000 lemmes. Il utilise un désambiguïsateur stochastique. Par ailleurs, Keogh *et al.* (2004) et Greene *et al.* (2004) présentent un système pour l'apprentissage de l'allemand basé sur l'étiqueteur *IMS Tree Tagger*. Wagner (2004) décrit un logiciel d'exercices autour de faux amis entre l'allemand et l'anglais, également basé sur cet étiqueteur. Enfin, Liu *et al.* (2005) présentent un générateur de textes à trous basé sur un étiqueteur.

En conclusion, nous pouvons affirmer que les étiqueteurs offrent des techniques suffisamment fiables pour une utilisation en ALAO, en tout cas comme aide à l'élaboration d'exercices. Un enseignant peut utiliser un étiqueteur pour préparer un texte et réviser facilement le résultat pour remédier aux erreurs commises par l'étiqueteur. On peut également utiliser les étiqueteurs comme aide à la lecture d'un texte, notamment comme désambiguïsateur de mots, par exemple pour aider à sélectionner la bonne entrée d'un lexique. L'apprenant devrait également avoir accès aux autres analyses possibles, afin de pallier les erreurs d'étiquetage. Par ailleurs, l'étiquetage, interactif ou non, est intéressant pour l'analyse des corpus, comme nous le verrons à la section 3.1.7.

3.1.3 Reconnaissance de la parole

Dans cette section, nous commençons par décrire les buts de l'utilisation de la reconnaissance vocale dans l'ALAO. Nous continuons avec les difficultés rencontrées pour cette technique, en particulier avec les productions

d'apprenants. Nous survolons ensuite les principales techniques utilisées pour cette technique. Puis nous évoquons quelques applications de reconnaissance vocale en ALAO. Nous discutons ensuite de la problématique de la rétroaction et de la remédiation. Enfin, nous concluons par des considérations sur l'application de cette technique et sur les besoins futurs.

Parler correctement passe par une bonne prononciation. Or les langues n'utilisent qu'un nombre limité de phonèmes parmi tous ceux qui peuvent être produits par l'appareil phonatoire humain. Très tôt dans le développement de l'enfant, le cerveau apprend à distinguer les phonèmes de la langue maternelle. Les enfants ont encore une forte capacité à apprendre à distinguer les phonèmes, ce qui favorise leur apprentissage des langues [7]. Puis cette capacité est fortement amoindrie vers l'âge de la puberté. Ainsi, un locuteur italophone ou francophone aura tendance à prononcer pareillement le mot anglais *thin* [θɪn] comme le mot *sin* [sɪn] (Aist, 1999).

Si un apprenant a des difficultés pour prononcer correctement, Defays et Deltour (2003) constatent un effet de *halo*, qui occulte les autres aspects de la langue et qui fait penser à ses interlocuteurs que l'apprenant s'exprime mal, alors que son vocabulaire et sa syntaxe sont bonnes. C'est pourquoi de nombreux logiciels utilisent des techniques plus ou moins sophistiquées de reconnaissance vocale comme aide à la prononciation (Aist, 1999; Nerbonne, 2003).

Eskenazi (1999a,b) et Probst *et al.* (2002) mentionnent quelques facteurs pour un entraînement fructueux de la prononciation : (i) les apprenants doivent produire une quantité considérable de phrases ; (ii) ils doivent obtenir un *feedback* pertinent ; (iii) ils doivent entendre différents locuteurs natifs pour aider leur compréhension ; enfin, (iv) la prosodie (amplitude, durée, ton) doit être accentuée. Menzel *et al.* (2001) voient la prononciation comme une tâche de production plutôt que de reproduction, dans un but communicatif, et soulignent la nécessité de fournir une rétroaction suffisante pour identifier les lacunes et donner des moyens de s'améliorer. Les outils de reconnaissance de la parole sont essentiels dans cette optique. Enfin, un apprenant doit apprendre le contraste entre différents phonèmes, notamment quand ils ne sont pas différenciés dans sa langue maternelle, comme [ʁ-l] pour les asiatiques, [s-z-ʃ-ʒ] pour les finnois, [b-v] pour les hispanophones,

7. Forts de ce constat, depuis quelques années, les programmes scolaires, en Suisse comme dans de nombreux autres pays, incluent l'apprentissage d'une première langue étrangère dès l'âge de 8 ans environ et d'une seconde à l'âge de 10 ans. D'autre part, cette capacité rapide d'apprentissage explique la faculté des enfants d'immigrés de première génération à s'exprimer très rapidement dans la langue du pays d'accueil pour arriver en quelques années à une maîtrise égale ou proche de celle d'un locuteur natif, alors que leurs parents gardent un fort accent et une maîtrise limitée.

etc. (Defays et Deltour, 2003).

Converser avec un locuteur natif est essentiel pour l'apprentissage des langues. Il peut être important de pouvoir entendre plusieurs variétés régionales ou sur le plan du registre (Hannahs, 2007). Cependant, les apprenants ont rarement l'opportunité de pratiquer activement la langue (Aist, 1999). Dans une classe, les apprenants ont nécessairement un temps de parole limité (Defays et Deltour, 2003). De plus, Witt et Young (1998) remarquent que l'apprenant a moins peur de se tromper seul face à une machine qu'en classe face à ses pairs. Un logiciel d'ALAO doté de reconnaissance de la parole et/ou de synthèse vocale (§3.1.4) peut être un bon palliatif à ces besoins communicatifs (Harless *et al.*, 1999), même si les résultats de la reconnaissance vocale ne sont pas parfaits.

A présent, nous passons aux difficultés rencontrées par la reconnaissance vocale. La reconnaissance de la parole est certainement un des problèmes les plus ardus du traitement automatique des langues. Elle consiste à transformer un signal (un mot ou un énoncé complet) prononcé dans un microphone en une représentation utilisable par la machine : cette représentation peut être une transcription phonétique (séquence de phonèmes), un mot, une phrase ou un texte correct sur le plan grammatical (pour une introduction générale au problème, v. Egan, 1999; Lamel et Gauvain, 2003). On trouve aujourd'hui de nombreuses applications de dictée vocale ou d'interrogation de serveurs, par téléphone ou devant des bornes interactives de renseignements.

Certains logiciels demandent une prononciation d'une phrase mot à mot en marquant bien les pauses. Dans ce cas, il suffit de faire une recherche de la chaîne phonétique obtenue dans un dictionnaire. D'autres logiciels acceptent une parole continue, plus difficile à traiter : en effet, lorsqu'un locuteur parle normalement, il se produit des phénomènes de co-articulation des mots ; par ailleurs, il faut aussi traiter les faux départs, les répétitions, les bégaiements etc.

Pour la parole continue, une même séquence phonétique peut être transcrite et découpée de diverses manières, comme le montrent les exemples suivants :

(3) a. cœur ↔ chœur
    b. eau ↔ aux ↔ au ↔ ô ↔ o etc.
    c. vieil Armagnac ↔ vieillard maniaque
    d. Leur livre traînait sur leur bureau. ↔ Leurs livres traînaient sur leurs bureaux. ↔ Leurs livres traînaient sur leur bureau. etc.

e. mon beau-frère est masseur ↔ mon beau-frère et ma sœur ↔ mon beau-frère hait ma sœur etc. (Habert, 2006)

Dans cette situation, un locuteur humain pourra utiliser sa connaissance du monde et du contexte pour retranscrire correctement, ce qui est difficile à modéliser pour un ordinateur.

Enfin, la prosodie est un élément essentiel du langage parlé, qui peut aider à déterminer le sens des énoncés en délimitant des unités de sens (Simon, 2001; Grobet et Simon, 2001) et qui dénote aussi nos émotions (Bänziger *et al.*, 2001). La prosodie peut être définie comme un contour de la phrase. Les trois éléments essentiels de la prosodie sont la fréquence fondamentale (F0, voix grave / aiguë), l'intensité et la durée des syllabes (Martin, 2004b). Mertens *et al.* (2001) prennent également en compte le rythme, la qualité vocale et les prises de souffle. Simon (2001) et Grobet et Simon (2001) définissent le concept d'unité prosodique comme une unité à la fin de laquelle la fréquence fondamentale diminue.

Pour entraîner le système, certaines applications demandent à l'utilisateur d'enregistrer une série de mots ou de phrases, afin d'entraîner le système à reconnaître les contours de la voix. D'autres systèmes peuvent fonctionner en mode multilocuteurs et ne demandent pas d'entraînement préalable.

Pour résumer, diverses contraintes peuvent être imposées au locuteur : parole continue ou discontinue (chaque mot prononcé isolément en marquant une pause), vocabulaire limité ou illimité, débit limité, milieu ambiant calme avec micro de bonne qualité, entraînement préalable du logiciel, etc. Moins les contraintes sont nombreuses, plus la tâche de reconnaissance est difficile.

Passons maintenant aux difficultés d'application de la reconnaissance vocale à l'ALAO et aux parades possibles pour pallier les problèmes rencontrés. Parfois difficile pour des énoncés de locuteurs natifs, la reconnaissance vocale l'est encore plus pour des énoncés parfois très éloignés d'une prononciation correcte.

Pour améliorer la qualité de la reconnaissance, la première possibilité est d'adapter le corpus d'entraînement pour la reconnaissance vocale en y incluant des phrases provenant d'apprenants d'une ou plusieurs langues maternelles (Ehsani et Knodt, 1998; Rypa et Price, 1999; Raux et Eskenazi, 2004; Morgan, 2004; Davidson et Isenberg, 2005; Chen *et al.*, 2009). La seconde tactique consiste à restreindre l'étendue de langue à reconnaître : exercices de vocabulaire (*The Audio Interactive Tutor*, Waters, 1994, 1995), micro-monde à scénarios restreints (§2.7.3, *Spoken Electronic Language Learning*,

Hiller *et al.*, 1994; Morton et Jack, 2005), syntaxe (mots ou phrases simples, *SANTIAGO,* LaRocca *et al.*, 1999) ou corpus de phrases entières (*MILT*, §B.3.3, Holland *et al.*, 1999). Une autre technique possible est d'aligner les phonèmes de la phrase de l'apprenant avec ceux d'une phrase-type stockée dans le système, comme pour le logiciel *WinPitch* (Martin, 2004a,b,c).

Généralement, la reconnaissance vocale utilise des méthodes stochastiques pour reconnaître les différents sons et obtenir une chaîne phonétique [8]. Nous citerons l'exemple très répandu des *Modèles de Markov Cachés* (*Hidden Markov Models, HMM*) qui sont basés sur une approche probabiliste du décodage acoustico-phonétique (Knill et Young, 1997; Ehsani et Knodt, 1998; Manning et Schütze, 2000) [9]. Le signal est découpé en intervalles temporels réguliers, qui correspondent à un état représentant un modèle phonétique. Le principe essentiel sous-jacent postule que chaque état du modèle dépend de l'état précédent. Ces modèles peuvent être des phonèmes ou des unités plus grandes comme des diphones, triphones [10], syllabes, des mots ou des phrases. Le principe essentiel des modèles de Markov cachés est de calculer les probabilités d'atteindre un autre état à un temps n + 1, n + 2 etc. Ces probabilités sont calculées à partir d'un apprentissage basé sur de grandes quantités de données, si possible représentatives de la tâche à effectuer et du public visé (variation dialectale et sociologique des locuteurs). Chen *et al.* (2009) décrivent les différentes techniques d'évaluation de la prononciation d'apprenants, basées notamment sur les HMM.

Décrivons à présent quelques applications basées sur des HMM. Le logiciel commercial *Dragon NaturallySpeaking* est intégrable à d'autres logiciels, d'où son succès commercial, et permet d'optimiser la reconnaissance de séquences répétitives grâce à des réseaux de transition (§3.3.4.1 [11]) dont les étiquettes d'arcs sont les mots à reconnaître. On citera également *Decipher*, *Nuance*, *Hidden Markov Model Toolkit* (HTK, Young *et al.*, 2006) et *Sphinx-II* (Huang *et al.*, 1993).

Examinons quelques exemples de logiciels qui utilisent l'ALAO. D'après Aist (1999), l'apprenant bénéficiera beaucoup plus d'une tâche concrète à réaliser (donner des ordres pour agir sur un micromonde) plutôt que des

8. Pour Ehsani et Knodt (1998) en revanche, cette tâche implique une grande variété de connaissances linguistiques, phonologiques, lexicales, sémantiques, grammaticales et pragmatiques.

9. Voir §3.1.2 pour l'application des HMM aux étiqueteurs.

10. Un diphone est la portion de parole comprise entre deux parties stables de phonèmes consécutifs. Ainsi *émigrante* est composée des diphones [e] + [em] + [mi] + [ig], etc. Le triphone est la partie stable entre trois phonèmes consécutifs.

11. Bien que cette section concerne l'analyse du langage, le principe des états et des transitions est le même.

exercices de répétition semblable aux exercices sur bande magnétique des laboratoires de langue. Les micromondes en sont le meilleur exemple (Hiller *et al.*, 1994; Rypa et Price, 1999; Levi *et al.*, 2004; Morgan, 2004; Mote *et al.*, 2004; Morton et Jack, 2005), avec des activités variées pilotées par la voix, ou des jeux (Pennington et Esling, 1996; Dalby et Kewley-Port, 1999). Des applications plus classiques existent, comme la dictée d'un texte (Coniam, 1998), comparaison de prononciation avec des locuteurs natifs (Fairfield, 1999), paires minimales (Eskenazi, 1999a; Wachowicz et Scott, 1999) ou aide à l'apprentissage de la lecture (Mostow et Aist, 1999; Mostow *et al.*, 2002).

Abordons maintenant les différents types de remédiation. D'après Ehsani et Knodt (1998, p. 50), la meilleure remédiation est "*a type of feedback that does not rely on the student's own perception*". Une grande partie des logiciels dotés de reconnaissance vocale se basent sur des rétroactions graphiques [12] – et parfois uniquement sur elles. Le graphique le plus courant est l'oscillogramme (v. fig. 3.1), qui retranscrit simplement les vibrations du signal, ce qui est une indication peu pertinente. L'exemple emblématique de logiciel utilisant un oscillogramme est *Talk to me* [13] (Hincks, 2003).

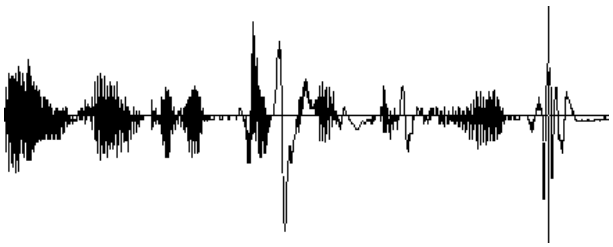


FIG. 3.1 – *Exemple d'oscillogramme*

Les spectrogrammes (fig. 3.2) marquent les vibrations sonores produites par le conduit vocal à différentes fréquences et permettent de distinguer l'énergie en décibels à certaines fréquences. Parmi les logiciels utilisant des spectrogrammes, citons *Athena* (§B.3.1) et *The Rosetta Stone* (Fairfield, 1999).

Aist (1999) prône l'utilisation de spectrogrammes simplifiés afin que le locuteur puisse distinguer les formants de sa prononciation et de celle d'un locuteur natif. Chaque son vocalique (ou voyelle) possède quatre formants [14],

12. On trouvera une revue des différentes courbes affichées par des logiciels chez Cazade (1999).

13. En guise de remédiation, le logiciel propose aussi courbes mélodiques et des figures montrent la position des lèvres et une coupe de la cavité buccale pour chaque phonème.

14. Les formants sont le reflet des résonances des cavités vocales qui renforcent le son émis

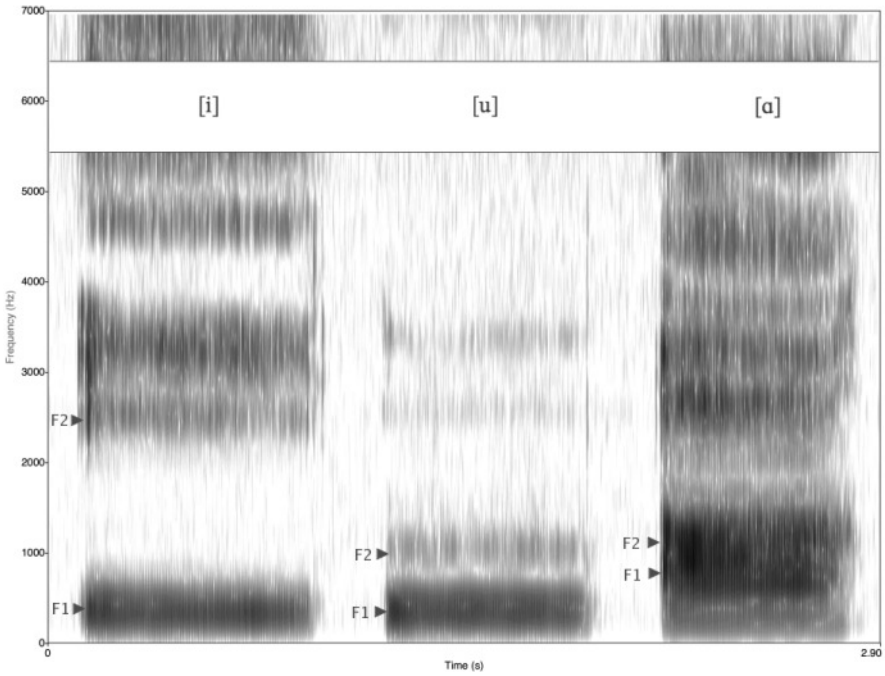


FIG. 3.2 – *Exemples de spectrogrammes avec marquage des deux premiers formants*

dont deux sont discriminants pour le distinguer des autres sons vocaliques. Les autres paramètres acoustiques à considérer sont la force, le tempo et la durée.

Il existe également d'autres outils : électropalatographe pour repérer le mouvement de la langue contre le palais (Gibbon *et al.*, 1991), coupe animée du conduit vocal (Carson-Berndsen, 1998), têtes animées (Kirschning, 2004), visualisation des contours intonatifs (Chun, 1998), histogrammes (Cazade, 1999) et calcul d'un score de similarité avec la prononciation correcte (Witt et Young, 1998; Aist, 1999; Harless *et al.*, 1999).

On trouve aussi des rétroactions orales sous forme de demande de répétition (Rypa et Price, 1999) ou de prononciation de l'énoncé correct en accentuant fortement les points posant problème (Raux et Eskenazi, 2004). La rétroaction peut être également non verbale, avec des agents dans un

par les vibrations des cordes vocales à des fréquences différentes selon la cavité mise en jeu. Ces cavités sont agrandies ou diminuées par la position des différents organes du conduit vocal (langue, glotte, pharynx, palais, mâchoire, lèvres). Le formant le plus bas en fréquence est appelé *fréquence fondamentale* (F0).

micromonde qui manifestent par des signes non verbaux qu'ils n'ont pas compris (Hiller *et al.*, 1994; Morton et Jack, 2005). Enfin, Wang et Seneff (2007); Seneff *et al.* (2007) demandent de traduire une phrase oralement d'anglais à mandarin et traduisent en anglais ce que le logiciel a compris.

Pour conclure cette section, la reconnaissance vocale est la technique de TAL la plus répandue dans les logiciels d'ALAO. Cette popularité est notamment due à la relative facilité de mettre en place des outils : les techniques en jeu sont relativement basiques, comme le repérage de phonèmes ; en outre, il existe de bons reconnaisseurs commerciaux dotés d'outils qui permettent leur intégration à d'autres logiciels. La popularité de la reconnaissance vocale en ALAO s'explique également par le besoin primordial d'apprendre à communiquer oralement.

Malheureusement, cette abondance de logiciels ne se traduit pas par une grande qualité pédagogique. Les aides à la prononciation sont davantage des gadgets *tape-à-l'œil* que des véritables apports pédagogiques. Ainsi, dans sa présentation de *Talk to me*, Hincks (2003) conclut que ce logiciel est mieux adapté aux débutants qui ont une prononciation mauvaise qu'aux apprenants plus avancés, qui bénéficient peu de ce genre de logiciel. Nous avons vu qu'il existe de nombreuses propositions pour apporter une rétroaction utile pour l'apprenant. Cependant, ces solutions ne sont pour l'instant qu'à l'état de prototype de recherche.

Au niveau pédagogique, les tâches proposées par les logiciels dotés de reconnaissance vocale existant sur le marché ne sont pas satisfaisantes. Les apprenants doivent répéter des mots et des phrases, ce qui n'entraîne pas leurs capacités à communiquer. Ces exercices sont parfois utiles au niveau débutant mais s'avèrent rapidement insuffisants. Il serait utile de faire réaliser des tâches à l'apprenant, qui dialoguerait avec un interlocuteur fictif qui lui donne un feedback sous forme de conversation.

Avec l'évolution technologique, il est probable que les outils de reconnaissance gagneront en performance dans un proche avenir. Les ordinateurs sont équipés de périphériques audio de qualité suffisante. Les processeurs deviennent de plus en plus performants et les ordinateurs standards ont de grandes capacités de disque et de mémoire. Grâce aux besoins en dictée vocale et en interfaces pilotées par la voix, le secteur de la reconnaissance vocale ne devrait pas manquer de moyens pour la recherche et développement. Logiquement, les logiciels d'ALAO devraient bénéficier de cet essor. Cependant, pour améliorer les performances des reconnaisseurs, il faut disposer de coûteux corpus de parole (§3.1.7), notamment avec des phrases d'apprenants, ce qui pourrait dissuader certains éditeurs de logiciels.

Dans le domaine de la recherche, il est souhaitable de réunir le plus possible de données sur les erreurs typiques des apprenants. Il devrait être possible d'en tirer des typologies qui permettent d'affiner les diagnostics, notamment en fonction de la langue première des apprenants.

Terminons par deux idées pour améliorer la rétroaction. Afin d'apprendre à marquer le rythme, nous suggérons, comme aide secondaire, d'utiliser des battements réguliers à la manière d'un métronome. Cela permet de mieux marquer les rythmes dans une phrase, qui sont parfois des indices d'émotion dans certaines langues. Cet outil pourrait aider les apprenants débutants. Par ailleurs, le diagnostic devrait être affiné grâce aux modèles de l'apprenant (§2.7.4) et, par exemple, la tolérance face aux erreurs devrait être diminuée au fur et à mesure des progrès des apprenants.

3.1.4 Synthèse vocale

Dans cette section, nous commençons par décrire les techniques et les problématiques de la synthèse vocale. Puis nous abordons les différentes applications de la synthèse pour l'ALAO. Nous poursuivons par certains exemples de logiciels. Enfin, nous terminons par un bilan et des perspectives de l'utilisation de la synthèse pour l'ALAO.

Commençons par aborder les techniques et problématiques de la synthèse vocale. Cette technique consiste à faire produire par une machine un énoncé oral à partir de de l'écrit, en passant par une transcription phonétique (Fuchs *et al.*, 1993). La synthèse vocale est un domaine assez ancien du TAL. Si les sons produits par les ordinateurs étaient très métalliques à l'origine, voire même difficilement compréhensibles, les résultats actuels sont tout à fait utilisables pour une utilisation large (Harashima, 2006).

Pour synthétiser une phrase, plusieurs étapes sont nécessaires [15] :

- analyse grammaticale ou étiquetage du texte, afin de désambiguïser les mots et de repérer les syntagmes ;
- représentation phonétique de l'énoncé, à partir d'un lexique contenant une représentation phonétique des mots et/ou de règles de phonétisation ;
- application de paramètres prosodiques ;
- génération du son par le synthétiseur.

15. D'autres approches sont possibles, notamment des méthodes stochastiques, mais nous ne les évoquerons pas ici.

Le repérage des syntagmes permet de résoudre la plupart des homographes hétérophones, comment *couvent* qui se prononce différemment s'il est substantif ou verbe (Gaudinat et Wehrli, 1997; Goldman, 2001). Les autres difficultés sont la prononciation des nombres, des mots inconnus (notamment les néologismes, les nom propres, etc.), les abréviations (*St.* en anglais abrège *Saint* et *street*), les sigles, etc.

Passons maintenant à l'application de la prosodie. Pour que la synthèse paraisse naturelle, il est indispensable que le son produit varie en fonction des paramètres prosodiques : hauteur, intensité et durée. Cette variation est indispensable pour produire un son acceptable pour les auditeurs et a un rôle fondamental dans la communication orale. Les marques d'intonation sont souvent utilisées par l'interlocuteur comme indice de l'information que le locuteur a voulu faire passer, par exemple l'accentuation d'un mot ou un groupe de mot. Les synthétiseurs vocaux doivent donc reproduire ces contours mélodiques afin que le son soit acceptable pour une oreille humaine [16]. D'autres paramètres interviennent, comme le débit, le rythme accentuel et la co-articulation à l'intérieur des mots et entre les mots (Dutoit et Stylianou, 2003). Le rapport du UMIST Centre for Computational Linguistics (2002) énonce les critères de qualité, de flexibilité et le caractère naturel de la synthèse pour une comparaison entre différentes techniques. Handley et Hamel (2005) ajoutent que la flexibilité du système doit aussi permettre de varier le registre (plus ou moins formel), la voix (masculin, féminin, jeune ou vieux) et le timbre.

A présent, nous abordons l'étape finale, la production de son par l'ordinateur. Différentes techniques permettent de synthétiser la voix humaine (Dutoit *et al.*, 2002; Dutoit et Stylianou, 2003). Citons parmi d'autres :

- la synthèse par règles, qui modélise la parole sous forme de spectre sonore, en jouant sur les harmoniques ;
- la synthèse par formants, où différents filtres sont appliqués aux sons pour simuler l'effet des cavités vocales sur l'amplification de certaines bandes de fréquences ;
- la synthèse par mots, où les différents mots ou séquences de mots sont pré-enregistrés dans le cadre d'une application à couverture restreinte ;
- la synthèse par concaténation de diphones, où des sons pré-enregistrés sont les transitions entre les zones stables de deux phonèmes de la langue ;
- la synthèse par sélection d'unités dans une grande base de données s'apparente à la technique précédente : elle recherche la meilleure unité

16. On se rappellera les robots des films de science-fiction au ton monocorde et métallique, qui lassent vite l'interlocuteur.

(diphone ou autre) dans une base d'enregistrements qui correspond le mieux au résultat voulu, en hauteur et en longueur ; ainsi, le même diphone sera enregistré plusieurs fois avec différentes variantes.

Parmi les techniques évoquées, la dernière obtient les meilleurs résultats, mais demande un espace disque considérable. De plus, la concaténation des sons est rendue plus difficile car il faut calculer le meilleur chemin parmi tous les sons disponibles. Toutefois, le prix des disques durs n'est plus un obstacle important et les processeurs sont de plus en plus performants.

Notons pour terminer qu'aux débuts de la micro-informatique, certains micro-ordinateurs étaient parfois dotés d'un synthétiseur intégré, comme le *Commodore 64* (Hackenberg, 1985) ou le *Victor 9000* (Fischer, 1986).

Nous passons maintenant à l'utilisation de la synthèse vocale en ALAO. Si un locuteur natif peut être tolérant à propos d'erreurs de prononciation, ce n'est pas le cas pour un apprenant d'une langue. La synthèse de la parole est utilisée comme aide à la prononciation (Nerbonne, 2003) ou comme partenaire virtuel pour simuler une conversation (Handley et Hamel, 2005), afin de simuler une immersion dans la langue, de pouvoir écouter n'importe quelle phrase d'un logiciel et d'éviter les coûts élevés d'un enregistrement sonore professionnel (Hamel et Wehrli, 1997). Skrelin et Volskaja (1998) et Harashima (2006) ajoutent qu'il est utile de synthétiser les consignes des exercices, que les apprenants ont tendance à éviter ou survoler. D'après Defays et Deltour (2003), la compréhension orale est une tâche fondamentale. Les apprenants abordent des difficultés et des aspects tels que la différenciation des phonèmes, le vocabulaire, les formes grammaticales, les niveaux de langue, l'intonation, le débit, l'accent, le rythme et les paramètres non verbaux (gestuelle) le cas échéant. De même, chaque langue a un contour mélodique différent (Aist, 1999). Il est donc nécessaire de varier aussi les différents modes comme l'exclamation, l'interrogation, l'expression d'émotions etc., ainsi que le timbre avec des voix d'hommes, de femmes et d'enfants (Aist, 1999; Probst *et al.*, 2002; Esling, 1992). Eskenazi (1999b) ajoute qu'il est souhaitable d'écouter des énoncés aux contours prosodiques exagérés (amplitude, durée, ton) et Skrelin et Volskaja (1998) qu'il est important que les apprenants puissent régler le rythme.

L'aide à la prononciation est aussi une application importante. Il est en effet difficile de passer de la prononciation de *to laugh*, de *night*, de *women*, de *Monsieur*, *maintenant* etc. à la version écrite et inversement. Certaines assimilations phonétiques posent aussi des problèmes, comme par exemple dans *exact*, prononcé [ɛgzakt], alors que la lettre x se prononce [ks]. Il n'y a pas d'autres moyens que d'apprendre par cœur ce lien entre prononciation

et graphie. La synthèse vocale permet à l'apprenant d'entendre les mots au lieu de devoir déchiffrer la transcription phonétique dans un dictionnaire, au risque de mal l'interpréter [17].

Poursuivons maintenant par la descriptions de quelques logiciels qui utilisent la synthèse vocale. Cette caractéristique existe depuis longtemps grâce à l'inclusion de synthétiseurs dans certains modèles de micro-ordinateurs (*Generate*, Hackenberg, 1985, §B.2.9, Fischer, 1986, Cohen, 1993). Les micromondes bénéficient beaucoup des synthétiseurs (*Herr Kommissar*, DeSmedt, 1995, §B.3.5, Douglas, 1995; Tomlin, 1995, *LingWorlds*). *Ordictée* (Guyomard *et al.*, 1997) est une application de dictée vocale à correction automatique. En outre, on peut aussi citer les logiciels *SAFRAN* (§B.2.30, Hamel, 1996), le logiciel de Skrelin et Volskaja (1998), *VINCI* (Thomas *et al.*, 2004a, §B.2.38) et *ITSPOKE* (Forbes-Riley et Litman, 2005, §B.2.41).

Pour conclure, nous passons à un court bilan de l'application de la synthèse vocale et aux perspectives d'avenir. L'intelligibilité de la synthèse vocale est aujourd'hui suffisante pour une application pédagogique. Les synthétiseurs vocaux permettent de choisir plusieurs voix, et il devient plus difficile de distinguer une voix de synthèse d'une voix naturelle. Cette technique convient parfaitement pour lire un énoncer, dicter un texte, donner la prononciation d'un mot ou d'une phrase, etc.

Par contre, la technique n'est sans doute pas encore parfaitement mûre pour entraîner la compréhension orale ou pour des systèmes de dialogue réaliste. Il faudrait faire varier la prosodie en fonction de l'énoncé, faire ressentir les émotions, varier les registres, etc. L'application de la prosodie a certainement une grande marge de progrès. On pourrait passer par une analyse sémantique de l'énoncé, mais cette solution est encore irréaliste. Cependant, vu le coût de production de séquences pédagogiques audio ou vidéo ou d'acquisition de documents authentiques, la synthèse vocale est un bon compromis.

Enfin, signalons une piste prometteuse du côté du paramétrage fin de la synthèse vocale. Du côté de l'enseignant, il serait judicieux de pouvoir donner des indications au synthétiseur pour accentuer un aspect particulier. Le modèle de l'apprenant (§2.7.4) pourrait aussi fournir des indications utiles sur les lacunes de l'apprenant et sur les aspects à mettre en valeur. Enfin, l'apprenant lui-même devrait pouvoir intervenir sur certains aspects de l'énoncé, pour observer l'effet des modification, ou simplement pour ralentir

17. Cependant, dans une évaluation, Handley et Hamel (2005) constatent que la synthèse vocale est davantage appréciée comme partenaire de conversation ou comme système de lecture que comme aide à la prononciation.

ou accélérer le débit ou accentuer ou diminuer l'amplitude.

3.1.5 Génération syntaxique

Dans cette section, nous commençons par définir les principes et techniques de génération automatique. Ensuite, nous décrivons les possibilités d'utilisation de la génération syntaxique en ALAO avec exemples de logiciels à l'appui et nous terminons par quelques réflexions en conclusion.

La génération automatique est un processus qui consiste à produire un énoncé correct en langue naturelle à partir d'une représentation sémantique ou d'informations tirées d'une base ou d'une source de données (Danlos et Roussarie, 2000). Elle est utilisée en traduction automatique (§3.1.6), en génération de résumés (§3.1.8) et en dialogue homme-machine (§3.1.9), par exemple pour l'interrogation de bases de données en langue naturelle.

Bateman et Zock (2003) distinguent trois domaines d'expertise nécessaires pour la génération du langage : connaissance du domaine (quoi dire), connaissance de la langue (lexique, grammaire, sémantique) et connaissances rhétoriques stratégiques (comment atteindre les buts communicatifs, style, etc.). La génération de la langue nécessite de planifier l'information qui doit être communiquée, puis de la lexicaliser et enfin de produire la structure de surface de la phrase en insérant les mots au bon endroit et dans une forme correcte. Pour sa part, Roussarie (1998) souligne l'importance d'une organisation soignée du discours. Il est nécessaire que le texte généré apparaisse le plus naturel possible. Nous décrirons des systèmes de représentation sémantique en (§3.4) et des systèmes de représentation lexicale en (§3.5)

Passons maintenant à l'utilisation de la génération en ALAO : elle peut être utilisée pour générer des dialogues dans des micromondes (§2.7.3), pour générer des exercices, pour manipuler des représentations sémantiques et pour reformuler des phrases. A présent, nous décrivons plus en détail chacun de ces domaines d'utilisation.

Commençons par la génération de dialogue. A partir des informations internes sur l'état du micromonde et les actions entreprises par l'apprenant (ordres oraux ou écrits, manipulation d'objets, etc.), le système peut générer des phrases en fonction du but à atteindre, pour aider l'apprenant à résoudre la tâche à accomplir ou pour lui apporter une rétroaction. *Die Sprachmaschine* (Harroff, 1986, §B.3.10) est un micromonde où les apprenants sélectionnent des mots dans des listes. Le programme génère ensuite une phrase en allemand et peut répondre à des questions prédéterminées sur

la nature de certains mots. Hambuger *et al.* (1999) utilisent un générateur pour simuler une conversation. Ainsi, selon eux, l'apprenant apprend à manier la langue et cette conversation pallie le manque d'immersion dans la langue avec des textes ou des locuteurs authentiques. Enfin, le micromonde *LingWorlds* (Douglas, 1995; Tomlin, 1995) cherche à entraîner la compréhension orale chez des apprenants débutants en manipulant des objets après avoir entendu des instructions orales données par synthèse vocale (§3.1.4). Le but est d'entraîner les structures simples de la langue et de connaître les prépositions spatiales. Les instructions sont générées à partir de représentations internes de l'état du micromonde.

Passons maintenant à l'utilisation de la génération pour l'élaboration d'exercices. Le système peut générer des phrases agrammaticales correspondant à des fautes fréquentes et les faire corriger par l'apprenant ou construire des phrases correctes pour des exercices de grammaire. La génération est également utile pour des exercices de transformation de phrases, les textes à trous ou la complétion de phrases (Bates et Ingria, 1981, *ILIAD*, Wilson, 1986; Loritz, 1995, *PARNASSUS* Neuwirth, 1989, *PILEFACE* Lelouche, 1991, B.2.26, *ILLICO*, Pasero et Sabatier, 1998, §B.5.8, Zamorano Mansilla, 2004). Pour enseigner les structures grammaticales, *PARSER* (Bailin et Thomson, 1988, §B.2.24) génère automatiquement une phrase et demande à l'apprenant d'identifier certains constituants.

Maintenant, passons à la manipulation de structures sémantiques. Les systèmes de génération utilisent des représentations du sens des phrases qui sont ensuite transformées en énoncés grammaticaux. Il peut être utile de manipuler des structures sémantiques, par exemple sous forme de graphes conceptuels, afin d'aider l'apprenant à exprimer des idées. Zock (1992) utilise une représentation des phrases sous forme de graphes conceptuels dénotant les relations prédicat-arguments comme entrée d'un générateur pour le logiciel *SWIM* (§B.4.33). Zock (2006) propose d'utiliser un dictionnaire électronique pour aider l'apprenant à organiser ses idées et de générer ensuite automatiquement une phrase à l'aide de mécanismes simples. Citons encore *TAEMA* (*Traitement Automatique de l'Écriture de Mots Affectifs*, Buvet et Issac, 2006).

Enfin, la génération peut servir à reformuler des phrases incorrectes. Wilks et Farwell (1992) prônent la génération d'une phrase en langue première de l'apprenant afin de lui montrer comment le système a compris sa phrase erronée en langue seconde et de l'aider à formuler une phrase correcte. C'est notamment le cas des logiciels du projet *Athena* (§B.3.1). Vandeventer (1998) propose d'utiliser un générateur pour produire une forme correcte à partir d'une phrase agrammaticale, comme le fait le correcteur *Arboretum*

(Bender *et al.*, 2004, §C.4).

Nous pouvons conclure que la génération syntaxique est une technique suffisamment éprouvée pour être utilisée en ALAO. Il est en effet plus facile de contrôler la production de phrases par un système que les productions des apprenants eux-mêmes. Les micromondes permettent aussi de restreindre l'étendue de langue à couvrir. Cependant, les coûts de production de ces systèmes expliquent leur rareté dans des systèmes commerciaux.

3.1.6 Traduction automatique

Dans cette section, nous abordons la traduction automatique et son application à l'ALAO. Nous évoquons d'abord la problématique de la traduction automatique. Nous poursuivons brièvement par les différentes techniques de traduction et les résultats qu'elles atteignent. Nous parlons ensuite de leur application à l'ALAO. Enfin, nous concluons par quelques réflexions de bilan.

La traduction automatique est l'un des domaines les plus étudiés du traitement automatique du langage. Le terme consacré français de *traduction automatique* (TA) est improprement traduit de l'anglais *Machine Translation* (MT), qui désigne indifféremment la traduction par ordinateur avec ou sans assistance humaine, selon Hutchins (2003).

Les ambiguïtés syntaxiques des langues rendent difficile les traductions. L'attachement des syntagmes prépositionnels est un problème bien connu :

(4) a. Notre groupe était composé d'enseignants de langues et de mathématiciens.
b. La réponse à la question de cet étudiant était mauvaise.

Dans la phrase (4a), il ne fait pas de doutes pour un humain que *et* coordonne *enseignants* et *mathématiciens*, et non pas *langues* et *mathématiciens*. Pourtant, la seconde combinaison serait tout à fait possible syntaxiquement, mais incorrecte sémantiquement – notamment parce que les *langues* ou les *mathématiques* s'enseignent, mais pas les *mathématiciens*. Dans le cas de (4b), *de cet étudiant* peut être attaché à *question* ou à *réponse* [18]. Dans

18. Dans ce dernier cas, il serait préférable d'utiliser la préposition *par*, mais, selon nous, cette interprétation est également grammaticale avec *de*.

certaines langues, il est possible de conserver certaines ambiguïtés dont regorge la langue source, mais dans des langues à cas, les désinences seront différentes et il faudra faire un choix. Un autre problème connu est la résolution de l'anaphore (Gaiffe, 2006). Les pronoms sont souvent marqués par le genre. Or celui-ci est parfois arbitraire et varie d'une langue à l'autre (Ayoun, 2007).

De plus, la plupart des mots ont plusieurs traductions, qui produisent des nuances voire même des sens totalement différents. En particulier, les expressions idiomatiques sont des locutions figées, parfois argotiques, qui revêtent une signification particulière. Ainsi, *casser sa pipe* doit être rendu avec un sens non littéral, en anglais *to kick the bucket* (pour une discussion plus complète, voir Wehrli, 1998, 2000; Nerima *et al.*, 2006).

Enfin, la langue a tendance à produire des combinaison de mots récurrentes et arbitraires, que l'on nomme *collocations*. Ces combinaisons concernent des associations nom-adjectif, adjectif-nom, verbe-nom, nom-verbe, nom-préposition-nom et verbe-préposition-nom. Ces collocations ne peuvent être extraites correctement qu'à partir d'une analyse syntaxique (Nerima *et al.*, 2006; Seretan et Wehrli, 2006, 2007; Seretan, 2009). Si l'on utilise des synonymes ou quasi-synonymes, les phrases produites paraîtront étranges ou presque incorrectes pour les locuteurs natifs. Examinons l'exemple suivant, donné par Nerima *et al.* (2003) :

(5) a. exercer une profession.
b. to practice a profession.
c. ?pratiquer une profession.

La collocation en (5a) est traduite par la collocation anglaise en (5b) ; en revanche, la traduction directe de (5b) en (5c) semblera étrange, voire incorrecte, bien que l'on en comprenne le sens.

Venons-en maintenant aux techniques de traduction par ordinateur (Boitet, 2000; Lapalme et Macklovitch, 2006). Les premiers traducteurs mot par mot ont très vite atteint leurs limites. C'est alors que des approches indirectes ont été développées : le texte d'entrée est converti vers une représentation sémantique, qui sert ensuite de base à la génération en langue cible. L'*approche par langage-pivot* (ou *interlangue*) permet de s'affranchir de la dépendance de la syntaxe. Le même langage pivot est théoriquement utilisable pour la traduction de plusieurs paires de langue. L'*approche par transfert* convertit un texte en une représentation R. Un module de transfert convertit alors cette représentation vers une représentation R' en langue

cible, qui est utilisée par le module de génération. Le modèle des représentations sémantiques est moins abstrait et plus facile à réaliser que l'approche des langages-pivot.

Depuis le début des années 1990, une nouvelle approche empirique, la *traduction automatique basée sur l'exemple* (*Example-based MT*), est apparue (Somers, 2003), basée sur des corpus de textes brut et leur traduction. Elles consistent à rechercher des séquences de mot déjà traduites ou les plus proches possible de la phrase ou du segment de phrase à traduire et de les recombiner automatiquement. Quant à la traduction automatique statistique, elle est aussi basée sur des corpus bilingues mais utilise des statistiques sur l'ordre des mots dans la langue cible et sur les correspondances entre mots (Somers, 2003). Des systèmes hybrides combinent traduction par règle, traduction basée sur l'exemple et/ou traduction par statistique.

Terminons ce survol des techniques par un constat mitigé. Malgré l'évolution technique des ordinateurs, les résultats atteints par les traducteurs automatiques sont encore pour la plupart insatisfaisants, mis à part quelques systèmes fonctionnant sur des domaines restreints comme le traducteur de bulletins météorologiques *TAUM-Météo* (Chevalier *et al.*, 1978). Pour la plupart des système, une révision profonde (ou post-édition) par un traducteur humain est indispensable.

Passons maintenant à l'application de la traduction automatique à l'ALAO. Nous avons recensé trois domaines d'utilisation :

- aide à l'expression dans la langue seconde ;
- apprentissages des techniques de traduction et de post-édition, notamment avec les mémoires de traduction ;
- utilisation d'outils de traduction en contexte.

Commençons par l'application de la traduction comme aide à l'expression pour débutants. Wang et Seneff (2004) présentent un environnement restreint de conversation de parole à parole pour l'apprentissage du mandarin pour anglophones. Les domaines sont très spécifiques, comme par exemple la recherche d'informations météorologiques. Le système aide l'apprenant à interagir en lui offrant la possibilité de s'exprimer dans la langue seconde ou de traduire ses paroles exprimées dans sa langue première. Le tuteur d'apprentissage de l'anglais pour apprenants chinois XTRA-TE (*English Chinese Sentence TRAnslator to Teach English*, Chen et Kurtz, 1989; Danna, 1997, §B.2.42) propose aux apprenants de traduire des phrases d'une langue à l'autre.

Poursuivons par l'apprentissage des techniques de traduction. *Bon Accord* (Farrington, 1994) est un logiciel d'apprentissage des techniques de traduction, basé sur une technologie simple de mémoire de traduction. D'après Somers (2004), la traduction automatique peut être utilisée pour illustrer des contrastes entre L1 et L2, grâce à la préservation des structures de la L2 lors du transfert vers la L1. D'autres exemples sont donnés par La Torre (1999), Decrozant et Voss (1999), Winiwarter (2004) et Niño (2005).

Terminons ce survol par l'utilisation d'outils d'aide à la traduction de mots en contexte. *COMPASS* (*COMPrehension ASSisstant*, Breidt *et al.*, 1996; Breidt et Feldweg, 1997) est un système d'aide à la traduction de textes en ligne, qui offre la traduction de mots en contexte. *GLOSSER* (§B.2.11, Dokter et Nerbonne, 1998) permet d'aider à déterminer le sens des mots dans leur contexte. De même, *Twic* (*Translation of Words In Context*, Wehrli, 2006, §5) est un autre exemple d'aide à la traduction qui permet de faciliter la lecture des documents en ligne. De tels outils peuvent s'avérer très utiles pour des apprenants assez avancés, en les aidant à comprendre des textes.

Pour conclure, la traduction automatique n'est pas beaucoup utilisée dans le domaine de l'ALAO, principalement pour deux raisons : premièrement, les résultats ne sont pas à la hauteur pour une utilisation pédagogique ; deuxièmement, un apprentissage doit se faire le plus possible par immersion, sans recours à la langue première. Toutefois, il est parfois acceptable que des apprenants débutants fassent appel à un traducteur vers leur langue première pour mieux comprendre certaines phrases. L'apprenant se rendra facilement compte si une phrase est vraisemblable, grâce aux intuitions qu'il a dans sa propre langue. L'opération inverse est en revanche à proscrire, car les intuitions des apprenants seront insuffisantes. Les outils d'aide à la traduction en contexte sont également une piste récente qui mérite d'être explorée.

3.1.7 Concordanciers et autres outils de traitement de corpus

Dans cette section, nous commençons par définir la notion de corpus et de concordancier. Nous discutons ensuite de l'utilité des corpus pour l'apprentissage des langues, puis poursuivons par une description de logiciels. Nous terminons par quelques remarques de conclusion.

Les corpus sont de gros recueils de données linguistiques (orales ou écrites) destinés à attester l'utilisation de certaines structures ou mots de la langue. De nos jours, les corpus sont accessibles par ordinateur, ce qui facilite l'uti-

lisation et le traitement des données. Pour le traitement des langues, ils constituent le carburant brut à partir duquel on construit des applications ou la batterie de test à partir de laquelle on teste ces applications (McEnery, 2003).

Définissons maintenant les différentes sortes de corpus. Certains corpus récoltent des données orales ou écrites de locuteurs, natifs ou non. Les données orales peuvent être sous forme d'une banque de sons ou d'une transcription écrite de l'oral [19]. D'autres corpus récoltent des textes d'un genre particulier (articles de journaux, textes littéraires, etc.) ou des mélanges de genres. Il existe des corpus multilingues, avec des textes dans différentes langues. Quant aux corpus parallèles, ils contiennent le même texte dans plusieurs langues, par exemple des dépêches d'agence de presse ou des textes de loi dans les pays plurilingues.

A côté du son et de l'écrit, on trouve également des transcription phonétiques et des textes annotés avec des informations linguistiques, telles que les parties du discours, voire même avec des informations sémantiques (McEnery et Wilson, 1993; Véronis, 2000; Habert, 2006). Ces annotations peuvent être faites manuellement ou par un étiqueteur (§3.1.2). Le format *XML* (§2.7.5) devrait s'imposer de plus en plus dans le monde des corpus grâce à son caractère standard, ouvert, flexible et évolutif et grâce à l'existence de nombreuses librairies de traitement standard dans les environnements de programmation. Les corpus arborés (*treebanks*) sont des corpus annotés avec des informations syntaxiques, qui sont utiles notamment pour les corpus d'entraînement des méthodes stochastiques d'analyse (§3.3.3.3) mais sont impossibles à annoter automatiquement (Véronis, 2000), notamment à cause de l'ambiguïté de certains attachements de syntagmes ; une alternative peut être une annotation partielle par un analyseur superficiel (§3.3.4.3). L'annotation sémantique, quant à elle, cherche soit à distinguer le sens des mots, soit à marquer les phénomènes discursifs tels que la résolution des anaphores (Véronis, 2000).

La taille des corpus est extrêmement variable, mais on recherche avant tout la représentativité et la qualité de l'échantillonnage des données (McEnery et Wilson, 2005). Un choix judicieux de textes est essentiel pour garantir la représentativité du vocabulaire et son adéquation avec le but pédagogique et le niveau des apprenants (Lamy *et al.*, 2005). Un bon corpus de locuteurs représentera les deux sexes équitablement ainsi que les différentes couches

19. Dans ce cas, il faut noter les paramètres tels que les hésitations, les reprises, etc. Par ailleurs, la tâche d'interprétation du transcripteur est considérable, car il se heurte aux mêmes difficultés évoquées à la section 3.1.3 pour les reconnaisseurs vocaux. Les reconnaisseurs vocaux sont d'ailleurs parfois utilisés pour transcrire des corpus oraux. Pour plus de détails sur cette problématique, on se référera à Véronis (2000).

sociologiques, différents âges, différents dialectes, etc. Un corpus littéraire représentera différents genres, différentes époques, etc.

Définissons maintenant la notion de concordancier : il s'agit d'un moyen d'accès à un corpus de textes pour montrer l'usage d'un ou plusieurs mots en contexte ou d'une partie du discours, dans un but de recherche littéraire, stylistique ou même linguistique (Flowerdew, 1996; Lamy *et al.*, 2005). Les concordanciers présentent généralement l'occurrence du terme de recherche (mot, partie du discours) soit entourée d'une fenêtre de quelques mots, soit dans son paragraphe ou soit dans la phrase qui la contient. Les concordances peuvent être affinées par l'utilisation de dictionnaires de quasi-synonymes. Un lemmatiseur (§3.1.1.4) permettra également de trouver par exemple les occurrences d'un verbe à tous les temps.

Pour terminer le survol des techniques, mentionnons les systèmes d'indexation de textes dans des bases de données ; même si ces bases ne sont pas des corpus à proprement parler, leur utilisation s'en approche. Ces systèmes d'indexation permettent de charger des textes dans une base de données et de les retrouver par mots-clés, à l'instar des moteurs de recherche par *Internet*. Chaque document est indexé en fonction des mots qu'il contient par un vecteur (ou plusieurs vecteurs lorsque l'unité textuelle est le paragraphe) dont chaque dimension représente un lemme ; chaque dimension représente la fréquence du mot dans le document et la répartition du mot dans la base de données textuelle (Fluhr, 2000, p. 242). Une base de données est indexée par une matrice de valeurs, où une dimension représente les lemmes et l'autre dimension les documents ou paragraphes. Les techniques de calcul vectoriel permettent par exemple de regrouper les documents les plus proches d'après leur sens. La proximité entre les documents est donnée par le cosinus des vecteurs qui le représentent.

Passons maintenant à l'utilisation des corpus en ALAO. Ils sont utiles à deux titres, comme le soulignent McEnery et Wilson (2005) :

> "Corpora can provide the basis of accurate, empirically justified, linguistic observations on which to base CALL materials. Additionally, the corpora themselves, typically via concordancing, may become the raw material of CALL-based teaching itself."

Les corpus de textes d'apprenants peuvent servir de base à l'élaboration de logiciels d'ALAO (Granger *et al.*, 2001) : ils servent à mieux cibler le contenu didactique et les exercices sur des problèmes fréquemment rencontrés par les apprenants. D'autre part, ils servent à adapter les outils de

TAL pour traiter des erreurs spécifiques des apprenants. Les auteurs relèvent l'importance d'un codage spécifique des erreurs : bien que le processus soit coûteux et fastidieux, les analyses sont plus précieuses qu'un simple texte brut ou annoté automatiquement. *ExoGen* (Blanchard *et al.*, 2009, §B.2.21) utilise un corpus étiqueté pour générer des exercices à la volée.

Maintenant, abordons l'utilisation des concordanciers et des outils d'alignement bilingue en ALAO. En utilisant des corpus, d'après McEnery et Wilson (1997), les apprenants parcourent les contenus à leur propre rythme et découvrent les caractéristiques de la langue par eux-mêmes. Ils peuvent confronter leurs expériences à celles de leurs pairs de manière à élargir et compléter leur compréhension de certains phénomènes. Les corpus sont un outil à travers lequel les apprenants peuvent apprendre, non pas directement à travers le contenu, mais en analysant les données. Un corpus annoté permet d'obtenir de meilleurs exemples qui fournissent un apport décisif pour l'enseignement de la grammaire (Clear, 2000). En outre, dans les méthodes modernes d'apprentissage, le vocabulaire ne peut être dissocié des constructions syntaxiques et expressions idiomatiques (v. p. 66) associées aux mots (Defays et Deltour, 2003), dont l'acquisition est facilitée par l'utilisation de concordanciers pour voir des exemples d'utilisation en contexte.

Les concordanciers permettent de mettre sur pied des tâches authentiques avec la langue réellement parlée par les locuteurs natifs. Lamy *et al.* (2005) listent des activités possibles à l'aide d'un concordancier :

- devinette d'un mot-mystère, remplacé par un non-mot dans le texte ;
- tâches stylistiques : repérer les différents contextes d'utilisation d'un mot ;
- tâches syntaxiques : repérer les règles d'utilisation de certaines parties du discours comme les adverbes ;
- dérivation par induction de la construction d'un verbe (p. ex *s'agir* est un verbe défectif qui ne peut avoir d'autre sujet qu'un *il* explétif) ;
- faux amis entre deux langues ;
- informations culturelles (compter le nombre d'occurrences de mots, comme café et thé dans des textes français, et en déduire les habitudes culturelles) ;
- évolution de la langue : des mots quasi-synonymes à une époque comme *anglais* et *britannique* peuvent acquérir des sens différenciés par la suite, ou encore le mot *mail* chez Balzac ou dans un quotidien actuel ;
- auto-évaluation du style de l'apprenant, à l'aide de liste de mots à repérer et de suggestions de variantes.

Nous ajoutons également que les concordanciers permettent aux enseignants de présenter des textes authentiques et récents aux apprenants. Les étiqueteurs permettent de réduire considérablement la tâche d'annotation des corpus. Grâce à l'abondance de textes sur *Internet*, les enseignants peuvent offrir et étudier des textes très récents et renouveler et varier leurs matériaux pédagogiques.

Passons maintenant à l'alignement de textes plurilingues. Cette technique facilite notamment le repérage d'expressions idiomatiques (v. p. 66) et de constructions différentes d'une langue à une autre (Nerbonne, 2003). D'après Lamy *et al.* (2005), les concordanciers bilingues ou parallèles permettent de distinguer les emplois de divers mots ou prépositions en retrouvant l'équivalent dans la langue de l'apprenant. Ce processus peut être fait manuellement par un expert ou automatiquement (Lapalme et Macklovitch, 2006). L'alignement de textes est utilisé également pour assister le processus de révision de textes, par exemple dans un texte technique ou législatif régulièrement traduit. Les changements dans le texte original sont repérés, puis le processus d'alignement met en évidence les parties à changer dans le texte cible.

Enfin mentionnons l'utilisation des outils secondaires des concordanciers. Les lemmatiseurs associés aux concordanciers sont parfois utilisés directement par les apprenants, par exemple pour corriger leurs erreurs en fonction du contexte d'utilisation d'un mot. Quant aux enseignants, ils les utilisent pour vérifier l'emploi d'un mot, par exemple dans un type de texte particulier, comme l'écriture scientifique.

Nous passons maintenant à la description de quelques logiciels d'ALAO qui utilisent des corpus. Cobb (1999) décrit l'utilisation d'un concordancier pour l'apprentissage de vocabulaire anglais académique en contexte. Weber (2001) décrit l'utilisation d'un corpus de mémoires légaux rédigés par des étudiants anglophones de droit pour des apprenants d'anglais légal. Chambers et O'Sullivan (2004) présentent une étude sur l'utilisation d'un corpus du français dans un cours de premier cycle universitaire, avec des exercices de composition.

Les moteurs de recherche peuvent être considérés comme des concordanciers. Smrž (2004) décrit l'utilisation d'un système de recherche de document en langue naturelle pour la langue tchèque. Le système *Exills* (Segond et Parmentier, 2004, §B.2.8) affine une recherche sur Internet en utilisant un modèle de l'apprenant (2.7.4.2). Hubbard (2004) ou Chinnery (2008) utilisent le moteur de recherche *Google* comme concordancier brut.

Passons maintenant aux concordanciers bilingues avec outils d'alignement. *Glosser-RuG* (§B.2.11, Dokter et Nerbonne, 1998) permet aux apprenants d'apprendre du vocabulaire en contexte en étudiant un texte. *MARKALISTeR* (*MARKing, ALIgning and Searching TRanslation equivalents*, Paskaleva et Mihov, 1998) est un autre concordancier du projet GLOSSER. On citera encore *Sinorama* (Liou, 2004) et *TOTALrecall* (Chan et Liou, 2005) pour anglais-chinois, *NEDERLEX* (Deville et Dumortier, 2004) pour le vocabulaire juridique néerlandais pour francophones.

Nous abordons maintenant les outils d'apprentissage de la grammaire et des collocations par extraction de patrons syntaxique. *INTEX* (Silberztein, 1994) traite des corpus afin d'extraire des patrons syntaxiques en construisant un graphe grammatical qui représente un transducteur (§3.3.4.1). *Nooj* (§B.2.23, Silberztein et Tutin, 2004) est un système de traitement de corpus dans un but pédagogique, basé sur *INTEX*. On citera encore les travaux de Qiao et Sussex (1996), McEnery *et al.* (1997), Whistle (1999), Saxena et Borin (2002) et Smrž (2004). Le *Littératron* (Ganascia, 2001; Audras et Ganascia, 2004, 2005, 2006) est un outil d'analyse stylistique qui permet aux apprenants de retrouver des structures syntaxiques dans un corpus de textes pour en analyser l'emploi en contexte. Enfin, *Collocator* (Wible *et al.*, 2006), intégré au navigateur de l'apprenant, identifie les collocations dans une page *Internet* en temps réel.

Passons maintenant à la création d'exercices à partir de corpus. Wilson (1994, 1997) propose de générer automatiquement des exercices d'utilisation des mots et de sous-catégorisation basée sur des corpus annotés. Pour leur part, Coniam (1997) et Foucou et Kübler (1999, 2000) utilisent des textes étiquetés automatiquement, d'un concordancier pour repérer les collocations et d'un dictionnaire de fréquence des mots pour la génération de textes à trous. Enfin, $\mathcal{A}$lexi$\mathcal{A}$ (§B.2.1, Selva et Chanier, 2000) utilise un corpus spécialisé de 400 textes étiquetés automatiquement et indexés par mots-clés. Le logiciel dispose d'un analyseur morphologique, d'un dictionnaire de 200 mots liés aux thèmes du corpus et d'un générateur d'exercices lexicaux.

Nous finissons ce tour d'horizon en mentionnant *Native English Writing Assistant* (§B.5.16, Hu *et al.*, 1998), un système d'apprentissage de l'anglais qui dispose d'un corpus de textes et fournit une assistance à l'écriture et à la lecture avec un détecteur de faux-amis grâce à une base d'erreurs.

Pour conclure, nous pouvons souligner une fois encore l'importance des outils de traitement de corpus pour l'apprentissage des langues. Les concordanciers et les autres outils sont suffisamment fiables pour être utilisés en ALAO. Les techniques de TAL nécessaires pour annoter les corpus ou élargir

la recherche à tous les lemmes offrent des résultats satisfaisants, qui peuvent être facilement révisés par des experts. De nombreux outils de traitement de corpus existent dans le domaine de l'ALAO. Ils sont essentiellement développés et utilisés dans un domaine universitaire, car ils bénéficient avant tout à des apprenants de niveau moyen à avancé. De nombreux systèmes seront encore développés dans l'avenir. Comme pistes de développement de ce secteur de l'ALAO, nous pouvons mentionner l'amélioration des outils d'extraction des collocations et l'amélioration de la fiablité des annotations syntaxiques et sémantiques.

3.1.8 Outils automatiques de résumé et d'évaluation

Dans cette section, nous commençons par survoler les différentes techniques de résumé et d'évaluation automatique. Puis nous décrivons l'application de ces techniques à l'ALAO. Nous poursuivons par une description de logiciels d'ALAO. Enfin, nous apportons quelques remarques de conclusion.

Commençons par un survol des techniques. Pour résumer un texte, Hovy (2003) distingue trois phases :

- *Identification des sujets.* Les critères varient : position (les premières phrases ou premiers paragraphes sont importants), repère de marqueurs, fréquence de mots et syntagmes, mots du titre, etc.
- *Interprétation ou fusion des sujets.* Les sujets extraits du texte sont reformulés dans de nouveaux concepts.
- *Génération du résumé.* Une fois les sujets extraits, un planificateur doit organiser l'information qui doit servir de base au système de génération.

Desclès et Minel (2000) décrivent le résumé scolaire comme un instrument de contrôle des connaissances. Ils distinguent plusieurs méthodes de résumé automatique :

- méthodes fondées sur la *compréhension*, qui se servent d'outils d'intelligence artificielle pour "comprendre" et condenser un texte ;
- méthodes par *extraction*, qui recherchent les unités saillantes du texte, par calcul de scores (fréquence des mots, etc.), par calcul de similarité (repérage statistique des thèmes du texte et des paragraphes qui les résument le mieux), ou encore par repérage de phrases prototypiques telles que les marques discursives ;

– méthode par *filtrage sémantique*, grâce au repérage d'éléments textuels et de règles d'exploration du texte.

Décrivons maintenant quelques formalismes et techniques servant aux résumés de textes. La *Théorie de la Structure Rhétorique* (*Rhetorical Structure Theory, RST*, Mann et Thompson, 1987; Hovy, 1988, §3.4.4) est un outil de description de la structure du discours. L'*Analyse Sémantique Latente* (*Latent Semantic Analysis, LSA*, Dumais *et al.*, 1988; Zampa et Lemaire, 2002; Zampa, 2004, 2005) est une technique courante d'analyse statistique d'un texte destinée à en inférer le sens [20] pour vérifier la compréhensibilité, la cohérence et et l'exhaustivité d'un texte (Dodigovic, 2005). Elle consiste à réduire l'espace sémantique des textes à 200 mots.

Enfin, des outils statistiques évolués destinés à évaluer les performances en traduction automatiques peuvent être utilisés. Le *Dynamic Programing matching* (*DP matching*), consiste à comparer une phrase de référence jugée correcte et la phrase à évaluer en calculant les mots substitués, insérés et effacés. Quant au *BLEU-score* (*BiLingual Evaluation Understudy*, Papineni *et al.*, 2002), il consiste à comparer des segments de la phrase de référence avec ceux de la phrase à évaluer et de compter le nombre de convergences, indépendamment de la position dans la phrase.

Nous passons maintenant à la description de quelques logiciels d'ALAO utilisant des techniques de résumé et d'évaluation automatiques. Le système *e-rater* d'évaluation automatiques de rédactions (Burstein et Chodorow, 1999; Burstein et Marcu, 2000) utilisent une technique de résumé de textes basée sur le test standard GMAT [21] qui sert de test d'entrée dans les écoles d'affaires. On signalera aussi les travaux de Gurevich et Deane (2007), basé sur des résumés de textes faisant partie du TOEFL [22].

Par ailleurs, le système d'aide à l'écriture *HARRY* (§B.5.6, Holdich *et al.*, 2004) utilise le vérificateur stylistique *Check Text*, qui fonctionne à l'aide d'outils statistiques, en comptant le nombre de mots, la richesse du vocabulaire etc. REAP (Heilman *et al.*, 2007) évalue automatiquement la lisibilité des textes d'un corpus à la fois pour des apprenants de langue maternelle et de langue seconde, en se basant sur les caractéristiques lexicales (difficulté sémantique des mots) et syntaxiques (présence de certaines constructions).

20. Dans le cas de la recherche d'information, la LSA est aussi appelée *Latent Semantic Indexing* (LSI).
21. *Graduate Management Admission Test*. V. `http://www.gmat.org/`, dernier accès le 16 mai 2006.
22. *Test Of English as a Foreign Language*, examen d'entrée pour les étudiants non anglophones dans les Universités nord-américaines.

Un certain nombre de systèmes a recours à la LSA. *RAFALES* (§B.4.29, Zampa, 2004, 2005) indexe des textes destinés à l'acquisition d'une langue étrangère ; Thomas *et al.* (2004b) évaluent de courts textes dans le domaine de l'apprentissage de l'informatique, en comparaison avec l'évaluation faite par un expert humain ; on citera également Angelova *et al.* (2004), Dodigovic (2005), Cosi *et al.* (2004), Dessus et Lemaire (2002), Robertson et Wiemer-Hastings (2002), et Ishioka et Kameda (2006). Utilisant des techniques d'évaluation de traducteurs automatiques, Yasuda *et al.* (2004) proposent de combiner les résultats du *DP matching* et du *BLEU-score* pour évaluer les aptitudes en anglais d'apprenants japonais. Enfin, l'outil de résumé automatique de textes de *Microsoft* sert de base à un outil automatique de génération de questions de compréhension de textes (§B.2.20, Schwartz *et al.*, 2004).

Enfin, dans un autre registre, Candido *et al.* (2009) présentent un logiciel de simplification des structures syntaxiques du portugais pour des apprenants qui ont des difficultés de lecture. Sans faire de résumé, à partir de l'analyse syntaxique, le système produit plusieurs paraphrases pour éliminer les structures complexes des phrases comme les appositions et relatives.

Terminons par quelques remarques conclusives. Comme la traduction automatique, les méthodes présentées ci-dessus ont encore des résultats mitigés. Comme le soulignent Desclès et Minel (2000), l'activité de résumé est très subjective et sujette à interprétation. La même remarque vaut bien entendu pour l'évaluation de texte d'apprenants, car celle-ci ne sera pas la même d'un expert humain à un autre. Les logiciels d'ALAO basés sur les résumés et l'évaluation automatique sont plutôt destinés à des apprenants avancés. Même à ce niveau, les erreurs grammaticales sont relativement fréquentes et faussent les performances des systèmes. Toutefois, ce domaine présente un grand intérêt pour les apprenants qui travaillent en autonomie, ou par exemple pour pré-évaluer un travail avant de le rendre à un enseignant.

3.1.9 Dialogue homme-machine

Dans cette courte section, nous commençons par décrire la problématique du dialogue homme-machine. Nous donnons ensuite quelques exemples d'application au domaine de l'ALAO. Nous terminons par quelques remarques de conclusion.

Commençons par un survol de la problématique du dialogue homme-machine : il consiste à construire un dialogue coopératif en langue écrite ou

orale entre l'utilisateur et l'ordinateur (Pierrel et Romary, 2000). Ces auteurs soulignent deux aspects incontournables dans les applications de dialogue : un traitement fin de la référence pour interpréter les énoncés et la détection des présupposés et intentions de l'utilisateur pour planifier le dialogue. Le système de dialogue doit ensuite construire une réponse grammaticale à l'aide d'un générateur (§3.1.5). Les dialogues homme-machine produisent souvent des malentendus et des incohérences auxquelles il faut remédier. Il faut aussi construire une représentation des données du problème.

Les systèmes de dialogue doivent être tolérants face aux inattendus. De plus, la langue orale est plus relâchée que la langue écrite. Ces systèmes doivent accepter les anaphores, les ellipses, les hésitations et les différents mécanismes de référenciation d'objets. Ils doivent s'adapter aux utilisateurs et appliquer une stratégie adéquate pour compléter au fur et à mesure les informations nécessaires manquantes.

Dans le domaine de l'ALAO, les techniques de dialogue homme-machine sont avant tout utiles pour le pilotage d'un micromonde (§2.7.3), notamment pour donner une rétroaction à l'apprenant. On pourra citer par exemple le micromonde *SHRDLU* (§B.3.8, Winograd, 1972), le jeu d'enquête policière *Herr Komissar* (§B.3.5, DeSmedt, 1995) ou le tuteur intelligent de physique *Why2-Atlas* (VanLehn *et al.*, 2002; Rosé *et al.*, 2002, §B.2.41) qui utilise des dialogues en langue naturelle. Forbes-Riley et Litman (2006) et Litman et Forbes-Riley (2006) décrivent le logiciel *ITSPOKE*, qui est la version en langage parlé de *Why2-Atlas* : le tuteur dispose de stratégies diverses pour guider le dialogue, comme la reformulation de concepts déjà introduits, la récapitulation, des indices (éléments de la réponse), etc.

Il est possible de programmer un système de dialogue très simple à l'aide d'expressions régulières (§3.1.1.1). Le système reconnaît des mots-clés et répond à son interlocuteur en fonction des besoins. Ces systèmes sont souvent appelés *robots de conversation* (ou *chatterbots*) et sont couramment trouvés dans les *chats* et d'autres outils (§2.7.5). Le système *Eliza* (Weizenbaum, 1966) est généralement considéré comme le premier robot de conversation. *Exills* (*Edutainment for Internet Language Learning Solution*, §B.2.8, Brun *et al.*, 2002; Segond et Parmentier, 2004) [23] est une plateforme d'apprentissage qui utilise des robots de conversation dans un *chat*.

Pour conclure, nous pouvons constater que le dialogue homme-machine est une technique de TAL bien répandue dans le domaine de l'ALAO. La principale difficulté est naturellement d'analyser les productions des apprenants, susceptibles de contenir des erreurs. Comme pour la plupart des tech-

23. `http://www.exills.com/`, dernier accès le 23 juillet 2006.

niques exposées dans les sections précédentes, nous pouvons souligner que la majeure partie des logiciels appliquant les techniques de dialogue homme-machine sont des prototypes de recherche. Ce fait est encore une fois certainement dû aux coûts élevés de développement de tels systèmes.

3.2 Traitement des erreurs orthographiques ou typographiques

Cette section n'aborde que l'orthographe des mots eux-mêmes, alors que l'orthographe grammaticale sera traitée à la section 3.3. Nous commençons par aborder la problématique de l'orthographe (§3.2.1). Nous poursuivons avec l'exposition de quelques typologies d'erreurs (§3.2.2). Puis nous décrivons les différentes techniques de traitement des erreurs orthographiques (§3.2.3). La section 3.2.4 traite des logiciels de correction orthographique. Enfin, nous concluons avec une brève discussion (§3.2.5).

3.2.1 Problématique

Commençons par une rapide discussion de la problématique de l'orthographe. Dans la littérature, on distingue généralement l'*orthographe d'usage*, qui concerne l'écriture d'un mot en lui-même, et l'*orthographe grammaticale*, qui concerne notamment les règles d'accord. L'orthographe du français est très complexe, vu que certaines lettres ne se prononcent pas (Catach, 1978). Ses règles sont parfois illogiques et présentent de nombreuses anomalies et irrégularités (Jaffré, 1992). Nous utilisons un alphabet qui a été conçu pour la langue latine et qui doit transcrire des sons qui n'étaient pas utilisés par le latin (Blanche-Benveniste et Chervel, 1978). Pensons au mot *femme* qui se prononce [fam] et non [fɛm] comme on devrait le prononcer en suivant les règles ordinaires de prononciation des lettres composant ce mot. L'anglais n'est pas en reste avec une prononciation qui diffère aussi fortement de l'écriture (Catach, 1989). L'orthographe du français doit donc être acquise mot par mot, vu le grand nombre d'exceptions aux règles (Blanche-Benveniste et Chervel, 1978). Si tout le monde commet des fautes, il n'en demeure pas moins que cet apprentissage est crucial pour maîtriser la langue, sous peine d'être fortement pénalisé : l'orthographe a acquis une importance majeure, plus grande que la syntaxe (Catach *et al.*, 1980; Jaffré, 1992).

Soulignons également que la frontière entre orthographe et grammaire est parfois floue. Ainsi, *haie*, *es*, *est* et *hais* se prononcent de la même manière

mais représentent des verbes et des substantifs. Comme nous le verrons à la section 3.3.3.2, ces erreurs sont détectées au niveau phonologique par les détecteurs d'erreurs grammaticales. Pourtant, certaines typologies classeront plutôt ces erreurs au niveau orthographique. Autre erreur parfois considérée comme orthographique, les substantifs *prémisses* et *prémices* sont facilement confondus. Lorsqu'un terme est confondu avec l'autre, la syntaxe ne suffit pas et des règles sémantiques doivent être appliquées ; dans la plupart des correcteurs grammaticaux, l'ambiguïté est signalée et l'utilisateur est invité à vérifier lui-même si le mot choisi est adéquat pour le contexte et à s'autocorriger manuellement.

Dans l'histoire de l'informatique, les techniques de correction orthographique sont apparues relativement tôt. Dans un premier temps, les systèmes se contentaient de signaler les mots absents du lexique ou une erreur potentielle grâce à des algorithmes qui calculaient la probabilité de l'appartenance d'une chaîne à la langue ; l'utilisateur devait ensuite corriger l'erreur à la main. Le premier correcteur orthographique commercial, sur système VAX, date de 1971 (Peterson, 1980). Développés grâce à l'essor de la microinformatique, les vérificateurs orthographiques sont désormais intégrés à tous les logiciels de traitement de textes et à bon nombre d'éditeurs de textes, clients pour courriel, navigateurs pour *Internet*, dictionnaires électroniques, moteurs de recherche etc. Les algorithmes sont généralement développés pour des fautes de locuteurs natifs et ne sont pas adaptés aux erreurs des apprenants.

Desmarais et Bisaillon (1998) décrivent l'utilisation d'un correcteur orthographique de traitement de textes pour l'amélioration de l'expression écrite. Elle souligne que la correction en cours de frappe est plus profitable pour l'apprenant que la remise d'une copie annotée par l'enseignant. Le commentaire des fautes en classe ainsi que l'élaboration en commun de règles d'usage est également utile. Le repérage automatique des fautes permet aux apprenants de repérer beaucoup plus de fautes que lors d'une relecture ; ensuite, les apprenants sélectionnent la graphie correcte. Nous détaillerons les techniques de corrections à la section 3.2.3 et quelques logiciels existants à la section 3.2.4.

3.2.2 Typologies des erreurs d'orthographe

Passons maintenant à une revue de différentes typologies des erreurs d'orthographe. Commençons par quelques typologies à dominante technique. Selon Damerau (1964), les fautes d'orthographe peuvent être décrites par la

typologie suivante :

- insertion de lettres (une lettre en trop) ;
- omission de lettres (une lettre manque) ;
- transposition de lettres (lettre ou groupes de lettres formant le même son, inversion de lettres) ;
- substitution de lettres (une lettre pour une autre, généralement la touche voisine sur le clavier).

Faulk (1964, cité chez Angell *et al.*, 1983 et Berghel, 1987) distingue trois types de mesures de similarité entre chaînes de caractères :

- similarité *matérielle*, où l'on compte le nombre de caractères communs à deux chaînes ;
- similarité *ordinale*, où l'on mesure à quel point les caractères sont dans le même ordre ;
- similarité de *position*, où l'on mesure à quel point les caractères sont dans une même position dans la chaîne.

Yannakoudakis et Fawthrop (1983b) constatent que les erreurs orthographique en anglais portent principalement sur les points suivants :

- utilisation des voyelles ;
- utilisation des lettres W, Y et H ;
- lettres doubles ou simple ;
- peu d'erreurs sont commises sur la première lettre ;
- transposition entre deux caractères voisins sur le clavier ;
- certaines consonnes sont plus fréquemment échangées.

Dans son classement des erreurs typographiques, Druard (1993) regroupe les erreurs typographiques citées plus haut (substitution, omission, inversion, insertion) en trois types :

i. coquilles : fautes qui sont détectées parce que le mot n'existe pas ou n'entre pas dans le contexte (ex : **journal* pour *journal*) ;
ii. doublons : mot écrit deux fois pour cause de frappe trop rapide (*je sais que *que tu viendras*) ;

iii. omissions : pour cause de frappe ou écriture trop rapide, on oublie un mot (*la sœur *Jean*).

La troisième catégorie n'est pas détectable sans analyse syntaxique. A côté de ces erreurs typographiques, Druard (1993) distingue les véritables fautes d'orthographe, où le mot est écrit de manière incorrecte mais qui ressemble phonétiquement au mot correct (*creindre* pour *craindre*).

D'autres typologies sont orientées vers la didactique et concernent les causes des erreurs. Ainsi, Pain (1981), pour son correcteur d'orthographe destiné aux enfants, propose une classification des erreurs centrée sur l'enfant :

- erreurs de perception visuelle ;
- apprentissage incorrect des correspondances phonème-graphème ;
- rationalisation incorrecte de règles particulières.

Pour sa part, Catach *et al.* (1986) proposent la typologie suivante :

- erreurs à dominante *phonétique*, liées à une mauvaise perception du son (**macasin* au lieu de *magasin*) ;
- erreurs à dominante *phonogrammique*, produites lorsque l'on connaît le son sans le retranscrire correctement (**farmacie* au lieu de *pharmacie*) ;
- erreurs à dominante *morphogrammique* produites quand l'orthographe des éléments (phonétiques ou non) au niveau des morphèmes grammaticaux ou lexicaux n'est pas respectée (**rapident* comme marque du pluriel de *rapide*) ;
- erreurs à dominante *logogrammique* produites sur des homophones, qui peuvent produire des erreurs au niveau du discours, du lexique ou de la grammaire (*chant* vs. *champ*) ;
- erreurs portant sur les *lettres non fonctionnelles* étymologiques ou historiques, comme des doubles consonnes ou le *s* de *toujours* ;
- erreurs concernant les *idéogrammes*, comme l'omission, l'adjonction ou la confusion d'une majuscule, d'une apostrophe, d'un trait d'union ou d'un signe de ponctuation.

James (1998) classe parmi les erreurs d'orthographe les erreurs de ponctuation, y compris l'oubli ou l'insertion d'un symbole de séparation (espace ou autre). On trouve aussi les erreurs typographiques, dont les erreurs spatiales (proximité sur le clavier), temporaires (mauvais ordre des lettres) et

d'anticipation d'un caractère. Les erreurs de dyslexie entrent également dans la catégorie des erreurs orthographiques, où la mauvaise lettre est utilisée pour produire un son. La confusion de mots existants ayant un son identique ou quasi-identique appartient également à cette catégorie (*prémices / prémisses*, *hospices / auspices*). Enfin entrent les erreurs phonétiques, dues à l'application erronée d'une graphie de la L1 ou d'une mauvaise application de règles de la L2.

De son côté, Pedler (2001) traite du problème particulier de la dyslexie, qui est mal prise en compte par les correcteurs commerciaux. Les erreurs typiques sont la confusion de phonèmes et l'omission ou l'insertion de syllabes entières. Bouraoui *et al.* (2009) proposent une typologie pour les textes produits par des handicapés.

Enfin, Rimrott et Heift (2005, 2008) constatent que les erreurs commises par des apprenants d'une langue ne sont pas les mêmes que celles des locuteurs natifs qui maîtrisent leur langue. Elles distinguent les erreurs de compétence et de performance. Dans la première catégorie, qui couvre 80% des erreurs, surviennent les erreurs d'influence de la langue première et les différences phonologiques et morphologiques. Les erreurs de performance sont des erreurs aléatoires et non systématiques telles que l'insertion ou l'omission de lettres. Elles constatent qu'une liste de propositions de correction doit être la plus courte possible et que le mot correct doit apparaître dans un ordre aussi haut que possible.

3.2.3 Techniques

Dans cette section, nous abordons les différentes techniques de la correction orthographique. Nous commençons par une description des principes de la correction et de la détection d'erreurs orthographiques (§3.2.3.1). Puis nous traitons des algorithmes suivants : recherche de mots par codage de chaîne (§3.2.3.2) ; recherche par bi- et trigrammes (§3.2.3.3) ; recherche de mots par similarité (§3.2.3.4) ; techniques de recherche et correction phonétique (§3.2.3.5) ; correction par règles (§3.2.3.6) ; techniques de séparation de mots (§3.2.3.7) ; listes de confusion (§3.2.3.8) ; correction par méthode stochastique et par réseaux de neurones (§3.2.3.9). Enfin, nous traitons des mesures de distance entre chaînes qui servent à éliminer les propositions les moins vraisemblables (§3.2.3.10).

3.2.3.1 Correction et détection d'erreurs orthographiques

S'il paraît évident aujourd'hui de disposer de propositions de correction, cette caractéristique n'a pas toujours été présente, pour questions de taille du lexique, de temps d'accès aux disques et de rapidité des processeurs. Ainsi, certains outils ne calculaient que la probabilité qu'une chaîne appartienne à la langue ; l'utilisateur, par ses connaissances, pouvait alors corriger ses fautes de frappe ou vérifier la bonne orthographe dans un dictionnaire.

Définissons maintenant le processus de correction orthographique : il s'agit de la recherche d'un mot dans un lexique, et, si ce mot n'y figure pas, de la recherche des mots les plus susceptibles de représenter l'orthographe correcte de ce mot (Ndiaye et Vandeventer Faltin, 2004). On trouvera un état de l'art de la correction orthographique chez Kukich (1992) et chez Mitton (1996).

La correction orthographique se passe généralement en deux temps : pour chaque mot de la phrase, le système tente de retrouver le mot dans une liste. Cette phase n'est pas triviale (Peterson, 1980; Berghel, 1987), car il faut identifier un mot potentiel en utilisant des délimiteurs (espaces, virgules, points, points-virgules, deux-points, §3.1.1.2). Il faut être particulièrement attentifs aux nombres, aux apostrophes et aux tirets. Les majuscules présentent aussi une difficulté, car elles sont parfois obligatoires, parfois interdites et parfois facultatives.

Si le mot n'est pas dans sa liste ou son lexique, un correcteur orthographique essaie de proposer des solutions alternatives grâce à différents algorithmes. La liste des mots utilisée par un correcteur contient généralement tous les mots de la langue (du moins le vocabulaire le plus courant) à toutes les formes (pluriel, formes déclinées, formes conjuguées, etc.). Heidorn *et al.* (1982) soulignent la nécessité de déterminer quelles sont les meilleures corrections grâce au contexte syntaxique. Pour leur part, Pollock et Zamora (1984) proposent de ne corriger que les mots ne contenant qu'une seule erreur, qui forment 90 à 95% des erreurs. Un problème particulier concerne les erreurs qui aboutissent à des mots qui existent dans le lexique. D'après Mitton (1987), cela concerne jusqu'à 16% des erreurs d'orthographe. Certains correcteurs traitent de ce problème, soit par analyse sémantique, soit par listes de confusion (§3.2.3.8). Les erreurs de séparation des mots demandent également un traitement particulier (§3.2.3.7). Enfin, en français, mentionnons le problème de l'apostrophe avec l'élision obligatoire du *e* devant un mot commençant par une voyelle pour les pronoms *je, te, le*, etc.

3.2.3.2 Recherche de mots par codage de chaîne

Pour retrouver les formes les plus proches du mot inconnu, la plupart des vérificateurs orthographiques utilisent une représentation codée d'un mot (Vandeventer et Ndiaye, 2002; Ndiaye et Vandeventer Faltin, 2003). Commençons par la technique des alpha-codes. Le terme d'alpha-code peut recouper différentes méthodes de calcul. Ndiaye et Vandeventer Faltin (2003) proposent de normaliser la chaîne (les accents sont enlevés et les caractères sont mis en minuscules) et ensuite d'écrire les consonnes du mot en ordre alphabétique, suivies des voyelles, également en ordre alphabétique ; chaque lettre n'est gardée qu'une fois. Un mot ne possède qu'un seul alpha-code, mais plusieurs mots peuvent partager le même alpha-code. Ainsi, un vérificateur orthographique pourra rechercher les mots qui contiennent le même alpha-code que le mot inconnu. Voici un exemple, tiré de Ndiaye et Vandeventer Faltin (2003) :

(6) a. *prèférré.
b. fpre.
c. préférer, préféré, préfère, préférée.

Le mot erroné en (6a) donne l'alpha-code en (6b). La recherche dans le lexique donne les propositions en (6c). Cette solution ne couvre cependant qu'une partie des erreurs possibles : diacritiques, inversions, duplications. La même technique permettra aussi de détecter l'omission ou l'insertion d'une lettre qui est présente dans une autre partie du mot, comme dans les exemples suivants qui obtiendront aussi l'alpha-code en (6b) :

(7) a. *péféré.
b. *prféféré.

Il faut également couvrir l'insertion d'une lettre qui n'appartient pas au mot et l'omission d'une lettre qui n'apparaît pas ailleurs dans le mot. Il est possible de créer de nouveaux codes, en insérant ou enlevant chaque lettre possible. Cela revient à rechercher les correspondances de vingt-six nouveaux alpha-codes. A nouveau, pour l'alpha-code de (6b), Ndiaye et Vandeventer Faltin (2003) donnent :

(8) a. pre, fpe. fre, fpr.
b. bfpre, cfpre, dfpre [...] frpae, fprei [...].

En enlevant des lettres, on obtient les quatre alpha-codes en (8a). En ajoutant une lettre, on obtient vingt-deux possibilités, dont les quelques exemples en (8b). Le nombre de propositions retrouvées par cette technique est alors élevé et l'on trouve beaucoup de propositions inadéquates (Pollock et Zamora, 1984). Nous verrons des techniques de calcul de distance entre chaînes de caractères à la section 3.2.3.10, qui servent à éliminer les chaînes trop éloignées.

Revuz (1991) propose une autre version d'alpha-code composée des lettres dans l'ordre alphabétique suivi du nombre d'occurrences de la lettres. Ainsi, pour *aimablement*, on obtiendra l'alpha-code *a2be2ilm2nt*. Cette fonction est plus proche d'une table de hachage, que nous décrivons un peu plus loin, et sert à la recherche dans un dictionnaire. Véronis (2004) propose de créer une clé en remettant les lettres dans l'ordre alphabétique en éliminant les doublons. Seules les clés de longueur n-1 et n+1 du mot inconnu sont prises en considération.

Pollock et Zamora (1984) proposent le calcul d'une *clé-squelette*, très proche de l'alpha-code, basée sur le principe suivant : on préserve la première lettre du mot, puis on liste toutes les consonnes uniques dans le mot en préservant l'ordre d'apparition, suivies des voyelles. Ainsi, *préféré* aurait pour code *prfe*[24]. Selon eux, il est en effet raisonnable de considérer que la première lettre est presque toujours correcte, dans plus de 90% des cas, ce qui est confirmé par Mitton (1987). L'ordre des consonnes est également préservé. Comme cette clé de recherche donne de mauvais résultats face aux erreurs portant sur les consonnes de début de mot, l'auteur propose une autre clé, la *clé d'omission*. Cette clé est basée sur la fréquence d'omission des lettres. Les statistiques démontrent que les consonnes sont omises dans l'ordre de fréquence décroissant suivant : *r s t n l c h d p g m f b y w v z x q k j*[25]. La clé d'omission classe les consonnes uniques dans l'ordre croissant de fréquence d'omission. Ces clés permettent de repérer des anagrammes. Zock (2002) présente la technique de l'*anacode*, qui est également proche de l'alpha-code. Les lettres sont simplement listées une fois dans l'ordre alphabétique. Pour le mot *préféré*, l'anacode sera *efpr*. Damerau (1964) propose d'utiliser un tableau booléen de 28 positions, une pour chacune des 26 lettres, une pour les nombres et une pour les autres symboles, qui encode chaque mot du lexique. Il ne corrige que les mots qui ne diffèrent que d'un caractère. Son système n'est pas interactif, car les ordinateurs existant à l'époque de la rédaction de l'article ne disposaient pas d'écrans ni de claviers.

24. Pollock ne considère que l'anglais et ne parle pas des caractères accentués, mais il nous a paru raisonnable de traiter les voyelles sans accents.
25. Cette fréquence pourrait être légèrement différente en français.

Terminons par la technique des tables de hachage (McIlroy, 1982; Kukich, 1992; Reynaert, 2004). Elles sont basées sur une fonction, appelée fonction de hachage (*hash function*), qui calcule une valeur numérique entière pour une chaîne. Un code est calculé pour chaque mot selon la méthode suivante : chaque lettre de l'alphabet reçoit une valeur numérique (par exemple sa valeur dans un code de caractère élevé à une puissance quelconque) et le hachage consiste à faire la somme de tous ces codes. Ainsi, tous les anagrammes d'une chaîne auront la même valeur de hachage. Cette valeur correspond à une entrée dans la table de hachage, qui contient une chaîne de caractère. Si plusieurs chaînes ont la même valeur de hachage, l'entrée du tableau contiendra également l'indice d'une autre case du tableau, qui contiendra une autre chaîne, et ainsi de suite à la manière d'une liste chaînée. Plus une fonction de hachage est efficace, moins il y a de collisions entre chaînes. La table de hachage contiendra également un grand nombre de cases vides. Cette technique est couramment utilisée car les valeurs retournées par la fonction sont indépendantes de la position des caractères dans la chaîne. Elle permet de retrouver immédiatement les inversions de lettres. L'insertion ou l'effacement de lettres se fait grâce à l'addition ou la soustraction de la valeur de hachage des lettres de l'alphabet. Quant à la substitution, elle soustrait successivement toutes les valeurs de lettres possibles et ajoute les valeurs des autres lettres. Le nombre de différences autorisées dépend de la longueur de la chaîne.

3.2.3.3 Recherche par n-grammes

Les n-grammes sont des groupes de n lettres constituant une sous-chaîne de mot. Les plus courants sont les bigrammes, constitués de deux lettres, et les trigrammes de trois lettres. Ainsi, le mot *trigramme* est constitué des trigrammes *tri*, *rig*, *igr*, *gra*, *ram*, *amm* et *mme*.

Morris et Cherry (1975) normalisent le texte en enlevant les apostrophes et les majuscules du texte et en extraient d'abord les statistiques de di- et trigrammes. Les mots sont ensuite classés dans un fichier et recherchés dans un lexique de 2726 mots techniques. Ensuite, le système calcule un score basé sur les fréquences de bi- et trigrammes pour les mots absents du lexique. Ainsi, on obtient un indice de particularité de la chaîne et l'utilisateur peut déterminer si le mot est correctement orthographié ou non.

Riseman et Hanson (1974) corrigent les mots erronés en se basant sur des matrices de n-grammes binaires. Chaque matrice D_{ij} dénote si une paire de lettre est attestée aux positions i et j dans un mot. Si une paire est attestée, sa probabilité est de 1, sinon, elle est de 0. Ainsi, pour caractériser tous

les mots de 6 lettres, 15 matrices $\left(\frac{6\times(6-1)}{2}\right)$ sont nécessaires. Une erreur est détectée pour un caractère si au moins deux matrices n'attestent pas la paire. Le système propose alors de remplacer le caractère erroné par le ou les caractères attestés par tous les digrammes. La méthode ne détecte pas forcément les erreurs pour des mots courts. En outre, elle ne permet pas de traiter l'omission ou l'insertion.

Peterson (1980) utilise un tableau de fréquences de trigrammes et bigrammes pour détecter une erreur potentielle. Il propose également d'enlever les suffixes et affixes courants comme *-ness* et de ne garder que la racine dans un dictionnaire. Cependant, de faux mots peuvent être générés et des erreurs morphologiques acceptées (comme **implyed*, construit sur *imply* + *-ed*, à la place d'*implied*, alors que *wanted* est correct).

Pour de Heer (1982, p. 235), les trigrammes sont les plus petites unités qui, combinées entre elles, sont significative de la signification de la langue. Il montre qu'il existe 36^3 trigrammes différents pour l'anglais, soit 46 656. Pour le français, outre les 26 lettres de l'alphabet, on ajoute *à, â, ç, é, è, ê, ë, ï, î, ô, ù, û* et *ü*, soit 39 symboles. A cela s'ajoutent les digraphes æ et œ, dont nous ne tiendrons pas compte puisqu'il s'agit de conventions d'imprimerie. 39^3 donne donc 59 319 trigrammes différents. De Heer (1982) estime que le nombre de trigrammes effectivement utilisés est de moins de 15%. Il estime donc le nombre de trigrammes différents dans un grand corpus de documents à 10 000.

Angell *et al.* (1983) présentent une méthode basée sur des trigrammes communs entre le mot inconnu et les mots du dictionnaire. Les mots candidats sont retrouvés grâce à un dictionnaire de trigrammes qui liste tous les mots qui contiennent le même trigramme. Un mot de longueur n sera représenté par un vecteur de longueur *n+2*, afin de noter aussi les espaces. Le mot *union*, de longueur cinq, sera représenté par un vecteur d à 7 dimensions contenant les trigrammes *##u, #un, uni, nio, ion, on#* et *n##*, où le signe # représente les espaces. d_i représente le trigramme commençant en position i dans la chaîne. Un vecteur similaire m, de longueur n', sera construit pour le mot inconnu. On définit c, le nombre de trigrammes communs aux deux chaînes, chaque trigramme ne pouvant être compté qu'une fois. c est donc le nombre de trigrammes pour lesquels $d_i = m_i$, où ni d_i, ni m_i ne sont déjà apparus lors d'une précédente égalité. Ceci permet de calculer l'indice de similarité, qui est connu sous le nom de *coefficient de Dice* :

$$\frac{2c}{n+n'} \tag{3.1}$$

Si un mot est entièrement inclus dans un autre, comme *sidère* et *considère*, le coefficient de Dice retourne une valeur trop élevée. Si la différence de longueur entre les deux chaînes est supérieure à 1, l'indice de similarité devient alors :

$$\frac{c}{\max(n,n')} \tag{3.2}$$

Enfin, Vosse (1992, §C.36) utilise également une technique de trigrammes et de triphones pour la partie orthographique de son correcteur grammatical du néerlandais. Les propositions sont ensuite ordonnées selon un système de score.

Passons maintenant à des méthodes statistiques de réaccentuation grâce à des trigrammes. Simard et Deslauriers (2001) présentent une méthode de réaccentuation de textes français. La phrase en (9a) peut être réaccentuée comme (9b) ou comme (9c).

(9) a. Ce chantier ferme a cause des emeutes.
b. Ce chantier fermé a causé des émeutes.
c. Ce chantier ferme à cause des émeutes.

Environ 85% des mots français ne contiennent pas d'accent. Plus de la moitié des éléments restants peut être réaccentuée de manière déterministe. Ainsi, un peu plus de 5% des mots correspondent à plus d'une forme valide. La méthode de Simard et Deslauriers (2001) se base sur une désambiguïsation par *Modèles de Markov Cachés* (*Hidden Markov Models, HMM*, v. 3.1.2 p. 49). Une première phase assigne des étiquettes aux différents mots. La seconde phase procède à la désambiguïsation.

Quant à Yarowsky (1994), pour résoudre le problème de l'accentuation des mots en espagnol et en français, il prône l'utilisation de méthodes générales de désambiguïsation sémantique grâce au contexte environnant. Le problème ne peut être résolu grâce à des techniques simples d'étiquetage lexical par trigrammes. Il est nécessaire de procéder à une analyse syntaxique voire sémantique. L'auteur calcule des listes de décisions basées sur le contexte des mots à ±n mots. Bien qu'il n'aborde la restauration de l'accentuation que dans un contexte de lettres capitales non accentuées, l'algorithme pourrait être utilisé dans un cadre d'apprentissage des langues.

3.2.3.4 Recherche de mots par similarité

Pour Pollock et Zamora (1984), la recherche par similarité consiste à vérifier une liste de mots similaires en tentant de recréer le mot erroné. Afin de gagner du temps de recherche et de l'espace disque, McIlroy (1982) propose d'éliminer les préfixes et suffixes pour trouver la racine d'un mot. Quelques règles particulières doivent être ajoutées pour traiter des phénomènes comme *cut* → *cutting*.

Berghel (1987) propose d'utiliser des axiomes de similarité, inspirés par Faulk (1964, cité chez Angell *et al.*, 1983 et Berghel, 1987), qui mesure la similarité matérielle, la similarité ordinale et la similarité de position (§3.2.2). En outre, Berghel (1987) prend en compte la double omission, la double insertion et la transposition autour d'un troisième caractère. La distance entre propositions est traitée par une mesure de Levenshtein (1966, §3.2.3.10).

Pour un correcteur de l'allemand destiné à des apprenants, Rimrott (2003) considère que les erreurs d'orthographe ne proviennent pas d'une incapacité à appliquer les règles orthographiques de la langue mais par une application erronée de règles morphologiques. En tant que mots inconnus, ils sont traités par un correcteur orthographique mais devraient plutôt être traités par un correcteur grammatical. Rimrott (2003) propose donc l'incorporation d'erreurs morphologiques fréquentes dans un correcteur orthographique. Elle utilise les expressions régulières pour enlever les préfixes et suffixes potentiels pour extraire les radicaux potentiels. Puis une recherche dans le lexique extrait les radicaux infinitifs. Ensuite, ces formes infinitives sont retournées au lexique pour proposer les différentes formes les plus proches de la forme erronée. Signalons finalement le système de classement des caractères japonais *kanji* par similarité graphique (Yencken et Baldwin, 2008).

3.2.3.5 Recherche et correction phonétique

Il existe de nombreuses techniques de recherche phonétique. *SOUNDEX* (Odell et Russell, 1918, 1922) est une technique très ancienne de calcul de proximité phonétique. La clé de recherche est constituée de la première lettre du mot, suivie de la valeur numérique des caractères d'après la liste suivante :

- A, E, I, O, U, H, W, Y → 0
- B, F, P, V → 1
- C, G, J, K, Q, S, X, Z → 2

- D, T → 3
- L → 4
- M, N → 5
- R → 6

Ainsi, la chaîne *Bush* aura pour valeur B020. Les zéros sont ensuite éliminés et les caractères répétés sont éliminés : *Bush* aura alors la valeur B2 et *Busch* également. Mais *Quayle* et *Kwail*, qui se prononcent de la même manière en anglais, ont respectivement la valeurs Q4 et K4. Cette technique est donc très sommaire. En outre, la clé n'est adaptée qu'à la prononciation de l'anglais. La technique de *métaphone* ou *double métaphone* (Philips, 2000) est une amélioration de la technique de *SOUNDEX*, qui tient davantage en compte les prononciations différentes des lettres (*c* prononcé [s] ou [k], selon les cas). Des versions existent pour d'autres langues comme le français.

Véronis (1988b, §C.35) propose une technique de correction phonétique du français. Il se base sur une table de similarité entre différentes sous-chaînes qui peuvent être substituées les unes aux autres. Si aucune sous-chaîne valide n'est retrouvée, l'algorithme teste s'il y a eu substitution, inversion, insertion ou omission d'un caractère. Les chaînes construites par l'algorithme sont ensuite recherchées dans un lexique de formes de base. En outre, le système vérifie les erreurs d'accord.

Tanaka et Kojima (1987) proposent une méthode de correction basée sur un fichier hiérarchique. Les phonèmes sont groupés dans quatre classes, qui sont à leur tour regroupées dans deux autres classifications, de la plus grossière à la plus fine. Un mot pourra donc être converti en trois différentes chaînes représentant la séquence des symboles des classes phonémiques le composant. Le fichier hiérarchique contient les chaînes phonétiques des mots classifiés de manière arborescente selon leurs classes, sur quatre niveaux de profondeur, de la plus grossière à la plus fine. Une chaîne erronée est convertie en chaînes de séquence de symboles phonétiques. La recherche dans le dictionnaire est ainsi facilitée.

Van Berkelt et De Smedt (1988) présentent une méthode de correction orthographique basée sur une analyse de triphones. Tout d'abord, les mots inconnus sont transformés en une ou plusieurs séquences phonétiques grâce à des règles. Puis les phonèmes sont regroupés en triphones. Un lexique inversé permet ensuite de retrouver tous les mots contenant un même triphone. Les mots retrouvés sont classés par ordre inverse de fréquence des triphones. Les auteurs présentent deux avantages de cette méthode : standardisation soit en anglais américain, soit britannique ; correction des erreurs orthographiques, qui concernent la correspondance entre orthographe et prononcia-

tion, par rapport aux erreurs typographiques (insertion, effacement) qui ne débouchent en général pas sur des homophones. Une technique basée sur les triphones et trigrammes est aussi utilisée par Vosse (1992, §C.36).

Courtin *et al.* (1991) utilisent un transducteur (§3.3.4.1) pour phonétiser une chaîne. Plusieurs propositions peuvent être retrouvées. Puis ces chaînes phonétiques sont à nouveau transcrites en chaînes orthographiques ; seules les propositions retrouvées dans le lexique sont gardées. De même, Segond et Parmentier (2004) décrivent un correcteur orthographique qui permet de prendre en compte les erreurs phonétiques. Par contre, Ndiaye et Vandeventer Faltin (2003, 2004) utilisent un système expert de 700 règles (Goldman, 2001) pour phonétiser la chaîne et recherchent ensuite les mots aux prononciations identiques ou similaires dans un lexique phonétique.

Enfin, Kempen (1992) propose une technique basée sur une analyse de triphones. Cette technique peut aussi bien traiter des erreurs de type visuel (*look-alike*), telles que les inversions, insertions et omissions de lettres, que de type phonétiques (*sound-alike*). Les mots sont d'abord convertis en phonèmes, puis un algorithme segmente la chaîne phonétique en triphones. Les mots contenant chaque triphone sont alors extraits du lexique. Enfin, un indice de similarité calcule le nombre de triphones communs entre la chaîne erronée et la proposition. En outre, l'algorithme de classement tient en compte la taille des chaînes et l'ordre des triphones communs.

3.2.3.6 Correction par règles

Yannakoudakis et Fawthrop (1983a,b) proposent deux correcteurs orthographiques par règles. L'un se base sur la recherche de mots proches dans un dictionnaire, en suivant certaines règles d'erreurs ; le second recherche dans le dictionnaire les mots qui diffèrent d'un ou deux caractères de la chaîne inconnue et vérifie si une règle d'erreur peut être appliquée.

Pijls *et al.* (1987) et Kempen (1992) proposent d'examiner une série de règles morphologiques et orthographiques afin de poser un diagnostic de l'erreur commise par l'apprenant. Si toutes les décisions de l'apprenant sont jugées correctes, le système tente de déterminer si une règle a été appliquée de manière incorrecte. Bos (1994) ajoute des règles d'erreurs pour traiter les principales erreurs commises par les apprenants (surgénéralisation de règles, application incorrecte d'une règle, application d'une règle inexistante etc.).

Agirre *et al.* (1992) et Aduriz *et al.* (1991) utilise un analyseur morphologique à deux niveaux pour la langue basque, en s'inspirant des travaux

de Koskenniemi (1994). Au premier niveau, un lexique gère les morphèmes (lemmes et affixes) et les règles morphotactiques de combinaison de ces éléments entre eux. Au second niveau, des règles traitent du passage du niveau lexical au niveau de surface. Oflazer et Güzey (1994) et Oflazer (2003, 1996) utilisent une technique similaire pour le turc.

Aldezabal *et al.* (1999) utilisent des transducteurs (§3.3.4.1) qui recherchent tous les lemmes possibles d'une forme. L'utilisation de dictionnaires d'utilisateurs permet d'étendre la couverture du correcteur. De même, Courtin *et al.* (1991) utilisent un transducteur pour procéder à une analyse morphologique d'un mot inconnu et retrouver une racine correcte. Un générateur morphologique permet ensuite de proposer une forme correcte.

Enfin, dans leur système expert (§C.12), Emirkanian et Bouchard (1989) utilisent des heuristiques pour corriger les erreurs d'orthographe. Le radical des mots est recherché dans un dictionnaire, qui contient également des informations sur le suffixe correct et sur les erreurs fréquentes de suffixation. Un dictionnaire de suffixes permet de trouver les suffixes valides de la langue. Les erreurs d'accentuation sont immédiatement corrigées. En cas d'erreurs phonétiques, 165 règles de substitution permettent de tenter de rétablir une bonne graphie.

3.2.3.7 Séparation de mots

La séparation de mots est un problème récurrent de la correction orthographique. Kukich (1992) se montre sceptique face aux problèmes de séparation de mots, qu'ils soient collés ou séparés de manière incorrecte. Les chaînes qui en résultent peuvent aboutir à un autre mot valide. Si l'on tente de joindre ou de séparer les mots, on aboutit rapidement à une explosion combinatoire. Certains correcteurs aboutissent à des solutions satisfaisantes mais ne traitent qu'un nombre limité de mots dans une tâche restreinte.

Fontenelle (2006) discute du problème de la segmentation des mots pour le correcteur de *Word* (§C.24). L'apostrophe est un séparateur sauf dans certains cas exceptionnel comme *aujourd'hui*, *prud'homme* et *presqu'île*. Le trait d'union est considéré comme un séparateur, à l'exception des quelques milliers de mots qui comportent un trait d'union (*chef-d'œuvre*, *porte-avions*, *cul-de-sac*, *brise-glace* etc.). Des mécanismes particuliers non dévoilés dans l'article permettent de traiter les nombreuses erreurs portant sur le pluriel comme **portes-avions* etc. Enfin, un traitement particulier est appliqué aux erreurs fréquentes qui impliquent une assez grande distance d'édition comme **éléfant* pour *éléphant*. Enfin, des erreurs de segmentation comme

**il straitent* sont traitées correctement, avec d'une part la correction de **straitent* pour *traitent*, et d'autre part la détection de l'erreur d'accord avec **il* pour *ils*.

3.2.3.8 Listes de confusion

Les listes de confusion consistent en une série de mots qui sont souvent confondus, comme *then / than* en anglais, *prémisses / prémices*, etc. (Mangu et Brill, 1997; Sitbon *et al.*, 2007).

Ces listes sont souvent employées dans les correcteurs orthographiques : lorsqu'un mot confondu apparaît dans un texte, un message d'avertissement prévient l'utilisateur en lui donnant un exemple pour l'aider à choisir l'élément correct. Dans certains correcteurs, cette option peut être désactivée. D'autres correcteurs proposent à l'utilisateur de désactiver uniquement certaines règles qu'ils estiment maîtriser.

Cette technique est extrêmement rudimentaire mais peut s'avérer utile. En effet, certains mots sont très souvent confondus, aussi bien par les apprenants que par les locuteurs natifs. De plus, un nombre non négligeable d'erreurs d'orthographe de tout type débouche sur un mot connu. Dans ce cas, il faudra utiliser des méthodes d'analyse syntaxique pour tenter de désambiguïser la phrase et proposer éventuellement une correction. Nous y reviendrons à la section 3.3.

3.2.3.9 Correction par méthode stochastique ou par réseaux de neurones

Kukich (1992) décrit une technique de réseaux de neurones pour la correction orthographique [26]. Cette technique d'intelligence artificielle simule le fonctionnement du cerveau humain. Un neurone reçoit les données d'un certain nombre d'autres neurones et combine les informations pour en former une nouvelle ; cette information sera ensuite utilisée par d'autres neurones. Le réseau doit d'abord être entraîné sur un corpus avant de pouvoir fonctionner sur des données nouvelles. Cette technique est efficace pour retrouver des mots à partir d'une chaîne incomplète ou incorrecte.

Dans le but de corriger les erreurs typographiques, Jones et Martin (1997) proposent une correction contextuelles basées sur l'Analyse Sémantique La-

26. V. §3.1.2 pour l'application des réseaux de neurones à l'étiquetage.

tente (LSA, v. 3.1.8 p. 75). Cette technique vise à détecter des erreurs qui débouchent sur d'autres mots comme *effect/affect*, *quiet/quite*, etc. En appliquant la LSA à des matrices décrivant des phrases tirés du corpus Brown, Jones et Martin (1997) procèdent à une description du contexte des mots qui prêtent à confusion. Les mots sont pris en isolation et deux par deux (bigrammes) pour une fenêtre de ± 7 mots autour du mot ambigu.

3.2.3.10 Mesures de distance entre chaînes et autre méthodes de filtrage

Les différentes méthodes de recherche de mots décrites précédemment trouvent souvent des mots trop éloignés du mot original. Le nombre de mots retrouvés peut être considérablement élevé, comme nous le verrons au chapitre 6. C'est pourquoi dans cette section, nous décrivons quelques méthodes de mesure de distance entre chaînes, destinées à filtrer les résultats.

La *distance lexicographique* entre deux chaînes de caractères (ou plus simplement deux mots) est le calcul du nombre minimal d'opérations nécessaires pour transformer une chaîne vers une autre. Les opérations possibles sont l'insertion, la substitution et l'effacement. La mesure de distance la plus connue, la distance de Levenshtein (Levenshtein, 1966), est une métrique simple entre deux chaînes, où chaque opération a un coût de 1. La distance est comprise entre 0 et la taille de la plus grande des deux chaînes comparées. L'algorithme est de complexité $\mathcal{O}(m \times n)$, ou m et n représentent la longueur des chaînes respectives. La distance de Levenshtein peut être pondérée par la longueur des chaînes comparées : le score obtenu est alors divisé par la somme des longueurs des deux chaînes comparées.

La figure (3.3) montre les opérations nécessaires à la transformation de *intention* vers *exécution*. Ainsi, la distance de Levenshtein entre ces deux chaînes est de 6. Une variante de cette métrique interdit l'opération de substitution et la remplace par une insertion suivie d'un effacement, ce qui revient à donner un coût de 2 à la substitution. Dans ce cas, la distance entre *intention* vers *exécution* sera de 10.

Jurafsky et Martin (2000, pp. 153 ss.) présentent un algorithme dynamique de calcul de distance entre une chaîne source et une chaîne cible, inspiré de Levenshtein (1966). Nous y reviendrons au paragraphe 6.1.2.1. Une telle mesure de distance est également applicable à la phonétique [27].

27. Voir notamment le correcteur orthographique en logiciel libre GNU-ASPELL, `http://aspell.net/`, dernière consultation le 13.04.10.

i. `intention`
ii. effacement de *i*
`ntention`
iii. substitution de *n* par *e*
`etention`
iv. substitution de *t* par *x*
`exention`
v. substitution de *e* par *é*
`exéntion`
vi. insertion de *u*
`exénution`
vii. substitution de *n* par *c*
`exécution`

FIG. 3.3 – *Transformation de* intention *à* exécution *pour calculer la distance entre deux chaînes*

Par contre, l'algorithme *Similar-Text* [28] (Oliver, 1993) retourne simplement le nombre de lettres qui se retrouvent entre deux chaînes et détermine un pourcentage de similarité. L'algorithme est de complexité $\mathcal{O}(\max(m,n) \times 3)$. Une autre mesure, la distance de la plus longue séquence commune (*longest common subsequence distance*, Needleman et Wunsch, 1970, cité par Navarro, 2001), n'autorise que les insertion et suppressions. Elle consiste à mesurer la plus grande séquence de caractères communs à deux chaînes en respectant leur ordre. La distance est le nombre de caractères sans correspondant dans l'autre chaîne.

La distance de Hamming (Hamming, 1950) compare deux chaînes de longueur égale et calcule le nombre de substitutions sur la même position. Ainsi, *b**ai**n* et *b**ie**n* auront une distance de 2, car les lettres *b* et *n*, qui sont à la même position, sont identiques, mais pas les autres lettres en gras.

La technique *three-way match* procède à au plus trois comparaisons pour calculer une distance. Un poids est calculé d'après la position du caractère dans l'erreur, d'après la proximité du caractère sur le clavier et d'après la proximité phonétique du son, d'après la taille de la chaîne et la distance d'édition. Soient S la chaîne inconnue, M le mot du lexique, n la position du caractère dans la chaîne inconnue et m la position dans le mot du lexique : si *caractère(n)* et *caractère(m)* ne sont pas identiques, on procède aux comparaisons entre *caractère(m)* et *caractère(n+1)* et entre *caractère(n)*

28. `http://php.benscom.com/manual/fr/function.similar-text.php`, consulté le 29 septembre 2006.

et *caractère(m+1)*. Par cette méthode, on peut retrouver les erreurs de permutation, effacement, insertions et substitutions. Les mots du lexique sont classés par ordre alphabétique de la première lettre et par ordre de longueur croissant.

Enfin, Church et Gale (1991) proposent de classer les propositions d'un correcteur à l'aide d'un filtre probabiliste bayésien, basé sur une estimation de probabilité du mot, sur des fréquences d'erreurs et sur le contexte droit et gauche du mot candidat (calculé sur des probabilités de co-occurrence).

3.2.4 Logiciels

Dans cette section, nous décrivons quelques logiciels de correction orthographique. Nous avons déjà passablement décrit les ouvrages des pionniers comme Morris et Cherry (1975), avec la recherche par trigrammes, ou *LADDER* (Hendrix *et al.*, 1978) ou Berghel (1987) avec la similarité par chaînes. Sitbon *et al.* (2007) décrivent un correcteur destiné aux personnes dysorthographiques, qui utilise des matrices de confusion phonétique et procède à l'aide de transducteurs (§3.3.4.1).

Dans le domaine de l'ALAO, Kempen (1992, §B.2.15) utilise l'analyse de triphones pour un correcteur du néerlandais couplé à un analyseur. Par contre, Vosse (1992, §C.36) utilise à la fois des triphones et des trigrammes pour sélectionner des mots appropriés ; ces mots sont ensuite ordonnés par un mécanisme de calcul de score et de classement. Véronis (1988b, §C.35) propose une technique de correction phonétique du français par substitution de chaînes phonétiques similaires. Enfin, pour le correcteur *CELINE* (§C.5), Courtin *et al.* (1991) passent par une double conversion graphèmes-phonèmes-graphèmes pour trouver des propositions plausibles.

Passons maintenant à des correcteurs par règles. Le tuteur intelligent *SPELLER* (de Haan et Oppenhuizen, 1994, §B.4.32) est destiné à apprendre aux apprenants néerlandophones de l'anglais à résoudre des problèmes d'orthographe. Le système traite 15 000 mots et se concentre sur les erreurs phonétiques et les erreurs typographiques (inversion, substitution et omission). Il identifie les stratégies et connaissances de l'apprenant et lui rappelle les erreurs du même type qu'il a déjà commises. Par ailleurs, Pijls *et al.* (1987), Kempen (1992) et Bos (1994) utilisent une correction par règles pour un conjugueur du néerlandais.

A cause de la richesse morphologique de la langue basque, le correcteur *XUXEN* (§C.37, Agirre *et al.*, 1992; Aduriz *et al.*, 1991) utilise un analy-

seur morphologique. Certaines erreurs fréquentes sont entrées directement dans le lexique. Quant aux autres erreurs typographiques, elles sont corrigées par analyse de trigrammes. Pour des raisons d'efficacité, seules les trois premières corrections retrouvées sont proposées à l'utilisateur. Successeur du correcteur précédent, *XuxenII* (Aldezabal *et al.*, 1999) est basé sur des transducteurs qui recherchent tous les lemmes possibles d'une forme. L'utilisation de dictionnaires d'utilisateurs permet d'étendre la couverture du correcteur.

Monson *et al.* (2004) présentent un correcteur orthographique pour le mapundungun, la langue des Mapuche, indigènes du Chili et Argentine. Cette langue a une morphologie très riche [29] en adaptant le correcteur orthographique *MySpell*, inclus dans le logiciel libre de bureautique *OpenOffice*. Quant à Enguehard et Mbodj (2004), ils discutent des problèmes liés à la correction orthographique des langues africaines, dont la morphologie est également complexe.

Le correcteur *SANTY* (§B.5.18 Rimrott, 2003) procède à une correction morphologique en tentant de retrouver les radicaux potentiel d'un mot inconnu. De même, le correcteur *Skryba* (§C.32, Nicholas *et al.*, 2004) corrige les erreurs d'orthographe grâce à des règles morphologiques et phonétiques.

Ben Othmane Zribi et Zribi (1999) évoquent les problèmes particuliers posés pour la correction de l'arabe. Les mots doivent parfois être voyellisés. De plus, l'arabe est une langue agglutinative qui utilises des affixes et des enclitiques (pronoms) et proclitiques (adverbes, prépositions et conjonctions). En outre, cette langue contient de nombreux lexèmes très voisins les uns des autres. Les propositions candidates pour la correction d'un mot peuvent donc être très nombreuses. Le correcteur est donc accompagné d'un analyseur morphologique, qui découpe les formes en proclitique, radical et enclitique.

3.2.5 Discussion

Terminons notre survol de la correction orthographique par quelques considérations. Cette technique est robuste et éprouvée et fonctionne de manière satisfaisante pour les utilisateurs. Toutefois, certaines caractéristiques comme la recherche phonétique et certaines règles d'erreurs ne sont pas adap-

29. La racine du verbe est suivie de marques de personne, de temps et de mode ; de plus, la forme peut être nominalisée ou adverbialisée ; enfin, entre la racine et les morphèmes finaux, on peut encore insérer des morphèmes indiquant l'aspect, le temps, la direction et l'accord avec l'objet.

tées aux apprenants d'une langue : ceux-ci confondent certains phonèmes (notamment les voyelles nasales), ils utilisent incorrectement les règles de conversion phonème-graphème, ils omettent les accents et les apostrophes, etc. Un bon correcteur doit être adapté à leurs besoins. Cependant, nous manquons de données fines sur les erreurs fréquentes des apprenants et sur les erreurs qui dépendent de la langue première. Ces indications seraient pourtant précieuses, notamment pour mieux ordonner les propositions. Nous reviendrons largement sur cet aspect au chapitre 6.

Par ailleurs, nous pouvons remarquer que la correction orthographique n'est pas un sujet de recherche très prisé. Il existe peu de publications comparé à d'autres domaines. Nous verrons plus tard que bien plus de projets de recherche portent sur la détection et la correction d'erreurs grammaticales.

Enfin, terminons par quelques considérations sur des pistes d'amélioration possible. Comme nous l'avons déjà souligné au paragraphe 3.1.1.4, il serait souhaitable d'utiliser un analyseur morphologique pour tenter de décomposer les mots inconnus, de retrouver la ou les racines potentielles et éventuellement d'aider l'apprenant, à travers un dialogue, à retrouver une forme correcte. Les outils comme les déclineurs et conjugueurs sont aussi une précieuse aide à la rédaction [30]. Par ailleurs, des accès à des dictionnaires monolingues et bilingues seraient recommandables, ainsi qu'à des lexiques conceptuels plus évolués comme nous l'étudions en 3.5.

3.3 Analyse syntaxique et détection d'erreurs

Dans les sections précédentes, nous avons survolé différentes techniques du traitement du langage et de vérification orthographique. Dans cette section, nous passons à la détection d'erreurs grammaticales.

Un analyseur (*parser*) est un programme informatique qui décompose une phrase en unités, vérifie si la phrase correspond aux règles d'une grammaire et en retourne la structure grammaticale (Holland *et al.*, 1993). Les langues sont toutes extrêmement ambiguës (au niveau lexical, syntaxique et sémantique) et leurs structures peuvent être extrêmement complexes. C'est pourquoi à l'heure actuelle, aucune grammaire ni aucun analyseur n'est capable d'analyser la totalité des phrases d'un texte d'une certaine complexité. Les analyseurs sont donc confrontés à de nombreuses difficultés, principale-

30. Malheureusement, l'accès aux dictionnaires est difficile pour cause de droits d'auteur.

ment :

- les nombreux mots inconnus, nom propres, mots d'origine étrangère, vocabulaire spécialisé, nouvelles productions, etc ;
- les structures ambiguës, lorsque règles de la grammaire prévoient que les constituants des phrases (syntagmes) peuvent être combinés entre eux de plusieurs manières, comme nous l'avons déjà mentionné à propos des exemples (1) p. 6 ;
- la surgénération des structures : certaines hypothèses d'analyse doivent être abandonnées en cours de route ; d'autres sont éliminées par filtres ou heuristiques, notamment au niveau sémantique ; enfin, les analyses restantes – parfois plusieurs centaines par phrase – doivent être triées selon des règles de vraisemblance et de préférences afin que la meilleure proposition soit sélectionnée [31].

On parle de la *couverture* d'un analyseur – ou plutôt d'une grammaire – pour décrire les structures de la langue qu'il est capable de décrire et analyser. Un analyseur *robuste* est capable de fournir une structure pour toute phrase, quitte à ce qu'elle soit incomplète, par exemple en cas d'erreur d'analyse, de couverture insuffisante ou d'erreur grammaticale.

Dans le domaine de l'ALAO, les apprenants commettent de nombreuses erreurs. Un analyseur classique ne peut pas donner de structure à une phrase mal formée et doit donc la rejeter. Cependant certains analyseurs sont capables d'accepter des phrases contenant des erreurs. S'ils marquent l'endroit et parfois la nature de l'erreur, on parle de *détection d'erreurs*. Si l'analyseur propose en outre une correction, on parle de *correction grammaticale*. Swartz et Yazdani (1992, introduction p. 3) proposent une autre définition de la tâche :

> "The parser must be capable of accepting divergent input strings from learners and be able to identify a plausible divergence from nonsensical language so as to be capable of reasoning properly about learners' attempts to use the L2."

Jensen *et al.* (1993) citent plusieurs caractéristiques d'un analyseur robuste pour traiter les phrases grammaticalement incorrectes. Un analyseur

31. Dans la plupart des cas, il est nécessaire de choisir une analyse préférentielle, par exemple pour la traduction automatique (§3.1.6) ou la synthèse vocale (§3.1.4). Dans d'autre cas, l'utilisateur voudra afficher tout ou partie des analyses.

doit :

i. pouvoir traiter plusieurs possibilités d'analyse ;
ii. être robuste pour pouvoir identifier le point où une erreur a été commise et identifier le type d'erreur ;
iii. pouvoir revenir en arrière et explorer d'autres possibilités d'analyse ;
iv. utiliser toutes les informations et identifier les analyses qui ne sont pas plausibles ; ceci peut être fait à l'aide d'une analyse sémantique, d'études de probabilité ou d'autres informations formelles ;
v. pouvoir se baser sur des heuristiques, qui sont fondées sur une analyse des causes des erreurs.

Dans cette section, nous commençons par décrire des typologies d'erreurs (§3.3.1). Puis nous abordons le concept de traitement des erreurs syntaxiques, de grammaire formelle et de contraintes en (§3.3.2). D'autres techniques de traitement des erreurs sont décrites en (§3.3.3). Nous abordons ensuite les techniques d'analyse syntaxique en (§3.3.4). Puis nous parlons des différents formalismes grammaticaux en (§3.3.5). Enfin nous terminons par une discussion en (§3.3.6).

3.3.1 Typologies des erreurs

Dans cette section, nous définissons la notion d'erreur et présentons quelques typologies. Cornu (1997) distingue les erreurs des locuteurs natifs de celles d'apprenant d'une langue seconde. Il définit une erreur comme "*un écart par rapport à l'usage bien défini de la langue*" (p. 3). Dans un premier temps, Cornu (1997) et Danna (1997) adoptent une analyse contrastive selon laquelle les apprenants d'une langue seconde ont tendance à transférer à la langue seconde des structures de la langue première. Ces erreurs sont appelées *interférences* ou *transfert négatif* :

– *Niveau orthographique : address* prend deux *d* en anglais et *adresse* un seul en français ;
– *Niveau morphologique :* des francophones écriront **the book of Jack* au lieu de *Jack's book* ;
– *Niveau syntaxique :* par exemple l'ordre des mots, avec *un *gris mur* par rapport à *a grey wall* ou *il veut être *un médecin* (← *he wants to be a doctor*) ;
– *Niveau lexical :* les faux-amis comme *library* (*bibliothèque*) ↔ *librairie* (*bookshop*).

Cependant, les apprenants commettent également des erreurs *intralangue*, pour lesquelles ils tentent d'appliquer des règles qui n'ont pas lieu d'être :

- *Surgénéralisation de règles* :
  - *Niveau morphologique :* règle régulière appliquée aux verbes irréguliers (**allerai* pour *irai*) ;
  - *Niveau syntaxique :* des règles d'inversion applicables aux interrogatives directes appliquées aux interrogatives indirectes (**je ne sais pas où iras-tu*) ;
  - *Niveau sémantique :* des mots sont appliqués à un contexte inapproprié (*?cette voiture est séduisante*, où l'on attribue des caractéristiques humaines à un objet ; l'acceptabilité de cette tournure est une question de style, de niveau de langue et d'interprétation, d'où le point d'interrogation) ;
- *Simplification excessive :* des éléments sont omis, comme dans **je garçon*, où le verbe *être* serait omis par un locuteur d'une langue où ce verbe peut être omis ou n'existe pas ;
- *Surélaboration :* cette construction est l'inverse de la précédente ; l'apprenant construit une structure trop compliquée (**la femme dont au sujet de laquelle je t'ai parlé*).

Les apprenants développent aussi des stratégies pour éviter les erreurs. Ils auront ainsi tendance à utiliser des périphrases pour pallier leur manque de vocabulaire, ils éviteront aussi certaines tournures difficiles à maîtriser.

Tasso *et al.* (1992) distinguent l'erreur de *commission* et l'erreur d'*omission*, qui sont deux faces de l'erreur commise par un apprenant. L'erreur de commission consiste à appliquer une mauvaise règle par méconnaissance des règles. L'erreur d'omission est le fait d'ignorer l'existence d'une règle. Ces deux problèmes doivent être pris en compte pour analyser les erreurs des apprenants.

Dans sa thèse, Keller (2000) propose une théorie sur les degrés de grammaticalité (*gradience*). Ce phénomène concerne selon lui tous les aspects de la grammaire. On trouve une autre théorie chez Prost (2008). Nous reviendrons sur cette notion par la suite.

Vandeventer Faltin (2003, pp. 4–5) liste les erreurs faites par des apprenants d'une langue :

- *Erreurs de dérivation :* création de mots inexistants en utilisant un mauvais schéma de dérivation. Ex : *nonpossible.

- *Erreurs d'homophonie :* confusion d'un mot avec un mot plus fréquent et proche phonologiquement. Ex : *reine* au lieu de *reigne*.
- *Erreurs d'ordre des mots :* les contraintes d'ordre ne sont pas respectées, souvent par analogie avec la langue maternelle de l'apprenant. Ex . **Il jamais n'a vu.*
- *Erreurs de genre des mots.* Ex : **la soleil.*
- *Erreurs d'auxiliaire.* Ex . **j'ai né.*
- *Erreurs d'euphonie.* Ex : **ce endroit.*
- *Erreurs de ponctuation.* Ex : ** il mange, et il boit.*

Payette (1990) définit plusieurs critères stylistiques pour juger une phrase :

- *éléments lexicaux :* il convient de choisir des mots appartenant à un registre adéquat suivant le type de texte ;
- *éléments syntaxiques :* l'organisation de la phrase et de ses différents membres reflète la clarté. Il faut utiliser les bons éléments pour une transition et varier les types de phrase ;
- *éléments rhétoriques :* il faut enrichir la prose en utilisant des expressions imagées et figuratives ;
- *éléments structuraux :* la structure du discours est essentielle. Un paragraphe doit être un groupe de phrases exprimant et développant une idée commune, en les co-articulant de manière adéquate.

Enfin, pour Heift et Schulze (2007), il est crucial de définir des classes d'erreurs qui conviennent à la vérification grammaticale, afin de rendre efficaces à la fois la détection des erreurs et la rétroaction apportée aux apprenants (§2.4). Ainsi, la description d'une erreur et la rétroaction seront tout deux basés à la fois sur l'erreur individuelle et sur la classe d'erreur. Une description cohérente aide aussi à définir les erreurs fréquemment commises par le même apprenant, grâce aux données contenues dans le modèle de l'apprenant (§§2.7.4.2, 2.7.4.3). Mais ces classifications posent souvent des problèmes de couverture et d'homogénéité. Nous pouvons aussi ajouter qu'il faut trouver un compromis entre les impératifs linguistiques, pédagogiques et computationnels : il est aussi important, comme nous le verrons par la suite, de ne pas alourdir le processus d'analyse ou de ne pas définir un type d'erreurs qui n'aurait qu'un bas taux de rappel des erreurs, alors qu'une granularité moins fine offrirait néanmoins une meilleure couverture. Pour le système *German Tutor* (§B.4.23), Heift et McFetridge (1999) utilise une classification hiérarchique qui permet différents niveaux de granularité, qui sont ensuite utilisés pour une rétroaction plus ou moins spécifique.

Pour terminer, mentionnons encore les ouvrages de Carbonell et Hayes (1984), Druard (1993), Schwind (1995), Ramírez Bustamante et Sánchez León (1996), James (1998), Cordier-Gauthier et Dion (2003) et Mirzaiean et Ramsay (2005), qui contiennent des typologies intéressantes.

3.3.2 Règles syntaxiques et traitement des erreurs

Dans une langue naturelle, une phrase et ses différents composants ou syntagmes sont soumis à des *règles de bonne formation*. Ainsi, on admettra généralement qu'une phrase est composée au minimum d'un verbe et son sujet, autrement dit respectivement d'un prédicat (une action ou un état) et d'un argument (celui qui mène l'action ou qui est affecté par l'état). Le sujet est généralement constitué d'un nom, commun ou propre. Dans de nombreuses langues, un nom commun est très souvent associé à un déterminant et forme un constituant appelé *syntagme nominal*, dont le nom est l'élément noyau (tête). Ces règles de bonne formation sont associées à des *contraintes* qui posent des conditions à la combinaison des constituants de la phrase, comme les règles d'accord.

Dans cette partie, nous commençons par décrire les principes de grammaire formelle ou indépendante du contexte en (§3.3.2.1) et terminons par les notions de contrainte et relâchement de contrainte (§3.3.2.2).

3.3.2.1 Grammaires formelles ou indépendantes du contexte

Les grammaires indépendantes du contexte sont des grammaires formelles utilisées par de nombreux analyseurs syntaxiques. Une règle de grammaire peut être exprimée sous forme d'une règle de réécriture indépendantes du contexte, comme celles (non exhaustives) que nous donnons en (10), qui correspondent à la syntaxe de la langue française. On trouvera une introduction aux grammaires hors contexte par Wehrli (1997, ch. 2). La flèche dénote la relation "se réécrit". Les tableaux (3.1) et (3.2) donnent les valeurs des étiquettes.

(10) a. S → NP VP
b. NP → Det N
c. NP → N
d. NP → Det Adj N
e. NP → NP PP

Catégorie	en français	en anglais	exemples
Det	déterminant	determiner	le, la, un, cet, ce
N	nom	noun	chien, arbre, Paul
V	verbe	verb	manger, dort
Adj	adjectif	adjective	bleu, petite
Adv	adverbe	adverb	gentiment, bien
Prep	préposition	preposition	avec, de
Conj	conjonction	conjunction	et, ou, que

TAB. 3.1 – *Grammaires hors contexte : catégories terminales ou lexicales*

Catégorie	en français	Catégorie	en anglais
P	phrase	S	sentence
SN	syntagme nominal	NP	noun phrase
SV	syntagme verbal	VP	verb phrase
SA	syntagme adjectival	AP	adjective phrase
SAdv	syntagme adverbial	AdvP	adverb phrase
SP	syntagme prépositionnel	PP	prepositional phrase

TAB. 3.2 – *Grammaires hors contexte : catégories non terminales ou syntagmatiques*

f. VP → V
g. VP → V NP
h. VP → V NP PP
i. PP → Prep NP

Les catégories *terminales* sont les éléments atomiques de la grammaire. Elles distinguent les mots ou éléments lexicaux, également appelés parties du discours (*part of speech*). Les éléments *non terminaux* sont des catégories plus complexes, qui regroupent des symboles terminaux et/ou non terminaux pour former des catégories plus complexes, appelées *syntagmes*. L'élément dominant du syntagme, qui donne son nom au syntagme, est appelé *tête*. Chaque syntagme comprend des catégories obligatoires, qui constituent un syntagme minimal. Un syntagme peut avoir des éléments facultatifs, nommés *ajouts* ou *modifieurs*, qui sont d'autres syntagmes attachés à certaines positions.

De plus, les prédicats varient en fonction du nombre et du type d'arguments qui les composent, qui est appelée *valence* (Kaplan, 2003). Ainsi, les verbes transitifs ont un prédicat à deux places, toutes deux formées de syntagmes nominaux. Tous les verbes ayant la même valence forment une

sous-catégorie, qui se distingue par une réalisation syntaxique différente, appelée *cadre de sous-catégorisation*. Certains verbes peuvent accepter plusieurs constructions et donc avoir plusieurs cadres de sous-catégorisation. Les noms et adjectifs peuvent également avoir un cadre de sous-catégorisation. En (11), nous énumérons quelques cadres de sous-catégorisation entre crochets ; le soulignement reprend l'élément lexical :

(11) a. dormir [_] (verbe intransitif)
b. manger [_ NP] (verbe transitif direct)
c. aller [_ PP(à)] (verbe transitif indirect)
d. donner [_ NP PP(à)] (verbe ditransitif indirect)
e. laver [se _] (verbe pronominal)
f. verre [_ PP(à)]

Ainsi, en (11b), le verbe *manger* ne pourra utiliser qu'une partie des règles de la grammaire, qui correspondent à son cadre de sous-catégorisation (dans notre grammaire, la règle (10g)). En (11c) et (11d), on stipule la préposition à utiliser dans le syntagme prépositionnel. (11e) illustre l'utilisation du verbe *se laver*. En (11f), nous illustrons les termes de *verre à vin* [32] et *verre à pied*.

Parmi tous les éléments non terminaux figure une catégorie initiale, ici S, qui est la catégorie de la phrase et qui doit être au début de toute dérivation. Une dérivation est une séquence de règles appartenant à la grammaire, à partir du symbole initial, qui permet de produire une chaîne du langage qui ne contient plus que des symboles terminaux. Prenons un exemple simple :

(12) a. Les pitbulls aboient.
b. S ⇒ NP VP ⇒ Det N VP ⇒ Det N V
c. $[_{\texttt{S}}\ [_{\texttt{NP}}\ [_{\texttt{Det}}$ les$]\ [_{\texttt{N}}$ pitbulls$]]\ [_{\texttt{VP}}\ [_{\texttt{V}}$ aboient$]]]$

La phrase en (12a) peut être analysée à l'aide de la grammaire en (10), ce qui donne la structure en (12c). On obtient la dérivation en (12b) en appliquant successivement les règles (10a), (10b) et (10f) [33]. Cette dérivation peut être représentée par un graphe arborescent, comme le montre la figure (3.4).

32. Un verre *à* vin désignera le contenant, un verre spécialisé pour contenir du vin. Avec la préposition *de*, on désignera le contenu. Cette différence pourra être marquée par les traits sémantiques.
33. Nous partons du principe que c'est le lexique qui donne les catégories lexicales des mots, et que notre grammaire n'a pas besoin de règles de réécriture comme D → les et N → pitbulls.

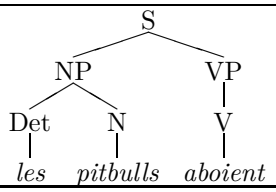


FIG. 3.4 – *Représentation arborescente de la phrase (12)*

Cette dérivation représente une stratégie d'analyse descendante, ou analyse dirigée par les hypothèses. L'analyse ascendante, dirigée par les données, part des mots et tente de remonter vers la catégorie initiale de la grammaire.

Outre la combinaison d'éléments lexicaux, les analyseurs doivent vérifier des contraintes pour valider ces phrases, en particulier les conditions sur les accords, dont nous parlons à la section 3.3.2.2. Signalons également qu'un analyseur traite généralement plusieurs possibilités d'analyse à la fois, vu que la langue est très ambiguë. Il est rare qu'une phrase n'ait qu'une seule analyse complète. Dans la pratique, elle en a des dizaines, voire des centaines. Ainsi, les règles applicables à un stade d'analyse sont activées et les règles activées qui ne peuvent plus être complétées sont abandonnées ; c'est ce que l'on appelle une analyse en parallèle.

Mentionnons encore quelques phénomènes qui nécessitent un traitement particulier et posent souvent des problèmes d'ambiguïté :

(13) a. Eric a écrit et débogué un analyseur.

b. Jacques aime sa femme, et moi aussi.

c. Sébastien commande encore une bière.

En (13a), on montre le problème de la coordination : *Eric* est à la fois sujet du verbe *écrire* et *déboguer*. Parfois les structures coordonnées sont fort complexes et il est nécessaire de produire une analyse vraisemblable. En (13b), nous illustrons le problème de l'ellipse, où des fragments de phrase sont omis pour éviter des répétitions lourdes. Ici, on peut compléter la phrase par *j'aime sa femme* ou *j'aime ma femme*. En (13c), la portée de l'adverbe *encore* peut concerner *commander* ou *bière*, selon le contexte. Enfin, rappelons l'exemple (1a) p. 6 qui illustre l'attachement des syntagmes prépositionnels.

3.3.2.2 Contraintes et relâchement de contraintes

Les contraintes sont des conditions qui doivent être vérifiées pour permettre la combinaison de constituants entre eux ; elles sont donc une restriction qui s'applique aux règles de réécriture vues précédemment. Le premier type de contrainte est celui des règles de réécriture elles-mêmes. Ainsi, à l'aide de la grammaire (10) p. 103, on ne pourra pas dériver les exemples suivants, qui sont agrammaticaux.

(14) a. * dormir chat gentil.
b. * chat le mange souris la.

D'après notre grammaire, aucune dérivation ne permet de débuter une phrase par un verbe, ce qui élimine (14a). Aucune règle ne permet de dériver N Det, ce qui élimine (14b). Le second type de contrainte s'applique aux règles de réécriture pour en restreindre la portée. Voici deux règles de (10) enrichies de contraintes sur leur application :

(15) a. S $\rightarrow$ NP VP, {accord en nombre, accord en personne}
b. NP $\rightarrow$ Det N, {accord en nombre, accord en genre}

L'accord entre les éléments d'un syntagme est vérifié par l'intersection des valeurs d'ensembles. L'exemple (16) montre pour la phrase (12a) que la règle d'accord de (15b) est vérifiée. Le nouvel ensemble de traits qui résulte de l'intersection est ensuite utilisé pour vérifier la règle (15a).

(16) les{pers = 3; gen=masc, fem; nb=pl} $\cap$ pitbulls{pers=3; gen=masc; nb=pl} = {pers=3; gen=masc; nb=pl}

En outre, le second type de contrainte consiste à respecter le cadre de sous-catégorisation d'un élément lexical, ou d'un de ses cadres s'il en compte plusieurs. Enfin, un troisième type de contrainte porte sur des critères de sélection sémantique grâce à des traits contenus dans le lexique, pour autant que celui-ci soit riche et que la grammaire prévoie un traitement sémantique de base. Considérons les exemples suivants :

(17) a. ? L'oreiller mord.
b. ? Le chat lit une BD.

c. ? L'agneau dévore le loup.

Les phrases en (17) respectent les règles d'accord. Pourtant, elles sont difficilement acceptables. Des contraintes sémantiques sont généralement associées aux verbes. Ainsi, en (17a), *mordre* possède un trait sémantique qui stipule que son sujet doit être animé. Si l'on n'est pas dans un monde de fiction, (17b) est impossible car *lire* demande un sujet humain. Enfin, (17c) relève du champ sémantique des mots : *dévore* implique un sujet carnivore [34].

Ainsi, la détection d'erreurs dépend d'un lexique riche et consistant. Dans le cadre de l'ALAO, pour que le but pédagogique puisse être atteint, il faut définir une couverture grammaticale et un lexique adapté à la tâche, afin de couvrir le plus exactement possible le champ grammatical à couvrir par l'application.

Passons maintenant aux techniques de relâchement de contraintes, qui permettent à l'analyseur syntaxique de produire une structure complète même en cas d'erreur de l'utilisateur. Dans ce cas, il est utile d'indiquer dans la structure grammaticale l'endroit et la nature de l'erreur. La rétroaction du système est alors plus efficace puisque l'apprenant sait précisément où il a commis des erreurs et peut y remédier. En outre, certains systèmes proposent une ou plusieurs corrections.

L'idée du relâchement de contraintes est très intuitive : dans une situation normale, un analyseur cherche à produire une analyse syntaxique complète d'une phrase en tenant compte de règles qui conditionnent les combinaisons des éléments de la phrase. Si l'analyseur ne peut donner une structure à la phrase complète, on peut tenter de désactiver une ou plusieurs contraintes sur ces règles, tant au niveau du mot qu'à celui du syntagme, pour obtenir une analyse complète. Cette technique est plus largement décrite chez Vandeventer Faltin (2003, chap. 3), dont nous nous inspirons.

De nombreux ouvrages traitent du relâchement de contraintes (Weischedel *et al.*, 1978; Weischedel et Black, 1980; Kwasny et Sondheimer, 1981; Hayes et Mouradian, 1981; Weischedel et Sondheimer, 1983; Granger, 1983; Carbonell et Hayes, 1984; Vandeventer, 2000; Vandeventer Faltin, 2003; Menzel, 2004). Afin de limiter le nombre d'analyses, les contraintes ne doivent être relâchées que si aucune analyse complète de la phrase peut être obtenue. Vandeventer Faltin (2003, p. 69) identifie deux actions à accomplir

34. Laissons également de côté le problème de la polysémie des mots : *dévorer* peut avoir un sens figuré comme dans *dévorer un livre* ou *dévorer des yeux*. *Loup* désigne également un poisson et est utilisé dans l'expression *vieux loup de mer* pour un marin expérimenté.

pour que le diagnostic soit efficace :

i. Le lieu où une contrainte a été relâchée doit être mémorisé pour faciliter le diagnostic ;
ii. Si les contraintes impliquent la vérification de valeurs de traits et le calcul de nouveaux traits pour le syntagme résultant de la combinaison (comme l'exemple (16) p. 107), en cas d'erreur, l'*intersection* des ensembles des traits sera vide et l'on ne pourra pas vérifier ce trait pour produire une phrase complète. Dans ce cas, on opérera l'*union* de l'ensemble. Dans d'autres cas, il sera nécessaire de définir un trait par défaut.

Passons maintenant à l'ordre d'application du relâchement et l'opportunité de relâcher certaines règles. D'un côté, les contraintes dures sont difficilement relâchables. Il est en effet peu souhaitable d'autoriser n'importe quelle combinaison de syntagmes, ce qui conduirait à une explosion du nombre de structures (Imlah et du Boulay, 1985; Vandeventer Faltin, 2003). Quelques règles d'ordre des adjectifs et des adverbes peuvent être néanmoins relâchées, car la structure varie peu. D'un autre côté, pour les contraintes douces ou conditionnelles, il est important de définir quelles sont les erreurs qu'il faut détecter et quelles contraintes doivent être relâchées.

Concluons par quelques réflexions sur les avantages et inconvénients posés par le relâchement de contraintes. Nous avons déjà évoqué le principal avantage : l'apprenant obtient une structure complète de sa production et peut ensuite utiliser les informations fournies pour remédier à son erreur. Une analyse complète montre aussi que la machine a pu donner un sens et une structure à une phrase, ce qui présente un aspect valorisant pour l'apprenant :

> "Being able to reach a complete parse, even if containing errors and having required the use of relaxed contraints, is also an indication that the sentence grammaticality, although not perfect, is not too far off the mark." (Vandeventer Faltin, 2003, p. 71)

Le principal inconvénient est la surgénération de structures. Vu l'ambiguïté du langage naturel, un analyseur produit parfois plusieurs dizaines de structures différentes pour une phrase simple. Des centaines d'autres phrases ont été éliminées par l'application de contraintes. En relâchant des contraintes, on augmente considérablement le nombre de structures produites. En effet, par effet boule de neige, une contrainte relâchée ouvrira la voie à une

analyse qui peut aboutir ensuite à d'autres relâchements de contraintes. D'autre part, parmi les différentes analyses, il est parfois préférable de sélectionner une analyse contenant des erreurs plutôt qu'une structure sans erreurs. Dans tous ces cas, il est alors nécessaire d'éliminer les structures les plus improbables par des heuristiques.

3.3.3 Autres techniques de traitement des erreurs

Dans cette section, nous abordons brièvement d'autres techniques de traitement des erreurs grammaticales. Nous commençons par l'approche par règles en (§3.3.3.1), puis viennent la réinterprétation phonologique (3.3.3.2) et les méthodes stochastiques (§3.3.3.3).

3.3.3.1 Approche par règles

Dina et Malnati (1993) proposent trois approches, outre le relâchement de contraintes :

- L'approche basée sur les règles : on combine des règles pour des entrées grammaticales et d'autres pour les entrées agrammaticales. La difficulté de l'approche consiste à anticiper les erreurs potentielles des apprenants ;
- L'approche basée sur des métarègles : si aucune règle de bonne formation n'est satisfaite, des contraintes peuvent être relâchées. Selon les auteurs, il peut y avoir des problèmes lors de multiples erreurs ;
- L'approche basée sur les règles de préférence : une grammaire qui surgénère les structures est accompagnée de règles de préférence. "... each time a formal condition is removed from a b-rule to make its applicability context wider, a preference rule must be added to the grammar. Such a p-rule must be able to state – in the present context of the b-rule – the condition that has been previously removed." (Dina et Malnati, 1993, p. 78).

Par ailleurs, il est possible de définir des règles de production d'erreurs (*mal-rules*) qui se basent sur des erreurs communes aux apprenants. Cette technique est surtout utilisée pour des règles d'ordre de mots ou de mots manquants, mais Schneider et McCoy (1998) les utilisent également pour des erreurs d'accord. Thurmair (1990) propose une technique similaire de règles de secours (*fallback rules*), qui ne s'appliquent que lorsque les règles

ordinaires ne donnent lieu à aucune analyse. Jensen *et al.* (1983, 1993) proposent une technique de *parse fitting*, qui consiste, en cas d'échec d'analyse, de placer un élément qui ne peut pas être combiné dans la position la plus plausible, afin de pouvoir poser un diagnostic. Clément *et al.* (2009) proposent également d'utiliser des règles d'erreurs pour paramétrer la détection d'erreurs en fonction de la langue maternelle de l'apprenant.

Vandeventer Faltin (2003) propose une méthode de réinterprétation de morceaux d'analyse (*chunk reinterpretation*) [35], où il s'agit de chercher à donner une structure complète à une phrase à partir des différents constituants partiels trouvés par un analyseur. Comme la méthode est basée sur des règles, elle est extrêmement dépendante de l'analyseur et de l'algorithme choisi.

Hermet et Alain (2009) présentent une approche basée sur deux techniques pour corriger les prépositions. D'une part, la traduction aller-retour de la langue seconde vers la langue première puis en revenant à la langue seconde, en utilisant un traducteur en ligne. D'autre part, un analyseur détecte la préposition erronée et le système génère des propositions de correction en substituant d'autres prépositions de la même famille (temporelles, locatives, etc.) et en recherchant le nombre d'occurrence des diverses propositions sur *Internet*. La meilleure correction est celle qui atteint le meilleur score en combinant les deux méthodes.

3.3.3.2 Réinterprétation phonologique

La confusion de mots qui ont une prononciation identique ou quasi-identique mais une orthographe différente est fréquente, chez les locuteurs natifs comme chez les apprenants. Ces mots sont appelés *homophones*. Vandeventer Faltin (2003) distingue (i) l'écriture phonétique, qui consiste à retranscrire de manière erronée un mot dont on entend la prononciation, et (ii) la substitution d'un mot erroné mais correctement orthographié à la place d'un autre. Considérons les exemples suivants :

(18) a. **Mossieu* Paul est arrivé!
b. *Les *prémices* de son raisonnement sont fausses.
c. *Je *suie* la voiture.

35. Cette méthode ne peut être rapprochée de l'analyse par morceaux (§3.3.4.3), qui consiste à n'isoler que certains constituants d'une phrase sans chercher à lui donner une structure complète et profonde.

En (18a), il s'agit d'une erreur orthographique, du type (i), où une transcription phonétique du son [məsjø] ou [mosjø], correctement orthographié *Monsieur*. Cette erreur peut être traitée par un correcteur orthographique. En revanche, en (18b), l'erreur, de type (ii), porte sur la confusion sémantique entre *prémisses* (proposition logique faisant partie d'un raisonnement) et *prémices* (premières manifestation d'un phénomène). Une telle erreur produit une phrase grammaticale. Par contre, en (18c), nous examinons une erreur de type (ii), où le substantif *suie* est incorrectement employé à la place de la forme verbale *suis* (du verbe *suivre*). Cette erreur (18c) n'aboutira pas à une analyse complète. Un pronom *je* ne peut être suivi par un substantif, et ce substantif ne peut être rattaché à un groupe nominal. Nous nous trouvons donc avec trois morceaux d'analyse.

La réinterprétation phonétique est une technique qui consiste à transcrire les mots d'une phrases en phonèmes et à y substituer une ou plusieurs alternatives en les recherchant dans un lexique phonétique (Vandeventer Faltin, 2003). Ensuite, les différentes propositions doivent être soumises à nouveau à l'analyseur syntaxique. Si une phrase entière peut être réinterprétée phonologiquement, un tel processus est très lourd, demande beaucoup de ressources et de temps et peut conduire à un très grand nombre d'analyse, comme dans le cadre de la reconnaissance vocale (§3.1.3). L'erreur de la phrase (18c) ne pourrait donc aboutir qu'à des suggestions de substitutions de mots. De plus, plus la phrase est longue, plus la réinterprétation prend du temps, car il s'agit d'étudier tous les cas de combinaisons de mots possibles. Dans les cas d'homonymie de mots de même catégorie lexicale, il est nettement plus efficace de dresser une liste de mots fréquemment confondus et de fournir un avertissement expliquant les différences sémantiques et demandant à l'utilisateur de vérifier qu'il emploie le bon mot.

Par conséquent, pour éviter ces inconvénients, il est préférable de se borner à ne substituer que certains mots. Comme les différents morceaux d'analyse trahissent souvent une erreur qui s'est produite à leurs extrémités, il est intuitif de ne remplacer que les mots qui se trouvent aux frontières des morceaux. Dans l'exemple (18c), les mots *je*, *suie* et *la* peuvent être réinterprétés.

La réinterprétation phonologique nécessite deux outils fondamentaux : (i) un lexique où une transcription phonétique est associée aux mots et où l'on peut rechercher des éléments par une chaîne phonétique ; (ii) un outil de transcription graphèmes-phonèmes (phonétiseur), tel que ceux que l'on utilise pour la synthèse vocale (§3.1.4) pour la prononciation des mots inconnus [36].

36. Idéalement, ce phonétiseur devrait générer toutes les prononciations possibles d'un mot, et non pas donner une seule solution. Par ailleurs, les phonèmes proches comme [aɑ],

Passons maintenant aux systèmes utilisant une réinterprétation phonologique. Nous avons déjà parlé de Vosse (1992, §C.36) et Courtin *et al.* (1991, §C.5) pour la partie orthographique de la correction (§3.2.4). De plus, Vosse (1992) utilise une réinterprétation phonologique pour les mots existants qui ont des homophones de catégorie lexicale différente. Enfin, avec le correcteur grammatical de *Microsoft Word* 97 (§C.24), Heidorn (2000) procède avec des paires d'homophones qui sont fréquemment confondus ; il n'y a donc pas de phonétisation de mots ou de recherche dans un lexique contenant la phonétisation des mots.

3.3.3.3 Méthodes stochastiques

Les grammaires indépendantes du contexte probabilistes (*Probabilistic Context-Free Grammar, PCFG*, Manning et Schütze, 2000) sont des règles indépendantes du contextes (§3.3.2.1), auxquelles on ajoute des probabilités d'occurrence des règles. Chen *et al.* (2002, §B.4.9) et Chen *et al.* (2005) décrivent une méthode POST-parsing (*part-of-speech tagging*) qui se base sur un analyseur écrit en PCFG. Le système compare la phrase de l'apprenant à des modèles de phrases stockée dans sa base de données et trouve la phrase modèle qui s'approche le plus de celle de l'apprenant.

Passons aux méthodes stochastiques appliquées à des textes étiquetés. Comme travail de diplôme, Naber (2003, §C.23) propose un correcteur grammatical destiné au logiciel libre de suite bureautique *OpenOffice* [37]. Il se base sur l'étiqueteur probabiliste *QTag* (Tufis et Mason, 1998) [38] basé sur un corpus d'entraînement d'un million de mots pour l'anglais et sur un petit corpus de 25 000 mots pour l'allemand. D'autres projets de logiciels libres basés sur un étiqueteur existent comme *CoGrOO* (§C.7) pour le portugais brésilien ou *GRAC* (§C.14) pour le français. Le correcteur libre *An Gramadóir* (§C.2) utilise un système analogue : les règles de désambiguïsation peuvent être soit écrites à la main, soit construites automatiquement par apprentissage à l'aide de l'algorithme de Brill (1995). Enfin, Intégré dans l'environnement d'apprentissage du suédois *Grim* (§B.2.13), le vérificateur grammatical *Granska* (Carlberger *et al.*, 2002; Knutsson *et al.*, 2002, 2003b,c, 2007) se sert également de la sortie d'un étiqueteur basé sur les Modèles de Markov Cachés (HMM, v. p. 49) et sur des règles d'erreurs ainsi que des règles d'aide et des

[eɛøœə], [oɔ], [uy] ou les voyelles nasales [ɑ̃ɔ̃ɛ̃œ̃] devraient également être substitués, car ils sont facilement confondus (Hannahs, 2007). D'ailleurs [ɛ̃] et [œ̃] ont une prononciation identique dans la plupart des régions francophones, sauf notamment en Suisse romande.

37. `http://fr.openoffice.org/`, dernier accès le 10 août 2006.

38. `http://www.english.bham.ac.uk/staff/omason/software/qtag.html`, dernier accès le 10 août 2006.

règles d'exception. Le système ESL-WEPS (Yi *et al.*, 2008, §B.5.22) utilise une analyse de surface au moyen d'HMM et des heuristiques pour détecter les erreurs d'apprenants.

Maintenant, nous passons aux méthodes basées sur des trigrammes. Golding et Schabes (1996) proposent un correcteur d'erreurs d'orthographe qui résultent en d'autres mots corrects, comme dans la phrase *Can I have a *peace of cake?*, où *piece* devrait être employé au lieu de *peace*. Ils se basent sur la combinaison de deux techniques : l'emploi de trigrammes de catégories lexicales, entraînés sur un gros corpus, et une méthode probabiliste bayésienne [39] basée sur (i) la présence d'un autre mot dans le voisinage immédiat (± n mots) et (ii) la présence de collocations (de mots particuliers ou certaines catégories lexicales). De son côté, Bigert (2004) décrit une méthode stochastique de détection d'erreurs résultant en des mots connus, comme *there*, *their* et *they're* en anglais, qui sont souvent confondus. Les fréquences de trigrammes sont une technique fiable mais qui peut déboucher sur la surdétection en cas de suites rares. Dans un autre registre, Liu *et al.* (2009) présentent un outil de correction d'erreurs lexicales basée sur des corpus de n-grammes hybrides, où les éléments peuvent être soit lexicaux, soit des étiquettes de catégorie lexicale.

Examinons ensuite les méthodes basés sur l'apprentissage. Torlakovic *et al.* (2004) présentent un système d'apprentissage des adverbes anglais. Celui-ci retrouve des exemples d'utilisation des adverbes dans certains contextes en se basant sur un modèle statistique. Le système est capable de corriger les phrases des apprenants : s'il existe plusieurs positions possibles pour placer l'adverbe, il signale la meilleure. Un autre système est proposé par Sun *et al.* (2007). Han *et al.* (2004) présentent un système d'apprentissage de l'utilisation de l'article en anglais, destiné à des apprenants locuteurs natifs de langues où l'article n'est pas utilisé (russe, chinois etc.). Le système utilise la théorie de l'entropie maximale [40]. Le système de Izumi *et al.* (2003) pour la détection d'erreurs en anglais parlé utilise la même théorie. C'est aussi le cas de *Dapper* (*Determiner And PrePosition Error Recogniser*, De Felice et Pulman, 2008, 2009), centré sur les préposition et entraînée sur un gros corpus afin de déterminer les contextes d'emploi des préposi-

39. Une règle bayésienne de probabilités stipule que : $P(X|Y) = \frac{P(Y|X) \times P(X)}{P(Y)}$, où P(X | Y) est la probabilité que X est le mot correct à la place de Y ; P(Y | X) est la probabilité conditionnelle d'observer Y lorsque X est le mot correct ; enfin, P(X) et P(Y) sont les probabilités de rencontrer X et Y de manière indépendante.

40. Dans le domaine du traitement informatique, l'entropie est une fonction mathématique qui stipule que plus une information est redondante, moins elle est importante ; l'entropie est maximale si tous les symboles de la source ont la même probabilité, autrement dit si l'information est identique. L'entropie est utilisée pour la compression des données, la cryptographie, l'étiquetage syntaxique, l'indexation d'informations, etc.

tions : une fenêtre de ± 3 mots est déterminée, et le dictionnaire *WordNet* (§3.5.3) permet de définir les classes sémantiques. La précision des résultats est d'environ deux tiers, ce qui est relativement faible. Mentionnons encore les travaux de *Lentillak* (Alegria *et al.*, 2006, §B.5.9), Nagata *et al.* (2006) et *COTiG* (Quixal *et al.*, 2008).

Foster (2004) présente une méthode stochastique basée sur des corpus parallèles, l'un de phrases agrammaticales et l'autre des phrases corrigées correspondantes. Les phrases agrammaticales sont analysées et reçoivent un score de similarité avec la ou les phrases grammaticales correspondantes. Elle utilise l'analyseur probabiliste à large couverture de Charniak (2000), basé sur l'entropie maximale.

Brockett *et al.* (2006) décrivent l'utilisation d'une technique de traduction automatique statistique de syntagmes pour la correction d'erreurs d'apprenants sinophones de l'anglais pour l'emploi de noms comptables.

Doll et Coulombe (2004) suggèrent d'utiliser les probabilités de la présence d'un mot dans un texte ; bien que les auteurs ne donnent pas de détails sur les techniques possibles, on peut proposer de demander à l'utilisateur de spécifier un style de texte et de considérer comme suspect un mot rarement utilisé dans ce contexte donné.

Pour le logiciel *CALIS* (§B.1.1), Borchardt (1987, 1995) présente succinctement une technique basée sur les réseaux neuronaux qui vise à détecter les réponses intentionnellement erronées (*bogus inputs*) commises par certains apprenants dans le but de connaître la bonne réponse pour ensuite effectuer un parcours sans faute et obtenir une bonne note. Les réseaux neuronaux ont été entraînés sur des phrases authentiques, avec ou sans erreurs intentionnelles. Le système ne commet pas de fausse détection mais laisse passer quelques erreurs intentionnelles.

Pour conclure, remarquons que les méthodes stochastiques n'ont qu'une couverture limitée des erreurs, à l'exception de la méthode basée sur les PCFG et des méthodes utilisant des étiqueteurs probabilistes. Dans ces cas, les probabilités ne sont qu'une aide à la désambiguïsation. Ensuite, des règles servent à obtenir un diagnostic. Toutefois, les méthodes stochastiques peuvent s'avérer un complément intéressant aux règles de grammaire pour des difficultés particulières de la langue dont cette section a donné quelques échantillons.

3.3.4 Algorithmes et techniques d'analyse

Après avoir traité des différentes techniques de détection d'erreurs, nous nous penchons brièvement sur quelques familles d'algorithmes utilisées pour les mettre en œuvre. Nous évoquerons successivement les automates (§3.3.4.1), les Grammaires d'unification et DCG (§3.3.4.2) et l'analyse par morceaux (§3.3.4.3). Pour un survol plus complet des techniques d'analyse, on se référera notamment aux ouvrages de Fuchs *et al.* (1993), Wehrli (1997) et Nazarenko (2006)

3.3.4.1 Automates, reconnaisseurs et transducteurs

Un automate est une machine informatique qui sert à gérer des informations à travers des états et la lecture de symboles (Winograd, 1983; Wehrli, 1997). Il est doté d'une tête de lecture et d'un ensemble fini d'états. Il déplace sa tête de lecture et/ou change d'état d'après des fonctions de transition d'un état à un autre qui prennent en compte la lecture d'un symbole donné ou l'état de certaines variables (Woods, 1980). Dans le cas des automates syntaxiques, les symboles lus sont les parties du discours (déterminant, nom etc.). Les états sont des étapes dans le processus d'analyse. L'état dans lequel se trouve l'automate au début du processus de reconnaissance est appelé état initial. Le ou les états finaux sont les états dans lequel l'automate doit se trouver à l'issue du processus de reconnaissance, afin qu'une phrase soit considérée comme correcte. Si aucune transition n'est disponible pour un symbole à un état donné, le processus échoue. Un *transducteur* est un automate qui produit une chaîne de sortie, qui représente par exemple l'analyse du mot ou de la phrase. Un *reconnaisseur* juge simplement la validité d'une entrée sans en fournir l'analyse. Les automates peuvent aussi servir d'analyseur morphologique (Keogh *et al.*, 2004, §3.1.1.4).

Les automates présentent l'avantage d'être une technique d'analyse relativement simple à implémenter et à maintenir. Ils s'avèrent particulièrement adéquats pour le traitement des phénomènes locaux. C'est pourquoi cette technique est très utilisée dans le domaine de l'ALIAO. Citons *Automated German Tutor* (Weischedel *et al.*, 1978, §B.4.8), *LADDER* (Hendrix *et al.*, 1978) pour l'interrogation de base de données, Kwasny et Sondheimer (1981), *ARCTA* (Kübler et Cornu, 1992; Kübler, 1995; Cornu *et al.*, 1996; Cornu, 1997, §B.4.5), *GPARS* (Loritz, 1992, 1995, §B.2.12), *ÉLÉONORE* (Renié et Chanier, 1996, §B.4.17), *Wo ist Hans?* (§B.3.12, Ward *et al.*, 1999) et *Formes Cachées / Hidden Shapes* (§B.5.7).

Par contre, nous pouvons remarquer que les automates ne sont pas efficaces pour une large couverture de la langue. Le traitement des dépendances à longue distance est malaisé. C'est pourquoi les outils décrits ici couvrent surtout des erreurs locales. Les automates sont aussi utiles pour une analyse par morceaux (§3.3.4.3), notamment après une phase d'étiquetage (§3.1.2) de la phrase à analyser.

3.3.4.2 Grammaires d'unification et Definite-Clause Grammars

Les grammaires d'unification sont basées sur des descriptions fonctionnelles (*functional description*, FD) contenant des séries d'attributs et valeurs, appelées traits (Kay, 1984, 1985, 1992). L'opération d'unification consiste à assembler deux FD pour en former une troisième, pour autant que les traits soient compatibles : l'intersection des valeurs des traits de ces objets doit former un ensemble non nul. Les catégories lexicales, fonctionnelles et syntaxiques apparaissent côte à côte dans les matrices. La figure (3.5) donne une analyse simple illustrant ce formalisme.

$$\begin{bmatrix} \text{cat} & \text{S} & & \\ \text{sujet} & \begin{bmatrix} \text{cat} & \text{N} \\ \text{nombre} & \text{sg} \\ \text{pers} & 3 \\ \text{det} & \text{def} \\ \text{tête} & \text{chien} \end{bmatrix} & & \\ \text{pred} & \begin{bmatrix} \text{cat} & \text{V} \\ \text{nombre} & \text{sing} \\ \text{pers} & 3 \\ \text{temps} & \text{présent} \\ \text{tête} & \text{court} \end{bmatrix} & & \end{bmatrix}$$

FIG. 3.5 – *Grammaires d'unification – analyse de la phrase* "le chien court."

Les traits du sujet et du prédicat sont identiques. Ainsi, l'unification peut être effectuée. Kay (1985) décrit un algorithme d'analyse basé sur le langage LISP (Gazdar et Mellish, 1989) [41].

Les DCG (*Definite-Clause Grammars*) sont une généralisation des grammaires indépendantes du contexte (§3.3.2.1) et permettent d'écrire facilement une grammaire et d'analyser des phrases en Prolog [42] (Pereira et Shie-

41. LISP (*List Processing*) est un langage de programmation basé sur des listes, très utilisé en intelligence artificielle (§2.7.4.1) et en analyse syntaxique.
42. Langage de programmation logique basé sur les prédicats logiques du premier ordre.

ber, 1987; Covington, 1994). Voici quelques règles simples :

(19) a. $sentence(s(NP,VP)) \rightarrow nounPhrase(NP),verbPhrase(VP)$.
b. $nounPhrase(np(D,N)) \rightarrow det(D),noun(N)$.
c. $verbPhrase(vp(V,NP)) \rightarrow verb(V),nounPhrase(NP)$.
d. $det(d(le)) \rightarrow [le]$.

Les règles de grammaire sont parfois codées accompagnées de listes de traits qui doivent s'unifier pour aboutir à une analyse complète. Les DCG peuvent donner lieu à des vérifications de traits extrêmement complexes. Dans des cas simples, une grammaire peut être écrite et testée très rapidement, ce qui explique la très grande popularité du formalisme. Voici quelques logiciels utilisant ce formalisme : le logiciel *Spion* et le correcteur *Syncheck* (Sanders et Sanders, 1987, §B.3.9), *VINCI* (Levison et Lessard, 1996, §B.2.38), *VP*^{2} (Schuster, 1986, §B.2.40), *LINGER* (Yazdani et Uren, 1988, §B.5.12), etc.

Remarquons la grande variété de systèmes qui utilisent les DCG. Comme nous l'avons déjà dit, le système séduit par sa simplicité, du moins lorsque l'on implémente une grammaire basique. De plus, de nombreux formalismes grammaticaux sont implémentés grâce aux DCG.

3.3.4.3 Analyse par morceaux ou analyse superficielle

L'analyse superficielle ou par morceaux (*chunk parsing* ou *shallow parsing*) consiste à ne repérer que certaines relations syntaxiques simples et locales et non pas tous les attachements et dépendances à longue distance. Blache (2005) définit un analyseur superficiel comme un outil qui fournit une structuration simple et non récursive. On peut définir également une analyse superficielle comme une tête autour de laquelle se trouvent des mots-fonctions qui correspondent à des schémas prédéterminés. Les verbes ne sont pas rattachés aux compléments, sauf les clitiques.

Selon Tschichold (2003), au lieu d'utiliser des techniques de TAL utilisant une analyse profonde qui conduit à des analyses erronées et à des résultats préjudiciables pour les apprenants, il devrait être moins risqué d'utiliser une analyse très locale. La maîtrise de petites unités est plus facile pour les apprenants, alors que les combinaisons et les attachements de syntagmes sont guidés par des sélections lexicales qui varient beaucoup d'une langue à l'autre.

Prolog est notamment utilisé pour l'analyse grammaticale et pour des systèmes experts d'intelligence artificielle (§2.7.4.1).

De plus, l'étendue du lexique nécessaire pour une application d'ALAO est plutôt faible (3000 mots).

Voici quelques logiciels utilisant une analyse superficielle : *Herr Kommissar* (DeSmedt, 1995, §B.3.5), *Cordial* (§C.8, Laurent, 1999; Campione *et al.*, 2005), *Didalect* (Hermet *et al.*, 2006, §B.2.6) et *LIPSTIC* (Pankhurst, 2005, §B.5.13).

Pour conclure, il peut sembler raisonnable d'utiliser des techniques robustes, qui ont par essence une large couverture de la langue vu qu'elles ne visent pas à l'exhaustivité des constructions, au lieu de formalismes grammaticaux complexes, sensibles aux erreurs et aux constructions qu'ils ne couvrent pas. Toutefois, il manque, à notre connaissance, des études sur l'efficacité de ces approches pour la détection d'erreurs et sur la qualité de rétroaction qui peut être atteinte à travers ces analyses moins riches, qui donnent donc moins d'indices sur les erreurs et les relations entre syntagmes.

3.3.5 Formalismes

Dans cette section, nous survolons quelques formalismes grammaticaux intéressants pour leur large couverture de la langue. La théorie du *Gouvernement et du Liage* (*Government & Binding*, GB) est issue des théories de la grammaire générative de Noam Chomsky (Chomsky, 1957, 1981; Tellier, 1995; Laenzlinger, 2003). Elle se base sur le schéma X-barre (fig. 3.6), où chaque catégorie lexicale (D, Adj, N, V, etc.) ou fonctionnelle (I, F) [43], en position X^0, est la tête d'une projection maximale XP (NP, DP, VP etc.), ce qui implique que ce formalisme est très lexicalisé. D'autres projections peuvent être attachées en position Spéc(ifieur) et Comp(lément) pour constituer finalement une phrase complète. Les niveaux XP et $\overline{X}$peuvent être dédoublés pour permettre les adjonction. Pour les langues latines, l'adjonction est également possible au niveau X^0 pour les clitiques. La théorie GB postule l'existence d'une structure profonde qui représente les structures fondamentales d'une phrase ; les constituants d'une phrase sont ensuite déplacés vers leur position finale en structure de surface. Ces mouvements permettent notamment l'interprétation des anaphores et des pronoms interrogatifs et relatifs, qui constituent les relations à longue distance, ainsi que le mouvement du passif. De plus, les rôles thématiques indiquent les types sémantiques des arguments des verbes (Agent, Thème, But, Intrument, Bénéficiaire, etc.).

43. I pour *Inflection* (avec les verbes conjugués) et F pour *Function* qui reçoit les propositions réduites du type *Je crois [cette décision inévitable].*

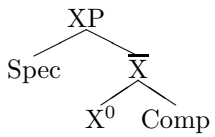


FIG. 3.6 – *GB – le schéma X-barre*

Voici quelques exemples d'analyse GB :

(20) a. $[_{\texttt{DP}}\ [_{\overline{\texttt{D}}}\ [_{\texttt{D}}$ l' $[_{\texttt{NP}}\ [_{\overline{\texttt{N}}}\ [_{\texttt{N}}$ étudiant] $[_{\texttt{PP}}$ en linguistique]]]]]]

b. $[_{\texttt{DP}}\ [_{\overline{\texttt{D}}}\ [_{\texttt{D}}$ l' $[_{\texttt{NP}}\ [_{\texttt{NP}}\ [_{\overline{\texttt{N}}}\ [_{\texttt{N}}$ étudiant]]]] $[_{\texttt{PP}}$ dans le parc]]]]

c. $[_{\texttt{IP}}\ [_{\texttt{DP}}$ les $[_{\texttt{NP}}$ Américains]] $[_{\overline{\texttt{I}}}$ torturent$_i$ $[_{\texttt{VP}}\ [_{\overline{\texttt{V}}}\ [_{\texttt{V}}$ t]$_i$ $[_{\texttt{DP}}$ leurs $[_{\texttt{NP}}$ prisonniers]]]]]]

d. $[_{\texttt{IP}}\ [_{\texttt{DP}}$ les $[_{\texttt{NP}}$ prisonniers]]$_i$ $[_{\overline{\texttt{I}}}$ sont $[_{\texttt{VP}}$ torturés $[_{\texttt{DP}}$ t]$_i$ $[_{\texttt{PP}}$ par $[_{\texttt{DP}}$ les $[_{\texttt{NP}}$ Américains]]]]]]

e. $[_{\texttt{CP}}\ [_{\texttt{PP}}$ A qui$_i^{+Wh}$] $[_{\overline{\texttt{C}}}\ [_{\texttt{DP}}$ Thierry$_j$] $[_{\overline{\texttt{C}}}\ [_{\texttt{C}^0}$ a$_k^{+Wh}$] $[_{\texttt{IP}}\ [_{\texttt{DP}}$ -t-il]$_j$ $[_{\overline{\texttt{I}}}\ [_{\texttt{I}^0}$ t]$_k$ $[_{\texttt{VP}}\ [_{\overline{\texttt{V}}}\ [_{\overline{\texttt{V}}}\ [_{\texttt{V}^0}$ payé] $[_{\texttt{DP}}$ une bière]] $[_{\texttt{PP}}$ t]$_i$]]]]]]]

f. $[_{\texttt{IP}}\ [_{\texttt{DP}}$ Catherine] $[_{\overline{\texttt{I}}}\ [_{\texttt{D}}$ l']$_i$ a $[_{\texttt{VP}}\ [_{\texttt{DP}}$ t]$_i$ $[_{\overline{\texttt{V}}}$ sermonnée $[_{\texttt{DP}}$ t]$_i$]]]]

En (20a), *en linguistique* précise le sens d'*étudiant* ; il peut être considéré comme faisant partie de sa sous-catégorisation et est attaché comme complément. En revanche, en (20b), le complément de lieu est une adjonction au NP. En (20c), le verbe conjugué est déplacé en position I^0 par le mouvement du verbe et une trace est insérée en V^0. En (20d), en contraste avec la phrase précédente, on présente le mouvement du passif, où le sujet de la phrase est liée à sa position canonique en complément du VP. En (20e), on illustre le mouvement des interrogatives avec inversion complexe du sujet. Enfin, en (20f), on montre le double mouvement des clitiques (Sportiche, 1996), où le DP repris par le clitique est dans un premier temps déplacé en spécifieur du VP, où a lieu l'accord du participe ; puis la tête est déplacée de D^0 en I^0.

Parmi les logiciels utilisant GB, on citera *BRIDGE* et de son successeur *MILT* (Garman *et al.*, 1993, §B.3.3), le projet *Athena* (Malone et Felshin, 1991, §B.3.1) ou FreeText (Vandeventer Faltin, 2003, §4).

La *Grammaire Lexicale Fonctionnelle* (*Lexical Functional Grammar, LFG*; Kaplan et Bresnan, 1982; Abeillé, 1993) décrit les phrases avec un couple de

structures, une *structure de constituants* (c-structures) sous forme d'arbres syntaxiques et une *structure fonctionnelle* (f-structures) sous forme de traits représentant les fonctions grammaticales (sujet, objet, etc.). Les f-structures sont des couples attribut-valeur représentant des traits sémantiques et syntaxiques et peuvent être incluses dans d'autres f-structures.

La figure (3.7) illustre une phrase simple avec l'équation qui unifie les traits du NP avec ceux du sujet de la phrase ; les flèches désignent l'unification des traits des nœuds avec ceux des niveaux inférieurs et supérieurs. Le passif est considéré comme une nouvelle règle fonctionnelle.

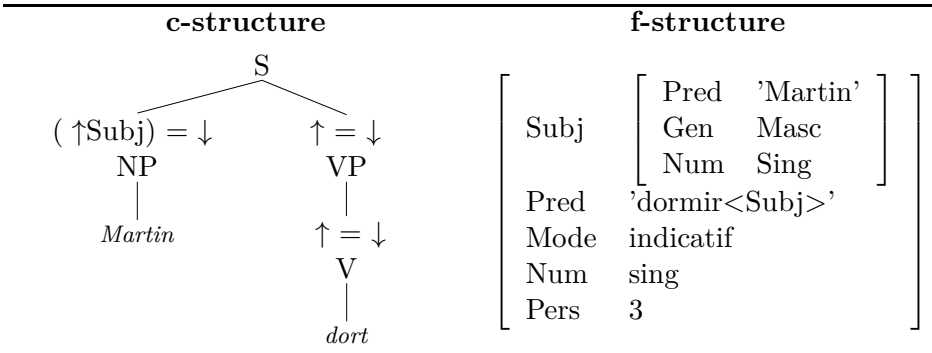


FIG. 3.7 – *LFG – Analyse d'une phrase*

Comme exemples de systèmes, citons Reuer (1999, §B.2.29), *MSLFG* (Cornu, 1997, §C.25), *CALLE* (Feuerman *et al.*, 1987, §B.4.11), *ALICE-chan* (Levin et Evans, 1995, §B.4.1) et *GETARUNS* (Delmonte et Dibattista, 2000; Delmonte, 2003, §B.2.10).

La *Grammaire Syntagmatique Guidée par les Têtes* (*Head-driven Phrase Structure Grammar*, HPSG; Pollard et Sag, 1994) est caractérisée par l'intégration des niveaux phonétique, syntaxique, sémantique et discursif dans une même structure de traits. Les structures de traits sont représentées sous la forme de matrices attribut-valeur (*attribute-value matrixes*, AVMs), qui contiennent des informations phonétiques, syntaxiques et sémantiques. La structure contenant ces informations est appelée *signe linguistique*. Les traits sont organisés de manière très hiérarchique. Les lexèmes sont par exemple classés en lexèmes flexionnels ou sans flexion, les noms sont décomposés en noms communs, pronoms et noms propres, etc. Chacun de ces types permet ensuite de définir des règles communes sous forme de contraintes. Le passif est une règle lexicale comme pour la LFG. Pour les relations à distance, les traits sont propagés à travers les structures. Comme exemples de systèmes basés sur ce formalisme, mentionnons *Textana* (Schulze, 1995, §B.4.35) et les systèmes de Heift et McFetridge (1999, §B.4.23).

Enfin, les grammaires de dépendance décrivent les relations entre mots, autrement dit les dépendances des mots les uns par rapport aux autres (Tesnières, 1959; Kahane, 2001). Ainsi, les mots de la phrase ne seront pas représentés de manière ordonnée. Les grammaires de dépendance consistent à associer une tête à un autre élément par un arc portant une étiquette qui définit une relation syntaxique appartenant à un ensemble fini de relations comme sujet, objet direct, auxiliaire, etc. (Schröder *et al.*, 2000). Mentionnons comme exemples d'utilisation des grammaires de dépendances les correcteurs commerciaux *Antidote* (Brunelle, 2004, §C.3) et *Correcteur 101* (§C.9) et les logiciels de Menzel et Schröder (1998b, §B.3.6).

3.3.6 Discussion

Cette section conclut la partie sur la syntaxe et la détection d'erreurs en esquissant un bilan et en proposant quelques perspectives.

Dans cette partie sur l'analyse, nous avons survolé de nombreuses techniques. Les applications présentées diffèrent au niveau de la profondeur d'analyse et de la couverture de la langue. Ainsi, certains systèmes présentent de bas taux d'erreurs sur un champ syntaxique et lexical réduit. Inversement, il est difficile de produire des applications à large couverture.

Cependant, la valeur pédagogique des systèmes se mesure essentiellement aux bénéfices d'apprentissage pour les apprenants. Ainsi, une analyse profonde et détaillée d'une phrase permet de fournir une rétroaction plus précise et de donner plus de détail sur les erreurs commises ou de présenter les notions de manière ludique (Kempen, 1992, §B.2.15). Par exemple, on peut comparer le système de Heift (2003, §B.4.23), qui fournit une analyse progressive et détaillée des erreurs, à *VINCI* (Levison et Lessard, 2004, §B.2.38), dont le diagnostic se borne à une comparaison mot à mot entre une phrase de référence et celle de l'apprenant, avec une analyse très limitée de l'erreur.

En outre, on peut également remarquer que la plupart des projets reprennent un analyseur existant et en adaptent la grammaire (Heift et Schulze, 2007). Ce fait s'explique en partie par les coûts élevés de développement. Il faut développer un algorithme d'analyse, construire des règles, acheter ou créer un lexique et enfin tester l'analyseur à l'aide d'un corpus de test. En ajoutant les résultats mitigés des performances et les fausses détections, il n'est pas étonnant que le nombre de projets soit plutôt limités.

Selon Bowerman (1990), les apprenants ont des notions grammaticales

vagues tels que sujet, objet et ordre des mots. De telles notions sont vagues et impossibles à traiter de manière informatique. Au contraire, les formalismes grammaticaux comme GB ou LFG (§3.3.5) sont adaptés au traitement informatique mais font appel à des notions difficiles à mettre en œuvre pour des apprenants. C'est pourquoi, selon lui, il est nécessaire de décrire des formalismes compréhensibles pour les apprenants tout en ayant une capacité générative qui permette le traitement informatique.

Suite à l'analyse de la couverture de nombreux logiciels, nous pouvons affirmer que l'approche syntaxique est indispensable à la détection d'erreurs. L'approche stochastique (§3.3.3.3) peut donner des résultats sur certaines construction très locales, où la distribution des éléments et leur pourcentage d'occurrences est important et où un apprentissage automatique sur corpus est possible. De telles techniques peuvent être utilisées en complément avec des méthodes par règles. En outre, nous pouvons remarquer un grand nombre d'analyseurs basés sur LFG (§3.3.5), HPSG (§3.3.5), les DCG (§3.3.4.2) et les grammaires de dépendances (§3.3.5). Nous partageons avec Heift et Schulze (2007) le constat que les meilleurs résultats sont atteints par des analyses à dominante lexicale, qui traitent les dépendances à longue distance. Ces analyses profondes permettent également un diagnostic efficace.

Cependant, les résultats mitigés de la correction grammaticale ne permettent pas encore son utilisation à grande échelle, selon certains spécialistes de l'ALAO. Ainsi, Tschichold (1999a,b) souligne le risque de voir des phrases grammaticalement correctes rejetées grâce au relâchement des contraintes et des phrases incorrectes acceptées. Elle en conclut que les performances des correcteurs sont encore insuffisantes et que les indications qu'ils fournissent sont trompeuses, voire carrément fausses. Elle propose donc de se contenter d'une aide sensible au contexte et de disposer à côté d'outils d'aide à l'apprentissage : grammaire de référence, dictionnaires bilingues et monolingues, corpora de textes adaptés à l'étude en contexte, etc. De plus, Tschichold (1999a) propose d'améliorer les performances des analyseurs en bloquant certaines entrées lexicales peu probables comme le verbe *to table*, qui est moins probable que le substantif *table*. De même, Tschichold (2006) fait remarquer que les techniques de TAL sont très sensibles à l'ambiguïté et que seules les applications tournant sur un vocabulaire et des structures restreints atteignent un bon taux de précision. Or le vocabulaire et les structures enseignés aux apprenants sont volontairement peu précis et ambigus [44].

Holland *et al.* (1993, p. 32) regrettent que la sémantique ne soit pas

44. On apprend notamment aux apprenants à utiliser des paraphrases pour pallier leurs lacunes en vocabulaire.

suffisamment développée, ce qui oblige les logiciels d'ALAO à se concentrer sur la forme plutôt que sur le fond, alors que l'approche pédagogique qui sous-tend la plupart des logiciels es tournée vers la communication de sens.

Par contre, Nerbonne (2003, p. 691) affirme que les utilisateurs sont intelligents et peu exigeants et peuvent bénéficier d'une technologie qui n'est pas fiable à 100%. Durel (2006) ajoute que lors d'une analyse partielle, il est important de le signaler aux apprenants, qui se méfieront alors davantage des indications données par le système. Pour *ISCA* (§B.5.12), Bolt et Yazdani (1998) propose une correction interactive.

De notre côté, nous estimons qu'il serait optimal d'allier une restriction du champ de l'analyse à une désambiguïsation interactive (Menzel, 2004). Ainsi, l'apprenant devrait pouvoir choisir entre plusieurs possibilités, aidé d'outils morphologiques et lexicaux. Diverses aides, visuelles ou sous forme de dialogues, peuvent être envisagées.

Par ailleurs, force est de constater que les composantes sémantiques des grammaires ne sont pas suffisantes pour une véritable évaluation automatique de la réponse. *De facto*, la plupart – si ce n'est la totalité – des vérificateurs grammaticaux que nous avons étudiés ici se limitent à une vérification grammaticale *stricto sensu*, sans s'attaquer au sens de la phrase. Les traits sémantiques sont tout juste suffisants pour une vérification rudimentaire de la cohérence d'un énoncé. Les expressions idiomatiques (v. p. 66) ne sont pas prises en considération. En outre, la vérification sémantique implique un lexique complet et consistant, ce qui est loin d'être garanti. Enfin, comme la syntaxe n'étudie que la phrase, la cohérence du discours n'est pas vérifiable. Dès lors, il n'est envisageable d'utiliser une vérification de la cohérence sémantique que sous la forme d'avertissement, qui signale par exemple que le système attendait un objet comestible pour le verbe *ronger* dans le cadre de l'expression *ronger son frein*. L'apprenant serait alors incité à utiliser des aides à l'apprentissage telles qu'un dictionnaire pour vérifier que sa production est correcte.

Pour conclure, nous pouvons remarquer une recrudescence des projets incluant des composantes de traitement du langage depuis quelques années. Heift et Schulze (2007) recensent environ soixante-dix projets dans la décennie 1985–1995 et seulement trente pour la décennie suivante ; actuellement, les auteurs constatent une recrudescence des projets. Parmi les causes de ce renouveau, on peut citer les progrès effectués en linguistique computationnelle et les performances accrues des ordinateurs, au niveau de la puissance de calcul, de la taille de la mémoire et de la taille et de la rapidité d'accès aux disques.

3.4 Formalismes sémantiques

Après avoir abordé la grammaire et l'orthographe, nous passons maintenant à une description de quelques formalismes sémantiques qui couvrent différents domaines. Nous essayons de voir en quoi la représentation du sens de la phrase peut contribuer à l'évaluation des productions des apprenants. Dans cette section, nous décrivons des formalismes sémantiques qui ont des portées et des caractéristiques différentes. Au §3.4.1, nous parlons de la *Segmented Discourse Representation Theory* (SDRT), qui est une théorie de représentation du discours. Ensuite, nous abordons deux formalismes sémantiques qui sont utilisés dans le domaine de la traduction automatique : les *Lexical Conceptual Structures* (LCS, §3.4.2) et les *Quasi-Logical Forms* (QLF, §3.4.3). D'autres familles de formalismes sont brièvement évoqués (§3.4.4) et une discussion générale termine la section (§3.4.5).

3.4.1 Segmented Discourse Representation Theory (SDRT)

La *Théorie de la Représentation Discursive Segmentée* (*Segmented Discourse Representation Theory*, SDRT) propose une théorie de représentation du discours qui intègre également une composante sémantique (Asher, 1993; Asher *et al.*, 1994; Asher et Lascarides, 2003). La SDRT étend la *Discourse Representation Theory* (DRT, Kamp, 1981) en ajoutant à ce formalisme des relations discursives.

Les éléments de base de la SDRT sont les DRSs (*Discourse Representation Structures*), empruntées à la DRT. Une DRS est une paire groupant des variables de discours et des conditions :

(21) $\mathrm{K} = < U_k, Con_k >$

- U_k est l'ensemble des référents du discours, comprenant :
  i. des individus, représentés par des variables x, y, z, etc. Les individus peuvent être des personnes ou des objets ;
  ii. des événements e_1, e_2, e_3, etc. ;
  iii. un moment d'énoncé de la parole n.
- Con_k est l'ensemble des conditions qui spécifient les référents du discours et leurs relations. Il s'agit des prédicats appliqués aux individus, ce qui définit un événement.

Examinons maintenant le court discours de l'exemple (22) :

(22) a. Jean-Philippe décolla de l'observatoire.
b. Il atterrit à Collonges.
c. Là, Céline l'attendait avec leurs enfants, Benjamin et Naëlle.

La figure (3.8) représente les DRS qui correspondent à ces phrases, sous forme schématique [45].

(22a) K_1

x, y, e_1, n
Jean-Philippe(x)
observatoire(y)
e_1: décoler-de(x,y)
$e_1 < n$

(22b) K_2

x, z, v, e_1, e_2, n
z = x
Collonges(v)
e_2: atterrir-à(z,v)
$e_2 < n$
$e_1 < e_2$

(22c) K_3

z', w, t, u, s, e_1, e_2, e_3, n
z' = x
t = v
Céline(w)
Benjamin(u)
Naëlle(s)
enfant-de(u⊕s, z'⊕w)
e_3: attendre(w⊕u⊕s, z')
dans(e_3, t)
$e_3 < n$
$e_3 < e_2$

FIG. 3.8 – *SDRT – DRSs correspondant aux phrases de l'exemple (22).*

Les DRS sont combinées à un ensemble (potentiellement vide) de relations de discours (*Discourse Relations, DR*) pour former une SDRS. Chaque constituant d'une SDRS est relié à un autre par une DR. La construction de la SDRS d'un texte est dynamique. Au fur et à mesure, de nouvelles DRS sont ajoutées et combinées aux autres. Une DRS peut être rattachées à plusieurs autres par des DRs, et plusieurs DRs peuvent relier deux mêmes SDRSs. Voici quelques DRs possibles selon Lascarides et Asher (1991, p. 263ss) et Asher (1993) ; les exemples sont inspirés de Matthiesen et Thompson (1988) et Dahlgren (1988).

- Narration(α,β). α précède β. Exemple : *Sébastien a bu beaucoup de bières. Il a chuté de vélo.*
- Explanation(α,β). β précède α et β cause α. Exemple : *Sébastien a chuté de vélo. Il a bu beaucoup de bières.*

45. Le symbole ⊕ dénote un regroupement d'individus.

- Elaboration(α,β). β est une partie de α. Exemple : *Le stade de Genève a été construit à coup de dizaines de millions. Les collectivités publiques ont dépensé de fortes sommes.*
- Background(α,β). α est un événement et β décrit les circonstances dans lesquelles il s'est produit. Exemple : *Sébastien a chuté de vélo. Il faisait nuit noire.*
- Result(α,β). L'événement α a causé l'événement ou l'état β. Exemple : *Bush a prononcé son discours. Le public a hué.*

Reprenons l'exemple (22) p. 126. L'événement noté par la phrase (22b) est clairement postérieur à (22a) et est une suite logique. On peut donc noter la relation Narration(K_1, K_2). Ensuite, la phrase (22c) reprend un événement antérieur. On pourra alors noter Background(K_2, K_3).

Avec la SDRT, il est ainsi possible d'obtenir une description abstraite d'un discours. Les relations de discours sont marquées de manière riche et non ambiguë. Les structures sont relativement simples et peuvent facilement être traitées dans des applications informatiques. En disposant de SDRS, on peut résoudre les ambiguïtés des anaphores. La traduction automatique (§3.1.6) pourrait être grandement facilitée et améliorée. Parmi d'autres applications possibles, citons encore le résumé automatique de textes (§3.1.8), la recherche avancée dans un corpus ou une base de données documentaire (§3.1.7), etc. Malheureusement, il est peu aisé d'extraire de telles structures dans l'état actuel des outils de traitement du langage et les applications existantes ont une couverture limitée (Amsili et Hathout, 1998; Danlos *et al.*, 2001; Danlos et El Ghali, 2002; Prévot *et al.*, 2002; Xuereb et Caelen, 2004; Asher *et al.*, 2004).

Dans le domaine de l'ALIAO, la SDRT pourrait être utilisée dans des applications orientées vers l'étude de textes. Le public cible serait alors plutôt de niveau moyen à avancé. Certains mots ou certaines expressions destinées à construire le discours pourraient être mises en évidence. On peut imaginer aussi des graphes dynamiques montrant l'organisation du texte, ou de demander à l'apprenant de construire ou de corriger de tels graphes.

Par ailleurs, les enseignants aiment produire des exercices à partir de textes divers et authentiques, comme des textes d'actualité tirés d'*Internet*. La création d'exercices est assez fastidieuse, même sans utiliser d'outils informatiques. Les SDRS pourraient être intégrées dans des outils automatiques ou semi-automatiques de création d'exercices, afin de marquer des relations du discours et la structuration de celui-ci. La diversité et l'actualité des textes serait alors un facteur de motivation pour les apprenants.

Enfin, la correction de petits textes produits par les apprenants pourrait être améliorée. A côté des erreurs de grammaire et d'orthographe, il serait possible de vérifier certains phénomènes de cohérence, notamment dans l'utilisation de connecteurs de discours.

Néanmoins, la SDRT manque encore d'applications informatiques à large échelle et les quelques pistes esquissées ici relèvent encore de l'utopie. Avant d'appliquer la SDRT à l'enseignement des langues, il faudrait disposer d'outils capables d'extraire des SDRS de textes grammaticalement corrects et suffisamment complexes.

3.4.2 Lexical Conceptual Structures (LCS)

Les *Structures Lexicales Conceptuelles* (*Lexical Conceptual Structures*, LCS; Jackendoff, 1975, 1983, 1990; Dorr, 1990, 1994) sont des représentations conceptuelles abstraites du sens des mots et des phrases. Elles se basent sur des structures stockées dans un lexique et construisent une structure abstraite indépendante des structures syntaxiques. Elles forment notamment la base du système de traduction automatique *UNITRAN* (Dorr, 1990, 1994, §3.1.6).

Les LCS sont basées sur des primitives (cause, permission, orientation etc.) de différents types (événements, états, position spatiale etc.) et restreintes par différents champs marquant le temps, le lieu etc. La principale difficulté causée par cette approche est la taille du lexique. En effet, il faudrait coder tous les mots des classes ouvertes de la langue pour avoir une couverture complète d'une langue. Considérons l'exemple (23) :

(23) a. I stabbed John
Je blesser-avec-arme-tranchante-passé-1s Jean

b. Yo le di puñaladas a Juan
moi clitique donner-passé-1s coup-de-poignard à Jean

c. 'J'ai poignardé Jean'

Ces trois langues utilisent des structures totalement divergentes. Néanmoins, les trois phrases auront la même LCS donnée en (24). L'action de poignarder est caractérisée par le fait d'être la cause qu'une personne pos-

sède une blessure due à un couteau [46].

(24) [$_{Event}$ CAUSE ([$_{Thing}$ I], [$_{Event}$ GO$_{Poss}$ ([$_{Thing}$ KNIFE-WOUND], [$_{Path}$ TOWARD$_{Poss}$ ([$_{Path}$ AT$_{Poss}$ ([$_{Thing}$ KNIFE-WOUND], [$_{Thing}$ John])])])])]

Le formalisme LCS est aussi utilisé dans le domaine de l'enseignement des langues pour le projet *MILT* (§B.3.3), qui utilise un formalisme GB (§3.3.5). La LCS d'une phrase produite par un apprenant est comparée à la réponse attendue par le concepteur de l'exercice (Dorr *et al.*, 1995).

Considérons les deux phrases suivantes, données par Dorr *et al.* (1995).

(25) a. John ran to the house 'Jean a couru à la maison'
Jean courir-passé-3s à la maison

b. John went to the house 'Jean est allé à la maison'
Jean aller-passé-3s à la maison

Les LCS correspondantes sont données en (26) :

(26) a. [$_{Event}$ GO$_{Loc}$ ([$_{Thing}$ JOHN],[$_{Path}$ TO$_{Loc}$ ([$_{Position}$ AT$_{Loc}$ ([$_{Thing}$ JOHN],[$_{Property}$ HOUSE])]),[$_{Manner}$ RUNNINGLY])]]

b. [$_{Event}$ GO$_{Loc}$ ([$_{Thing}$ JOHN],[$_{Path}$ TO$_{Loc}$ ([$_{Position}$ AT$_{Loc}$ ([$_{Thing}$ JOHN],[$_{Property}$ HOUSE])]))]]

Si la réponse attendue est (25a) et que l'apprenant répond (25b), le système détectera qu'il manque [$_{Manner}$ RUNNINGLY] et qu'il n'y a pas d'éléments superflus. La réponse est donc incorrecte. A l'inverse, si la réponse attendue est (25b) et l'apprenant répond (25a), la réponse est acceptée, car l'apprenant n'a rien oublié mais ajouté un élément superflu. Un *feedback* pourra signaler l'oubli ou l'ajout à l'apprenant. Dorr (1997) décrit l'utilisation de LCS pour donner des ordres à l'agent de *MILT* (§B.3.3). Les phrases entrées au clavier par l'utilisateur sont ensuite transformées en LCS et comparée avec une réponse attendue.

46. Les primitives sont en majuscules. Les éléments en italiques après les primitives sont les champs qui restreignent les primitives. Quant aux types logiques, ils sont notés en indice après les crochets ouvrants.

Pour conclure, nous avons décrit ici le seul système d'ALAO à composante sémantique de cette envergure qui a été développé à notre connaissance. L'utilisation d'un micro-monde permet de restreindre suffisamment la couverture syntaxique. Ce système nécessite un lexique important et cohérent. Bien que nous n'avons trouvé aucune indication à ce sujet, il est à prévoir que le traitement d'une phrase nécessite un long temps de réponse. Remarquons également que le système ne traite pas le discours. Cependant, la bonne qualité du traitement des réponses est encourageant et pousse à poursuivre les recherches dans le domaine.

3.4.3 Quasi-Logical Forms (QLF)

Les *Formes Quasi-Logiques* (*Quasi Logical Forms*, QLF) sont des structures sémantiques abstraites utilisées par le système de traduction automatique *Core Language Engine* (Alshawi et van Eijck, 1989; Alshawi, 1992). Elles sont un ensemble sous-spécifié des *Logical Forms* (LF; van Eijck et Alshawi, 1992). Les QLF représentent donc un résultat intermédiaire, après une première phase d'analyse d'une phrase, indépendamment de l'influence du contexte. Les questions de portée et la résolution des références restent sous-spécifiées. Nous commençons par décrire les LFs pour ensuite décrire les QLF.

Les LF sont des formes logiques complètement spécifiées qui représentent une des significations possibles d'une phrase. Elles sont basées sur des prédicats et arguments, dans une notation appropriée pour un traitement informatique. Le formalisme reprend la notation de liste de Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Les variables commencent par une majuscule, les constantes par une minuscule. Une formule est notée sous la forme :

(27) `[<pred>, <arg1>, ..., <argn>]`

A côté d'opérateurs et connecteurs logiques de premier ordre, il existe des extensions d'ordre supérieur, marquant la modalité (déclaratives, interrogatives, etc.), les modaux, le temps, etc. Les *Quasi Logical Forms* (QLF, Alshawi, 1990) sont basées sur les LF et contiennent des constructions additionnelles pour les quantificateurs à portée non résolue et pour des termes et relations non résolus :

– Les expressions quantifiées à portée non résolue sont dénotées par `qterm(<cat>, <var>, <form>)`. Par exemple, on les utilise pour des

quantificateurs ambigus (*aucun assistant, chaque femme*) ou pour des expressions définies comme *la rectrice* ou *ces cinq policiers*.

- Les références non résolues sont dénotées par `a_term(<cat>, <var>, <form>)`. On les utilise pour les pronoms, les réfléchis et les expressions indexicales comme *aujourd'hui* ou *moi-même*.
- Les relations implicites sont dénotées par `a_form(<cat>, <var>, <form>)`. Il s'agit de complément du nom (*le doyen de la faculté*) et des ellipses non résolues (*il en a acheté une plus chère*).

Examinons maintenant l'exemple suivant :

(28) Elle a rencontré un ennemi de George.

```
[past, [imperf, [rencontrer,
    a_term(<t=ref, p=pro, l=elle, n=sing, a=<> >, Y, [femme, Y]),
    qterm(<t=quant, n=sing, l=un>, X,
        a_form(<t=pred, p=genit>, R
            [and, [ennemi, X], [R, George, X]]
        )
    )
]]].
```

FIG. 3.9 – *QLF – représentation de la phrase (28)*

La QLF de la figure (3.9), adaptée de Alshawi *et al.* (1991), commence par les traits de temps, suivi du prédicat. Un premier argument `a_term` dénote le pronom féminin *elle*, attribué à une *femme*, qui est définie par la variable Y. Le deuxième argument est une expression quantifiée indéfinie `q_term`, qui définit l'*ennemi* par la variable X. Cette `q_term` contient elle-même une expression relationnelle `q_form` qui introduit une troisième variable R pour *George*. *George* et *ennemi* sont associés par l'opérateur `and` qui indique qu'ils sont associé par une relation nom/complément (marqué par le trait `genit`).

On constate que la structure des QLF est riche et permet de dénoter les relations entre les éléments porteurs de sens de la phrase (verbe, noms, adjectifs, adverbes, les catégories ouvertes de la langue) en attribuant aux catégories fermées une valeur plus abstraite. De telles structures peuvent être dérivées indépendamment du formalisme syntaxique utilisé. Enfin, il faut noter que l'ordre des arguments est primordial, car la fonction des membres du prédicat n'est pas spécifiée.

Alshawi et Carter (1994) décrivent une application des QLF pour une tâche de désambiguïsation. A notre connaissance, il n'existe aucune application d'ALIAO qui utilise les QLF. Cependant, ce formalisme est suffisamment abstrait et général pour envisager des applications comme celles décrites pour les LCS.

3.4.4 Autres formalismes

Passons maintenant à d'autres formalismes sémantiques utilisés dans des logiciels d'ALAO. Pour son système *ILTS* (§B.4.25), Schwind (1988) décrit des règles simples basées sur des prédicats et des compléments, qui permet de vérifier la cohérence sémantique basique des phrases des apprenants.

De son côté, la *Théorie de la Structure Rhétorique* (*Rhetorical Structure Theory, RST*, Mann et Thompson, 1987; Hovy, 1988; Roussarie, 1998; Sabah et Grau, 2000) permet de représenter la structure de textes, comme outil de planification pour la génération de textes (§3.1.5). Elle peut être rapprochée dans une certaine mesure de la SDRT (§3.4.1). Un texte est découpé en unités reliées par une structure arborescente à l'aide de relations. Les relations sont appelées *Preuve*, *Antithèse*, *Justification*, *Concession*, *Condition*, *Circonstance*, *Motivation* etc. Rizzo *et al.* (2002) utilisent aussi la RST dans un outil intelligent d'aide à l'apprentissage de l'écriture de textes structurés pour enfants locuteurs natifs de l'anglais.

Herr Kommissar (§B.3.5 DeSmedt, 1995) est bâti sur un formalisme sémantique *Knowledge Representation System* (*KRS*). Les KRS sont un langage de haut niveau qui permet d'encoder les connaissances de manière explicite (Van Marcke, 1987). Ces structures se basent sur une ontologie de 2 500 *concepts* organisés hiérarchiquement et reliés entre eux par des relations unidirectionnelles appelées *sujets*. Les concepts sont basés sur une architecture orientée objet qui permet un mécanisme d'héritage. Le système *Alice* (§B.4.1, Cerri *et al.*, 1992) utilise aussi les KRS.

Delmonte (2004a) décrit un système sophistiqué d'interprétation du discours pour le système *GETARUNS* (§B.2.10), basé sur la *Sémantique Situationnelle*. Le système est capable de faire des interprétations temporelles et d'inférer des indications spatiales. En outre, Frederiksen *et al.* (1992) utilisent une *sémantique relationnelle* dans le cadre d'un système d'ALIAO.

Pour le système *STASEL* (§B.5.19), Payette (1990) utilise un formalisme *frame statement* qui est dérivé d'une représentation sémantique du système *Absity* (Hirst, 1987). Les phrases sont décomposées en cadres représentant

différents rôles (agent, patient, lieu, provenance, etc.) ainsi que le type énonciatif (question). Payette (1990) adapte ce formalisme afin de décrire la structure stylistique d'une phrase.

SWIM (§B.4.33), Zock (1992) utilise un système de *graphes conceptuels* pour exprimer des idées et les soumettre à un générateur de phrases. Ces graphes représentent le sens sous forme de prédicats et arguments qui peuvent être reliés entre eux par des dépendances hiérarchiques.

Pour interagir avec le micromonde *SAMPRAS* (§B.3.7), Michel et Lehuen (2004) utilisent un *lexique conceptuel*. Comme le but est de réaliser une recette de cuisine, les mots sont classés selon leur fonction (ustensile, rangement, récipient, ingrédient, etc.). Le moteur d'inférences est implémenté en *Jess* (*Java Expert System Shell*)[47], qui est un système expert capable de raisonner selon des règles déclaratives sous forme de faits. Le système est capable de reconnaître l'action à effectuer (et de déterminer si les conditions pour l'effectuer sont présentes) ou de poser des questions pour la réaliser. Le système ne détecte pas les erreurs syntaxiques mais vérifie néanmoins la cohérence.

Didalect (Hermet *et al.*, 2006, §B.2.6) cherche à évaluer la compréhension de l'écrit d'apprenants du français en leur posant des questions sur le texte qu'ils ont lu. Le concepteur de questions doit sélectionner une à trois phrases du texte, selon les cas, qui contiennent la réponse à la question. Le système évalue les réponse en constituant trois listes de mots, une contenant les mots présents à la fois dans la réponse et dans la/les phrases modèle, les deux autres contenant les mots présents uniquement dans la phrase modèle ou dans la réponse de l'apprenant. Le dictionnaire des synonymes permet de détecter les relations de synonymie, antonymie, hyperonymie, hyponymie, etc. Le système utilise aussi une analyse par morceaux de la phrase (§3.3.4.3). L'évaluation sémantique de la réponse consiste à évaluer la présence de constituants importants : les ajouts temporels sont considérés comme facultatifs, les adverbes sont moins importants que les adjectifs etc.

Pour un logiciel d'apprentissage du basque (§C.37), Díaz de Ilarraza *et al.* (1998) prônent une structure d'interlangue (§3.1.6) pour modéliser la connaissance des apprenants et certains phénomènes courants d'erreurs de déviation morphologique. Wilks et Farwell (1992) proposent d'utiliser une interlangue dans un logiciel d'ALIAO. Ils utilisent le formalisme de la sémantique préférentielle (*Preference Semantics*, Wilks, 1975; Wilks et Farwell, 1992; Wilks et Fass, 1992), qui consiste en un assemblage de 80 à 100 primitives qui décrivent des entités, des états, des qualités et des actions. Le

47. `http://herzberg.ca.sandia.gov/jess/`, consulté le 12 novembre 2004.

sens des mots est décrit par des assemblages de primitives appelés formules. Le but est de choisir une signification dans un ensemble de significations possibles. Il est possible d'éliminer certaines significations incompatibles avec le contexte. Ces mécanismes seraient évidemment très utiles dans le cadre de l'ALIAO.

3.4.5 Discussion

Nous remarquons qu'à quelques exceptions près, les systèmes présentés dans cette section utilisent des formalismes très simples, qui ne vont pas au-delà du niveau de la phrase. Quant aux formalismes plus élaborés, ils nécessitent de grands lexiques, qui ne sont malheureusement pas encore disponibles.

Parmi les logiciels présentés, *MILT* (§B.3.3) est certainement le logiciel le plus élaboré. Il date déjà de plus de 10 ans. Dans le cadre d'applications limitées, les formalismes sémantiques pourraient revenir au goût du jour, grâce aux progrès techniques des processeurs et des mémoires, qui raccourcissent le temps de traitement.

Par ailleurs, l'avenir se trouve peut-être dans l'application de méthodes purement stochastiques comme l'analyse sémantique latente (LSA, §3.1.8 p. 75). Cependant, elles sont utilisées dans le cadre de techniques de traduction ou de résumés et n'ont pas été utilisées pour corriger de relativement courtes productions.

3.5 Formalismes lexicaux

Le lexique est un instrument crucial de notre capacité à produire des énoncés. Il permet de formuler des idées et des concepts et de les associer à des mots. Dans le domaine de l'apprentissage des langues, l'acquisition de vocabulaire est l'un des problèmes centraux avec la maîtrise des structures grammaticales, puisque l'apprenant doit d'abord acquérir un certain nombre de mots, à l'écrit et à l'oral, avant de pouvoir exprimer des idées et communiquer.

Il existe de nombreuses formes de lexiques. On peut distinguer lexiques monolingues et bilingues ou plurilingues. L'information contenue dans les lexiques varie. On trouvera généralement des informations sur la partie du

discours et sur la sous-catégorisation syntaxique du mot (compléments du verbe, prépositions utilisées etc.), sur la morphologie (forme canonique et formes fléchies) et sur la phonétique (prononciation). La représentation du sens est délicate : à côté des définitions classiques des dictionnaires, il existe de nombreuses manières de représenter ces informations, dont nous présentons quelques exemples dans cette section.

Les langues divergent fortement au niveau de leurs manières d'exprimer les idées. Les mots ne recouvrent pas forcément les mêmes concepts. Là où nous utilisons un seul mot pour le concept de *neige*, les Inuits ont une vingtaine de termes, voire plus, ce qui s'explique aisément par le contexte où ils vivent. Le français, pour sa part, utilisera des adjectifs modifiant *neige*, comme *mouillée* ou *poudreuse*. Nous avons également déjà largement évoqué la problématique des collocations (p. 66).

Remarquons pour finir que les lexiques ne sont pas une technique de traitement automatique des langues à proprement parler. Ils sont plutôt des ressources qui permettent aux applications de fonctionner. En outre, à l'inverse, certaines techniques d'analyse syntaxique permettent d'extraire certaines informations comme les collocations. Enfin, l'analyse statistique des fréquences de mots est aussi une indication précieuse pour construire les lexiques, pour la traduction et même pour l'analyse syntaxique.

Dans cette section, nous étudions un formalisme puissant de description lexicale, le *Lexique Génératif* (§3.5.1). Nous poursuivons avec les aspects sémantiques de la *Théorie Sens-Texte* (§3.5.2). Ensuite nous abordons un réseau sémantique *WordNet* (§3.5.3). Puis nous décrivons brièvement quelques autres dictionnaires (§3.5.4). Enfin, nous terminons cette section par une discussion générale (§3.5.5).

3.5.1 Le Lexique Génératif

Le lexique génératif (*Generative Lexicon*, Pustejovsky, 1995) présente le lexique comme un ensemble dynamique, où le sens des mots change en fonction du contexte. Pustejovsky postule l'existence d'un ensemble-noyau de sens des mots, qui peuvent être combinés entre eux en syntagmes et clauses afin de former de nouveaux sens. Contrairement aux LCS (§3.4.2), ce noyau n'est pas basé sur des primitives, ni sur des relations entre mots. Chaque mot est constitué par une structure argumentale, une structure événementielle, des relations d'héritage lexical et des *structures de qualia* (*Qualia structure*), qui représentent les différents modes de prédication d'un élément lexical et les relations arguments et événements. Ces structures sont décrites comme "*a*

system of relations that characterize the semantics of nominals, very much like the argument structure of a verb" (Pustejovsky et Boguraev, 1993, p. 204). Ces structures comportent quatre rôles. Toute catégorie comprend une structure de qualia, mais tous les éléments lexicaux n'ont pas de valeur pour chacun des rôles. :

- *rôle constitutif* (*constitutive*) : relation entre un objet et ses parties constitutives. Ici, on trouve notamment les matériaux, le poids, les parties et composantes.
- *rôle formel* (*formal*) : propriété qui distingue l'élément dans un domaine plus grand. On trouve ici l'orientation, la magnitude, la forme, la dimensionnalité, la couleur et la position.
- *rôle télique* (*telic*) : but et fonction de l'objet. On trouve ici le but qu'un agent a de réaliser une action et les fonctions ou buts incorporés dans l'objet qui spécifient certaines activités.
- *rôle agentif* (*agentive*) : facteurs impliqués dans l'origine de l'élément ou de sa cause. Ici on trouve le créateur, l'objet, l'élément naturel et la chaîne causale.

Pour encoder la variation polysémique des mots, Pustejovsky propose un formalisme nommé *Lexical Conceptual Paradigm* (LCP), qui caractérise une entrée lexicale comme une méta-entrée, ce qui permet de saisir les ambiguïtés lexicales. Les LCP ne considèrent pas les mots comme ayant des sens distincts, mais comme différents aspects d'une même méta-entrée lexicale. Le terme *fenêtre* représente des notions d'objet physique, d'ouverture ou les deux à la fois. Dès lors, *fenêtre* aura la LCP suivante :

(29) `phys_obj.aperture_lcp =`
`{phys_obj.aperture, phys_obj, aperture}`

Pour terminer, examinons deux verbes :

(30) a.

tuer		
EVENTSTR =	E_1 =	**e_1:process**
	E_2 =	**e_1:state**
	RESTR =	$<_\alpha$
	HEAD =	e_1
ARGSTR =	ARG1 =	1 [**ind**]
		FORMAL = **physobj**
	ARG2 =	2 [**animate_ind**]
		FORMAL = **physobj**
QUALIA =	**cause_lcp**	
	FORMAL =	**dead(e_2,** 2**)**
	AGENT =	**kill_act(e_1,** 1**,** 2**)**

b.
$$\left[\begin{array}{ll}
\textbf{bâtir} & \\
\text{EVENTSTR} = & \left[\begin{array}{ll} \text{E}_1 = & \textbf{e}_1\textbf{:process} \\ \text{E}_2 = & \textbf{e}_2\textbf{:state} \\ \text{RESTR} = & <_\alpha \\ \text{HEAD} = & \text{e}_1 \end{array}\right] \\
\text{ARGSTR} = & \left[\begin{array}{ll} \text{ARG1} = & \boxed{1}\left[\begin{array}{ll} & \textbf{animate_ind} \\ \text{FORMAL} = & \textbf{physobj} \end{array}\right] \\ \text{ARG2} = & \boxed{2}\left[\begin{array}{ll} & \textbf{artifact} \\ \text{CONST} = & \boxed{3} \\ \text{FORMAL} = & \textbf{physobj} \end{array}\right] \\ \text{D-ARG1} = & \boxed{3}\left[\begin{array}{ll} & \textbf{material} \\ \text{FORMAL} = & \textbf{mass} \end{array}\right] \end{array}\right] \\
\text{QUALIA} = & \left[\begin{array}{ll} & \textbf{create_lcp} \\ \text{FORMAL} = & \textbf{exist}(\textbf{e}_2, \boxed{\textbf{2}}) \\ \text{AGENT} = & \textbf{build_act}(\textbf{e}_1, \boxed{\textbf{1}}, \boxed{\textbf{3}}) \end{array}\right]
\end{array}\right]$$

En (30a), ARG1 est un objet indéfini, ARG2 est un objet vivant et animé. L'acte de *tuer* est donc de causer la mort. En (30b), l'acte de *bâtir* est réalisé par un objet animé qui construit un artefact, c'est-à-dire un objet réalisé par un être vivant à l'aide de matériaux. Les matériaux sont un objet par défaut. Après la réalisation de l'acte de bâtir, l'objet existe. Grâce aux règles de composition, des phrases complètes peuvent être décrites au moyen du lexique génératif.

Il n'existe qu'un nombre restreint de réalisations informatiques du Lexique Génératif, qui sont limitées à un domaine peu étendu. Nous citerons les ouvrages de Claveau *et al.* (2001), Claveau et Sébillot (2004) et Namer *et al.* (2007). A notre connaissance, aucune application d'ALAO n'utilise un Lexique Génératif. Dans ce cadre, les relations décrites dans le modèle du lexique génératif pourraient être utiles à plusieurs niveaux. Pour l'acquisition du vocabulaire, les différentes relations encodées dans ce lexique pourraient servir à générer des exercices de vocabulaire. Dans le cadre d'exercices de compréhension de textes, l'accès à un tel lexique permettrait de mieux comprendre le contexte d'utilisation d'un mot. Un analyseur syntaxique pourrait aider à désambiguïser le mot et à choisir la bonne entrée dans le lexique. Inversement, les relations du lexique sont utiles à la désambiguïsation syntaxique, puisque les informations sémantiques contenues dans le lexique sont riches. Nous avons déjà souligné l'importance des relations sémantiques pour l'analyse syntaxique à la section 3.3 de ce chapitre. Enfin, pour des descriptions sémantiques comme la SDRT (§3.4.1), les LCS (§3.4.2), les QLF (§3.4.3) ou les PSS (§7.1), les informations contenues peuvent aider à la construction des structures ou peuvent aussi aider à la génération d'autres phrase en déterminant des synonymes, antonymes, hyponymes ou hyperonymes, (v. définition p. 141).

Comme tous les formalismes lexicaux, le lexique génératif nécessite une taille de stockage et un nombre d'informations considérables. L'aspect compositionnel du sens et le caractère ouvert du lexique ajoutent aux difficultés de l'approche. Ce formalisme est encore récent, ce qui peut expliquer le faible nombre de réalisations. Nous n'avons par trouvé de référence d'un lexique réellement implémenté, y compris pour la langue anglaise. La richesse des informations contenues dans ce lexique font qu'il est très difficile d'extraire automatiquement des informations exhaustives.

3.5.2 Théorie Sens-Texte

La *Théorie Sens-Texte* (TST, Mel'čuk, 1997; Polguère, 1998; Mel'čuk, 2001) est une théorie linguistique complète qui offre un formalisme aux niveaux sémantique, syntaxique, morphologique et phonétique. A l'exception du côté sémantique, chacun de ces niveaux est divisé en structures profondes et structures de surface. Des processus permettent le passage d'un niveau à l'autre, du sens au texte, c'est-à-dire de concepts à une phrase écrite ou orale. Le niveau syntaxique de surface est représenté sous forme de grammaire de dépendance assez complexe (§3.3.5). Une représentation sémantique est constituée de trois structures :

- Une *structure sémantique* qui constitue le sens propositionnel de la phrase, de laquelle dépendent les deux autres structures ;
- Une *structure sémantico-communicative*, qui reflète le sens communicatif ;
- Une *structure rhétorique*, qui reflète les intentions du locuteur (ironie, pathétique, niveaux de langage...).

La structure sémantique est un graphe orienté dont les nœuds sont formés d'unités sémantiques ou sémantèmes. Chaque unité est un sens particulier (ou lexie) non ambigu. Kahane (2001) montre que *cheval* (mot simple) *pomme de terre* (mot composé) et *prendre le taureau par les cornes* (expression idiomatique, v. p. 66) forment des sémantèmes, et que des parties du discours de catégorie différente peuvent avoir le même sémantème, comme *partir / départ*, *durer / pendant*.

Les fonctions lexicales sont un concept clé de la TST. Elles constituent un outil de modélisation des phénomènes collocationnels. Une soixantaine de fonctions modélisent les relations de cooccurrences lexicales. Ces fonctions sont universelles et suffisent à décrire la plupart des collocations dans toutes

les langues. Elles servent à donner la ou les lexies qui permettent d'exprimer une fonction particulière en fonction d'une autre lexie. Les fonctions sont nommées d'après des noms latins. Le tableau (3.3) montre quelques fonctions lexicales.

Fonction	**Utilisation**	**Exemples**
Magn	intensification	**Magn**(*dormir*) = profondément, comme une souche... **Magn**(*blessé*) = gravement, grièvement. **Magn**(*amour*) = ardent. $\mathbf{Magn}_{temps}$(*amour*) = éternel
Bon	bon, tel que le locuteur approuve	**Bon**(*conseil*) = bon, précieux. **Bon**(*temps*) = beau. **Bon**(*se porter*) = comme un charme
AntiBon	contraire de Bon	**AntiBon**(*victoire*) = à la Pyrrhus. **AntiBon**(*temps*) = de chien
Loc	localisation standard spatiale ou temporelle	$\mathbf{Loc}_{in/ad}$(*gare*) = en gare_{train}, à $[\text{art}]_{personnes}$
Oper	verbes support	**Oper**(*remarque*) = faire. **Oper**(*méfait*) = perpétrer. **Oper**(*applaudissements*) = recueillir
Func	autres verbes support	**Func**(*réunion*) = est en cours. **Func**(*responsabilité*) = incombe à. **Func**(*danger*) = menace
Labor	verbes support	**Labor**(*liste*) = mettre N sur ART.
Incep	début action	
Caus	être cause de	

TAB. 3.3 – *TST: quelques fonctions lexicales*

Ces fonctions peuvent être combinées, comme par exemple
Magn+Oper(amour) = brûler, se consommer, mourir [d' ∼ pour N].

Le cœur de la Théorie Sens-Texte est le *Dictionnaire Explicatif et Combinatoire*, qui encode les informations lexicales. Le dictionnaire est explicatif car il décompose un mot en un ensemble d'éléments constitutifs et il est combinatoire car il décrit la sous-catégorisation du mot, ainsi que les fonctions lexicales qui peuvent lui être associées. Le dictionnaires devrait contenir environ un million d'entrées pour le français contemporain. Les quatre volumes parus (Mel'čuk *et al.*, 1984, 1988, 1992, 1999) comprennent les descriptions de 510 vocables du français et 1583 définitions lexicales (Altman et Polguère, 2003), dont la liste est donnée sur le site de l'*Observatoire de Linguistique Sens-Texte* [48].

48. `http://www.olst.umontreal.ca/`, consulté le 10 août 2004.

Passons aux réalisations informatiques. La *BDéf* est une base de données en cours de construction qui vise à informatiser le *DEC* (Altman et Polguère, 2003). *DiCo* (Polguère, 2003) est une base de données dérivée du DEC. Elle décrit les dérivations sémantiques de mots avec d'autres et les *collocations* de la langue. Il existe une application grand public de DiCo appelée *Lexique actif du français* (*LAF*), qui a pour but d'enseigner l'acquisition du lexique. Citons encore les travaux de Rambow et Korelsky (1992) et Visser (1999).

Passons maintenant aux applications possibles de la TST pour l'ALAO. Milićević et Hamel (2005) présentent *Dire autrement*, un dictionnaire de reformulation basé sur la TST. Ce dictionnaire est destiné aux apprenants du français de niveau moyen à avancé, afin qu'ils puissent surmonter les difficultés d'une utilisation idiomatique de la langue en utilisant des paraphrases. Pour cela, il est nécessaire de connaître les relations de collocation d'un mot avec un verbe, avec un intensificateur etc. Il est également utile de connaître les relations d'antonymie, synonymie, nominalisation, verbalisation etc.

Avec des dictionnaires complètement implémentés, une telle théorie permettrait de vérifier des collocations dans des applications d'analyse. Le degré de grammaticalité d'une phrase pourrait ainsi mieux être évalué et des expressions équivalentes pourraient être admises comme correctes. En tant que dictionnaire d'apprentissage, le DEC ou ses dérivés seraient très utiles. Enfin, pour la traduction automatique ou comme aide à la rédaction, des dictionnaires bilingues indiquant les équivalences de traduction seraient une contribution essentielle. Pour le français, l'étendue du lexique pourrait éventuellement s'avérer suffisante pour une utilisation dans une application d'ALAO.

3.5.3 WordNet

WordNet (Fellbaum, 1998; Fellbaum et Miller, 2003) est la base de données lexicales la plus importante pour l'anglais. Les noms, adjectifs, verbes et adverbes sont organisés en ensemble de synonymes qui représentent des concepts sous-jacents (*synsets*). Pour les noms, on dénombre 114 648 mots organisés en 79 689 concepts [49]. *EuroWordnet* est un projet dérivé de *WordNet* pour le néerlandais, l'italien, l'espagnol, l'allemand, le français, le tchèque et l'estonien, achevé en 1999. Ce projet est géré par un consortium européen centralisé à Amsterdam. Il utilise les mêmes *synsets* que *WordNet*. La base française compte 23 000 *synsets* et 49 000 relations entre ces termes.

49. Chiffre disponibles sur le site `http://www.cogsci.princeton.edu/~wn/`, dernier accès le 5 juillet 2004.

Certains mots sont organisés en collocations. Les concepts sont organisés de manière hiérarchique. *WordNet* encode de nombreuses relations :

- *synonymie/antonymie* : des sens sont synonymes si la substitution de l'un pour l'autre *dans un contexte donné* ne change pas la valeur de vérité. Un mot F est antonyme d'un G si non-G signifie F. Cependant, *petit* et *grand* sont antonymes mais quelque chose qui n'est pas petit n'est pas forcément grand.
- *hyponymie/hyperonymie* : ces relations sont aussi connues sous le terme de général/particulier. F est un hyponyme de G si F est une sorte de G. A l'inverse, F sera un hyperonyme de G.
- *méronymie/holonymie* (relation partie/tout) : F est un méronyme de G si F est une partie de G.
- *enchaînement causal* : c'est une relation entre verbes. Une action résulte d'une autre, comme par exemple *tuer* et *mourir*.
- *troponymie* : c'est une autre relation entre verbes. Si V1 est une manière de faire V2, alors, V1 est un troponyme de V2.

WordNet est utilisé pour l'extraction automatique de nouveaux concepts et de nouvelles relations entre concepts. Il est également utile pour l'extraction de concordances sémantiques, de taxonomies, pour l'indexation de textes etc. *Wordnet* est aussi utilisé dans le domaine de la correction stylistique (Hirst et St-Onge, 1998; Hirst et Budanitsky, 2005) : le système détecte le mauvais emploi de mots en tenant compte du contexte.

L'utilisation de *WordNet* dans le domaine de l'ALIAO est évoqué par Miller et Fellbaum (1992). Ils relèvent que les mots de deux langues ne se recoupent pas exactement et qu'il est souvent nécessaire de recourir à des adjectifs ou des périphrases pour un concept qui n'existe pas dans la langue cible ou, à l'inverse, que des mots ont plusieurs équivalents. De plus, les mots ne recouvrent parfois pas exactement le même concept ni le même cadre de sous-catégorisation. Pour une aide à l'apprentissage, Miller et Fellbaum (1992) préfèrent l'utilisation d'un réseau sémantique monolingue en langue seconde pour favoriser l'acquisition lexicale. Quant à Agirre *et al.* (1998), ils utilisent un calcul de proximité de sens pour présenter le meilleur choix en premier dans un correcteur d'orthographe (§C.1).

TAEMA (*Traitement Automatique de l'Écriture de Mots Affectifs*, Buvet et Issac, 2006) est un outil d'apprentissage des expressions affectives (domaine de l'amour, de la sympathie etc.) qui nécessitent l'emploi d'expressions figées. Le système est capable de générer une série de phrases représentant la même idée. Zock (2006) propose quant à lui d'utiliser un dictionnaire qui

dispose de liens associatifs pour aider les apprenants à exprimer leurs idées à l'aide d'un outil de génération (§3.1.5). *TAGARELA* (§B.4.34, Bailey et Meurers, 2006) compare les lemmes de mots qui ne se trouvent pas dans la réponse attendue par le système afin de détecter les synonymes. Enfin, pour le système *SigmaStar* (§B.2.31) d'enseignement sur téléphones mobiles, Ott *et al.* (2005) utilisent le dictionnaire *GermaNet* (Lemnitzer et Kunze, 2002) dérivé de *WordNet*.

Pour conclure, nous constatons que *WordNet*, le formalisme lexical le plus répandu et le plus développé, est également le plus utilisé dans le monde de l'ALAO. L'étendue des lexique est essentielle pour une utilisation pédagogique ; plus il y a de mots dans un lexique, plus il y a de chances de retrouver les mots d'un texte.

3.5.4 Autres outils lexicaux

Décrivons maintenant quelques autres outils lexicaux. Commençons par les applications de dictionnaires. Le *Trésor de la Langue Française* (TLF) est le dictionnaire électronique le plus complet (Dendien et Pierrel, 2003; Bernard *et al.*, 2004), disponible en CD-ROM ou sur *Internet*. Il compte 100 000 mots, 450 000 entrées et 500 000 citations précisément identifiées. Le *Dictionnaire d'Apprentissage du Français Langue Etrangère ou Seconde* (DAFLES) est conçu comme un outil d'apprentissage innovant qui facilite l'accès à l'information de manière différente que les traditionnels dictionnaires sur papier (Selva *et al.*, 2003) [50]. Les informations sont filtrées selon les besoins. On trouve de nombreuses collocations et des listes associatives d'actants, comme *scrutin, vote, élu, corps électoral, électeur* etc. pour *élection*. *Alfalex* (*Actieve Leeromgeving Frans voor Anderstaligen - LEXicon*, Verlinde *et al.*, 2003) est un environnement actif d'apprentissage du français langue étrangère pour néerlandophones, inspiré du DAFLES et d'$\mathcal{A}$lexi$\mathcal{A}$ (§B.2.1, Selva et Chanier, 2000).

Le projet multilingue *Papillon* [51] (Mangeot et Thevenin, 2004) est une base lexicale multilingue de l'allemand, l'anglais, du français, du japonais, du malais, du lao, du thaï, du vietnamien, du chinois, etc. établi sur une base collaborative (§2.3), dont Zock (2006) propose une application pédagogique. Citons encore les travaux de Tschichold et ten Hacken (1998); ten Hacken et Tschichold (2001) et Hamel *et al.* (1995); Singleton *et al.* (1998). Enfin, les réseaux sémantiques pourraient également fournir des aides intéressantes

50. Le DAFLES est disponible à l'adresse `http://www.kuleuven.ac.be/dafles/`, dernier accès le 30 juillet 2004.
51. `http://www.papillon-dictionary.org/`, consulté le 30 octobre 2004.

pour l'apprentissage (Wilks et Farwell, 1992; Ji *et al.*, 2003; Ji et Ploux, 2003; Ploux et Ji, 2003).

3.5.5 Discussion

Dans cette section, nous avons brièvement discuté de l'utilité des lexiques dans l'apprentissage des langues. Ils sont importants à deux titres : (i) en tant que tels, ils peuvent constituer une aide à la rédaction et à l'apprentissage ; (ii) combinés à d'autres outils de TAL, ils peuvent aider à la correction automatique et vérifier si la réponse de l'apprenant est compatible avec la réponse attendue par le système.

Parmi les formalismes lexicaux d'une certaine envergure, *WordNet* (§3.5.3) est le plus étendu. Le *Lexique Génératif* (§3.5.1) est un formalisme très compliqué et difficile à bâtir, qui n'a pas été réellement implémenté sur ordinateur. Le lexique de la Théorie Sens-Texte (§3.5.2) est également difficile à construire et à définir, mais pour le français, le lexique a une étendue qui pourrait s'avérer utilisable en ALAO. Enfin, nous pouvons aussi mentionner le Trésor de la Langue Française, qui est un dictionnaire performant et utile. D'autres applications fournissent des lexiques de moindre envergure. Les applications de type dictionnaire fournissent des indications utiles sur les collocations et les relations telles que l'hyponymie ou l'hyperonymie. Par contre, les réseaux sémantiques permettent de découvrir des relations de proximité entre mots. A notre sens, les deux types d'outils sont utiles et peuvent se compléter. Relevons enfin que les liens hypertextes et les possibilités d'affichage et de représentation des informations offrent une plus-value indéniable aux supports électroniques par rapport aux versions papier classiques.

3.6 Conclusion

Dans ce chapitre, nous avons examiné les techniques de traitement du langage, et en particulier leur application au domaine de l'ALIAO. Nous avons pu constater que, plus les applications sont de haut niveau, plus elles sont sensibles aux erreurs, qui sont légion dans des productions des apprenants. Issac et Hû (2002) soulignent qu'un petit gain en performance des outils d'intelligence artificielle entraîne un très fort accroissement de la complexité du système. Or il est impératif de pouvoir disposer d'outils fiables, robustes et rapides.

En outre, dans de nombreux domaines, les outils de TAL se heurtent à l'écueil de la compréhension des énoncés et de la connaissance du monde. Certes, les performances des outils de TAL sont plus probantes si l'on restreint le vocabulaire, les structures syntaxiques disponibles et les tâches à effectuer. Pourtant, les apprenants d'une langue étrangère ont toujours besoin de capacités générales de production et de compréhension, et cette capacité doit notamment être acquise par la consultation de documents authentiques, écrits ou oraux. Or il est plutôt préjudiciable à l'enseignement et frustrant pour l'enseignant de devoir choisir des documents en fonction du degré de couverture des outils et non en fonction des capacités du public cible ou du niveau que les apprenants doivent atteindre.

Ainsi, de nombreux efforts ont été déployés dans le domaine du TAL, avec des progrès de niveau très inégal en fonction des sous-domaines. Les domaines les plus utilisés dans l'ALAO sont par conséquent la synthèse (§3.1.4) et la reconnaissance vocale (§3.1.3), l'étiquetage (§3.1.2), les concordanciers (§3.1.7), les applications lexicales et, dans une moindre mesure, l'analyse syntaxique et la détection d'erreurs (§3.3). De manière un peu pessimiste, Bailin (1995) souligne que les coûts de production de systèmes d'ALIAO sont considérables et qu'ils apportent encore peu de bénéfices et d'acquis pour l'enseignement. Quant à Goodfellow *et al.* (2001), ils constatent que l'analyse de bas niveau au niveau lexical peut constituer une bonne alternative à la trop faible qualité des outils de TAL disponibles pour l'ALAO. Ils proposent de se concentrer sur les mots de vocabulaire utilisés par les apprenants en se basant sur les fréquences de mots.

Par contre, dans un bilan des projets parus dans le domaine du traitement des langues appliqué à l'ALIAO de 1978 à 2004, Heift et Schulze (2005) constatent que les programmes affichent des ambitions plus réalistes. L'accent est désormais mis sur la modélisation de l'apprenant et la remédiation, tout en poursuivant la tâche initiale de repérer les erreurs des apprenants. Des agents intelligents qui ciblent des tâches précises valent mieux que de grands systèmes qui visent une couverture étendue de la langue.

Pour notre part, nous soulignons que des progrès significatifs ont été faits dans la plupart des domaines évoqués. L'ALAO et le TAL sont des disciplines encore jeunes qui n'ont pas encore atteint la maturité du fait de la complexité de la langue. Les techniques utilisées sont encore expérimentales. C'est pourquoi nous n'avons présenté pratiquement que des prototypes de recherche, à l'exception de certains correcteurs commerciaux, du logiciel *Herr Kommissar* (§B.3.5) et de logiciels basés sur la reconnaissance vocale. Nous préférons pourtant montrer un optimisme relatif et pointer les bénéfices potentiels ou réels plutôt que d'insister sur les faiblesses des outils et

leur caractère inachevé.

Comme pistes de recherche pour l'avenir, mentionnons d'abord l'utilisation de ressources de bas niveau comme aides à l'élaboration d'exercices, comme le fait le projet *MIRTO* (§B.2.21). Ces ressources permettent un gain de temps appréciable pour les concepteurs d'exercices et peuvent apporter une aide à l'apprentissage pour les apprenants. Les techniques de correction basées sur les étiqueteurs sont aussi une approche intéressante, qui permet assez rapidement à des non-informaticiens de bâtir des règles d'erreurs.

Par ailleurs, à partir d'un certain niveau, les apprenants peuvent se montrer des alliés efficaces pour aider les outils de diagnostic. En répondant à des questions simples, ils peuvent préciser leurs intentions et aider la désambiguïsation. Il faut toutefois éviter que l'apprenant perde trop de temps dans ce processus.

Enfin, didacticiens, informaticiens, linguistes et psychologues doivent travailler ensemble pour améliorer l'efficacité des interfaces et des diverses représentations des phrases, afin d'améliorer l'apprentissage. L'informatique permet de présenter rapidement de nombreuses informations sur la structure des phrases et sur les mots, ce qu'un correcteur humain n'a pas le temps de faire sur le papier. De plus, des mises en évidence par des animations remplacent avantageusement les dessins au tableau noir ou au rétroprojecteur.

Chapitre 4

Le projet FreeText

Dans ce bref chapitre, nous présentons le projet FreeText [1], dans le cadre duquel les recherches présentées dans cette thèse ont été réalisées en grande partie. Ayant duré d'avril 2000 à mars 2003, FreeText a consisté à développer un prototype de logiciel d'ALIAO pour apprenants du français langue étrangère, de niveau intermédiaire à avancé (Vandeventer, 2001; Granger *et al.*, 2001; Hamel et Girard, 2004; L'haire, 2004, etc.). Le logiciel était destiné tant à des apprenants dans un cadre privé qu'à des institutions éducatives ou même à des entreprises, plutôt orienté vers la lecture et l'écriture et peu vers la compréhension et l'expression orale. Sa grande originalité était de prévoir l'emploi de nombreux outils de TAL comme aides à l'apprentissage : conjugueur, correction orthographique, analyse syntaxique, détection d'erreurs syntaxiques, traduction automatique, comparateur de phrases et reformulateur de phrases.

Le projet FreeText a réuni les partenaires suivant :

– Département d'Ingénierie de la Langue du *University of Manchester Institute of Science and Technology* (UMIST), chargé du contenu di-

1. Voir `http://www.latl.unige.ch/freetext/`, dernière consultation le 2 août 2009. Le projet FreeText relève du programme IST du 5^e programme-cadre de la Commission Européenne, contrat IST-1999-13093. Le volet suisse du projet FreeText reçoit le soutient financier de l'Office Fédéral de l'Education et de la Science, contrat 99.0049. Le contenu de cet article n'engage que ses auteurs et ne représente pas l'opinion de la Communauté Européenne. La Communauté Européenne n'est pas responsable de l'usage qui pourrait être fait des données figurant dans cet article. Les informations de ce document sont présentées telles quelles et aucune garantie n'est donnée que ces informations sont adéquates pour n'importe quelle utilisation particulière. Dès lors, l'utilisateur emploie ces informations à ses propres risques.

dactique et de la coordination administrative et financière ;
- Département de Linguistique, Université de Genève, chargé du développement des outils de traitement automatique des langues (à l'exception du correcteur orthographique et de la traduction automatique) et de la coordination scientifique ;
- Le *Centre for English Corpus Linguistics* de l'Université Catholique de Louvain (UCL), chargé de la récolte et de l'étiquetage du corpus et de l'évaluation du logiciel et des outils ;
- Softissimo SARL, Paris, partenaire industriel, chargé des outils de correction orthographique et de traduction automatique et de la réalisation du prototype.

La suite de ce chapitre se présente comme suit. Tout d'abord, nous présentons l'ensemble du projet (§4.1). Puis nous détaillons les différentes composantes (§4.2). Enfin nous terminons par un bilan et des discussions (§4.3).

4.1 Présentation générale

Dans cette section, nous décrivons le projet FreeText dans son ensemble. Le logiciel est basé sur une approche communicative (§2.3) et se concentre sur les marques linguistiques qui distinguent et organisent différents types de textes qui ont diverses intentions énonciatives (Hamel et Girard, 2004). FreeText contient quatre tutoriels [2] présentant 16 documents multimédias authentiques, qui servent à illustrer divers actes de langage et autour desquels des activités sont proposées. Les textes choisis proviennent de diverses régions de la francophonie. Les quatre tutoriels sont :

- *(s')informer* : les textes de type informatif sont destinés à donner une information au destinataire. On y trouve un *curriculum vitæ*, un reportage télévisé, un article de magazine et un article d'encyclopédie sur la francophonie ;
- *(faire) réagir* : les textes argumentatifs sont destinés à faire acheter un produit ou à faire adhérer le destinataire à l'opinion qui y est présentée. On y trouve un extrait de site *web* d'un mouvement politique, un

2. Un cinquième tutoriel, *(se) parler*, devait aborder le dialogue, notamment les marques de l'oral dans les textes écrits. Le consortium a renoncé à produire ce tutoriel, notamment en l'absence d'un logiciel de reconnaissance vocale (§3.1.3) et de la difficulté de produire une rétroaction intelligente à l'aide de cette technique. Nous pouvons ajouter qu'il aurait été difficile de tenir la comparaison avec des logiciels commerciaux disponibles sur le marché, qui bénéficient de moyens sans commune mesure avec ceux d'un projet de recherche.

éditorial de quotidien, un extrait de film de fiction et une publicité télévisée ;
- *(se) raconter* : les textes narratifs sont destinés à rapporter des événements réels ou fictifs qui ont eu lieu dans un passé proche ou lointain. On y trouve un extrait de roman, un extrait de site *web*, un extrait de film et un article de presse sur un fait divers ;
- *(faire) agir* ; les textes injonctifs ou persuasifs sont destinés à transmettre des conseils, des ordres, des recommandations, etc. On y trouve un extrait de site web, un article d'hebdomadaire, un extrait de guide touristique et un extrait de film.

Les différents documents sont présentés par une capture d'écran, un *fac simile* ou une vidéo. Une transcription interactive, y compris des documents écrits, permet à l'apprenant de visualiser une version écrite augmentée et standardisée des documents et de disposer d'aides lexicales. Les enseignants ont également la possibilité d'introduire de nouveaux documents dans un tutoriel personnalisé et de créer de nouveaux exercices. Autour de chaque texte, de nombreuses activités sont proposées. Au total, FreeText compte 550 exercices de 16 types différents. Les activités appartiennent à quatre domaines :

- *Compréhension* : les activités concernent le sens du texte :
  - *Contexte de production* : l'apprenant doit identifier l'auteur, son intention, le public auquel le texte s'adresse, le médium (sous quelle forme le document est transmis), le motif pour lequel le document a été produit et la fonction du document. Il doit répondre aux questions par un court texte à l'aide des outils à disposition et une ébauche de solution lui est fournie lorsqu'il demande l'affichage de la solution.
  - *Vocabulaire* : l'apprenant doit donner une définition de certains termes qui apparaissent dans les textes, également sous forme de textes libres.
  - *Jeux de mots* : divers jeux sont proposés, comme des mots croisés.
  - *Dictée* : l'apprenant doit retranscrire un texte audio enregistré par un locuteur natif.
  - *Culture* : l'apprenant peut consulter divers documents explicatifs en relation avec le texte présenté.
- *Exploration* : à travers le texte, l'apprenant doit découvrir divers points de grammaire et identifier certains contenus à l'aide de questionnaires à choix multiples. Les points à identifier sont le destinateur, le destinataire, la mise en contexte pour remplir la fonction du texte (emploi des

modes, adjectifs, adverbes, etc.), l'organisation de la phrase (phrases simples ou complexes, ponctuation, etc.) et celle du texte (structure globale, repères spatio-temporels).
- *Manipulation* : ici, on trouve les exercices traditionnels des tutoriels (QCM, textes à trous, etc.).
- *Création* : l'apprenant doit écrire des textes de longueur moyenne en pastichant le texte présenté.

Un corpus d'erreurs authentiques d'apprenants du français (*French Interlanguage Database*, FRIDA, Granger *et al.*, 2001; Granger, 2003) a également été récolté, avec des erreurs d'apprenants anglophones, néerlandophones et d'autres provenances, avec différents niveaux de langue et niveaux d'étude (universitaires ou écoles secondaires). Ce corpus contient 492 808 mots. Les erreurs ont été partiellement balisées sous la forme de balises *XML* [3] d'après une typologie précise des erreurs. Au total, 322 964 mots ont été inclus dans le corpus balisé, dont 46 241 comportaient une erreur, soit 14,3%.

Les dix erreurs les plus fréquentes sont, par catégories d'erreur :

- Erreurs de forme :
  - erreurs d'orthographe autres que les diacritiques, la casse, l'agglutination et les homonymes ;
  - erreurs de diacritiques ;
  - erreurs de casse.
- Erreurs de grammaire :
  - erreurs de genre ;
  - erreurs de nombre ;
  - erreurs de classe : confusion de catégories ou sous-catégories lexicales.
- Erreurs lexicales :
  - erreurs de signification.
- Erreurs de syntaxe :
  - mots manquants ;
  - mots redondants.
- Erreurs de ponctuation :
  - oubli de ponctuation.

3. V. §2.7.5. Les documents ne sont toutefois pas conformes aux spécifications de validité du formalisme *XML*.

Ce corpus a été utilisé dans le projet de deux manières :

- création d'outils de référence et d'exercices ciblés vers les erreurs fréquemment commises par les apprenants ;
- adaptation des outils de diagnostic d'erreurs vers les erreurs les plus fréquentes et création d'un corpus de test pour évaluer l'efficacité de ces outils.

Remarquons pour terminer que ce projet a bénéficié d'une approche pluridisciplinaire (§2.2) et a fait l'objet d'une phase d'évaluation comme préconisé au paragraphe §2.5. Nous reviendrons sur ces aspects en fin de chapitre.

4.2 Un tutoriel intelligent

FreeText offre des outils de traitement automatique des langues et peut être considéré par conséquent comme un logiciel d'ALIAO. Nous avons donc choisi de présenter le logiciel comme un tutoriel intelligent (§2.7.4) et de décrire les différents modules qui composent traditionnellement les logiciels intelligents, même si le logiciel n'est pas construit selon cette architecture et ne fait pas collaborer ses différentes composantes. Nous reviendrons sur ces aspects à la section 4.3.

Dans cette section, nous décrivons le module expert (§4.2.1), le module de l'apprenant (§4.2.2), le module pédagogique (§4.2.3) et le module d'interface (§4.2.4).

4.2.1 Module expert

Le module expert est la composante essentielle des tutoriels intelligents. Afin de fournir des outils d'apprentissage efficaces et d'apporter une rétroaction intelligente aux apprenants, FreeText dispose d'une série d'outils de traitement des langues. Le module expert permet de fournir une rétroaction intelligente aux apprenants pour les exercices dont les réponses sont libres. Par ailleurs, d'autres outils de traitement automatique des langues offrent des aides diverses à l'apprentissage.

Tout d'abord, un correcteur orthographique (§3.2) permet de corriger essentiellement les erreurs typographiques et phonétiques. Nous utilisons le

correcteur orthographique du correcteur grammatical *Hugo* (§C.19), développé par le partenaire *Softissimo* et très légèrement adapté pour des apprenants du français langue étrangère [4]. Puis un analyseur syntaxique procède à l'analyse grammaticale et à la détection d'éventuelles erreurs. Nous présenterons en détail ces modules au chapitre 5. Enfin, le projet prévoyait un vérificateur "sémantique" et un reformulateur de phrase [5]. Le fonctionnement des outils est résumé par la figure (4.1).

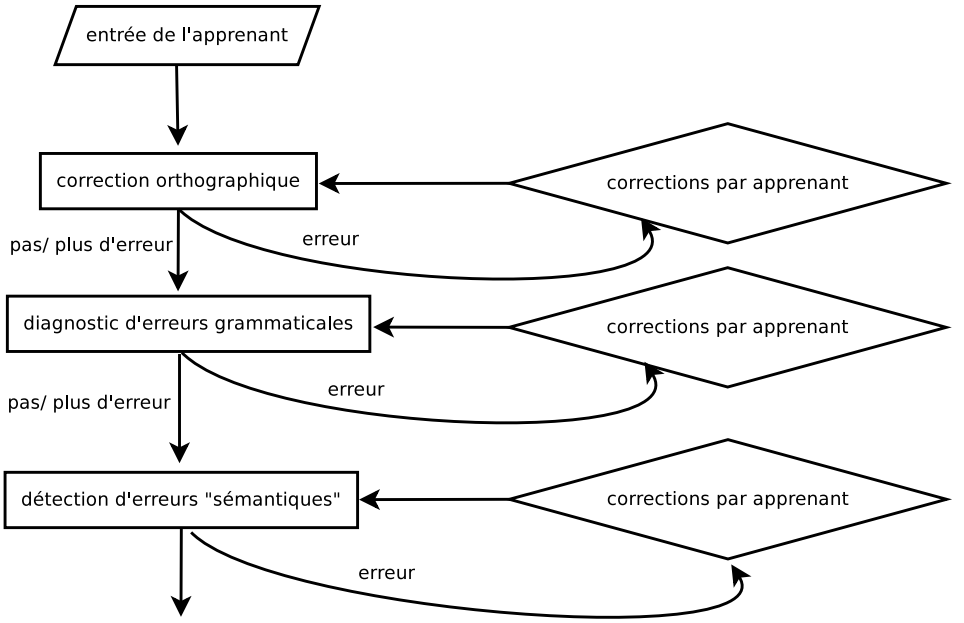


FIG. 4.1 – *FreeText: fonctionnement des outils de diagnostic*

Parmi les aides à l'apprentissage, un conjugueur (§3.1.1.4) permet d'accéder aux conjugaisons des verbes. L'apprenant peut entrer une forme verbale quelconque et consulter la conjugaison complète du ou des verbes concernés. Par ailleurs, l'outil de visualisation d'analyse syntaxique en couleurs (§5.3.2) et le diagnostic d'erreurs (§5.3.3) renvoient tous deux aux conjugaisons des verbes de la phrase. En outre, l'apprenant peut soumettre n'importe quelle phrase au synthétiseur vocal *FipsVox* (§§3.1.4, 5.1), dont nous avons ralenti le débit et augmenté l'amplitude prosodique. Enfin, l'apprenant dispose en tout temps d'un lien vers le traducteur automatique en ligne *Reverso*

4. En parallèle, une autre équipe a développé un correcteur expérimental *FipsCorr* (Ndiaye et Vandeventer Faltin, 2003, 2004), qui n'a finalement pas été retenu par décision du consortium du projet. Nous avons continué le développement de ce correcteur sur le nom de *FipsOrtho*, qui fait l'objet du chapitre 6.

5. Nous avons commencé à développer une partie de ces outils (§7), mais le consortium a considéré que l'ampleur du développement d'autres parties du projet avait été sous-estimée et que l'effort devait être porté sur les outils-phares du projet plutôt que sur des développements expérimentaux.

(§3.1.6), également développé par *Softissimo*, qui ne fonctionne que dans le sens français vers anglais, espagnol et allemand.

Pour conclure, nous pouvons remarquer que ce module est très étendu et tire parti des nombreux outils développés par les divers membres du consortium. Il offre une richesse et un nombre d'outils considérables par rapport aux logiciels que nous avons étudiés (§2). Dans les chapitres suivants, nous reviendrons sur ces outils et sur leur potentiel d'utilisation, en détaillant leur efficacité.

4.2.2 Module de l'apprenant

Le module de l'apprenant a pour rôle de récolter les données de l'apprenant afin d'adapter les stratégies du logiciel. Dans FreeText, toutes les activités de l'apprenant sont répertoriées dans une base de données. On y trouve les réponses aux exercices, le nombre de tentatives, le pourcentage d'exercices effectués, ainsi que le contenu des appels aux outils d'aide et de la grammaire de référence. La date et l'heure de chaque élément est notée. Par contre, les erreurs détectées par le diagnostic d'erreurs ne sont pas enregistrées. Malheureusement, aucune information personnelle sur les apprenants eux-mêmes n'est enregistrée dans la base, à part le groupe dont fait partie l'apprenant dans le cadre d'une utilisation en classe. La figure (4.2) illustre la consultation des statistiques par un apprenant.

FIG. 4.2 – *FreeText: consultation des statistiques des exercices par apprenant*

Ce module ne remplit qu'une petite partie des tâches habituellement

remplies – ou, plutôt, qu'on aimerait voir remplies – par le module de l'apprenant d'un logiciel d'ALIAO. Ainsi, les informations sur les apprenants ne sont pas assez fines. Il serait bon d'avoir des informations sur le sexe, l'âge et la langue première de l'apprenant, ne serait-ce qu'à des fins de recherche. Ensuite, lorsqu'un outil offre un diagnostic, celui-ci devrait être enregistré, avec le type d'erreur, la ou les catégories lexicales en jeu, l'identifiant de l'exercice et de la réponse, afin de pouvoir retrouver le contexte [6]. De même, les appels à la grammaire de référence devraient être référencés, avec un formalisme le plus proche possible du marquage du diagnostic d'erreurs, afin de pouvoir mesurer comment l'apprenant utilise les différents outils et sa progression éventuelle.

4.2.3 Module pédagogique

Le module pédagogique gère habituellement les outils d'aide à l'apprentissage et la stratégie pédagogique des logiciels intelligents. Les outils intelligents de FreeText sont sa principale aide à l'apprentissage, avec essentiellement la visualisation de la structure des phrases et la visualisation des erreurs. Le prototype fournit aussi un glossaire, des explications de vocabulaire et une grammaire de référence complète destinée aux apprenants, adaptée du formalisme grammatical des Principes & Paramètres (Chomsky et Lasnik, 1995, §3.3.5), liée aux textes et exercices du tutoriel et ciblée sur les erreurs typiques des apprenants d'après l'étude du corpus [7] (Walther Green, 2002). Enfin, un mentor sous forme d'un crayon animé guide l'apprenant en lui suggérant des activités à faire ou l'utilisation des différents outils linguistiques.

FreeText dispose de 16 types d'exercices différents :

- *Boites textes* :
  - *Création* : l'apprenant doit écrire, en production libre, un texte répondant aux instructions de l'énoncé, à l'aide du diagnostic d'erreurs, comme l'illustre la figure (4.3) ;
  - *Question ouverte* : l'apprenant doit écrire, en production libre, un texte répondant aux instructions de l'énoncé, avec l'aide de tous les outils disponibles. Les deux premiers cas ne se distinguent que par le type d'aide ;

6. Idéalement, les enseignants devraient même pouvoir marquer un diagnostic incorrect, voire le corriger, afin de permettre de mesurer le taux d'erreur des outils. De même, si une erreur n'a pas été marquée, il faudrait également pouvoir la signaler.

7. Les textes des tutoriels n'étaient hélas pas tous choisis et le corpus n'était pas complet au moment de la rédaction de la grammaire, au début du projet.

FIG. 4.3 – *FreeText: exercice à réponse ouverte*

  - *Réécriture* : l'apprenant doit réécrire ou corriger un texte ;
  - *Dictée* : l'apprenant doit retranscrire un texte audio ;
  - *Enumération* : l'apprenant doit saisir sa réponse sous forme de liste énumérée.
- *Texte lacunaire* :
  - choix sous forme de *boîte de sélection* ;
  - mots d'une *liste à recopier* et éventuellement adapter (conjuguer) dans les lacunes, une seule réponse possible ;
  - même principe mais *plusieurs réponses* possibles ;
  - choix par *glisser-déplacer* ;
  - choix par *appariement* de plusieurs listes sous forme de boîte de sélection.
- *Association* :
  - *glisser-déposer* : l'apprenant doit répondre aux instruction en glissant des étiquettes et en les déposant au bon endroit ;
  - *glisser-déposer multiple* : l'apprenant doit répondre aux instruction en glissant des étiquettes et en les déposant au bon endroit, avec la bonne combinaison.
- *Méli-mélo* :
  - *glisser-déposer* : l'apprenant doit remettre des étiquettes dans le

bon ordre pour reconstituer des phrases ;
  - *puzzle* : même principe, mais pour recomposer un texte.
- *Questionnaire à Choix Multiples* (QCM) :
  - à réponse *unique* ;
  - à réponses *multiples* ;
- *Mots croisés.*

Seuls les exercices de la première catégorie sont corrigés par les outils de diagnostic. Nous reviendrons au chapitre 5 sur les diverses aides à l'apprentissage offertes grâce au diagnostic d'erreur et sur les améliorations possibles. Bornons-nous pour l'instant à regretter que l'outil de synthèse vocale n'offre pas une transcription de la phrase prononcée avec l'alphabet phonétique international [8], comme c'était le cas dans le projet SAFRAN (§B.2.30).

Pour conclure, nous pouvons constater que FreeText n'a pas de réel module pédagogique. En effet, le logiciel est organisé en tutoriels thématiques sur une approche communicative et non grammaticale. Il n'y a pas de progression ni d'objectifs chiffrables à atteindre et le contrôle de la navigation est entièrement laissé aux apprenants. Il serait dès lors difficile de proposer des activités en fonction des résultats de l'apprenant. Par contre, des améliorations sont possibles : comme nous le verrons au chapitre 5, il serait utile d'utiliser les résultats du diagnostic d'erreurs pour orienter l'apprenant directement vers des explications pour l'aider à remédier à ses erreurs et à les corriger.

4.2.4 Module d'interface

FreeText est à la fois disponible sur *Internet* et en application autonome tournant sous le système d'exploitation *Windows* [9]. Pour les utilisations en

8. `http://www.langsci.ucl.ac.uk/ipa/`, dernière consultation le 16.8.2009.

9. Le logiciel tourne grâce au paquet *EasyPhp* (`http://www.easyphp.org/`, dernière consultation le 12.8.09). Ce paquet installe et intègre un serveur *web Apache* (`http://www.apache.org/`, dernière consultation le 12.8.09), un serveur de base de données *MySQL* (`http://www.mysql.org/`, dernière consultation le 12.8.09) et l'interpréteur de scripts *PHP* ; enfin, l'interface de gestion de base de données *MySQL PhpMyAdmin* permet de gérer facilement les bases, les tables et les utilisateurs. La combinaison *Apache-MySQL-PHP* est très répandue car elle est entièrement basée sur des logiciels libres ; on connait cette combinaison sous les sigles WAMP avec *Windows*, LAMP avec *Linux* et MAMP avec *Macintosh*. Les différents outils fonctionnent grâce à des programmes exécutables *CGI* (V. note 16 p. 34) qui font appel à des bibliothèques externes de liens dynamiques *DLL* (*Dynamic Link Library*, fonctions partagées à disposition d'une ou plusieurs applications informatiques) qui contiennent les différents outils linguistiques. Ensuite, les scripts

salle de classe (*intranet*) ou sur *Internet*, le logiciel doit être installé sur un PC ou un serveur sous *Windows* branché sur un réseau informatique [10].

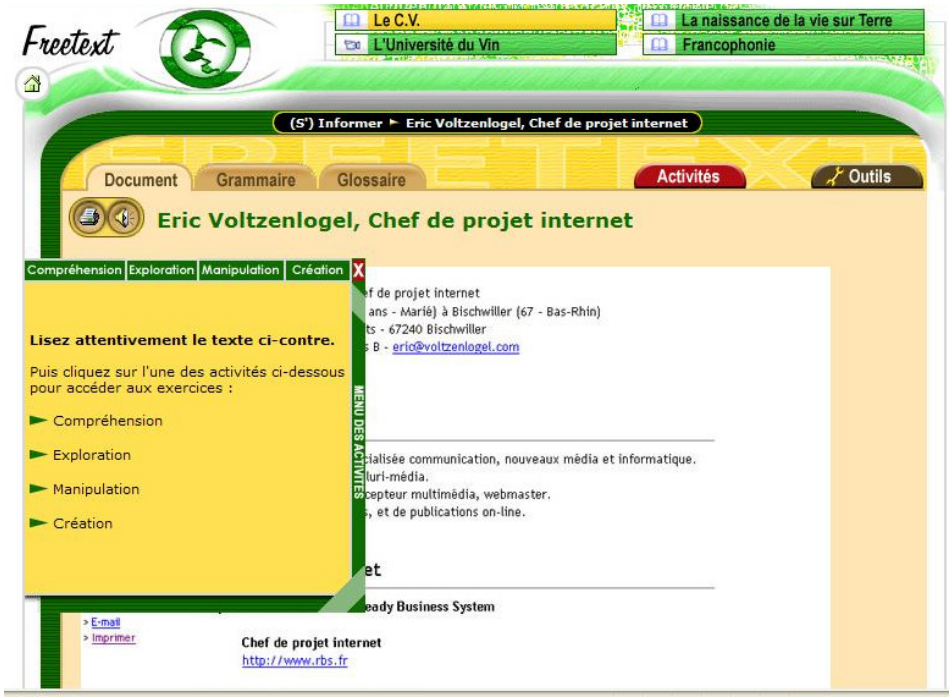


FIG. 4.4 – *FreeText: tutoriel*

Pour les tutoriels, un menu escamotable permet aux apprenants d'accéder aux différentes activités, comme l'illustre la figure (4.4). Des onglets permettent de basculer entre les différents domaines d'activité (§4.1). Les outils linguistiques sont accessibles en tout temps dans une seconde fenêtre du navigateur. Outre les tutoriels, le logiciel dispose d'un module statistique, accessible aux apprenants et aux enseignants, ainsi que d'une interface de mise à jour pour que les enseignants puissent introduire de nouveaux documents. Enfin, une visite guidée offre un tour d'horizon des caractéristiques du logiciel pour les apprenants à travers des captures d'écran animées et commentées.

Pour conclure, en tant que lien entre les différents modules, le module d'interface joue un rôle central. Il tire plutôt bien parti des technologies à disposition au moment de sa conception. Toutefois, il est regrettable que le

PHP formatent les différents résultats. Les animations sont en *Flash*, les interactions des exercices et des tutoriels sont gérées pour la plupart en *Javascript* ou en *DHTML* (V. note 18 p. 35). Pour des raisons techniques, le logiciel n'est compatible qu'avec le navigateur *Internet Explorer*.

10. Le logiciel a été testé sur de nombreuses versions du système d'exploitation *Windows*, tant pour des serveurs que pour des postes de travail.

logiciel ne puisse pas fonctionner sur tous les navigateurs. De plus, le code n'est pas conçu de manière optimale et modulaire, ce qui empêche l'évolution vers des versions ultérieures de *PHP*, ce qui est souvent nécessaire, voire impératif, pour corriger les failles de sécurité [11].

4.3 Bilan et discussion

FreeText a fait l'objet d'évaluations variées, tant à l'interne qu'à l'externe du projet. Commençons par les évaluations externes. Tout d'abord, une version préliminaire du prototype a été évaluée par des enseignants. Par ailleurs, certains enseignants ont accepté de faire utiliser le logiciel par leurs apprenants. Toutes les interactions ont été enregistrées par le système de traçage du logiciel. En outre, les apprenants ont été invités à remplir un questionnaire sur divers aspects du logiciel. Les principaux enseignements de cette évaluation ont été le bon accueil des exercices, à part les exercices de création. On a également noté un accueil mitigé pour les exercices dont les réponses étaient évaluées par comparaison avec une réponse modèle. Les outils de diagnostic ont été jugés intéressants mais pas totalement fiables. Globalement, le logiciel a été jugé agréable à utiliser et les apprenants ont jugé à 70% qu'il pouvait les aider à s'améliorer.

En outre, des enseignants familiers avec l'ALAO, des experts en technologie de l'information et un expert en ALAO ont été sollicités pour tester le prototype dans son état quasi final. Leur évaluation a souligné quelques insuffisances techniques dues aux difficultés d'accéder à certains outils ou à des vidéos trop longues à télécharger. L'approche pédagogique a reçu des avis positifs, à la nuance près que l'aspect de la compréhension et de l'expression orale était insuffisamment abordé. Les outils de vérification orthographique et de diagnostic d'erreur ont été davantage critiqués. Le niveau a été jugé trop élevé pour des apprenants de niveau intermédiaire. Mais globalement le prototype a été jugé adéquat pour un complément à l'utilisation en classe et pour des exercices de remédiation. Des professeurs de français ont également testé le synthétiseur (Cassart *et al.*, 2002) et ont déploré la mauvaise qualité de la synthèse ainsi que de trop nombreuses erreurs de prononciation, qui n'apportaient aucune plus-value au logiciel, voire pouvaient induire les apprenants en erreur.

11. Malheureusement, les scripts *PHP* ont été développés avec la version 4.0.6 du langage et nécessiteraient une adaptation conséquente pour fonctionner avec les versions actuelles du paquet *EasyPhp* (*PHP* version 5.3), notamment à cause des librairies *XML* (§2.7.5) qui sont incompatibles.

Par ailleurs, une évaluation comparative a été réalisée par une experte externe, qui a comparé FreeText avec deux logiciels commerciaux d'apprentissage du français, *Tell me More* [12] et *Interaktive Sprachreise* [13]. Enfin trois évaluations qualitatives, pour chaque année du projet, ont été produites par un expert externe.

Passons aux évaluations internes : de nombreuses validations ont été effectuées par d'autres équipes que celle responsable de la partie évaluée. Les tutoriels eux-même ont été notamment évalués sur le plan de la pertinence des contenus, des erreurs, de l'adéquation des exercices avec les différents buts, ainsi que de la cohérence des notions avec la grammaire de référence. Le correcteur orthographique et le diagnostic d'erreurs ont également été validés à l'interne : à deux reprises, des phrases extraites de l'évaluation du prototype par des apprenants ont été soumises aux outils et leurs résultats ont été comparés avec une correction manuelle. L'évaluation a notamment montré que le diagnostic d'erreurs fonctionnait correctement pour l'orthographe, la flexion, l'euphonie, le nombre et la personne, mais que des progrès devaient être faits pour la classe, l'ordre des mots, le genre et la confusion.

En outre, la synthèse vocale a aussi été testée, en demandant à des apprenants de comparer des phrases prononcées par la synthèse vocale avec la même phrase prononcée par des locuteurs natifs, en transcrivant phonétiquement et orthographiquement ce qu'ils entendaient [14]. Cette évaluation a relevé des erreurs de prononciation, notamment pour des mots entièrement en majuscules non accentuées ou pour des erreurs de résolution d'abréviations et d'homophones, ainsi que des lacunes dans le lexique. Le test a également révélé que la synthèse vocale convenait mal pour prononcer des dialogues. Par contre, le synthétiseur est adéquat pour lire des textes et les instructions du logiciel. Cependant, l'évaluation a relevé qu'il était hautement recommandé de tester la synthèse en introduisant de nouveaux document et exercices, afin de vérifier la présence d'erreurs. Elle a aussi souligné qu'il serait souhaitable de tirer partie des indications fournies par la mise en forme du texte, telles que les listes ou les titres. Enfin, il est paru indiqué que différentes options de paramétrage de la synthèse soient disponibles, comme la lecture des signes de ponctuation pour les dictées, mais aussi un style plus ou moins formel.

Passons maintenant à notre évaluation personnelle du logiciel. Comme

12. `http://fr.tellmemore.com/`, dernier accès le 16.11.2009.
13. `http://www.digitalpublishing.de/`, dernier accès le 16.11.2009.
14. L'étude n'a pas testé les progrès des apprenants, tout en relevant qu'une telle évaluation serait nécessaire. Le dispositif de test devrait diviser, comme c'est l'usage, les apprenants en deux groupes, l'un utilisant la synthèse vocale et l'autre des fichiers enregistrés par des locuteurs natifs. Les performances seraient mesurées à l'aide d'un pré- et d'un post-test.

nous l'avons souligné, les modules les plus importants de FreeText sont le module expert et le module d'interface. Les autres modules sont beaucoup plus réduits. Par ailleurs, il n'y a pas ou peu d'interaction entre les différents modules, bien qu'elle soit possible et relativement aisée à mettre en place, comme nous le montrerons au chapitre 5 : les erreurs détectées par la détection d'erreurs pourraient servir à guider l'apprenant vers la grammaire de référence et vers des exercices. De plus, des statistiques sur les erreurs courantes des apprenants peuvent aisément être extraites et exploitées par les enseignants. Enfin, à des fins de recherche, il est regrettable que les données ne soient pas exploitées pour un corpus [15].

Les objectifs du projet ont été globalement tenus de manière satisfaisante, bien qu'il a fallu les revoir à la baisse, notamment pour le nombre de tutoriels et le développement de certains outils. Au niveau pédagogique, le logiciel donne l'occasion à l'apprenant d'aborder une série de thèmes variés et offre de nombreux moyens de remédiation.

Poursuivons ce bilan par une évaluation technique. Entre le début et la fin du projet, la bande passante d'*Internet* et le taux d'équipement des écoles et des ménages en général ont été considérablement améliorés. Le choix de logiciels libres était tout aussi indiqué, afin d'éviter les problèmes de licence. Sur ces points, le choix était tout à fait judicieux. Par contre, *Internet* a pour inconvénient que les serveurs *web* doivent partager les ressources entre de nombreux utilisateurs simultanés. Par conséquent, les serveurs *web* sont configurés de manière à limiter la disponibilité de mémoire et de temps pour les processus. Il aurait été possible de paramétrer l'installation du logiciel en fonction de l'utilisation sur un ordinateur personnel autonome ou sur un serveur *web* destiné à de nombreux utilisateurs, qui peut de plus être utilisé par plusieurs applications. Néanmoins, les ressources disponibles seraient de toute manière moindre qu'avec un logiciel complet et autonome, directement installé sur un système d'exploitation sans dépendre d'un navigateur ni d'un serveur. En outre, le code des applications de TAL n'a pas été suffisamment optimisé et les processus prennent un temps considérable, voire aboutissent à un dépassement de la limite de limite de temps d'exécution (*timeout*). Nous en concluons qu'il était et qu'il est toujours peu indiqué d'utiliser des applications *Internet* pour des tâches aussi gourmandes en ressources.

Par ailleurs, FreeText n'est disponible que sous *Windows* et l'installation n'est donc pas possible sur d'autres systèmes d'exploitation. De plus, l'installation du logiciel a posé de nombreux problèmes techniques, notam-

15. Afin de garantir la cohérence des données et pour pallier aux faiblesses et à certains manques de fiabilité de la détection, un tel corpus devrait être révisé par un expert humain, comme nous l'avons fait pour nos recherches sur le correcteur orthographique (§6.4.2).

ment dans les institutions [16]. Enfin, nous avons déjà souligné les problèmes d'adaptation des scripts de FreeText aux versions ultérieures de *PHP*. Toutes ces limitations n'ont pas été correctement anticipées et ont malheureusement eu un impact négatif sur les phases de test du logiciel et sur sa pérennité.

Pour conclure, malgré les aspects relevés ici, FreeText apporte une contribution significative au domaine de l'ALIAO. Malgré l'envergure moyenne du projet, le nombre d'outils en jeu, leur complexité et leur couverture sont considérables. Relevons aussi que ce projet interdisciplinaire a nécessité de nombreuses compétences et a suscité de nombreux échanges passionnants du point de vue scientifique. C'est pourquoi FreeText a eu un écho important au sein de la communauté scientifique.

16. Comme nous l'avons souligné, FreeText est gourmand en ressources. Il est dès lors recommandé de le faire tourner sur un serveur puissant, surtout pour une utilisation simultanée en classe, ce qui a posé notamment les problèmes suivants :

i. les serveurs dans les institutions tournent plutôt sous *Linux* que sous *Windows* ;
ii. les serveurs *Windows* font tourner le module de serveur Web *Internet Information Services* (IIS), ou d'autres serveurs *Apache*, ce qui nécessite une configuration personnalisée du logiciel et des compétences dont certaines institutions ne disposaient pas ;
iii. si un simple ordinateur personnel était utilisé comme serveur pour FreeText, il ne disposait pas forcément de la puissance requise (processeurs, mémoire et accès disque) et saturait rapidement.

Chapitre 5

L'analyseur *Fips* et son application à l'ALAO

Dans ce court chapitre, nous présentons l'analyseur *Fips* qui a servi de base à nos travaux d'application du TALN à l'ALAO, notamment dans le cadre des projets FreeText (§4) et SAFRAN (§B.2.30). Dans un premier temps, nous décrivons l'analyseur et son fonctionnement (§5.1). Ensuite, nous décrivons la détection des erreurs (§5.2). Nous poursuivons par une description de la sortie de l'analyseur et son utilisation pour des applications pédagogiques (§5.3) et terminons par une discussion finale (§5.4).

5.1 Description de l'analyseur

Développé à partir de 1991 et en constante évolution depuis, l'analyseur *French Interactive Parser System* [1] (*Fips*, Laenzlinger et Wehrli, 1991; Wehrli, 1997; Goldman *et al.*, 2000; Berthouzoz, 2000; Wehrli, 2004b, 2006, 2007; Wehrli et Nerima, 2009) est un analyseur tabulaire (*chart parser*) qui implémente la théorie du gouvernement et du liage (*Government & Binding*, §3.3.5) et sa version des *Principes & Paramètres* (Chomsky et Lasnik, 1995; Tellier, 1995; Laenzlinger, 2003), avec des emprunts à la théorie mi-

1. Les premières versions de l'analyseur étaient dotées de modules interactifs de désambiguïsation de l'analyse, mais cette fonctionnalité a été rapidement abandonnée.

nimaliste (Chomsky, 1995) [2], à la LFG (§3.3.5) et à la *Syntaxe Simplifiée* [3] (*Simpler Syntax*, Culicover et Jackendoff, 2005, 2006). *Fips* est un analyseur ascendant, guidé par le lexique (*data driven*) de type validation (*licensing parser*, Abney, 1989) [4] et traite les analyses alternatives en parallèle. Ainsi, toutes les variantes possibles sont analysées avant d'être éliminées par des filtres descendants tels que le module thématique, qui vérifie que les rôles thématiques (v. p. 119) correspondant aux arguments du verbe sont bien remplis.

Après un découpage de la phrase en unités lexicales, toutes les lectures possibles d'un mot (les lexèmes) sont projetées en structures syntaxiques dans un schéma X-barre (v. 3.6 p. 120) simplifié donné en (5.1).

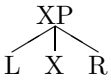


FIG. 5.1 – *Schéma X-barre simplifié*

Au lieu des habituels Spécifieurs, Compléments et positions d'ajout, le schéma simplifié dispose simplement de deux listes de sous-constituants à gauche (L, *left*) et à droite (R, *right*). Les deux listes et la tête X peuvent être vides. Le niveau barre, qui sert notamment à distinguer ajouts et compléments, est supprimé. Les catégories sont Adv (Adverbe), A (Adjectif), N (Nom), D (déterminant), V (verbe), P (préposition), C (Conjonction), Interj (Interjection), T (Inflection / Temps) [5]. Après le processus de projection, chaque candidat est alors inséré dans un graphe ou agenda, appelé

2. Il s'agit d'une simplification de la théorie des *Principes & Paramètres* qui prône l'économie de *dérivation* et l'économie de *représentation*. Les mouvements et transformations n'ont lieu que pour ajouter du sens. Les structures profondes et structures de surface sont éliminées. La *forme logique* représente le sens de la phrase (les clauses), et la *forme phonologique* contient les paramètres phonétiques et prosodiques de la phrase. Dans le schéma X-Barre (v. p. 120), les niveaux XP et X-Barre ne sont représentés que s'ils contiennent des constituants. La grammaire ne propose que deux types d'opération, le mouvement (*move*), qui représente le déplacement des constituants et leur transformation morphologique, et la fusion (*merge*), qui représente la combinaison des constituants suivant des vérifications de contraintes.

3. Cette théorie postule que non seulement la syntaxe, mais encore la phonétique et la sémantique ont une capacité générative et dépendent les unes des autres de manière non hiérarchique. La syntaxe simplifiée laisse un rôle plus important à la sémantique et ne postule pas de catégories vides ou de copies de constituants.

4. Les têtes peuvent valider des relations à leur gauche comme à leur droite.

5. La catégorie T correspond à la catégorie I de GB présentée au §3.3.5. Lorsqu'une forme verbale est lue, un syntagme verbal VP est projeté. Dans tous les cas sauf les participes, une autre projection TP est projetée, avec une tête contenant un marqueur de temps et un VP en complément. En français, pour les temps simples, le verbe remonte en T^0 et laisse une trace en V^0, mais pas en anglais. Ainsi, *raconte* donne lieu à une projection $[_{TP}\ [_{T}$ raconte$]$ $[_{VP}\ [_{V}\ e]]]$, où e représente la trace laissée par le mouvement de la tête verbale (v. p. 120). Des projections complexes ont aussi lieu pour les DP à valeur adverbiale, répondant à des

chart.

Fips traite les analyses en parallèle de gauche à droite. Chaque mot nouveau inséré dans le graphe peut être associé à son contexte gauche. *Fips* tente donc de combiner le nouveau constituant :

- soit en attachant le nouveau constituant à droite d'un constituant du contexte gauche ; cette stratégie est appelée algorithme du *coin droit* ;
- soit en attachant le constituant du contexte gauche comme spécifieur ou ajout du nouveau constituant.

L'opération de combinaison est appelée *Merge*. Ces combinaisons sont régies par des règles grammaticales associées à des contraintes, notamment sur l'accord, les traits lexicaux ou la structure argumentale. L'attachement à droite peut se faire sur n'importe quel nœud actif, c'est-à-dire tous les nœuds susceptibles de recevoir un attachement ; cette opération est possible, que ce soit comme attachement direct au constituant gauche ou comme attachement à un des sous-constituants actifs. Enfin, différents processus vérifient la grammaticalité par stratégie descendante, notamment :

- vérification des tables d'arguments (essentiellement des verbes) ;
- module du liage (ou de déplacement, *Move*), qui lie l'élément extraposé (comme les pronoms interrogatifs) à un élément vide dans la position canonique ;
- modification des structures argumentales des prédicats (ajout, effacement ou modification d'arguments lors des structures passives ou causatives).

Nous décrirons plus en détail le processus d'analyse de *Fips* au paragraphe 5.2.1. Afin de filtrer les alternatives, un score est alloué à chaque analyse, selon des critères de préférence psycholinguistique, comme l'attachement du groupe prépositionnel au site d'attachement le plus bas. Par défaut, l'analyseur ne retourne comme résultat que l'alternative ayant le plus bas score, mais il est possible d'afficher plusieurs alternatives. En cas d'échec de l'analyse, *Fips* présente plusieurs morceaux d'analyses (*chunks*, §3.3.4.3). Par ailleurs, les mots constituant potentiellement une collocation (v. p. 66) sont identifiés dans le lexique ; si de tels mots apparaissent dans les positions syntaxiques pouvant contenir des collocations, *Fips* consulte une base de données de collocations pour valider le choix et préférer une analyse. Comme une collocation peut contenir une autre collocation, le choix est récursif comme pour *travail à temps partiel*, *tomber en panne sèche*, etc.

contraintes spécifiques : par ex., *la semaine dernière* ou *le lundi* sont analysés comme des DP, mais aussi des AdvP avec DP comme complément.

Wehrli (2004b, 2007) décrit l'orientation récente de *Fips* vers une analyse multilingue, pour l'anglais, le français, l'allemand, l'italien, le castillan et le grec. Les principes de la grammaire GB permettent de tirer parti des caractéristiques communes des langues. L'ajout d'une nouvelle langue doit être facile et ne pas perturber le système. L'analyseur a donc été réécrit de manière modulaire en utilisant la programmation orientée-objet (Meyer, 2008). Ainsi, les objets décrivant les langues italienne et française héritent d'un objet décrivant les langues romanes, qui partagent un certain nombre de caractéristiques communes. L'objet contenant un item lexical (nom, nom composé, expression idiomatique [6]) peut être étendu pour une langue particulière, afin de traiter de phénomènes qui lui sont propres. Il en va de même de l'objet décrivant les projections, qui, pour les langues romanes, contient des listes supplémentaires pour traiter les clitiques.

Fips est utilisé comme analyseur et comme étiqueteur (§3.1.2). Ses analyses servent entre autres de base à la synthèse vocale *FipsVox* (Gaudinat et Wehrli, 1997; Gaudinat et Goldman, 1998; Goldman, 2001, §3.1.4), à la traduction automatique (Ramluckun et Wehrli, 1993; L'haire *et al.*, 2000; Laenzlinger *et al.*, 2000, §3.1.6), à l'extraction de collocations (Seretan, 2008) et à la traduction de mots et de collocations en contexte (Wehrli, 2004a, 2006; Seretan et Wehrli, 2006; Nerima *et al.*, 2006; Seretan et Wehrli, 2007).

5.2 Détection d'erreurs syntaxiques

Dans cette section, nous décrivons la détection d'erreurs syntaxiques à l'aide de l'analyseur *Fips*. Pour de plus amples informations, on lira une description détaillée des techniques dans la thèse de Vandeventer Faltin (2003), qui a rédigé les spécifications du relâchement de contraintes, de la réinterprétation phonologique ainsi que les algorithmes de classement des résultats. D'autres publications sont également parues sur le sujet (Vandeventer, 2000, 2001; L'haire et Vandeventer Faltin, 2003a,b; L'haire, 2004; Walther Green, 2004). Le tableau (5.1), repris de Vandeventer Faltin (2003), décrit les classes d'erreurs détectées par *Fips*.

Cette taxonomie des erreurs est fortement inspirée par celle du corpus *FRIDA* (Granger, 2003, §4). Examinons ces catégories plus en détail. Les erreurs d'auxiliaire (AUX) concernent la confusion entre auxiliaire *être* et *avoir*. La détection de l'emploi erroné du verbe *avoir* à la place du verbe *être* est rendue délicate par le cas des tournures passives. De même, les erreurs de voix (VOI), peu fréquentes dans le corpus, concernent principalement un

6. V. p. 66.

Cat	Description	Exemple	R	P	A
AUX	Mauvais choix d'auxiliaire	**Il* ***a*** *venu hier*	X		
CLA	Emploi d'une mauvaise classe de mot (catégorie lexicale ou sous-type de catégorie)	*** ***Ce*** *me plaît.*	X		
CPA	Complément d'adjectif incorrect	**Il est* ***fier à*** *Marie.*	X		X
CPV	Complément verbal incorrect	**Il* ***lui*** *regarde.*	X		X
EUF	Erreur d'euphonie	**J'expose* ***le*** *argument principal.*	X		X
GEN	Erreur d'accord en genre	*** ***Elle*** *est* ***venu*** *hier.*	X	X	X
HOM	Utilisation d'un homophone incorrect	**Il faut* ***maître*** *au monde des enfants.*		X	
MAN	Elément manquant dans la phrase	**Ils ont de* _ *force.*	X		X
NBR	Erreur d'accord en nombre	*** ***Ils pose*** *les questions.*	X	X	X
NEG	Négation superflue ou manquante	**C'*_ *est pas un problème.*	X		
ORD	Mauvais ordre des mots	**Une* ***intelligente*** *femme.*	X		X
OUB	Oubli de ponctuation	**Les pommes* _ *les poires et les raisins*		X	X
PER	Erreur d'accord en personne	*** ***Je dort***	X	X	X
VOI	Erreur de voix : utilisation incorrecte de la lecture pronominale d'un verbe	**J'* _ *évanouis.*	X		

La colonne R reprend les erreurs détectées par relâchement de contraintes, la colonne P les erreurs détectées par relâchement phonologique et la colonne A les erreurs détectées par la méthode *ad hoc*.

TAB. 5.1 – *Fips – erreurs détectées et techniques en jeu*

mauvais emploi ou l'absence d'emploi de la variante pronominale d'un verbe. Par contre, les erreurs de classe (CLA) sont de nature différente. Dans cette catégorie entre la confusion entre article défini et indéfini (*le / un*) ou entre pronom démonstratif et possessif (*ces / ses*), etc.

De leur côté, les erreurs de complément d'adjectif (CPA) et de verbe

(CPV) concernent les prépositions des compléments ou d'autres erreurs de type [7]. Les erreurs d'euphonie (EUF) sont très fréquentes et englobent les erreurs d'élision et de contraction.

Quant aux erreurs d'ordre des mots, elles impliquent le cœur même d'un analyseur syntaxique, puisque celui-ci dépend des règles d'assemblage des constituants et de l'algorithme d'analyse. Dans notre système, les règles d'ordre sont subdivisées entre règles sur l'ordre des adjectifs (ORDAJ) et sur l'ordre des adverbes (ORDAV). En outre, dans une moindre mesure, l'ordre des verbes et des pronoms est traité dans certains cas. Comme nous l'avons déjà maintes fois signalé, le relâchement de contraintes doit être manipulé avec précaution, sous peine de voir une surgénération d'analyses.

En revanche, comme nous l'avons déjà constaté dans cet ouvrage, les erreurs d'accord sont très fréquentes et particulièrement bien traitées par relâchement de contraintes. On y trouve les erreurs de genre (GEN), de nombre (NBR) et de personne (PER) : les apprenants ont souvent des difficultés à apprendre les genres des mots et se trompent également fréquemment de conjugaison. Quant aux erreurs d'homonymie (HOM), aussi très présentes, elles sont des confusions entre mots qui ont des prononciations analogues (*prémisses / prémices*, *son / sont*, etc.). Cependant, comme nous l'avons vu à la section 3.3.3.2, seules les erreurs portant sur des mots de différentes catégories lexicales peuvent être détectées à l'aide d'un analyseur syntaxique.

Enfin, nous pouvons ranger dans la même catégorie les erreurs d'élément manquant (MAN), d'oubli de ponctuation (OUB) et de négation (NEG). Bien que fréquentes, les erreurs d'élément manquant sont difficiles à détecter car il faut suppléer un élément qui n'est pas présent. Les erreurs de ponctuation, très fréquentes, concernent particulièrement l'oubli de la virgule ; l'emploi de la ponctuation est essentiellement une question de style et les règles en jeu ne sont pas très précises. Par contre, les erreurs de négation concernent l'omission des particules *ne* ou *pas*.

Dans cette section, nous décrivons la technique de relâchement de contraintes (§5.2.1). Puis nous poursuivons par la réinterprétation phonologique (§5.2.2) [8]. Ensuite, nous présentons l'ordre d'application des techniques (§5.2.3). Enfin, nous décrivons l'algorithme de classement des résultats (§5.2.4).

7. *FRIDA* étiquette également les erreurs d'adjectifs prédicatifs comme des CPA.
8. Prévue au début du projet, la méthode *ad hoc* de réinterprétation de morceaux présentée à la section 3.3.3 a finalement été abandonnée.

5.2.1 Relâchement de contraintes

Commençons par examiner comment *Fips* utilise le relâchement de contraintes, dont les principes sont énoncés à la section 3.3.2.2. Les entrées lexicales sont détaillées à l'annexe D. Prenons la phrase suivante, tirée du corpus FRIDA (Granger, 2003, §4) :

(31) *L'héritage du passé est très forte et le sexisme est toujours présent.

Dans la première partie de la phrase coordonnée, le syntagme adjectival *très forte* doit s'accorder avec le sujet *l'héritage du passé*. Le tableau en (§D.2) représente les entrées lexicales du mot *forte*. L'emploi adverbial, prononcé [fɔʁte], est utilisé dans un contexte musical (jouer *forte*). Le mot *héritage* contient une seule entrée, présentée en (§D.1). Enfin, le mot *est* correspond à pas moins de huit entrées, illustrées en (§D.3).

Examinons le processus d'analyse de la phrase (31). Ce processus est légèrement simplifié pour raisons de clarté.

i. L'analyseur lit l'élément *l'* et projette un constituant DP, ainsi qu'un autre constituant pour le clitique.
ii. Aucun élément n'existe dans l'agenda et aucune combinaison ne peut avoir lieu.
iii. Le mot *héritage* est lu et est projeté dans un NP, avec les informations présentées en (§D.1).
iv. Le constituant NP est adjoint à droite du DP présent dans l'agenda.
v. Le mot *du* est lu et projeté dans un constituant $[_{\texttt{PP}}$ du $[_{\texttt{DP}}$ $]]$.
vi. Le PP est attaché à droite du NP précédent.
vii. Le mot *passé* est lu. Il est projeté en NP.
viii. Ce NP attaché à droite du DP le plus profond dans la liste. La structure analysée est
$[_{\texttt{DP}}$ l' $[_{\texttt{NP}}$ héritage $[_{\texttt{PP}}$ du $[_{\texttt{DP}}$ $[_{\texttt{NP}}$ passé$]]]]]$
ix. Le mot *est* est lu. Chacune des huit variantes listées en (§D.3) donne lieu à une projection. Toutes les variantes verbales sont notamment projetées dans un constituant $[_{\texttt{TP}}$ est$_i$ $[_{\texttt{VP}}$ $e_i]]$.
x. Le DP complexe dans l'agenda peut être attaché à gauche des TP présents dans l'agenda, ce qui donne la structure
$[_{\texttt{TP}}$ $[_{\texttt{DP}}$ l' $[_{\texttt{NP}}$ héritage $[_{\texttt{PP}}$ du $[_{\texttt{DP}}$ $[_{\texttt{NP}}$ passé$]]]]]$ $[_{\texttt{T}}$ est$_i]$ $[_{\texttt{VP}}$ $e_i]]$. Les autres lectures de *est* ne permettent pas de combinaison.
xi. Le mot *très* est lu. Il donne lieu à une projection AdvP.

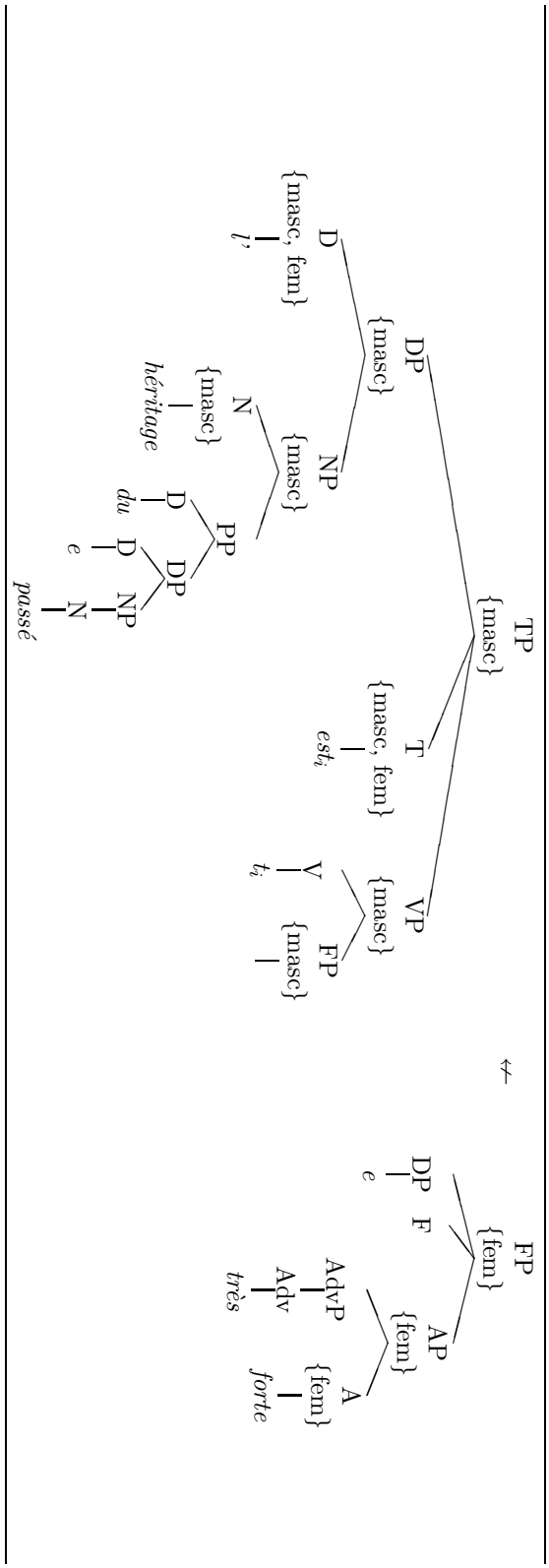


FIG. 5.2 – *Fips – échec d'analyse de la phrase (31)*

xii. Ce constituant AdvP ne peut pas être combiné avec les constituants précédents.

xiii. Le mot *forte* est lu. Chacune des variantes en (§D.2) est projetée en des constituants AdvP et AP. Par ailleurs, comme les adjectifs peuvent donner lieu à des propositions réduites [9], un constituant $[_{\texttt{FP}}\ [_{\texttt{DP}}\ \text{e}]\ [_{\texttt{F}}\ [_{\texttt{AP}}\ [_{\texttt{DP}}\ \text{e}]\ [_{\texttt{A}}\ \text{forte}]]]]$ est également construit.

xiv. Le constituant $[_{\texttt{AdvP}}\ \text{très}]$ peut être adjoint à gauche des constituants AdjP et AP de l'étape précédente, y compris le AP inclus dans le FP.

xv. Le constituant FP peut être attaché à droite du VP projeté à partir de la lecture (3) du (§D.3). Cependant, l'accord en genre entre le VP et le FP ne peut être vérifié et l'attachement est impossible, comme l'illustre la figure (5.2).

xvi. Les contraintes d'accord sont relâchées. L'analyseur procède à l'union des traits de genre du verbe et de l'adjectif, au lieu d'en faire l'intersection. Les traits sont mis à jour et le constituant FP est attaché à droite du VP. Le diagnostic de l'erreur est déplacé vers le NP sujet. Ce procédé est illustré à la figure (5.3).

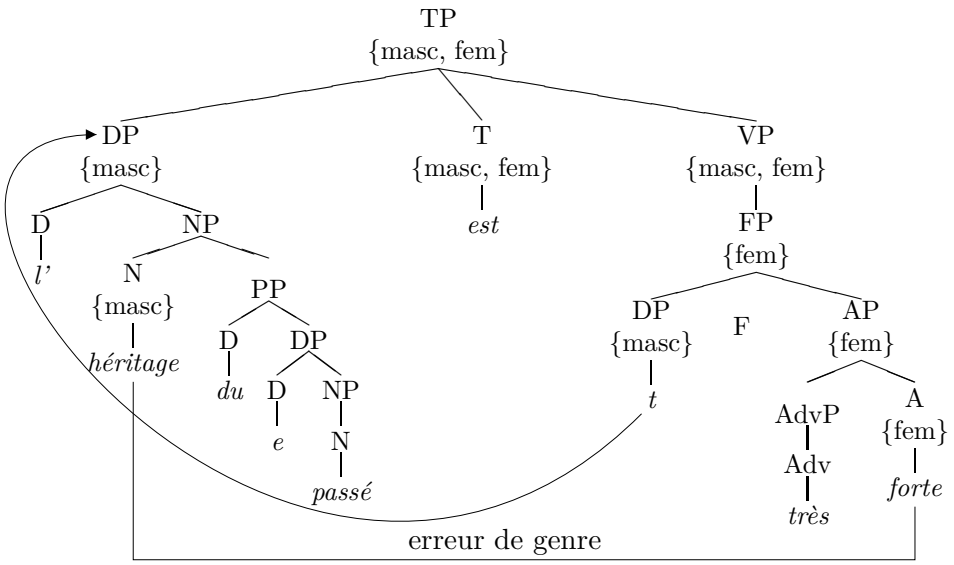


FIG. 5.3 – *Fips – analyse de la phrase (31) avec relâchement de contraintes*

Comme nous l'avons vu au paragraphe 3.3.2.2, il semble raisonnable de ne pas relâcher toutes les contraintes en même temps. Vandeventer Faltin (2003, p. 128) propose de sélectionner les catégories d'erreurs à traiter par

9. Les verbes au participe passé peuvent également donner lieu à une proposition réduite : *pressés par le temps, nous avons fait fausse route.*

relâchement de contraintes puis d'identifier les contraintes en jeu qui peuvent être relâchées. Ainsi, l'erreur de genre que nous venons d'analyser concerne l'accord sujet-adjectif prédicatif. Mais l'on trouve aussi des erreurs d'accord déterminant-syntagme nominal (**le maison*) et d'accord du participe passé (**la pomme que j'ai mangé*). Dans *Fips*, les contraintes ne sont relâchées qu'après avoir vérifié les conditions nécessaires pour préciser le contexte. Comme nous l'avons vu dans notre exemple, ces conditions sont associées à des actions qui permettent d'attacher les constituants (ici, l'union des traits de genre) et de préciser le diagnostic (attachement du diagnostic d'erreur au sujet plutôt qu'au verbe).

D'autres types d'erreur peuvent également être traités par relâchement de contrainte. Une description plus détaillées peut être consultée chez Vandeventer Faltin (2003). Examinons maintenant les différents types d'erreur et la couverture de celle-ci :

i. les erreurs d'emploi de l'auxiliaire (AUX) sont détectées de manière satisfaisante. Pour éviter une sur-détection, les erreurs d'emploi du verbe *être* à la place d'*avoir* ne sont signalées que pour les verbes qui n'acceptent pas la construction passive ;
ii. les erreurs de classe (CLA) sont difficiles à détecter sans traitement sémantique. Cependant, nous détectons l'emploi erroné du pronom *ce* en tant que sujet, ainsi que l'emploi d'un pronom de mauvaise catégorie (**qui sont **eux**?*) ;
iii. les erreurs de complément d'adjectif (CPA) causent une surdétection à cause de l'attachement du groupe prépositionnel : il est en effet difficile de distinguer compléments et ajouts ;
iv. les erreurs de complément verbaux (CPV) sont également difficiles à traiter. En effet, il est difficile de choisir le bon lexème du verbe lorsqu'il y a une erreur ; c'est pourquoi tous les lexèmes sont pris en considération et les arguments verbaux sont vérifiés un à un. En outre, dans ce cas aussi, l'attachement des groupes prépositionnels pose problème ;
v. les erreurs d'euphonie (EUF) couvrent les contraction préposition-déterminant (*du*, *au*), les élisions erronées et le *-t-* euphonique ;
vi. les erreurs d'accord en genre, nombre et personne (GEN, NBR, PER) sont traitées par paires de mots et dépendent des catégories lexicales des mots dont l'accord est vérifié ;
vii. les éléments manquants (MAN) ne sont pas détectés en tant que tels, mais entrent dans d'autres catégories (CPV, NEG, etc.) ;
viii. les erreurs de négation (NEG) sont détectées dans des cas simples, notamment lorsque l'un des éléments de la négation est présent et pas l'autre ;

ix. les erreurs d'ordre sont traitées catégorie par catégorie. L'ordre des adverbes (ORDAV) n'est que partiellement pris en compte, lorsque certains types d'adverbe sont attachés dans une position agrammaticale. Pour l'ordre des adjectifs (ORDAJ), nous détectons les adjectifs postnominaux utilisés en position prénominale (**une rouge pomme*) et vice-versa. L'ordre des verbes (ORD) est pris en compte partiellement ;

x. l'oubli de ponctuation (OUB) ne traite que le cas des traits d'union manquants, spécialement dans le cas de l'inversion du sujet dans une interrogative ;

xi. enfin, les erreurs de voix (VOI) ne sont que partiellement traitées : nous détectons les verbes essentiellement pronominaux qui sont utilisés sans pronom, ainsi que les verbes non pronominaux utilisés avec pronom réfléchi.

Le principal inconvénient de *Fips*, selon Vandeventer Faltin (2003), est paradoxalement sa sous-spécification des contraintes. En effet, l'analyseur est conçu pour être robuste et fournir une analyse complète ou partielle à la plupart des phrases. Ainsi, l'absence des arguments verbaux sont insuffisamment pris en compte. La phrase **je lui aide* est acceptée par la version standard de *Fips* ; le pronom clitique au Cas Datif est attaché comme ajout et l'absence d'un complément DP ou d'un clitique Accusatif n'entraîne pas de problème d'analyse. Tout au plus, le score de la phrase correcte *je l'aide* est nettement plus bas. Ainsi, pour fonctionner comme outil de diagnostic, de nouvelles contraintes ont dû être ajoutées. En outre, le lexique n'est pas toujours consistant. Par exemple, les adjectifs prénominaux ne sont pas toujours désignés comme tels et certaines sous-catégorisation de verbes manquent. Enfin, *Fips* rencontre également les problèmes de surgénération de structures et de sélection du bon diagnostic, dont nous avons parlé à la section 3.3.2.2. Nous reviendrons sur les solutions proposées au paragraphe 5.2.4.

5.2.2 Réinterprétation phonologique

Dans cette section, nous décrivons la technique de réinterprétation phonologique mise en place pour *Fips*. Cet analyseur dispose d'un lexique qui contient des correspondances phonétiques (§3.3.3.2) et d'un phonétiseur pour les chaînes de caractères qui ne sont pas dans le lexique (Gaudinat et Goldman, 1998; Goldman, 2001). De plus, *Fips* produit des analyses partielles en cas d'échec. Les conditions mentionnées par Vandeventer Faltin (2003) et reprises à la section 3.3.3.2 sont donc réunies. Prenons comme

exemple la phrase suivante :

(32) *Elles peuvent maître au monde des enfants.

La phrase (32) ne peut pas être analysée complètement. *Fips* retrouve les morceaux suivants :

i. [$_{DP}$ elles]
ii. [$_{TP}$ [$_{DP}$ e] [$_{T}$ peuvent [$_{VP}$]]]
iii. [$_{NP}$ maître [$_{AdvP}$ [$_{PP}$ au [$_{DP}$ [$_{NP}$ monde [$_{PP}$ des [$_{DP}$ [$_{NP}$ enfants]]]]]]]

La première étape du processus de réinterprétation est de trouver les mots en bordure des morceaux d'analyse. Les mots en tête et en fin de phrase ne sont pas pris en considération lorsqu'ils ne forment pas un morceau à eux seuls. Ensuite, on recherche dans le lexique toutes les prononciations alternatives de chaque mot. Pour notre phrase (32), nous retrouvons les éléments suivants :

i. elles [ɛl] : elles, aile, ailes.
ii. peuvent [pœv]
iii. maître [mɛtʁ] : mètre mètres, mettre maîtres.

Tous les mots réinterprétés sont insérés dans le graphe à condition qu'ils soient d'une autre catégorie lexicale que le mot original. Le pronom *elles* est de même catégorie (nom) que les autres propositions, dont aucune ne peut être retenue. Le verbe *peuvent* n'a aucun homophone. Par contre, le nom *maître* permet de trouver un verbe *mettre*. Ensuite, toutes les phrases possibles sont soumises à nouveau à l'analyseur, qui tente d'obtenir une ou plusieurs phrases complètes. Dans notre exemple, un seul élément peut être inséré dans le graphe. Ainsi, *Fips* peut analyser une phrase complète :

(33) Elles peuvent mettre au monde des enfants.

Soulignons que seules les analyses complètes sont retenues, car elles garantissent davantage un diagnostic correct. Par ailleurs, deux améliorations ont été apportées à cette technique. Le premier ajout concerne le traitement des consonnes latentes : si un mot est susceptible de contenir une consonne latente (§3.3.3.2, Hannahs, 2007), il est réinterprété à la fois avec et sans

consonne latente. Ceci permet de trouver plusieurs alternatives. Le second ajout concerne les mots multiples. Il arrive parfois que deux mots à la frontière d'un morceau d'analyse puissent être combinés en un seul mot.

(34) *[Vous pouvez] [peut être] [dormir].

Dans la phrase (34), les morceaux sont délimités par des crochets. La technique de réinterprétation de mots multiples consiste à rechercher dans le lexique phonétique les valeurs des deux mots à la frontière d'un morceau d'analyse. Cette technique permet parfois de retrouver des mots existants, comme *peut-être* dans notre exemple, qui permettent d'obtenir une analyse complète.

Terminons par quelque considérations sur les avantages et inconvénients de cette approche. Tout d'abord, elle rencontre les mêmes problèmes de surgénération de structures que le relâchement de contraintes. De plus, elle génère une ambiguïté lexicale supplémentaire en multipliant les alternatives. Il devient alors plus difficile de sélectionner la proposition de correction la meilleure possible. Pire, cette technique prend un temps d'analyse considérable si les phrases sont trop longues. C'est pourquoi nous avons dû la désactiver au-delà d'un certain seuil afin d'éviter l'échec de l'application pour dépassement de limite temporelle d'exécution (*timeout*) ; la méthode n'est pas activée pour les phrases de plus de 26 mots et / ou dont l'analyse lors d'une première phase donne plus de cinq morceaux d'analyse. En outre, il n'a pas été possible de construire des prononciations approchantes, basées sur des confusions fréquentes chez les apprenants, comme nous le proposions à la section 3.3.3.2. Par contre, la technique obtient un taux raisonnable de détection d'erreurs et fonctionne de manière satisfaisante.

5.2.3 Combinaison des techniques

Dans cette section, nous décrivons l'ordre d'application des techniques présentées dans les sections précédentes. Vandeventer Faltin (2003) énumère trois techniques pour combiner de manière optimale les méthodes de détection d'erreurs :

- *approche indépendante* : les méthodes sont appelées indépendamment les unes des autres, dans n'importe quel ordre ;
- *approche en cascade* : les techniques sont appliquées l'une après l'autre, en fonction des résultats obtenus (par exemple, aucune erreur détectée ou analyse partielle) ;

– *approche heuristique* : les techniques ne sont appliquées que si aucune analyse complète n'a pu être obtenue en désactivant les techniques de détection d'erreurs ; des heuristiques sur les morceaux d'analyse obtenus permettent de choisir les techniques de détection à utiliser.

Pour des raisons pratiques, l'approche en cascade a finalement été préférée aux deux autres. La réinterprétation phonologique n'est donc activée dans une seconde phase d'analyse que si aucune phrase complète n'a pu être obtenue avec le relâchement de contraintes ; comme la réinterprétation phonologique demande des analyses partielles, cet ordre a été simple à déterminer. Il a également été envisagé de permettre une application de la réinterprétation phonologique accompagnée du relâchement de contraintes. Cependant, cette solution générait trop de faux positifs, comme l'illustre l'exemple suivant :

(35) a. *set éléphants
b. sept éléphants
c. *cet éléphants

La phrase (35a) contient une erreur d'homonymie qui peut facilement être réinterprétée en (35b). Cependant, si le relâchement de contraintes est activé, l'analyse (35c) pouvait être sélectionnée alors qu'elle est nettement moins plausible à cause de l'erreur qu'elle contient. Comme le taux de faux positifs était particulièrement élevé, il a été décidé de désactiver le relâchement de contraintes lors de la réinterprétation phonologique.

5.2.4 Classement des résultats d'analyse et sélection de l'analyse préférentielle

Dans cette section, nous abordons le système de score qui permet de classer les résultats d'analyses et de sélectionner l'analyse et – le cas échéant – le diagnostic le plus vraisemblable. Selon Vandeventer Faltin (2003), il est primordial de trouver un équilibre entre la promotion d'une analyse sans erreurs – mais peu vraisemblable compte tenu du profil des apprenants et de la complexité de la structure syntaxique – et la promotion d'analyses plus vraisemblables mais contenant des erreurs. Il est aussi essentiel de ne pas détecter des erreurs inexistantes. L'idée est donc de mesurer un score qui tient compte de la vraisemblance de la structure syntaxique, des erreurs détectées et de la rareté des éléments lexicaux.

Tout analyseur non déterministe doit être capable d'ordonner les analyses qu'il produit afin de sélectionner une ou plusieurs analyses préférentielles. La fréquence des structures d'analyse dépend du genre de texte analysé (texte journalistique, littérature enfantine, texte scientifique, etc.). Cette fréquence est dérivée à partir de l'étude de corpus simple (§3.1.7), de l'analyse statistique de corpus annoté, ou simplement de l'observation et de l'expérience des personnes qui écrivent les spécifications de la grammaire. Le score dépend généralement de la complexité syntaxique et donc du nombre de constituants. Certains lieux d'attachements sont préférés à d'autres. *Fips* sélectionne les analyses préférentielles à partir de la combinaison des scores de ses constituants et en pénalisant ou favorisant certains attachements d'après des critères d'observations psycholinguistiques. Concrètement, le score est calculé en soustrayant et en additionnant des points, en se basant sur les attachements, sur les mots eux-mêmes et sur certaines caractéristiques lexicales et syntaxiques.

Quant aux fréquences d'erreurs, elles sont basées sur l'étude du corpus FRIDA (Granger, 2003). Vandeventer Faltin (2003) a proposé un score selon la formule suivante :

$$\sum_{i=1}^{|e|} log(f(e_i)) \tag{5.1}$$

Ici, e est une erreur dans la phrase et $|e|$ est le nombre d'erreurs dans la phrase. f est une fonction qui retourne la fréquence de l'erreur, entre 0 et 10%. Le logarithme de la fréquence, calculée sur quatre décimales, donne de petits nombres pour les erreurs fréquentes et des nombres plus élevés pour de petites fréquences [10]. Avec ce score, les analyses sans erreurs sont préférées, de même que les phrases contenant des erreurs fréquentes.

Maintenant, passons au calcul basé sur les fréquences de mots. Cette mesure permet de sélectionner la meilleure catégorie lexicale en cas d'ambiguïté. Là encore, les données statistiques doivent être extraites d'un corpus annoté. Les fréquence des mots aussi bien que des lexèmes (forme de base) sont utilisées. Les fréquences des mots sont additionnées, mais si un mot peut avoir plusieurs lexèmes, le score du mot est pondéré par la fréquence du lexème sélectionné dans l'analyse.

Ces techniques ont été testées sur un extrait du corpus FRIDA représentant environ 500 phrases ou 10 000 mots. Ces phrases ont été d'abord

10. L'échelle logarithmique permet d'espacer les petites valeurs et de rapprocher les grandes.

analysées par *Fips* et toutes les analyses possibles ont été entrées dans une base. La meilleure analyse a été ensuite sélectionnée manuellement par une experte. Enfin, l'analyse préférée a été comparée aux autres analyses afin de déterminer l'adéquation de chaque technique de classement, à travers deux séries de tests, l'un basé sur des statistiques et l'autre sur des inférences floues [11]. Ces test n'ont malheureusement pas permis de déterminer le calcul de poids optimaux pour combiner les différentes méthodes de classement, notamment à cause d'une trop forte surgénération d'analyses. En revanche, les tests ont montré que le score calculé par *Fips* à l'origine donnait les meilleurs résultats.

Finalement, Potter (2002) et Vandeventer Faltin (2003) proposent les heuristiques suivantes :

i. blocage des constituants DP, PP et DP incomplets, des TP sans sujet, des sujets sans Cas et des chaînes A-barre incomplètes ;
ii. blocage du relâchement de contraintes sur l'auxiliaire si *avoir* est attaché à un FP contenant un participe passé ;
iii. le score de *Fips* n'est pas pénalisé lors du relâchement d'une contrainte ;
iv. préférence donnée aux analyses sans erreurs, sauf lorsque le score de *Fips* de la meilleure analyse sans erreur vaut plus du double du score de la meilleure analyse contenant des erreurs ;
v. les analyses contenant des erreurs sont classées d'après leur score d'erreur ;
vi. à score égal – score de *Fips* ou score d'erreur selon les cas – la préférence sera donnée à la meilleure analyse selon l'autre méthode de score.

5.3 Rétroaction

Dans cette section, nous examinons l'utilisation pédagogique des techniques présentées dans ce chapitre. Dans un premier temps, nous décrivons la sortie *XML* de l'analyseur en 5.3.1, puis nous décrivons les différentes applications qui en sont tirées et l'application pédagogique qui en est faite [12] : grammaire en couleurs (§5.3.2), diagnostic d'erreurs (§5.3.3) et sortie d'arbre syntaxique (§5.3.4). Mentionnons que ces outils sont accessibles pour l'apprenant en tout temps et en production libre et que les différentes étapes sont

11. La logique floue (*fuzzy logic*) est une technique, utilisée notamment en intelligence artificielle, dont la principale caractéristique est d'accepter d'autres états de vérité que *vrai* et *faux*.
12. Précisons que notre participation personnelle au projet FreeText a porté particulièrement sur cet aspect et sur la vérification sémantique par comparaison de phrases (§7).

précédées par une phase de vérification orthographique, basé sur le vérificateur d'orthographe du correcteur *Hugo* (§C.19), développé par le partenaire commercial du projet et légèrement adapté pour la correction d'erreurs d'apprenants. Une fois les éventuelles erreurs d'orthographe corrigées, l'analyse et – le cas échéant – le diagnostic d'erreurs sont lancés.

5.3.1 Sortie XML

Dans cette section, nous décrivons la sortie *XML* produite par l'analyseur *Fips*. Nous avons vu au paragraphe 2.7.5 que le langage *XML* est flexible et évolutif, mais aussi suffisamment contraint pour permettre la validation de documents. Une première sortie *XML* sommaire de l'analyse de *Fips* existait avant le projet FreeText, notamment pour la synthèse vocale. Nous avons fortement étendu ce formalisme au cours du projet. Ainsi, un document *XML* complet et valide est produit à chaque activation de l'analyseur. En principe, l'analyseur est utilisé pour analyser une seul phrase et retourne l'analyse préférentielle ; cependant, la sortie *XML* est également prévue pour fournir toutes les analyses d'une phrase ou pour contenir l'analyse d'un document complet.

```
<PROJ cat="DP" tree="GN">
<HEAD cat="D" tool="yes" gender="masc" number="sin"
lexeme="le" ortho="le" colorGrammLex="Det">
Le
</HEAD>
<PROJ cat="NP" attachedAs="comp" tree="hidden">
<HEAD cat="N" tool="no" gender="masc" number="sin"
lexeme="chat" ortho="chat" colorGrammLex="Nom">
chat
</HEAD>
</PROJ>
</PROJ>
```

FIG. 5.4 – *Extrait de la sortie* XML *pour* le chat

La figure (5.4) illustre une sortie *XML* simple. Ainsi, les projections syntaxiques sont représentées par la balise <PROJ>. La catégorie de la projection est représentée dans l'attribut *cat*. L'attribut *tree* contient une étiquette simplifiée pour des applications pédagogiques, dont nous parlerons dans les paragraphes suivants. Les têtes lexicales sont représentées par la balise <HEAD>, dont les attributs illustrent les traits lexicaux. L'attribut *lexeme* permet d'afficher la forme de base du mot. Quant à *colorGramLex*, il contient une étiquette lexicale simplifiée pour les applications pédagogiques.

Examinons maintenant la figure (5.5), qui représente la sortie d'une phrase contenant une erreur.

```
<PROJ cat="TP" tree="Ph">
<PROJ cat="DP" attachedAs="spec" colorGramm="sujet" tree="GN">
<HEAD cat="D" tool="yes" gender="masc" number="sin"
lexeme="le" ortho="le" colorGrammLex="Det">
<ERROR index="Aa01" manypart="yes" category="NBR">
Le
</ERROR>
</HEAD>
<PROJ cat="NP" attachedAs="comp" tree="hidden">
<HEAD cat="N" tool="no" gender="masc" number="plu"
lexeme="chat" ortho="chats" colorGrammLex="Nom">
<PARTERROR antecedent="Aa01">
chats
</PARTERROR>
</HEAD>
</PROJ>
</PROJ>
<BAR cat="T" colorGramm="pred" tree="GV" index="AAAAA"
number="sin" person="3" tense="passeCompose" mode="indicatif"
voice="active" lexeme="dormir">
<HEAD cat="V" tool="yes" coref="AAAAA" number="sin"
person="3" tense="present" mode="indicatif" lexeme="avoir"
ortho="a" colorGrammLex="Aux">
a
</HEAD>
<PROJ cat="VP" attachedAs="comp" tree="hidden">
<HEAD cat="V" tool="no" coref="AAAAA" gender="masc"
number="sin" tense="participePasse" lexeme="dormir"
ortho="dormi" colorGrammLex="Verbe">
dormi
</HEAD>
</PROJ>
<PUNC key="fullstop">.</PUNC>
</BAR>
</PROJ>
```

FIG. 5.5 – *Extrait de sortie* XML *pour* *le chats a dormi

Comme nous avons une phrase complète, les fonctions grammaticales sont représentées dans l'attribut *colorGramm* de <PROJ>, pour les fonctions sujet, complément d'objet direct, prédicat, etc. La balise <PUNC> marque tous les signes de ponctuation. La balise <BAR> permet d'afficher un niveau intermédiaire, qui a disparu de la représentation X-barre simplifiée de *Fips* (fig. 5.1 p. 164), mais qui peut être utile pour afficher des informations

pertinentes :

- le verbe de la phrase principale est au passé composé (*a dormi*). Or, l'information lexicale des têtes porte sur les deux parties de ce verbe, respectivement un auxiliaire et un participe passé. Dans une application pédagogique, il nous a paru important de pouvoir afficher qu'il s'agit d'un verbe à un temps composé. Cette information est donc contenue dans les informations de la balise <BAR>. De plus, l'attribut *index* contient un identifiant qui est repris par les attributs *antecedent* des deux parties du verbe. Ainsi, l'information peut être facilement accessible, même si des éléments, comme un adverbe, sont insérés entre l'auxiliaire et le participe.
- le niveau barre de la phrase est un niveau important au niveau syntaxique, car c'est là qu'apparaît le syntagme verbal dans la plupart des formalismes, comme nous l'avons constaté à la section 3.3.

Enfin, deux balises permettent le marquage des erreurs. <ERROR> est inclus dans la balise <HEAD> pour marquer la position des erreurs détectées et contient l'attribut *cat* pour marquer la catégorie de l'erreur. L'attribut *manypart* permet de signaler si une erreur porte sur plusieurs éléments. En outre, *warning* signale si l'erreur doit être marquée comme un avertissement (§5.3.3). L'attribut *correction* contient la proposition du système de diagnostic pour la réinterprétation phonétique (§5.2.2). Enfin, *index* contient l'identifiant de l'erreur. Comme une erreur porte généralement sur plus d'un élément, une balise <PARTERROR> marque le ou les éléments suivants et est coindicée avec la première partie de l'erreur à l'aide de l'attribut *antecedent*. Ainsi, il est possible de marquer plusieurs erreurs dans une même phrase et de repérer quel élément fait partie de quelle erreur.

En conclusion, la sortie *XML* tire parti des informations fournies par l'analyseur et par le lexique. De nombreuses indications sont disponibles afin de tirer partie de l'analyse riche et profonde fournie par *Fips*. Enfin, le format *XML* s'avère tout à fait adéquat par son caractère souple et flexible et par sa large utilisation dans le monde informatique, qui permet ensuite un traitement des résultats par de nombreux outils et formalismes informatiques.

5.3.2 Grammaire en couleurs

Décrivons à présent la grammaire en couleurs. Le principe est d'illustrer les différentes parties d'une phrase en utilisant différentes couleurs [13]. Examinons la figure (5.6).

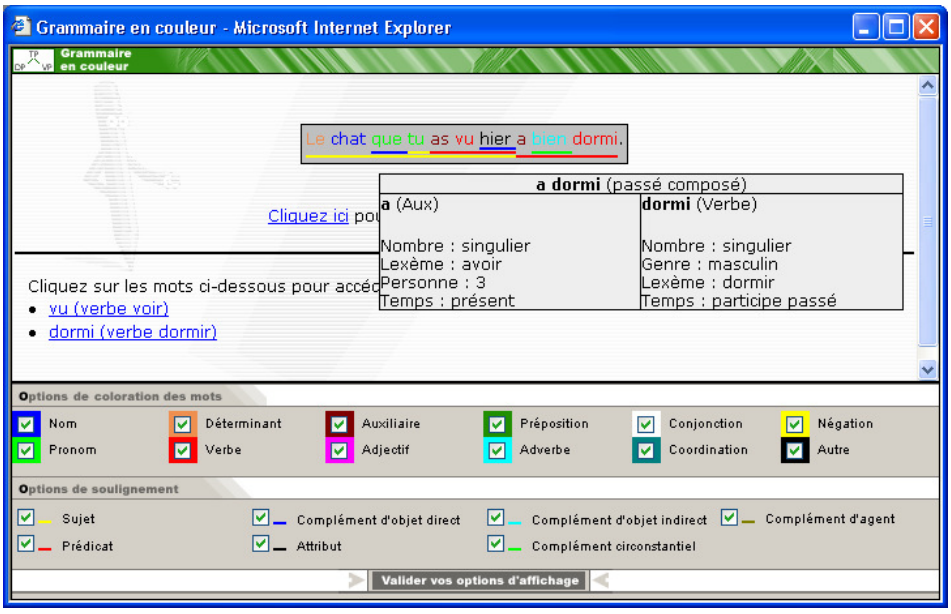


FIG. 5.6 – *Grammaire en couleur pour la phrase* le chat que tu as vu hier a bien dormi.

Les différentes catégories lexicales sont distinguées en changeant de couleur de police. Pour les fonctions grammaticales (sujet, prédicat, compléments circonstanciels), nous utilisons des soulignements de différentes couleurs. Différents niveaux de soulignement permettent de distinguer les niveau de subordination : la proposition relative *que tu as vu hier* est à un niveau inférieur et le sujet de la phrase principale est bien le syntagme nominal complexe *le chat que tu as vu hier*.

L'interface de la sortie est dynamique. En tout temps, l'apprenant peut sélectionner les catégories à afficher ou masquer grâce à l'interface, pour éviter d'être submergé par trop d'informations. L'apprenant peut vérifier sa phrase en cliquant sur un lien pour accéder au diagnostic d'erreurs. De plus, si la phrase contient des erreurs, un avertissement est affiché. En passant le curseur de la souris sur les différents mots, une infobulle affiche les informations lexicales. Remarquons dans notre exemple comment sont affichées les

13. Cette méthode est en vigueur pour l'apprentissage de la grammaire chez les apprenants du français langue première, notamment dans les écoles primaires du canton de Genève.

informations sur les verbes conjugués aux temps composés, alors que l'adverbe *bien* est inséré entre l'auxiliaire et le participe. Enfin, les différents verbes de la phrase sont listés et l'apprenant peut accéder à un conjugueur dynamique qui affiche leurs formes verbales à tous les temps et à tous les modes (§3.1.1.4).

Un test plus approfondi serait sans doute nécessaire pour améliorer la qualité du marquage des éléments. La sélection des mots et des lexèmes serait sans doute améliorée par une adaptation des statistiques de fréquences du lexique, notamment grâce à un apprentissage sur un corpus de phrase d'apprenants. Considérons la phrase suivante :

(36) Je suis un beau garçon.

Dans cet exemple, en l'absence du contexte de la phrase, il est impossible de déterminer notamment de l'identité du locuteur, si la forme *suis* provient du verbe *être* ou du verbe *suivre*. Il serait donc utile d'attirer l'attention de l'apprenant sur cette ambiguïté. Par ailleurs, il arrive malheureusement parfois que le lexème choisi soit erroné ; on éviterait certainement la plupart de ces erreurs en affichant toutes les formes possibles compatibles avec l'analyse préférentielle.

Passons maintenant aux améliorations possibles. En premier lieu, il serait souhaitable que les concepteurs d'exercices puissent paramétrer l'affichage et masquer les fonctions inutiles. Ensuite, il faudrait également pouvoir analyser plusieurs phrases dans une même phase d'analyse et les afficher séparément. Par ailleurs, si une phrase n'a pas pu être analysée complètement, l'utilisateur devrait en être clairement informé, afin qu'il ne soit pas induit en erreur par un affichage incorrect. Les frontières de segments d'analyse pourraient être clairement indiquées. L'apprenant pourrait alors simplifier la phrase, afin d'obtenir une analyse complète. Par ailleurs, les phrases d'exemple de la grammaire de référence devraient pouvoir être analysées directement en cliquant sur un lien. Enfin, les couleurs devraient être mieux contrastées, par exemple grâce à différentes couleurs d'arrière-plan.

5.3.3 Diagnostic d'erreurs

Nous décrivons maintenant le diagnostic d'erreurs en soi, dans lequel les erreurs détectées dans la phrase sont énumérées et décrites. La figure (5.7) montre l'interface du système de diagnostic.

FIG. 5.7 – *Diagnostic d'erreurs pour la phrase* *Le chats que tu a vues hier ont bien dormi.

Lorsque l'on survole le mot *erreur* avec le curseur de la souris, comme dans la figure ci-dessus, les différents mots impliqués dans l'erreur apparaissent en surbrillance. Lorsque le curseur de la souris survole les mots de la phrase, les informations lexicales sont affichés comme pour la grammaire en couleurs (§5.3.2). Des liens invitent également l'apprenant à consulter le conjugueur et la grammaire en couleurs, afin de pouvoir corriger sa production.

Parfois, il est difficile de poser un diagnostic sûr. C'est notamment le cas de l'ordre de l'adverbe et de l'adjectif, comme l'illustre l'exemple suivant :

(37) a. ?Souvent il mange des glaces.
b. Un grand homme /?un homme grand

Ces phrases peuvent paraître fausses, ou du moins étranges, auprès de certains locuteurs natifs. Il s'agit d'un jugement normatif. Dans ce cas, le diagnostic n'affiche qu'un avertissement.

Enfin, nous avons vu au paragraphe 5.2.2 que lorsqu'un mot était réinterprété phonétiquement, une correction était proposée à l'apprenant. Malheureusement, dans le prototype final, cette proposition n'est pas signalée de manière suffisamment visible.

Poursuivons maintenant par des propositions d'amélioration. En premier lieu, il serait nécessaire de corriger les défauts d'affichage et de rendre le système un peu plus convivial. Comme la grammaire de référence a été finement indexée, il conviendrait de faire un lien direct vers le point de grammaire le plus précis possible, compte tenu du type de l'erreur et des parties du discours impliquées. En outre, il est important qu'un champ permette de modifier la phrase dans la fenêtre même du système de diagnostic, plutôt que de devoir jongler avec de multiples fenêtres. Ceci facilitera l'autoremédiation, en ayant l'ensemble des données disponibles au même endroit.

Par ailleurs, il serait judicieux que les concepteurs d'exercices et même les apprenants puissent n'afficher que certains types d'erreurs. Il serait aussi envisageable de définir des erreurs prioritaires qui seraient affichées systématiquement, alors que la présence d'autres erreurs pourrait simplement être signalée ; l'apprenant serait alors libre de demander leur affichage.

Bien entendu, l'approche pédagogique pourrait être améliorée. Un diagnostic progressif, comme celui proposé par Heift (2003, §2.4, B.4.23), serait envisageable : en affichant une seule erreur à la fois, dans un premier temps, le ou les lieux de l'erreur seraient mis en surbrillance ; dans un second temps, on afficherait également le type de l'erreur ; ensuite, les informations lexicales seraient affichées. D'autres niveaux pourraient être proposés, comme une correction automatique de l'erreur, dans les cas où il est possible et fiable de corriger l'erreur. Quant à elles, Girard et Voce (2003) proposent un diagnostic en trois temps : premièrement, la présence d'une ou plusieurs erreurs serait signalée (étape de *rétroaction*) ; puis l'étape de *localisation* signalerait la position des erreurs, avec accès aux infobulles lexicales ; enfin l'étape d'*explication* donnerait un diagnostic du type "*vérifiez que vous avez bien accordé en genre (et en nombre) tous les mots en gras*" et donnerait des liens vers un point de la grammaire de référence, la grammaire en couleurs et le conjugueur.

Enfin, vu que plusieurs diagnostics d'erreurs peuvent souvent être proposés, on pourrait proposer plusieurs alternatives de diagnostic. Il serait cependant nécessaire de valider cette proposition à travers un test approfondi à l'aide d'un corpus, afin d'éviter un trop grand nombre d'analyses erronées, qui serait contre-productif.

Pour valider le diagnostic, il conviendrait d'en faire une analyse approfondie, afin d'évaluer notre stratégie pédagogique. Enfin, il devrait être possible d'affiner un peu le diagnostic en tenant compte du profil général des apprenants. Un apprenant germanophone aura tendance à utiliser **aider à quelqu'un*, par traduction directe de *helfen + datif*. Un profil de l'appre-

nant pourrait éventuellement influencer le choix de la meilleure analyse en fonction des erreurs fréquemment commises par l'apprenant.

5.3.4 Arbre syntaxique

Nous passons maintenant à l'arbre syntaxique. Ce type de représentation illustre les dépendances syntaxiques entre les différents syntagmes de la phrase. Une sortie arborescente de *Fips* existait déjà depuis plusieurs années et il nous est incombé d'adapter et d'améliorer cette sortie [14]. Les arbres ne sont disponibles qu'en cas d'analyse complète de la phrase. La figure (5.8) montre la fenêtre de l'arbre syntaxique comme affichée pour l'apprenant.

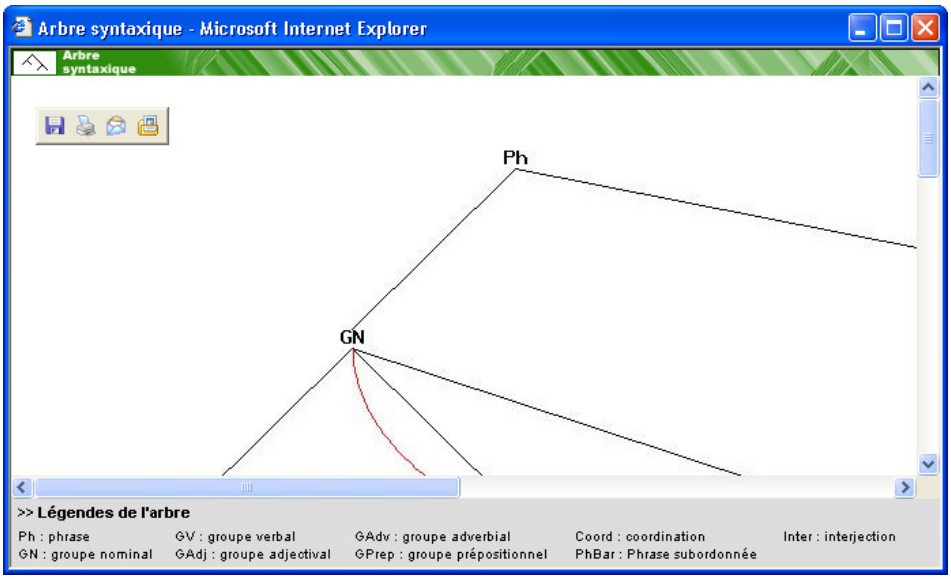


FIG. 5.8 – *Fenêtre de la sortie d'arbre syntaxique*

Nous utilisons une version simplifiée et francisées des étiquettes de *Fips*. La figure (5.9) montre l'arbre dans son ensemble.

Des arcs de couleurs soulignent les dépendances entre différents éléments comme le pronom relatif, son antécédent et la position canonique au sein de la subordonnée.

Nous n'avons malheureusement pas eu tout le temps nécessaire pour

14. Le partenaire industriel ne disposait pas de personne compétente dans le domaine de la construction dynamique de graphiques.

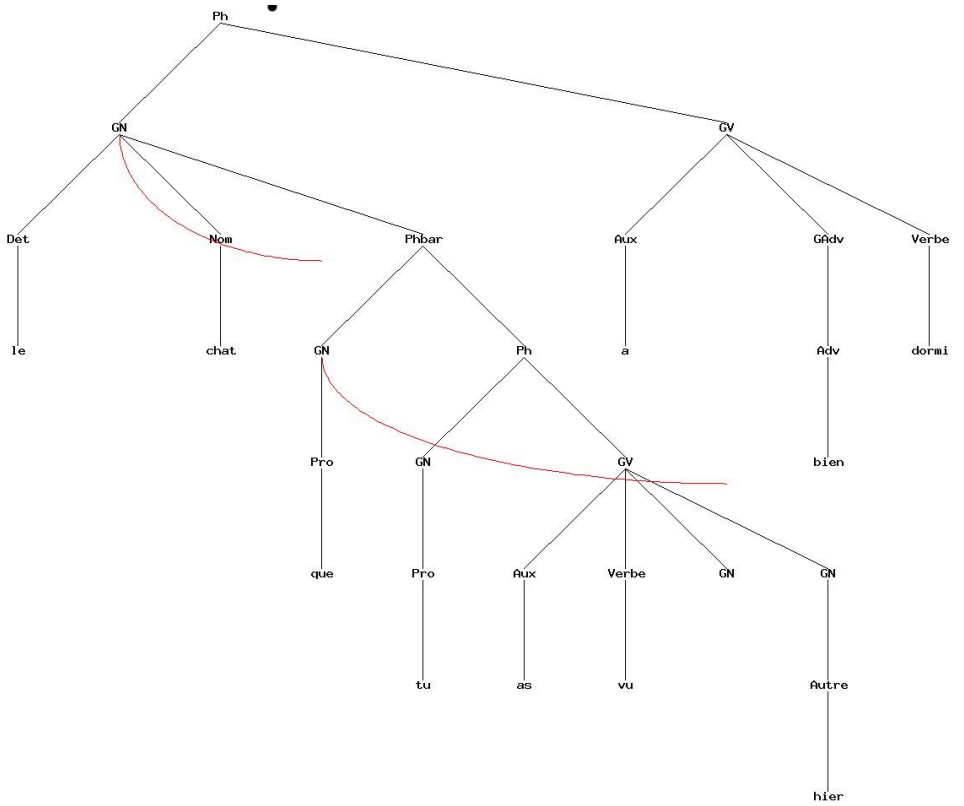


FIG. 5.9 – *Arbre syntaxique pour la phrase* Le chat que tu a vu hier a bien dormi.

peaufiner cette sortie. Il nous a manqué également certaines compétences pour maîtriser le placement des étiquettes et des mots ainsi que le traçage des lignes et des courbes. Ainsi, pour éviter des chevauchements, l'arbre a dû être considérablement élargi par rapport à la version originale. Il s'ensuit un manque de lisibilité lorsque la phrase a une certaine complexité comme la phrase d'exemple. De plus, les arcs ne sont pas affichés correctement. Dans la figure (5.9), le premier arc devrait relier le groupe nominal *le chat* à celui contenant le pronom relatif *que* ; le second arc partirait du même nœud vers le groupe nominal objet de la subordonnée, qui est un nœud vide sans élément lexical.

Comme pistes d'amélioration, nous pourrions envisager d'afficher un arbre dynamique, dont seuls les éléments essentiels seraient détaillés au départ [15]. L'apprenant pourrait ensuite regarder les détails à sa guise. Les nœuds les

15. Les linguistes illustrent souvent leurs exemples syntaxiques en réduisant les syntagmes non pertinents et en affichant leur contenu sous un triangle.

plus intéressants pédagogiquement pourraient être mis en évidence, comme par exemple les nœuds contenant des traces.

5.4 Discussion

Pour conclure ce chapitre, nous faisons un rapide bilan de l'utilisation de *Fips* pour un diagnostic grammatical. Tout d'abord, le pari de modifier un analyseur à large couverture et à but généraliste a visiblement réussi (Vandeventer Faltin, 2003). L'utilisation d'un analyseur robuste permet d'obtenir au moins une analyse partielle des phrases. La large couverture et le lexique riche permettent également de traiter une grande variété de constructions et de pouvoir utiliser le système de diagnostic d'erreurs dans de nombreuses situations sans restreindre la difficulté des textes ou des exercices. Avec ces résultats, nous confirmons l'adéquation de la théorie GB pour les applications de diagnostic et pour le relâchement de contraintes (§3.3.5), ou du moins la nécessité d'une analyse profonde qui tient en compte les dépendances à distance. Nous confirmons aussi les observations de la conclusion de la partie sur les théories grammaticales (§3.3.6), à savoir la nécessité d'un analyseur ascendant, basé sur l'analyse lexicale.

Quant à la couverture des erreurs, elle est globalement satisfaisante. Une évaluation interne a donné des résultats encourageants. La détection d'erreurs d'accord est encore problématique pour les structures complexes, notamment les DP coordonnés, les appositions, les mots inconnus etc. L'absence de ponctuation rend aussi parfois l'analyse fragile. Dans certains cas, le diagnostic fonctionne en plusieurs étapes : une première erreur est détectée et corrigée par l'apprenant ; lors d'une seconde analyse, le diagnostic d'autres erreurs peut être établi. Par ailleurs, dans certains cas, les lacunes de *Fips* empêchent un diagnostic correct :

(38) Je donne ma sœur une voiture.

Dans la phrase (38), calquée sur la structure de *to give* en anglais, le diagnostic détecte une erreur de complémentation avec *une voiture*. En effet, *ma sœur* peut être un complément valide de *donner*. Le filtre thématique n'a pas attribué le rôle de destinataire au bon constituant. Même avec un lexique cohérent et complet qui contiendrait tous les traits sémantiques, il serait probablement impossible d'établir des règles sûres qui permettent d'accepter toutes les combinaisons correctes et d'éliminer toutes les combinaisons incorrectes.

Par ailleurs, le système a quelques problèmes de stabilité ou ne fournit pas de résultat dans un temps acceptable, notamment à cause de la surgénéralisation de structures (Vandeventer Faltin, 2003). Le diagnostic en soi est parfois trompeur, comme dans le cas de l'accord de l'adjectif postposé avec le nom, où il est indiqué qu'il y a une erreur d'accord nom-adjectif, alors qu'on souhaite le contraire. En outre, le diagnostic est parfois erroné, à cause du fonctionnement de l'analyseur et de la technique du relâchement de contraintes : dans **toutes les étudiants étrangers*, une erreur est détectées entre *les* et *étudiants*. En effet, ce déterminant peut être à la fois féminin et masculin et se combine parfaitement avec *toutes*, puis le mot *étudiants* est lu et ne peut être combiné. Il faudrait introduire des stratégies spécifiques pour revenir en arrière en cas de constituant complet sans erreur dans un DP complexe.

Au niveau pédagogique, le système fournit une analyse détaillée qui permet de nombreuses applications. Sur ce plan, FreeText offre une variété d'outils inégalée à notre connaissance. Les interfaces de grammaire en couleur et d'arbre syntaxique profond sont également inédites et semble-t-il appréciées par les apprenants et les enseignants. Pour les améliorer encore, de nouvelles techniques, en particulier sur *Internet*, permettent de présenter les informations de manière plus attractive et dynamique. Des technologies comme *Flash* (§2.7.5) permettraient de créer des animations sur la base d'analyses dynamiques. Parmi les outils utiles à intégrer pour un prochain projet, citons un dictionnaire avec définitions, muni d'un outil de recherche phonétique comme celui du DAFLES (§3.5.4). Enfin, mentionnons encore la nécessité d'un profil de l'apprenant, qui pourrait utiliser la richesse des informations contenues dans la sortie *XML* pour tracer un profil précis des lacunes de l'apprenant.

Poursuivons par quelques mots sur l'évaluation. Une version intermédiaire du prototype a été testée par des professeurs de français entre fin 2001 et début 2002 (Cassart *et al.*, 2002). Il en est ressorti que de tels outils seraient grandement utiles mais que des problèmes de fiabilité grevaient encore trop le prototype. Parmi les problèmes relevés par les professeurs figuraient des problèmes de mauvais diagnostic, un manque de lisibilité de la grammaire en couleurs (contrastes, soulignements difficiles à distinguer, etc.) et l'absence de liens vers la grammaire de référence pour permettre à l'apprenant de remédier à ses erreurs. En outre, un testeur regrettait l'absence d'une véritable correction grammaticale, bien que cette option ait été délibérément écartée du concept pédagogique du logiciel. Par contre, la version finale des outils n'a pas pu être évaluée par des apprenants et des enseignants et n'a fait que l'objet d'une évaluation interne au projet, ce qui nous a empêchés de mesurer les progrès accomplis dans les outils de diagnostic et leur interface.

En conclusion, nous pouvons affirmer qu'il vaudrait la peine de poursuivre la recherche dans le domaine de la détection d'erreurs grammaticales. La structure actuelle de *Fips* est bien plus modulaire, avec une modélisation objet plus aboutie que lors de la réalisation du prototype de FreeText. Des améliorations du lexique et de la rapidité de traitement sont possibles, notamment grâce à une optimisation des algorithmes. Une seconde phase permettrait certainement de poursuivre la validation et l'amélioration des outils et d'affiner le traitement pédagogique.

Chapitre 6

FipsOrtho : un correcteur orthographique

Dans ce chapitre, nous décrivons le correcteur orthographique *FipsOrtho*. Notre apport personnel pour ce projet a consisté à poursuivre le développement du correcteur orthographique expérimental *FipsCorr*, destiné aux apprenants d'un langue étrangère (Ndiaye et Vandeventer Faltin, 2003, 2004), qui a été développé en parallèle au projet FreeText (§4) [1]. De 2005 à 2009, nous avons ensuite repris le travail, amélioré les techniques, récolté et annoté un corpus de phrases d'apprenants. Cette recherche est présentée ici.

Nous commençons par décrire les différentes techniques utilisées par le correcteur (§6.1). Nous poursuivons brièvement avec une description de la sortie *XML* du correcteur (§6.2). Ensuite, nous présentons une description générale du système (§6.3). Puis nous exposons les résultats des évaluations de notre correcteur (§6.4). Enfin, nous terminons ce chapitre par une discussion (§6.5).

6.1 Techniques de correction

Dans cette section, nous présentons les techniques de correction mises en œuvre dans le cœur de *FipsOrtho*, le correcteur orthographique proprement dit. La figure (6.1) présente le flux des techniques de correction, que nous

1. Le correcteur a été développé pendant le projet, mais le consortium a décidé d'utiliser une version du correcteur orthographique intégré au correcteur grammatical Hugo (§C.19) légèrement adaptée pour des apprenants du français langue étrangère.

allons décrire par la suite.

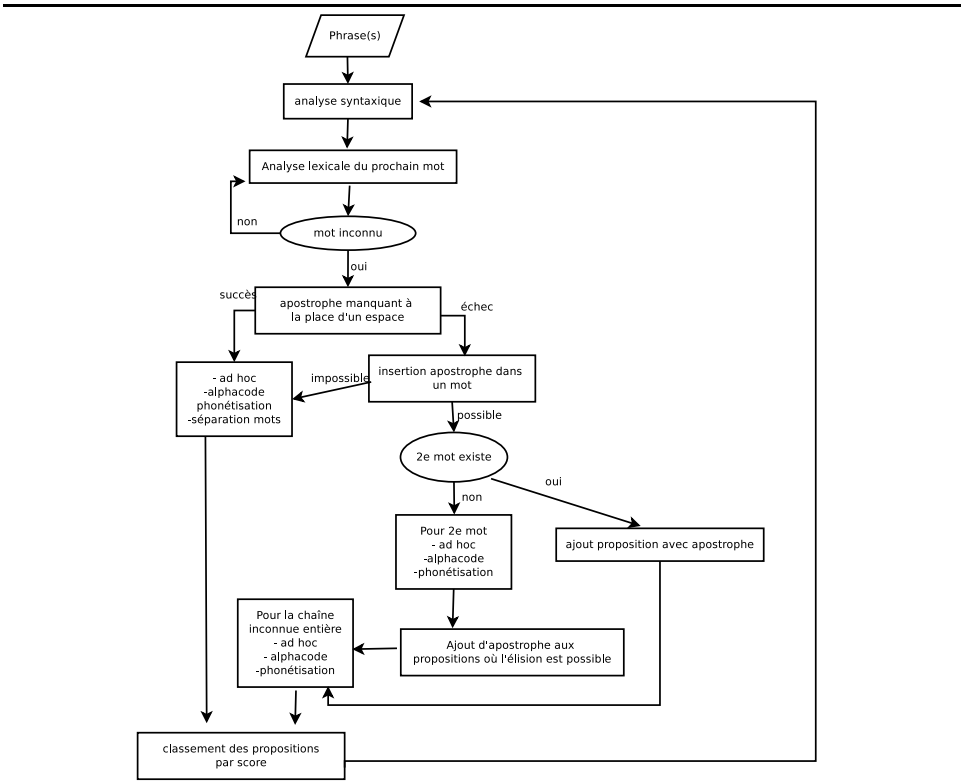


FIG. 6.1 – *Flux des techniques de correction orthographique*

Considérons la phrase suivante :

(39) Les travails* sont difficiles.

Cette phrase contient une erreur orthographique, plus précisément une erreur morphologique sur *travails** au lieu du mot correct *travaux*. Tout au long de notre description des techniques de correction, nous nous servirons de cet exemple comme fil rouge pour illustrer le traitement d'une erreur.

Cette section est organisée comme suit. Nous examinons l'analyse syntaxique et lexicale (§6.1.1). Ensuite, nous décrivons les techniques fondamentales de correction, la recherche par alpha-code (§6.1.2), la réinterprétation phonétique (§6.1.3) et la méthode *ad hoc* (§6.1.4). Puis nous survolons les techniques mineures de détection d'apostrophe manquante (§6.1.5), de séparation des mots (§6.1.6) et de mise en majuscules (§6.1.7). Enfin, nous terminons par notre algorithme de classement des propositions (§6.1.8).

6.1.1 Analyse syntaxique et lexicale

L'analyse syntaxique et lexicale précède les techniques de correction proprement dites. Il peut sembler curieux, au premier abord, de commencer le processus de correction par une analyse syntaxique (Berghel, 1987). Comme nous l'avons vu aux sections 3.1.1.2 et 3.2.3.1, la segmentation des unités lexicales n'est pas un processus trivial et nécessite un traitement pour délimiter les phrases ou segments de phrases. Il semblerait *a priori* logique d'utiliser les routines d'analyse lexicale de *Fips* qui précèdent l'analyse syntaxique, afin de repérer les mots inconnus. Par contre, il est moins évident d'utiliser une analyse complète. Cependant, nous avons repris la technique de *FipsCorr*, qui reprenait l'analyse syntaxique d'une phrase.

Voici la justification de cette technique : l'analyseur lexical attribue aux mots inconnus toutes les catégories lexicales ouvertes, à savoir nom, adjectif, adverbe et verbe ; un analyseur robuste doit en effet pouvoir faire face mots inconnus ou aux erreurs et variantes orthographiques, qui sont très fréquents dans une phrase, notamment avec les noms propres. Lors du processus d'analyse, *Fips* tente de combiner chacun de ces éléments pour fournir une ou plusieurs analyses syntaxiques à la phrase. Ainsi, une catégorie lexicale est assignée au mot inconnu. Même si les résultats de cette heuristique ne sont pas très précis, cette indication est néanmoins utilisable et précieuse : comme nous le verrons à la section 6.1.8, cette catégorie est prise en compte par l'algorithme de classement des mots.

Ainsi, pour la phrase (39), nous obtenons la structure suivante :

(40) [$_{\texttt{TP}}$ [$_{\texttt{DP}}$ [$_{\texttt{D}}$ Les] [$_{\texttt{NP}}$ [$_{\texttt{N}}$ travails*]]] [$_{\texttt{T}}$ sont] [$_{\texttt{FP}}$ [$_{\texttt{DP}}$ e] [$_{\texttt{F}}$ [$_{\texttt{AP}}$ [$_{\texttt{DP}}$ e] [$_{\texttt{A}}$ difficiles]]]]]

Dans notre exemple, le mot inconnu est considéré de préférence comme un nom. Après l'analyse syntaxique, nous poursuivons le processus de correction orthographique et recherchons chaque mot dans le lexique, qui compte environ 200 000 formes lexicales (Ndiaye et Vandeventer Faltin, 2004). Si le mot existe, alors nous passons au mot suivant ; s'il n'existe pas, alors nous déclenchons le processus de correction, à la recherche de propositions.

6.1.2 Recherche par alpha-code

Comme nous l'avons vu à la section 3.2.3.2, la recherche par alpha-code consiste à classer les lettres d'un mot dans un certain ordre et de retrouver dans le lexique des mots ayant le même alpha-code ou un alpha-code approchant. Cette technique permet de trouver des propositions pour des erreurs d'insertion, d'omission et de substitution.

Nous avons repris la méthode et l'implémentation de *FipsCorr* (Ndiaye et Vandeventer Faltin, 2003, 2004). La chaîne est mise en minuscules et désaccentuée, puis les consonnes du mot sont listées par ordre alphabétique, suivies par les voyelles classées dans l'ordre alphabétique également ; chaque caractère n'est listé qu'une fois. Chaque entrée du lexique contient l'alpha-code du mot, qui peut être utilisé comme clé de recherche. Pour retrouver des alpha-codes proches, nous créons vingt-six nouveaux alpha-codes en introduisant une par une les lettres absentes de l'alpha-code original, puis en retirant une par une les lettres présentes [2]. Pour un exemple concret, nous renvoyons le lecteur aux exemples (6), (7) et (8) p. 84. Nous avons appelé l'élargissement de l'alpha-code *alpha-wide* (W) et la restriction *alpha-narrow* (N).

Par ailleurs, nous avons envisagé de calculer la substitution d'une lettre par une autre, pour traiter le cas où une erreur de substitution remplace une lettre correcte par une autre qui ne se trouve pas dans l'alpha-code, comme **troglodite* pour *troglodyte* (où les alpha-codes sont respectivement *dglrteio* et *dglrteoy*). Cette méthode reviendrait à introduire $(n \times (26 - n))$ nouveaux alpha-codes, où n est la taille de l'alpha-code d'origine. Ainsi, pour un alpha-code de 7 caractères, il y a 133 nouveaux alpha-codes (7×19), ce qui ralentirait considérablement le processus et multiplierait le nombre de propositions inutiles. Nous avons donc renoncé à introduire cette méthode.

Reprenons donc le mot *travails** de notre phrase d'exemple (39), qui donne les alpha-codes suivants :

(41) a. lrstvai

b. *alpha-wide* : blrstvai, clrstvai, dlrstvai, lrstvaei, flrstvai, glrstvai, hlrstvai, jlrstvai, klrstvai, lmrstvai, lnrstvai, lrstvaio, lprstvai, lqrstvai, lrstvaiu, lrstvwai, lrstvxai, lrstvaiy, lrstvzai ;

c. *alpha-narrow* : rstvai, lstvai, lrtvai, lrsvai, lrstai, lrstvi, lrstva.

2. Dans *FipsCorr*, la méthode de restriction d'alpha-code n'était pas présente.

Ainsi, pour chaque mot, vingt-sept recherches sont lancées dans le lexique. Pour notre exemple, nous retrouvons 148 mots, 6 par alpha-code, 93 par *alpha-wide* et 49 par *alpha-narrow*. Voici quelques exemples de mots retrouvés, accompagnés de la méthode utilisée :

- travail (N) ;
- travailla (N) ;
- travaillai (N) ;
- travaillais (A) ;
- travaillas (A) ;
- travaillasse (W) ;
- travaillât (N) ;
- travaillées (W) ;
- travaillés (W) ;
- travailles (W) ;
- allitératives (W) ;
- ravitaillais (A) ;
- etc.

La technique de l'alpha-code génère beaucoup de propositions, dont certaines sont très éloignées de la chaîne originale. Nous devons donc calculer le degré de proximité entre le mot inconnu et les propositions et déterminer un seuil au delà duquel les propositions ne sont pas retenues. Cette mesure est appelée *distance lexicographique* et fait l'objet de la section suivante.

6.1.2.1 Distance lexicographique

Nous avons ici également repris et amélioré la méthode de distance lexicographique utilisée dans *FipsCorr*. A la section 6.1.2, nous avons constaté que la méthode de l'alpha-code générait énormément de propositions différentes, dont certaines étaient trop éloignées du mot inconnu. Pour pallier ces défauts, Ndiaye et Vandeventer Faltin (2003, 2004) proposent une méthode de filtrage basée sur un calcul de distance, la *distance lexicographique*, basée sur la distance de Levenshtein (Levenshtein, 1966). Cette méthode simple, déjà décrite à la section 3.2.3.10 p. 94, détermine le nombre minimal d'opérations d'insertion, de suppression ou de substitution de lettres pour passer de la chaîne source à la chaîne cible. L'insertion ou la suppression d'une lettre ont un coût de 1, tandis que la substitution a un coût de 2 [3].

3. Contrairement à ce qu'annonçaient Ndiaye et Vandeventer Faltin (2003), *FipsCorr* ne prévoyait qu'un coût de 1 pour les substitutions.

Nous avons adapté la méthode de calcul des distances en utilisant l'algorithme de Damerau-Levenshtein (Damerau, 1964; Levenshtein, 1966) présenté chez Wagner et Fischer (1974), Lowrance et Wagner (1975) et Jurafsky et Martin (2000, pp. 153 ss.). L'algorithme est appelé ainsi car la transposition est ajoutée aux opérations d'insertion, de suppression et de substitution [4].

L'algorithme crée une matrice de distances, où chaque symbole de la chaîne source représente une colonne et chaque symbole de la chaîne cible une ligne. Ainsi, la cible sera sur l'axe vertical et la source sur l'axe horizontal. Chaque cellule de la matrice *distance*[*source*,*cible*] contient la distance entre les *cible* premiers caractères de la cible et les *source* premiers caractères de la source.

La première ligne et la première colonne de la matrice sont initialisées avec les valeurs des indices, la première cellule ayant les coordonnées [0,0] ; dans la première matrice du tableau (6.1), ce sont les lignes marquées par le signe # ; nous avons volontairement omis ces valeurs dans les autres matrices du tableau pour des raisons de lisibilité. Ces valeurs correspondent à la comparaison d'une chaîne non nulle avec une chaîne nulle. Elles permettent de calculer la valeur de la distance selon la méthode suivante : chaque cellule est remplie en fonction des cellules avoisinantes, comme l'illustrent le tableau (6.1) ; pour remplir une cellule de la matrice, l'algorithme additionne un coût à la valeur des cellules à gauche, au dessus et en diagonale (en haut à gauche) de la cellule à remplir ; la valeur sélectionnée est la plus petite de ces trois valeurs.

Subs

	#	c	o	u
#	0	1	2	3
c	1	**0**	1	2
o	2	1	**0**	1
n	3	2	1	**1**

Eff

	a	t	r	e
a	**0**	1	2	3
t	1	**0**	**1**	2
e	2	1	1	**1**

Perm

	a	p	s
a	**0**	1	2
s	1	1	1
p	2	1	(2) **1**

Ins

	a	t	e
a	**0**	1	2
t	1	**0**	1
r	2	**1**	1
e	3	2	**1**

Les chemins sont marqués par les nombres en gras souligné.

TAB. 6.1 – *Opérations pour le calcul de distance entre deux chaînes*

Si l'algorithme sélectionne l'*effacement*, on additionne un certain coût au nombre immédiatement à *gauche* ; pour l'*insertion*, c'est le nombre immédiatement *au-dessus* ; pour la *substitution*, c'est le nombre dans la *diagonale* ; pour la *transposition* ou permutation de lettres, on utilise le *deuxième nombre dans la diagonale*, s'il est plus petit que le nombre obtenu grâce à

4. Pourtant, aucune des références citées ici ne mentionne ou ne cite ni Damerau, ni Levenshtein.

la substitution (dans le tableau (6.1), le nombre entre parenthèses); enfin, si les lettres sont *identiques*, le nombre dans la diagonale est *copié*.

Dans la littérature, l'insertion et l'effacement ont généralement un coût de 1 ; la substitution a un coût de 1 ou 2 ; la transposition a généralement un coût de 1 ; enfin, si les lettres sont identiques, le coût est de 0. La distance finale correspond au nombre le plus en bas à droite de la matrice.

Nous avons modifié l'algorithme après une première phase de tests sur une liste de mots présentée à la section 6.4.1 et à l'annexe G. Tout d'abord, lorsqu'une des chaînes comportait une double consonne et l'autre une seule, l'algorithme comptait une insertion ou un effacement selon les cas. Cette confusion entre consonnes doubles et consonnes simples est très courante. Ainsi, des chaînes moins proches selon notre jugement obtenaient une distance plus faible. Pourtant, intuitivement, la différence entre une consonne et une consonne doublée n'est pas aussi forte que l'insertion ou l'effacement d'une lettre différente de la lettre voisine.

C'est pourquoi nous avons ajouté à l'algorithme deux méthodes *ad hoc*. Ainsi, pour l'insertion d'une double consonne, on ajoute un coût de 0,1 au nombre immédiatement supérieur, comme l'illustrent les exemples du tableau (6.2) ; pour l'effacement d'une double consonne, on ajoute 0,1 au nombre immédiatement à gauche. Comme pour la transposition, on ne remplace le nombre obtenu par la méthode générale de calcul que si le nombre obtenu par l'ajout du score faible est inférieur. Ainsi le coût entre **addresse* et *adresse* est de 0,1 au lieu de 1, et de 0,3 au lieu de 3 entre **proffesionel* et *professionnel*.

Ins dbl cons

	a	s	u
a	**0**	1	2
s	1	**0**	1
s	2	(1) **0,1**	1
u	3	2	**0,1**

Eff dbl cons

	a	s	s	u
a	**0**	1	2	3
s	1	**0**	(1) **0,1**	2
u	2	1	1	**0,1**

Les chemins sont marqués par les nombres en gras souligné.

TAB. 6.2 – *Opérations pour le calcul de distance entre deux chaînes avec simple/double consonne*

Nous avons également opéré d'autres adaptations. Tout d'abord, la casse, les espaces, les apostrophes et les traits d'unions ne sont pas pris en compte : **angleterre* a une distance de 0 avec *Angleterre*. Par ailleurs, nous avons introduit un coût très faible, de 0,1 au lieu de 1, pour les substitutions entre

caractères accentués ou possédant un signe diacritique (ç, ñ, é, è, à, û, ù, etc.). En effet, une trop forte pénalité frappait ces fautes courantes, spécialement pour les mots courts, comme **tres* pour *très*. Nous avions envisagé dans un premier temps de ne pas pénaliser du tout la substitution des caractères accentués, ce qui aurait donné une distance de 0 entre ces deux chaînes. Cependant, la solution d'un coût plus léger permet de souligner que les chaînes ne sont pas identiques mais très proches, par rapport à l'insertion ou la substitution d'un caractère, comme entre *tres* et *êtres*. Nous détaillerons ces exemples au tableau (E.1) p. 446.

Les algorithmes complets sont donnés à l'annexe E en (1) et (2) p. 447. La distance est pondérée en divisant l'addition des coûts par l'addition de la taille des deux chaînes comparées. L'annexe E contient aussi des exemples complets de matrices des distances.

Par ailleurs, il nous reste à définir le seuil au-delà duquel une proposition doit être rejetée. Le correcteur que nous avons adapté proposait la distance suivante :

$$\mathcal{D} = \frac{2}{longueur\ source + longueur\ cible} \tag{6.1}$$

Ce seuil correspond donc à deux opérations de modification de la chaîne. Remarquons que la proposition *professionnel* pour le mot erroné **proffesionel* était rejetée, car elle dépassait le seuil de 0.08.

La version initiale du correcteur rejetait les propositions dont la distance était égale au seuil. Après étude de nos corpus de test, il nous a paru préférable d'accepter ces propositions à la limite du seuil, bien que le nombre de propositions – et, par conséquent, le taux de propositions inopportunes – augmente parfois de manière significative. Nous avons aussi augmenté le seuil comme suit :

$$\mathcal{D}\prime = \frac{2.3}{longueur\ source + longueur\ cible} \tag{6.2}$$

Toutefois, nous appliquons deux restrictions :

i. Les propositions retrouvées par alpha-code et alpha-code élargi ne sont acceptées que si elles commencent par la même lettre que le mot inconnu et ont une distance inférieure ou égale au seuil ;

ii. Les propositions retrouvées par alpha-code restreint ne sont retenues que si elles sont en dessous du seuil et qu'elles commencent par la même lettre que le mot inconnu.

Ces deux restrictions s'appuient sur l'observation faite par de nombreux chercheurs qu'une erreur porte très rarement sur la première lettre de la phrase (Kukich, 1992). Nous avons aussi constaté que les propositions retrouvées par alpha-code restreint étaient souvent éloignées de la chaîne d'origine.

Examinons maintenant les résultats retrouvés par alpha-code pour *coutumace*. Les résultats sont présentés à l'annexe H, p. 507. Sur 41 propositions, seule une est finalement retenue, soit environ 2,44%. Elle est trouvée par alpha-code élargi, avec l'ajout de la lettre *n*. La proposition *couteau*, trouvée par alpha-code restreint, est rejetée car la distance est égale au seuil. La méthode des alpha-codes génère toujours 27 clés de recherche. La méthode restreinte donne un nombre de d'alpha-codes égal à la taille de l'alpha-code du mot recherché (ici, 7). La méthode élargie donne toujours un nombre d'alpha-codes de 26 moins la taille de l'alpha-code original (ici 19).

Pour notre exemple (39) avec **travails*, nous retenons 10 propositions sur 148 résultats. Dans les détails, 2 propositions sur 6 sont retrouvés par alpha-code simple (33,3%), 4 propositions sur 93 par alpha-code élargi (4,3%) et 4 propositions sur 49 par alpha-code restreint (8,1%).

Nous parlerons en détail des résultats de notre approche à la section 6.4, qui se base sur les résultats des tests présentés aux sections G et H.

6.1.3 Réinterprétation phonétique

Comme nous l'avons déjà souligné à la section 3.2.3.5, beaucoup d'erreurs proviennent d'une écriture phonétique des mots : le mot est écrit comme il est prononcé, en utilisant une des manières d'écrire un son. Ainsi, en français, *ph* et *f* transcrivent le même son [f], de même que le son [o] se retranscrit *o*, *au*, *eau* [5]. Ndiaye et Vandeventer Faltin (2003) proposent un algorithme de réinterprétation phonologique qui consiste à créer une transcription phonétique du mot mal orthographié et de tenter de retrouver le mot correct à partir de sa prononciation stockée dans le lexique. Considérons l'exemple

5. Et *aux*, *ault*, *ot*, *ots*, *os*, *oc*, *ocs*, *aulx* etc. en fin de mot

suivant, tiré de ce même article :

(42) a. *sau.
b. [so].
c. saut, seau, sceau, sot, sauts, seaux, sceaux, sots.

L'exemple (42a) est retranscrit phonétiquement en (42b). Une recherche dans le lexique donne les homophones en (42c). Certaines variantes au pluriel semblent moins plausibles que les variantes au singulier et les mots plus rares comme *sceau* pourraient être écartés suivant le niveau des apprenants.

Pour améliorer cet algorithme, nous pouvons partir du constat que les apprenants ont souvent des difficultés à distinguer certains phonèmes. On peut penser notamment aux paires minimales phonologiques *pomme/paume* ([pɔm] / [pom]), *pré/près* ([pʁe] / [pʁɛ]), etc. Les voyelles nasales sont sources de maintes erreurs entre [õ / ɑ̃ / ɛ̃ / œ̃]. Certaines consonnes ne sont pas bien distinguées par certains locuteurs, notamment [b / v] pour les hispanophones et [ʁ / l] pour les locuteurs de certaines langues asiatiques. Il serait alors judicieux de pouvoir tenir compte de ces confusions possibles dans la réinterprétation phonologique, voire de paramétrer le correcteur en fonction de la provenance du locuteur. Il serait aussi nécessaire de dériver plusieurs chaînes phonétiques d'un même mot erroné. Les *-s* et *-t* finaux peuvent être prononcés ou non selon les cas. Enfin, il serait bon de tenir compte des graphies de certains sons dans les langues maternelles des apprenants et des confusions de phonèmes[6].

L'analyseur *Fips* (§5) dispose d'environ 700 règles ordonnées de phonétisation (Gaudinat et Goldman, 1998; Goldman, 2001; Ndiaye et Vandeventer Faltin, 2004) qui procèdent à une phonétisation du mot grâce à une stratégie déterministe. Ce phonétiseur sert à la synthèse vocale *FipsVox* (§3.1.4) pour prononcer les mots inconnus ; il sert également à phonétiser automatiquement les nouveaux mots insérés dans le lexique, afin de gagner du temps ; le cas échéant, cette phonétisation doit être revue par un expert afin d'éviter les erreurs.

La réinterprétation phonétique consiste donc à phonétiser la chaîne inconnue et de rechercher cette chaîne dans le lexique avec cette clé de recherche.

6. Les propositions esquissées ici ne sont basées que sur quelques observations et intuitions empiriques. Si l'on voulait les mettre en œuvre de manière efficace, l'analyse d'un large corpus d'erreurs d'apprenants serait indispensable pour pouvoir établir une typologie des erreurs en fonction de la provenance.

Par ailleurs, pour améliorer la recherche, nous étendons la recherche en substituant certains sons de la chaîne originale par d'autres sons approchants :

- [a / ɑ] ;
- [ɔ / o] ;
- [e / ɛ] ;
- [ø /œ / ə] ;
- [ɔ̃ / ɑ̃] ;
- [œ̃ / ɛ̃].

En ce qui concerne cette méthode, nous nous sommes bornés à adapter *FipsCorr* en modifiant légèrement l'élargissement de la recherche phonétique. Pour notre exemple (39), **travails* est phonétisé [tʁavaj]. Nous retrouvons les mots suivants :

- travail ;
- *travaille* ;
- *travaillent* ;
- travailles.

Les deux propositions en italiques sont de nouvelles propositions ; les deux autres ont également été retrouvées par alpha-code.

6.1.4 Méthode *ad hoc*

La méthode *ad hoc* de traitement d'erreurs morphologiques est un algorithme rudimentaire (Ndiaye et Vandeventer Faltin, 2003, 2004), adopté à défaut de temps pour adapter l'analyseur morphologique de *Fips* pour le traitement de mots inconnus. Comme nous l'avons vu à la section 3.2.3.6, ces erreurs sont très fréquentes chez les apprenants, qui créent de faux mots, par exemple en ajoutant un morphème inadéquat à une racine. L'erreur peut porter sur la déclinaison, la conjugaison etc. Ndiaye et Vandeventer Faltin (2003, 2004) décrivent les exemples suivants :

(43) a. *animals
b. *devé
c. *changeage

L'exemple (43a) est causé par une mauvaise dérivation flexionnelle, en appliquant la règle régulière du pluriel en *-s* à des noms en *-al*, dont le pluriel est *-aux*, suivant une dérivation exceptionnelle. L'exemple (43b) est une erreur de dérivation du verbe devoir, qui a un participe passé irrégulier *dû, due, dus* et *dues*. Ces deux premiers exemples peuvent être traités par des règles *ad hoc* : les mots en *-al* et *-ail* ont des pluriels en *-aux* ; l'erreur (43b) est traitée par une règle spécifique pour un verbe très courant de la langue.

Le troisième exemple (43c) est une mauvaise dérivation nominale à partir du verbe *changer*, avec le suffixe *-age* au lieu de *-ment*. Ndiaye et Vandeventer Faltin (2003, 2004) traitent également l'erreur courante suivante :

(44) a. *allerons
b. all-
c. -erons

Le verbe *aller* compte pas moins de quatre radicaux (*all-, ir-, va-* et *aill-*). Il est donc fréquent de trouver (44a) pour la forme correcte *irons* (indicatif futur 1^e personne du pluriel). Un analyseur morphologique devrait trouver (44b) et (44c), respectivement une racine du verbe aller et une terminaison de l'indicatif futur 1^e personne du pluriel des verbes du premier groupe.

Nous avons donc repris et amélioré la méthode présentée par Ndiaye et Vandeventer Faltin (2003, 2004) [7] qui tente de corriger les erreurs les plus fréquentes en retrouvant certains patrons dans un mot et en le remplaçant par une autre sous-chaîne, afin de trouver un mot connu. La table (6.3) liste les patrons que nous tentons de retrouver :

Le signe # marque un début de mot lorsqu'il est à gauche et une fin de mot lorsqu'il est à droite. Lorsqu'il est absent, c'est tout le mot qui est remplacé.

Avec cette technique, il est fréquent de produire d'autres non-mots. C'est pourquoi nous recherchons chaque proposition dans le lexique et ne l'insérons dans la liste que si le mot existe.

La liste d'erreurs est bien évidemment très partielle et lacunaire. Elle est susceptible de croître avec le temps, au fur et à mesure de nouvelles

7. Contrairement à ce que spécifient Ndiaye et Vandeventer Faltin (2004), l'application des autres méthodes de recherche n'est pas désactivée si la méthode *ad hoc* trouve des propositions.

Patron	Chaîne remplacement	Exemple / commentaire
als#	aux	chevals* → chevaux
ails#	aux	travails* → travaux
#aller	ir	allerez* → irez
devé	dû	trouvé chez Mogilevski (1998)
#tenir	tiendr	teniras* → tiendras
#venir	viendr	venirais* → viendrais
#voir	verr	voirai* → verrai
#fair	fer	fairais* → ferais
age#	ment	changeage* → changement
ment#	age	repassement* → repassage

TAB. 6.3 – *Liste de sous-chaînes pour la méthode* ad hoc

observations dans un corpus d'apprenants.

Pour finir, examinons le résultat de la méthode *ad hoc* pour la phrase (39). La proposition *travaux*, qui n'a pas été trouvée par les autres méthodes, peut être ajoutée à la liste des propositions.

6.1.5 Apostrophe manquante

Comme nous l'avons souligné au paragraphe 3.2.3.7, l'apostrophe est un signe particulier, qui nécessite un traitement en tant que tel. Elle se retrouve en français dans *aujourd'hui*, *prud'homme*, *prud'hommal* et *presqu'île*. En outre, elle peut être insérée après les lettres *c*, *d*, *j*, *l*, *m*, *n*, *s* et *t* devant une voyelle ou un *h* muet (*l'habitation, d'habitude, l'arbre*, etc.). Enfin, certains mots comme *que*, *lorsque*, *jusque*, etc. peuvent être élidés devant une voyelle.

Une élision n'est possible que dans des cas précis :

- le mot après l'élision commence par une voyelle ou la lettre *h* ;
- si le mot est un adjectif ou un nom, il est au singulier et la consonne qui le précède est *l*, *c* ou *d* ;
- si le mot est un verbe, la consonne doit être *l*, *j*, *t*, *s*, *m*, *n* ou *c*.

Dans l'apprentissage des langues, l'étude de corpus d'apprenants permet de constater que l'apostrophe est souvent remplacée par un espace [8]. Parfois

8. Nous n'avons pas trouvé ailleurs de théories expliquant ce phénomène. Nous faisons

également, les mots ou parties de mots sont simplement collées ensemble, comme dans *aujourdhui**.

Pour traiter ce problème, nous utilisons, comme nous pouvons le constater à la figure (6.1) p. 192, deux méthodes. La première méthode intervient lorsqu'un mot inconnu contient un espace. Dans ce cas, nous insérons une apostrophe et tentons de trouver ce mot dans le lexique. Ce type d'erreur intervient parfois lorsqu'un début de mot contenant une apostrophe est lue par l'analyseur lexical de *Fips* (§§3.1.1.2 et 5) : ainsi, les chaînes "*prud homme*"* ou "*aujourd hui*"* sont considérées comme un seul mot inconnu [9].

Par contre, la seconde méthode examine si la première lettre d'un mot inconnu fait partie des lettres qui peuvent être suivies d'une apostrophe, que nous avons énumérées ci-dessus, ou si le mot commence par les lettres "*qu*". Si un des critères est rempli, nous regardons si le reste de la chaîne après les candidats à l'élision est trouvé dans le lexique. Si c'est le cas, nous insérons la proposition avec une apostrophe. Si le restant de la chaîne n'est pas trouvé dans le lexique, nous tentons d'appliquer les méthodes *ad hoc*, alpha-code et phonétique ; si un mot est trouvé, nous y ajoutons une apostrophe si l'élision est valide et filtrons la chaîne complète grâce à la distance lexicographique. Par ailleurs, les méthodes *ad hoc*, alpha-code et phonétique sont également appliquées à la chaîne complète sans apostrophe.

Mentionnons encore pour terminer que cette méthode ne trouve pas de proposition pour la phrase (39).

6.1.6 Séparation de mots

La méthode de séparation de mots est appliquée après les méthodes *ad hoc*, alpha-code et phonétique. Nous avons vu à la section 3.2.3.7 que ce problème est loin d'être trivial. Pour notre part, nous ne traitons que le cas où deux mots séparés sont collés l'un à l'autre.

Notre méthode procède comme suit. Nous séparons la chaîne en deux parties à chaque endroit possible. Nous tentons d'insérer un trait d'union (*portemonnaie** → *porte-monnaie*) et une apostrophe (*prudhomme** →

l'hypothèse de l'inaccessibilité relative de ce symbole sur certains claviers configurés pour des langues où il est rarement employé. Nous proposons également l'existence d'une mauvaise habitude engendrée par la cadence rapide de frappe des apprenants lors d'autres activités comme le clavardage ou l'envoi de courriels (§2.7.5).

9. Étonnamment, la recherche phonétique ne donne pas de proposition pour ces chaînes, sans doute par suite d'un dysfonctionnement du phonétiseur.

prud'homme) et recherchons la chaîne dans le lexique. Par ailleurs, si la première sous-chaîne existe dans le lexique, nous recherchons également la seconde (*veuxpas** → *veux pas* [10]) et insérons les deux mots séparés par un espace comme proposition, le cas échéant.

Pour terminer, précisons encore que, comme la précédente, cette méthode ne trouve pas de proposition pour l'exemple (39).

6.1.7 Insertion de majuscule

La méthode d'insertion de la majuscule est triviale. Nous nous basons sur l'analyse de *Fips* et vérifions si le premier mot de la soumission est en majuscule. De plus, si le mot inconnu est mis en majuscule, les propositions éventuelles seront elles aussi mises en majuscule.

Cette méthode n'est pas idéale, mais nous avons jugé utile de la conserver. L'analyse lexicale de *Fips* ne se base pas sur les majuscules pour fixer les limites de phrases, afin de ne pas être déroutée par la lecture de textes informels comme un courriel. Par contre, comme l'analyse lexicale présuppose que le texte est grammatical, elle perd de la précision avec des textes comportant des erreurs d'apprenants. Il nous a paru risqué de tenter de restituer les limites de phrases nous-mêmes et nous n'avions pas le temps ni les ressources linguistiques pour améliorer l'analyseur lexical pour le traitement d'erreurs. Il serait par contre important d'attirer l'attention de l'apprenant sur la faiblesse du traitement des erreurs de majuscules, par exemple en utilisant un avertissement demandant à l'apprenant de vérifier la correction [11].

Précisons pour finir que notre phrase (39) a déjà une majuscule au début et que *Fips* considère la phrase comme complète. Il n'y a donc aucune proposition pour cette méthode.

10. ?*Je veux pas* est une phrase admise dans un registre oral informel, mais qui est généralement considérée comme agrammaticale à l'écrit et doit être corrigée dans une application d'ALIAO. Ceci n'est néanmoins pas possible avec un correcteur orthographique.

11. D'une part, l'apprenant utilise souvent d'autres correcteurs grammaticaux et orthographiques dans sa langue première, qui ont certainement des performances supérieures aux nôtres à ce niveau. D'autre part, habitués à rédiger dans un style informel pour des courriels, des messages courts par téléphone mobile (textos ou sms), etc., ils oublient de s'adapter à un style plus formel. Enfin, les langues diffèrent dans l'usage de la majuscule, comme l'allemand qui l'emploie systématiquement pour les substantifs, ou l'anglais dans les titres de livres ou d'articles.

6.1.8 Ordre des propositions

Il est important de proposer les corrections dans un ordre de pertinence, comme nous l'avons vu à la section 3.2, et que la liste soit la plus courte et la plus pertinente possible (Rimrott et Heift, 2005). Ndiaye et Vandeventer Faltin (2003) proposent un nombre de cinq propositions dans un premier temps. Intuitivement, un faible nombre de proposition est souhaitable, car les utilisateurs ne veulent pas être submergés d'informations.

Pour *FipsOrtho*, nous avons largement repris et amélioré la méthode de *FipsCorr* (Ndiaye et Vandeventer Faltin, 2003, 2004) [12]. Nous classons les propositions par score en tenant compte de la méthode ou des méthodes utilisées pour les retrouver, de la distance lexicographique et de l'adéquation du mot avec l'analyse de la phrase (partie du discours, valeurs d'accord). Les propositions sont ensuite classées par score décroissant et par distance lexicographique croissante.

Méthode	**Valeur**	**Méthode**	**Valeur**
Ad hoc	12	Apostrophe	10
Séparation	9	Phonologique	6
Alpha-code	5	Alpha-code élargi	3
Alpha-code restreint	2	Majuscule	0

TAB. 6.4 – FipsOrtho*: valeurs de score par méthode*

Le tableau (6.4) reprend les valeurs de score des méthodes de *FipsOrtho*. Si une proposition est retrouvée par plusieurs méthodes, les scores sont additionnés. Les valeurs ont été fixées d'après des observations sur la liste de mots (§§6.4.1 et G) et sur les phrases du corpus (§6.4.2). La méthode *ad hoc* a le plus gros score. Il s'agit de favoriser une correction par règles, destinée à pallier les défauts de la recherche de mots par clés alpha-code ou phonétique. Ensuite, les méthodes d'insertion d'espace, d'apostrophe ou de tiret ont aussi un score élevé, car ce type d'erreurs est courant et les propositions devaient ressortir. La méthode phonologique obtient un score légèrement plus haut que l'alpha-code. Enfin, l'élargissement et la restriction de l'alpha-code, générateurs de nombreuses propositions, sont clairement défavorisés.

Par ailleurs, nous avons favorisé les propositions les plus proches du mot inconnu. Ainsi, nous augmentons de 8 le score des propositions dont la distance lexicographique est inférieure à 0,1.

12. Contrairement à ce qui est affirmé dans ces articles, *FipsCorr* ne limite pas le nombre de propositions à 5 et n'élimine pas les propositions trouvées par plusieurs méthodes.

Enfin, comme nous l'avons dit à la section 6.1.1, nous favorisons les propositions qui s'adaptent le mieux à l'analyse syntaxique de *Fips*. Ainsi, nous comparons certaines valeurs de traits attribuées au mot inconnu avec celles de chaque proposition. Les propositions d'une catégorie lexicale différente de celle attribuée par *Fips* ne sont pas écartées, à cause de la trop faible fiabilité de l'analyse. Le tableau 6.5 donne les valeurs de score par similarité de trait.

Trait	**Valeur**	**Trait**	**Valeur**
Catégorie	3	Nombre	3
Genre	3	Personne	2

TAB. 6.5 – FipsOrtho*: valeurs de score par similarité de trait*

Pour conclure, examinons l'ordre des propositions pour la phrase (39). Le mot **travails* est considéré comme un nom, masculin ou féminin, au pluriel et à la troisième personne. Le tableau (6.6) donne la liste des propositions ainsi que leurs valeurs respectives.

Proposition	Cat.	Gen.	Nb.	Pers.	Méth.	Distance	Seuil	Score
travaux	N	m	P	3	AH	0.2	0.15333	23
travail	N	m	S	3	P, N	0.06666	0.15333	22
travailles	V	m, f	S	2	P, W	0.06111	0.12777	20
travaillés	V, A	m	P	1-3	W	0.06111	0.12777	19
travaille	V	m, f	S	1-3	P	0.06471	0.13529	17
travaillas	V	m, f	S	2	A	0.06111	0.12777	16
travaillent	V	m, f	P	3	P	0.16315	0.12105	14
travailla	V	m, f	S	3	N	0.06471	0.13529	13
travaillées	V, A	f	P	1-3	W	0.11052	0.12105	11
travaillais	V	m, f	S	1-2	A	0.11052	0.12105	8
travaillasse	V	m, f	S	1	W	0.11	0.115	6
travaillai	V	m, f	S	1	N	0.11666	0.12777	5
travaillât	V	m, f	S	3	N	0.11666	0.12777	5

TAB. 6.6 – FipsOrtho*: liste des propositions pour* *travails *(phrase 39)*

Au total, nous trouvons treize propositions, dont les plus pertinentes sont en tête. Nous reviendrons sur les résultats de nos listes de proposition au paragraphe 6.4.

6.2 Sortie XML

Comme pour *Fips* , la sortie de *FipsOrtho* est un document *XML* complet (§5.3.1) pour chaque soumission au correcteur. Nous verrons à la section 6.3 que ce document est ensuite enrichi pour être inclus dans le corpus. La figure (6.2) montre un extrait de la sortie *XML* de *FipsOrtho*.

```
<LATLCORR xml:lang="fr">
  <SUBMISSION>
    <SENTENCE sentenceId="1">
      <ITEM index="i00001" pos="1" projcat="DP" gender="masc
      fem" number="plu" pers="6">
        <ORIGINAL itemTag="i00001">Les</ORIGINAL></ITEM>
      <PUNC key="space"/>
      <ITEM index="i00002" pos="2" projcat="NP" gender="masc
      fem" number="plu" pers="6">
        <ORIGINAL itemTag="i00002">travails</ORIGINAL>
        <PROPS itemTag="i00002">
          <PROPOSAL index="p00001" itemTag="i00002" cat="N"
          gender="masc" number="plu" pers="6" method="ad_hoc"
          dist="0.2" thresh="0.1533333333333333" score="23">
                travaux</PROPOSAL></PROPS>
      </ITEM>
      <PUNC key="space"/>
      <ITEM index="i00003" pos="3" projcat="TP" gender="masc
      fem" number="plu" pers="6">
        <ORIGINAL itemTag="i00003">sont</ORIGINAL></ITEM>
      <PUNC key="space"/>
      <ITEM index="i00004" pos="4" projcat="AP" gender="masc
      fem" number="plu" pers="4 5 6">
        <ORIGINAL itemTag="i00004">difficiles</ORIGINAL>
      </ITEM>
      <PUNC pos="5">.</PUNC>
    </SENTENCE>
</SUBMISSION></LATLCORR>
```

FIG. 6.2 – *Extrait de sortie* XML *pour la phrase (39)*

Chaque soumission au système est incluse dans une balise <SUBMISSION> et chaque phrase délimitée par *Fips* est marquée par une balise <SENTENCE>. La balise <ITEM> contient chaque élément de la phrase. Nous incluons dans cette balise les valeurs de traits et d'analyse filtrées par l'analyseur. Par ailleurs, chaque élément dispose d'un identifiant unique dans l'attribut *index*.

Pour chaque élément de la phrase, nous incluons la chaîne originale dans

l'élément <ORIGINAL>, qui est co-indicé avec <ITEM>. Si le mot est inconnu ou contient une erreur, nous ajoutons une balise <PROPS> qui contient la liste des propositions. Chaque proposition de correction est incluse dans la balise <PROPOSAL>, qui est co-indicée avec <ITEM> grâce à l'attribut *itemTag* et reçoit également un identifiant unique. Dans cette balise, nous ajoutons également les valeurs de la proposition, la ou les méthodes utilisées pour la retrouver, la distance lexicographique avec l'original (§6.1.2.1) et le seuil calculé entre ces deux éléments.

Après la correction interactive par l'apprenant, le document *XML* est modifié avant d'être sauvegardé dans une base de données. Si la proposition incluse est sélectionnée par l'apprenant, la balise <PROPOSAL> contient l'attribut *selected="yes"*. Si l'apprenant garde la proposition originale, cet attribut est ajouté à <ORIGINAL>. Si une correction est entrée manuellement, elle est incluse dans la balise <HUMAN_CORR>.

La validation pour le corpus (§6.3) donne lieu à une nouvelle modification du document *XML*. Si le document est sélectionné pour le corpus, on y ajoute les informations ajoutées et validées par l'expert. L'attribut *correctchoice="yes"* est ajouté aux balises <PROPOSAL> ou <ORIGINAL> de la proposition sélectionnée ; si une correction manuelle est entrée, l'attribut *expert="yes"* est ajoutée à la balise <HUMAN_CORR>.

Enfin, si l'utilisateur – l'apprenant – s'est enregistré, nous ajoutons son âge, son pays, sa langue première, son niveau et la date et l'heure de la soumission dans la balise <SUBMISSION>.

6.3 Description du système

Dans cette section, nous décrivons le système du correcteur *FipsOrtho* en général. Nous commençons par une description de l'architecture du système (§6.3.1). Puis nous décrivons brièvement la typologie des erreurs dont nous nous sommes servi pour annoter notre corpus (§6.3.2).

6.3.1 Architecture générale

FipsOrtho peut être librement testé sur *Internet* [13], tant en mode individuel, libre et anonyme, ou comme utilisateur authentifié dans le cadre

13. `http://latlcui.unige.ch/spellchecker`, dernier accès le 24.3.2009.

d'une utilisation en classe. L'annexe (§F) décrit en détail les interfaces offertes par le système. La figure (6.3) illustre l'utilisation du correcteur par un apprenant.

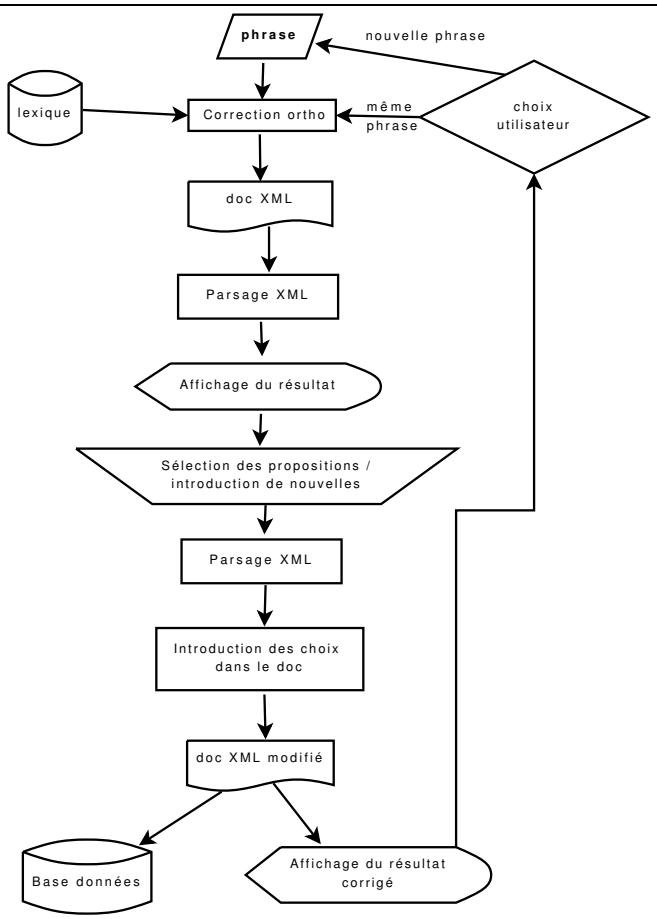


FIG. 6.3 – FipsOrtho *: vue d'ensemble du système, utilisation par l'apprenant*

Après le processus de correction, le correcteur orthographique produit un document *XML* (§6.2). Ce document est alors traité par un script *PHP* (§2.7.5) qui en affiche les propositions. Une fois leur phrase corrigée, les apprenants doivent valider leurs corrections. Le document *XML* de la phrase est alors à nouveau parsé et la proposition choisie est marquée ou la correction manuelle y est introduite. Le document est également inséré dans une table de la base de données. La phrase corrigée est réaffichée. L'apprenant a alors le choix entre proposer une nouvelle phrase ou corriger à nouveau sa production.

Passons maintenant à la partie corpus de notre système, dont le fonctionnement est illustré par la figure (6.4). Chaque envoi au correcteur (segment[s]

de phrase ou phrase[s]) est passé en revue par un expert et éventuellement inséré dans un corpus, selon le choix de l'expert d'après ses propres critères. Ainsi, les phrases de linguistes, les tests manifestes du correcteur et les phrases similaires ou identiques à d'autres entrées du corpus peuvent être écartés. Si la phrase est retenue, les choix des apprenants sont évalués par l'expert. En outre, celui-ci caractérise l'erreur en fonction d'une typologie que nous présenterons à la section 6.4. L'expert peut aussi baliser des erreurs non détectées. Ensuite, le système stocke chaque erreur et chaque proposition dans des tables de la base de données, afin de permettre de calculer des statistiques et pour en donner l'accès aux utilisateurs.

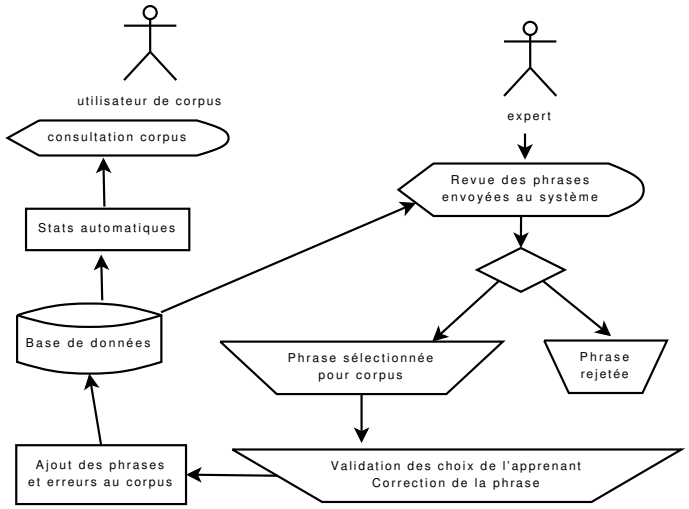


FIG. 6.4 – FipsOrtho *: récolte et consultation du corpus*

6.3.2 Typologie d'anotation du corpus

Comme nous l'avons vu à la section précédente, un expert est chargé d'annoter chaque mot de la phrase qui contient une erreur. Chaque erreur peut être annotée avec plusieurs catégories. Le tableau (6.7) liste les différentes catégories, avec leur code, leur description et des exemples. Lorsque cela était possible, nous avons regroupé les catégories dans des groupes. Les catégories séparées par des doubles lignes ne sont pas groupées avec d'autres et forment un groupe à membre unique.

Cette typologie a été inspirée par Catach *et al.* (1986), Cordier-Gauthier et Dion (2003) et Ndiaye et Vandeventer Faltin (2003). Précisons encore que certaines catégories sont détectées exclusivement par l'ordinateur, d'autres exclusivement par les humains, mais la majeure partie peut être détectée

Code	Désignation	Commentaire	Exemple
		Erreurs typographiques	
INS	Insertion	Caractère superflu	*cherval** → *cheval*
OMI	Omission	Caractère manquant	*a_bre** → *arbre*
SUB	Substitution	Touche voisine sur le clavier	*progrqme** → *programme*
INV	Inversion		*agneda** → *agenda*
		Erreurs lexicales	
LEX	Erreur lexicale	Mot existant mais inapproprié	*fonds* → *fondé*
NPR	Nom / adj. propre	Mot existant et correct, inconnu par le lexique	*Plymouth*
INC	Mot inconnu	Mot correct absent du lexique	*dravidien*
EMP	Emprunt	Emprunt de la langue maternelle	*trade* → *commerce*
		Erreurs phonétiques	
PHG	Err. phonogrammaticale	Mot inexistant mais prononciation correcte	*fonétique** → *phonétique*
PHO	Err. phonétique	Prononciation incorrecte d'un mot existant ou non	*londi** → *lundi*, **macasin* → *magasin*, **suivent* → à *suivant*
HPO	(quasi-)homophone	Mot existant inapproprié	*prémisses* → *prémices*, *est* → *et*
LNF	Lettres non fonctionnelles	Caractères non prononcés présents pour raisons historiques et étymologiques	*toujour** → *toujours*
DIA	Diacritiques	Accentuation	*Meme** → *même*
MOR	Err. morphologique	Erreurs morphologiques de conjugaison, formation de mots, marque du pluriel etc.	*rapident** → *rapides*
AGR	Accord		*les enfants sage**
CPL	Complémentation		*J'attends sur* Anne*
		Erreurs verbales	
AUX	Auxiliaire		*les invités sont* dansé*
TPS	Temps	Temps verbal	*joué* → *a joué*
MOD	Mode		*Je veux que tu viens**
		Erreurs de mots	
MAN	Mot manquant		*colonie _ tabac*
SUP	Mot superflu	Redondance	*les pêcheurs ~~entre~~ avec les Amérindiens*
		Erreurs de signe	
CAS	Casse	Min./maj. incorrecte	*Français** (langue) → *français*
PNC	Ponctuation		
SPC	Séparation par espace	Espace manquant ou superflu	*fauxsauniers** → *faux sauniers*
SEP	Séparation par autre signe		*sinstaller** → *s'installer*
ORD	Ordre des mots		*arrives-tu* → *tu arrives*
BRU	Bruit	Fausse détection	Ex. : majuscule inadéquate

TAB. 6.7 – *Typologie des erreurs du corpus d'erreurs orthographiques*

par les deux, comme nous le verrons en détail à la section 6.4.2.

6.4 Evaluation

Dans cette section, nous présentons les phases de test que nous avons fait passer à *FipsOrtho*. Nous présentons tout d'abord les résultats du test d'une liste de mots (§6.4.1), puis nous présentons notre corpus de phrases d'apprenants (§6.4.2). Enfin, nous terminons par quelques remarques finales sur l'évaluation (§6.4.3).

6.4.1 Test sur liste de mots

Nous avons en premier lieu testé notre correcteur avec une liste de 238 mots, tirée de Dinnematin *et al.* (1990), Burston (1998) et Softissimo (2002) avec des variations orthographiques de notre main. La liste et les résultats complets peuvent être consultés à l'annexe G. Le tableau (6.8) récapitule les résultats globaux du test :

Méthodes	**Score**	**%**	**Tot. cumulé**	**%**
Alpha + phono	57	23,95		
Alpha-wide + phono	14	5,88		
Alpha-narrow + phono	10	4,2		
Alpha	57	23,95	114	47,9
Alpha-wide	13	5,46	27	11,34
Alpha-narrow	11	4,62	21	8,82
Phono	25	10,5	106	44,54
Pas de proposition	20	8,4		
Pas de proposition correcte	31	13,03		
TOTAL:	238	100		

TAB. 6.8 – *Résultats globaux de la liste d'erreurs*

Seules les méthodes alpha-code et phonologique ont été prises en compte pour ce test, car il a été mené en grande partie au début de nos travaux, afin de mettre au point la distance lexicographique ainsi que de paramétrer la réinterprétation phonétique. Dans cette section, nous commentons les résultats globaux dans un premier temps puis nous revenons sur certains cas particuliers.

Commençons par traiter des résultats globaux. Sans surprise, les méthodes les plus efficaces sont l'alpha-code et la phonétique, tant seules qu'en conjonction avec d'autres méthodes, comme le montre la troisième colonne du tableau (6.8), qui donne le total cumulé des méthodes, seules et combi-

nées. A l'inverse, bien qu'utiles pour l'effacement ou l'insertion d'une lettre, les méthodes alpha-code élargi et restreint donnent des résultats plus faibles. De plus, nous échouons à donner une correction pour 51 mots (dont 20 sans proposition), soit 21,43%. Par ailleurs, 81 mots (34,03%) sont retrouvés par deux méthodes.

La méthode alpha-code fournit en moyenne 14,2 propositions, dont 1,61 passent le filtre de la distance lexicographique (11,37%). Pour l'alpha-code restreint, 57,9 propositions sont formulées en moyenne et 1,66 sont retenues (2,87%). En moyenne, 65,97% des alpha-codes restreints retournent des propositions, ce qui donne 10,64 propositions par alpha-code. Enfin, la méthode alpha-code élargie donne en moyenne 94,6 propositions, dont 2,56 sont sélectionnées (2,71%). En moyenne, 38,18% des alpha-codes élargis fournissent une proposition, ce qui donne une moyenne de 5,45 proposition par alpha-code. En considérant uniquement les chiffres, le rendement de la méthode peut sembler médiocre ; nous montrerons par la suite que les objectifs d'efficacité sont néanmoins atteints car les corrections souhaitées sont souvent sélectionnées.

Quant à la méthode phonétique, elle ne donne qu'une moyenne de 1,5 propositions, dont 0,61 nouvelles. L'algorithme génère entre 1 et 108 chaînes phonétiques à rechercher [14], soit une moyenne de 6,71 chaînes phonétiques par mot. Cependant, seulement 0,6 chaîne en moyenne retrouve une proposition, ou 18,77% des chaînes. Fréquemment, la bonne correction est également retrouvée par une méthode alpha-code, car il ne s'agit pas d'erreurs phonétiques proprement dites, mais d'erreurs d'insertion ou d'omission. Par ailleurs, 111 mots n'aboutissent à aucune proposition de correction par méthode phonétique, soit 46,64% ; sur les 127 mots aboutissant à une ou plusieurs propositions, 59 ne trouvent aucune proposition supplémentaire par rapport aux alpha-codes, soit 46,46%. Le mot **gaité* aboutit à 8 propositions supplémentaires trouvées par recherche phonétique, et **pens* à 9, ce qui sont des cas exceptionnels sur lequel nous reviendrons.

En moyenne, 148,6 propositions sont trouvées par mot. Grâce au filtre de la distance lexicographique, la liste des propositions est réduite en moyenne à 6,49 mots (4,37%), ce qui constitue une moyenne raisonnable, comme nous l'avons discuté à la section 6.1.8. Le mot **pens* compte toutefois 44 propositions, ce qui est excessif.

14. La chaîne **developpenet* donne 108 chaînes phonétiques à partir de l'original [dəvəlopənɛ], avec la substitution du [o], de chaque [ə] et du [ɛ]. La chaîne **coordonateur* retourne 48 chaînes phonétiques car les trois [o] sont substitués, ainsi que le son [a] et [œ].

Passons maintenant aux détails de la liste. La liste contient des erreurs classiques, même chez les locuteurs natifs. Ainsi *accueil* est souvent orthographié **acceuil*. Pour la variante **aceuil*, la méthode alpha-code restreinte trouve *écueil*, qui a une distance inférieure au seuil, mais est rejetée par la contrainte de la première lettre identique. On peut aussi citer **addresse*, fréquente chez les anglophones, et **aigüe* pour *aiguë*.

Attachons-nous maintenant aux détails des méthodes. L'alpha-code s'avère une méthode efficace. Ainsi, elle est capable de corriger des inversions (**rénumération*, **aéropage*, **acceuil*), des substitutions (**absorpsion*, **subcidiaire*), des insertions (**réddhibitoire*, **subbit*) et des omissions (**asujettir*, **profesionel*). Les techniques en jeu ne sont pas toujours les mêmes par type d'erreur : ainsi la substitution de **absorpsion* est trouvée par élargissement de l'alpha-code par un *t*, celle de **subcidiaire* par le retrait de *c* et celle de **simptomatique* par l'ajout de *y* ; l'insertion de **réddhibitoire* est trouvée par l'alpha-code original, tandis que celle d'**infarctusse* l'est par le retrait du *e* ; l'omission de **chrysantème* est trouvée par alpha-code original tandis que **control* est trouvé par l'ajout du *e*. Par contre, lors de la substitution de *sinthèse* ou **troglodite*, *i* remplace *y* et il n'y a ni insertion, ni retrait de lettre dans l'alpha-code [15]. Cette erreur n'est donc trouvée que par méthode phonétique.

Notre modification de la distance de Levenshtein pour les doubles lettres donne satisfaction, par exemple autour de *professionnel*, d'*imbécillité*, *intensément*, *charrette*, *souffle* etc. La proposition *voyagera* passe la rampe pour **voyera*, ce qui est excessif[16]. Il nous a manqué du temps et des ressources pour faire une évaluation qualitative de la distance.

La correction phonétique n'est de loin pas optimale. Certains résultats sont satisfaisants : on peut citer **acolite* pour *acolyte* [17], **algorythme*, **cyprés* [18], **négligeamment*. Un mot retourne même huit nouvelles propositions par méthode phonétique : **gaité* donne *guettait*, *guettaient*, *guettais*, *guettai*, *guetté(e)(s)*, *guetter* et *guettez*. Par contre, il n'est pas rare que des chaînes n'obtiennent pas de proposition, comme **aigue*, **barete*, **bifteek*, **boîter*, **braîment*, **contigüe* [19], **piqure*, **psiquédélique*, **reswa*, **angletterre*, **pre-*

15. Nous avons renoncé à traiter ce type de substitution, comme expliqué en 6.1.2 p. 194.
16. Cependant, il s'agit de la seule proposition retenue, ce qui vaut mieux que rien. Cette erreur pourrait être traitée grâce à une règle supplémentaire de la méthode *ad hoc* (§6.1.4).
17. Relevons qu'une proposition *accolait*, trouvée par alpha-code restreint, est retenue, avec un score assez proche du seuil.
18. Pour cet exemple, la substitution de phonèmes [e / ɛ] a correctement fonctionné, mais pas pour **cipre*, car le son [ə], qui est prononcé dans certaines circonstances et peut être substitué dans notre algorithme, ne se trouve pas dans la chaîne phonétisée par notre système ; le même cas se présente pour **malgre* et **malgres*.
19. *Contigu* est retrouvé par méthode phonétique, mais pas *contiguë*, qui n'était pas dans

sentimen, **raissonable*, etc. De plus, le déterminisme du système expert de phonétisation pose parfois problème : il renvoie par exemple [cɔnɛksjɔ̃] pour **connection*, alors que [cɔnɛktjɔ̃] est tout aussi vraisemblable. Enfin le système commet des erreurs, comme **attrapper* ou **râtisser* dont la fin est phonétisée en [-ɛʁ], **environment* en [ɑ̃viʁõmɑ̃] ou **rest* en [ʁɛs].

Passons maintenant aux cas particuliers. En premier lieu, des mots de la liste existent dans le lexique : *appas* est considéré dans la liste comme une erreur pour *appât*, mais est en fait une variante orthographique pour une acception vieillie du mot [20]. A l'inverse, certains dictionnaires acceptent *pourcent*, qui est absent du lexique. Le lexique contient parfois aussi des locutions comme *c'est-à-dire* ou *il y a*. Par contre l'inversion *y a-t-il* n'est pas dans le lexique [21].

En revanche, certains mots n'aboutissent à aucune proposition : les mots rares *accessit*, *consonant*, *drolatique*, *séborrhée*, *becquée* [22], *appoggiature* [23] et *syzygie* ne sont pas dans le lexique. **Malapris* et **malapri* ne donnent pas de correction valable, car *malappris* est absent du lexique ; cependant on retrouve *malarias* et *malaria* par suppression du *p*. *Congrûment* est également absent du lexique, mais la proposition *congruent* est sélectionnée par effacement du *m*. *Entropie*, pour **anthropie*, est absent du lexique mais on propose *anthropoïde*. Le mot **cipre* ne donne pas de proposition correcte mais *chiper*, *cire*, *ciré*, *cirée* et *cirer*. Idem pour **gueto* qui donne *guet*, *guette* et *guetté(e)*. Enfin **emploiés* ne donne aucune proposition phonétiques, mais les propositions erronées *emplies*, *emplîmes*, *emplis*, *emploi*, *emploie*, *emploies* et *emplois* .

Pour le mot **cettes*, qui est en fait une erreur morphologique, la solution *ces* est trouvée par alpha-code restreint mais éliminée par la distance lexicographique. **Journals* retrouve *journal*, ce qui est un bon compromis, comme une correction morphologique devrait trouver la forme correcte *journaux*. Au lieu de la correction attendue *font*, **faisent* donne *faisaient* et **faissent fassent*. Parmi les erreurs morphologiques, la méthode phonétique trouve *croyant* pour **croyent* et *prévoyant* pour **prévoyent*. Pour **devont*, la méthode phonétique trouve *devant*. Pour **voirai*, on a *voir*.

le lexique.

20. Equivalent d'*attraits* ou de *charme*.

21. *Il y a* est parfois considéré comme une préposition (*il est mort il y a deux ans*) ou comme un présentatif (*il y a de la bière dans le frigo*).

22. *buée* est proposée pour **béquée* ; quant à *bec* pour **beckee*, il est rejeté à cause de la distance.

23. Certains dictionnaires admettent la variante **appogiature*, qui est considérée comme une erreur dans la liste.

Le mot **esplication* donne le pluriel *explications* par insertion de la lettre *x* dans l'alpha-code. **Quesque* donne *quelques* et *queue*, mais pas la correction retenue. Ici, une méthode morphologique ou la méthode *ad hoc* feraient parfaitement l'affaire [24].

Parmi les mots qui donnent lieu à de nombreuses propositions, citons **asujettir* pour *assujettir*, avec 22 propositions. Dans cette liste se retrouvent exclusivement les formes conjuguées du verbe, de même pour **address* et **addresse*. Quant à **marasm*, il donne 19 propositions, dont la plupart sont trop éloignées comme *marcs*, *marks*, *marais* ou *mari*.

Enfin, certains mots comme **san* ou **sau* donnent lieu à plusieurs candidats vraisemblables, selon le contexte, comme *sangs* et *cent* pour l'un et *sceau*, *sot* pour l'autre. **Puit* donne *puits* et *puis* et **touts* donne *tous* et *tout* qui sont aussi vraisemblables les uns que les autres.

Signalons pour terminer certains mots introduits délibérément dans la liste alors que les méthodes en jeu sont impuissantes pour les traiter. Les mots comme **nonpossibles*, **experiment* ou **votage* pourraient être corrigés par analyse morphologique.

6.4.2 Corpus

Notre corpus consiste en 362 phrases ou séries de phrases soumises au correcteur, qui proviennent de différentes sources :

- phrases authentiques d'apprenants de Jamaïque, d'Australie et du Canada, provenant de textes libres, fournies par deux enseignants de langue ;
- phrases tirées d'articles scientifiques (Mogilevski, 1998; Cordier-Gauthier et Dion, 2003) ;
- phrases d'apprenants tirées d'un banc de test pour des correcteurs grammaticaux (Sanz, 1992), constituées de dictées du concours télévisé d'orthographe de Bernard Pivot, truffées d'erreurs artificielles ;
- texte d'un courriel de locuteur natif ;
- phrases authentiques d'apprenants d'un niveau plutôt avancé, recueillies pour le corpus FRIDA (§4) ;
- phrases entrées par des utilisateurs du correcteur en libre accès.

24. Faute de données sur les fréquences d'erreurs, nous avons renoncé à traiter davantage d'erreurs par la méthode *ad hoc*.

Malheureusement, malgré un appel sur une liste de discussion d'une association professionnelle et différentes tentatives auprès de collègues, il n'a pas été possible de recueillir des phrases directement saisies et corrigées par des apprenants. Nous avons donc dû sélectionner la solution correcte nous-mêmes à la place de l'apprenant. Puis nous avons joué le rôle de l'expert en corrigeant et en annotant les phrases.

Le corpus a été conçu de manière à conserver les indications sur l'âge, le sexe, la langue première, le pays de provenance et le niveau de langue. Si le corpus atteignait une taille suffisante et un échantillon significatif de niveaux et de provenances des apprenants, il serait possible d'en tirer des indications pour paramétrer le correcteur orthographique, afin de déterminer un ordre optimal des propositions et d'améliorer nos techniques. Nous avons dû renoncer à analyser le corpus en tenant compte des caractéristiques des apprenants, parce qu'il n'avait pas une taille suffisante, et surtout parce qu'aucun apprenant n'a corrigé ses productions lui-même.

Au final, nous avons un corpus de 1065 phrases, représentant 14 494 mots ; en moyenne, chaque entrée compte 40,04 mots et chaque phrase 13,61 mots. On peut attribuer plusieurs types à chaque erreur, soit lorsque plusieurs erreurs portent sur le même mot, soit lorsqu'il est possible d'attribuer plusieurs causes d'erreur sans pouvoir trancher. Le tableau (6.9) montre les statistiques du corpus.

Un peu plus du tiers des erreurs sont détectées par le correcteur, le reste est corrigé par l'expert. Le correcteur retourne une proposition correcte pour plus de la moitié des erreurs repérées ; parmi celles-ci, huit erreurs sur dix sont corrigées par une seule méthode. Dans la suite de cette section, nous examinerons de manière les résultats des méthodes (§6.4.2.1). Nous passons aux erreurs plus problématiques (§6.4.2.2), où une erreur a été détectée mais n'a pas pu obtenir de correction ou n'a pas été détectée et corrigée manuellement. Enfin nous analysons les résultats par type d'erreur (§6.4.2.3).

6.4.2.1 Résultats par méthodes

Le tableau (6.10) détaille les résultats par méthode utilisée pour trouver la proposition correcte. Globalement, les performances du correcteur sont satisfaisantes : en moyenne 7,86 propositions sont retournées, avec un pic de 54 ; la proposition correcte se trouve en moyenne à la position 1,61.

Globalement, le corpus contient 6776 propositions en tout, pour 719 erreurs ; nous comptons 595 non-mots différents, dont parfois l'ordre des pro-

<table>
<tr><td colspan="6">Nombres d'erreurs dans le corpus : 2468</td></tr>
<tr><td colspan="5">Mots inconnus détectés: 861 (sur 14 494 mots, soit 5,94%)</td><td rowspan="3">Erreurs non détectées, correction par expert: 1607 (65,11%)</td></tr>
<tr><td colspan="2">Proposition automatique correcte grâce aux méthodes du correcteur: 460 (53,43%)</td><td colspan="2">Non-erreurs: 213 (24,74%)</td><td colspan="2">Erreurs corrigées manuellement: 188 (21,84%)</td></tr>
<tr><td>Par une seule méthode: 368 (80%)</td><td>Par 2 méthodes ou plus: 92 (20%)</td><td>Avec prop. de correction: 119 (55,87%)</td><td>Sans prop. de correction: 94 (44,13%)</td><td>Avec prop. de correction: 140 (74,47%)</td><td>Sans prop. de correction: 48 (25,53%)</td></tr>
</table>

TAB. 6.9 – *Statistiques du corpus*

Nom	Score	Seule	Combi	% seul	Ordre moyen
ad hoc	5 (1,09%)	3 (0,65%)	2	60	2,75
alpha-code	373 (81,09%)	241 (52,39%)	132	64,61	1,4
alpha-narrow	60 (13,04%)	37 (8,04%)	23	61,67	2,24
alpha-wide	71 (15,43%)	50 (10,87%)	21	70,42	2,4
majuscule	7 (1,52%)	7 (1,52%)	0	100	1
apostrophe	4 (0,87%)	4 (0,87%)	0	100	1
phonétique	191 (41,52%)	23 (5%)	168	12,04	1,33
séparation	5 (1,09%)	3 (0,65%)	2	60	1

TAB. 6.10 – *Corpus: résultats par méthode pour les propositions sélectionnées*

positions diffère, selon la catégorie lexicale attribuée au mot inconnu d'après le contexte. Le tableau (6.11) résume les résultats de chaque méthode.

Nom	Score	Seule	Combi	Moy. dist/seuil
ad hoc	5 (0,07%)	3 (60%)	2 (40%)	95,65%
alpha-code	1550 (22,87%)	1338 (86,32%)	212 (13,68%)	52,31%
alpha-narrow	1709 (25,22%)	1596 (93,39%)	113 (6,61%)	74,42%
alpha-wide	3341 (49,31%)	3205 (95,93%)	136 (4,07%)	76.03%
majuscule	27 (0,4%)	27 (100%)	0 (0%)	0%
apostrophe	14 (0,21%)	14 (100%)	0 (0%)	9.94%
phonétique	921 (13,59%)	464 (50,38%)	457 (49,62%)	74,74%
séparation	84 (1,24%)	82 (97,62%)	2 (2,38%)	0%

TAB. 6.11 – *Corpus: résultats par méthode pour l'ensemble des propositions*

Enfin, le tableau (6.12) établit les correspondances entre méthodes et types d'erreurs. Passons maintenant à l'examen détaillé des méthodes.

6.4.2.1.1 Alpha-code Nous pouvons considérer la méthode alpha-code comme la plus efficace, puisque elle permet de retrouver plus de huit propositions correctes sur dix, ou plus de la moitié des propositions si l'on ne considère que le cas où la méthode est seule à trouver la proposition correcte (dans près des deux tiers des cas). Pour le nombre global de propositions, la proportion est de plus d'une proposition sur cinq. Sans surprise, les propositions fournies par cette méthode sont les plus proches du mot inconnu, puisque les propositions trouvées par cette méthode sont par nature les plus proches de la chaîne erronée. L'ordre moyen des propositions correctes, légèrement supérieur à un, est également excellent.

En examinant le tableau (6.12), on constate que la méthode permet de retrouver de nombreuses erreurs de diacritiques, puisque l'alpha-code ne tient pas compte des caractères accentués. Le bon score des erreurs d'omission, d'inversion, d'insertion et de substitution, pour lesquels cette méthode

Code / Mét	AGR	AUX	BRU	CAS	CPL	DIA	EMP	HPO	INC	INS	INV	LEX	LNF	MAN
ad hoc	-	-	-	-	-	-	-	-	-	-	-	-	-	-
alpha	21 (5,63%)	-	2 (0,54%)	5 (1,34%)	1 (0,27%)	224 (65,05%)	22 (5,9%)	-	1 (0,27%)	60 (16,09%)	9 (2,41%)	26 (6,97%)	2 (0,54%)	11 (2,95%)
alpha-narrow	10 (16,67%)	-	-	1 (1,67%)	3 (5%)	9 (15%)	9 (15%)	-	1 (1,67%)	20 (33,33%)	1 (1,67%)	5 (8,33%)	1 (1,67%)	-
alpha-wide	11 (15,49%)	-	-	1 (1,41%)	-	7 (9,86%)	5 (7,04%)	-	-	-	-	4 (5,63%)	2 (2,82%)	3 (4,23%)
maj	-	-	2 (28,57%)	5 (71,43%)	-	-	-	-	-	-	-	-	-	-
apostrophe	1 (25%)	-	-	-	-	-	-	-	-	-	-	-	-	-
phonétique	12 (6,28%)	-	-	1 (0,52%)	-	83 (43,46%)	9 (4,71%)	-	1 (0,52%)	30 (15,71%)	1 (0,52%)	14 (7,33%)	5 (2,62%)	4 (2,09%)
séparation	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	MOD	MOR	NPR	OMI	ORD	PHG	PHO	PNC	SEP	SPC	SUB	SUP	TPS
ad-hoc	1 (20%)	5(100%)	-	-	-	-	-	-	-	-	-	-	1 (20%)
alpha	-	10 (2,68%)	2 (0,54%)	62 (16,62%)	2 (0,54%)	39 (10,46%)	58 (15,55%)	3 (0,8%)	12 (3,22%)	3 (0,8%)	7 (1,88%)	1 (0,27%)	1 (0,27%)
alpha-narrow	1 (1,67%)	6 (10%)	6 (10%)	2 (3,33%)	-	16 (26,67%)	18 (30%)	1 (1,67%)	-	1 (1,67%)	19 (31,67%)	-	2 (3,33%)
alpha-wide	2 (2,82%)	6 (8,45%)	-	42 (59,15%)	-	11 (15,49%)	28 (39,44%)	-	-	1 (1,41%)	10 (14,08%)	1 (1,41%)	3 (4,23%)
maj	-	-	-	-	-	-	-	-	-	-	-	-	-
apostrophe	-	-	-	1 (25%)	-	-	-	-	4 (100%)	-	-	-	-
phonétique	1 (0,52%)	7 (3,66%)	3 (1,57%)	37 (19,37%)	3 (1,57%)	49 (25,65%)	17 (8,9%)	-	9 (4,71%)	2 (1,05%)	23 (12,04%)	-	-
séparation	-	-	-	-	-	-	-	-	1 (20%)	4 (80%)	-	-	-

TAB. 6.12 – *Corpus: correspondances entre méthode et type d'erreur*

entre en jeu fréquemment, confirme nos assertions sur l'adéquation de cette méthode pour corriger des erreurs de ce type.

L'examen des propositions retenues nous permet également de constater que notre distance lexicographique fonctionne correctement et favorise souvent la bonne proposition. En effet, la méthode alpha-code trouve souvent les erreurs portant sur les doubles consonnes comme **languissament* ou **noctambulles*. Comme ces erreurs sont faiblement pénalisées et qu'une petite distance lexicographique donne un score plus élevé, ces propositions ressortent particulièrement.

Passons maintenant aux résultats globaux de la méthode. Comme nous avons déjà pu le souligner, la méthode permet de retrouver des erreurs variées : l'erreur d'insertion **millieurs* donne la proposition *meilleurs* et **develloped*s → *développés*, les erreurs phonétiques **Vegetarian* → *végétarien* et **primière* → *première*, et les erreurs d'omission **enchaterent* → *enchantèrent* et **politque* → *politique*, etc.

Dans certains cas, le nombre de propositions est excessif. Ainsi l'erreur **trés*, qui est un mot court, donne pas moins de 49 propositions, dont *tresses* ou *tressés*, dont la distance est très proche du seuil. De même, pour **sure*, le système retrouve *susurre*, *Laissas* et *Lisais* pour *Lisa* et *pressés* et *presser* pour **pres*, pour **aiseé Asie* et *aies*.

La distance lexicographique est également très grande dans le cas de *fessaient* pour l'erreur **faissaient*, qui retourne dix propositions. On peut également rejetter les propositions *défiant* et *défendait* pour l'erreur **définait* [25], qui sont également proches du seuil.

Examinons maintenant l'efficacité du seuil de distance pour sélectionner ou rejeter des propositions. Le seuil $\mathcal{D}$ (v. p. 198) accepterait de justesse *accueillera* et *accueillir* pour **acceuiller*, *meilleur* pour **mieulleur*, *travaille* sur **travillie* ou *traditions* pour **tradions*. Pour **coute*, on retrouve *couette*, qui est une proposition assez éloignée, mais néanmoins largement au dessous du seuil $\mathcal{D}$, à cause de la faible pénalisation des doubles consonnes. Par contre, ce seuil rejetterait *recommander* pour **recomendera*, qui sont néanmoins en dessous du seuil. Ces faits plaident donc plutôt pour le maintien du seuil $\mathcal{D}\prime$.

En ce qui concerne la règle de la première lettre identique, on peut regretter **studes* pour *études*, qui trouve la solution *sud-est* mais rejette la

25. Le contexte nous indique que l'apprenant voulait écrire *définit* - la première proposition de la liste qui a été sélectionnée - ou *définissait*.

proposition correcte qui se trouve cependant au dessous du seuil. Le même constat vaut pour **vaient* pour *avaient*.

Pour conclure, la méthode donne des résultats très bons ; elle se retrouve dans de nombreuses solutions, seule ou avec la méthode phonétique. L'ordre moyen des propositions retrouvées par alpha-code est aussi très satisfaisant.

6.4.2.1.2 Alpha-code restreint La méthode d'alpha-code restreint, où une lettre de l'alpha-code original est retirée, retrouve un peu plus de treize pourcent des propositions correctes. Dans plus de huit pourcent des cas, cette méthode est la seule à intervenir pour trouver la proposition correcte. En revanche, elle intervient dans plus de vingt-cinq pourcent du total des propositions. L'ordre moyen des propositions sélectionnées est de deux.

Les erreurs d'insertion sont les plus fréquentes pour cette méthode, du fait qu'un caractère superflu est retiré, notamment lors de fautes de frappe, comme **loa* pour *la*, **millieurs* pour *milliers* ou **certainq* [26]. Les erreurs phonogrammatiques et phonétiques arrivent en seconde position. Enfin, citons une proportion significative d'erreurs de substitution, comme **function*, **specialization* ou **précedante*.

Venons-en maintenant aux résultats globaux de la méthode. Grâce à une lettre présente par erreur, on trouve des erreurs phonétiques *désastre* pour **disastre* ou phonogrammatiques comme *exemple* pour **example*. La méthode trouve également beaucoup de mots au singulier pour des pluriels erronés, comme *journal* pour **journals*, *créneau* pour **crenaux* ou *année* pour **anées*. Même si la proposition n'est pas correcte, citons *intéressent* pour **interesant*, *À travers* pour **Autravers* et *tout* pour **touts*.

On constate aussi de nombreuses propositions qui devraient être rejetées mais sont gardées bien que d'une distance proche du seuil $\mathcal{D}\prime$. Pour **acceuiller*, la méthode donne *accueille* et *acculer* ; la première proposition atteint le seuil $\mathcal{D}$, mais la seconde proposition atteint une distance supérieure. On peut citer *fourrées* pour **fourures*, *test*, *terré* ou *tètes* pour **trés*, *unisson* pour l'erreur de séparation **unpoisson*, etc. Pour **tristent*, on obtient *triste* et *tristes*, toutes deux juste au seuil $\mathcal{D}$. En revanche, *défait* pour **définait* atteint ce seuil mais n'est pas une proposition souhaitable, de même que *Aérée*, *Aère* et *Aéré* pour *ARTE*. Pour **sure*, on retrouve 45 propositions, dont de nombreuses propositions comme *serre*, *suée* ou *sers* sont trop éloignées. Il serait judicieux de prévoir un traitement spécial pour les mots de faible

26. On constate le même cas pour l'erreur de substitution **cigareltes* ou inversion (**receuillis*).

longueur, où **coute* donne *coté*, *cotée* et *cote*, tous trois à distance inférieure ou égale au seuil $\mathcal{D}$.

En revanche, pour **éducationnel*, la méthode trouve l'unique proposition *éducation*, qui est au dessus de $\mathcal{D}$ [27]. Pour **partoger*, les six propositions *partager*, *partagera*, *partagea*, *partagée*, *partage* et *partagé*, toutes trouvées par cette méthode, peuvent être considérées comme pertinentes, bien que les cinq dernières soient soit égales, soit supérieures à $\mathcal{D}$. La remarque s'applique également à *prestigieuse* pour **préstigieux* et *sérieuse* pour **serieux*. Enfin, **finisent*, *finisse* et *finisses* sont au dessus de $\mathcal{D}$, en troisième et quatrième position. Ainsi, tant que les propositions sont peu nombreuses, moins de cinq par exemple, il est judicieux de garder même les propositions éloignées, au moins celles appartenant à un même lemme.

Pour **ingénière*, la méthode retrouve *ingénie*, au seuil $\mathcal{D}$ et *ingéniée*, qui le dépasse. Comme ces deux propositions appartiennent au verbe pronominal *s'ingénier*, une heuristique permettrait de les éliminer en fonction du contexte où il n'y a pas de pronom personnel.

La règle de pénalisation moindre des erreurs entre double et simple consonnes donne parfois lieu à des corrections absurdes: *déférente* pour **differente* est très largement au dessous de $\mathcal{D}$, alors que *déférent* en est très proche ; de même, *réelle* est donné pour **revélé*. En revanche, la règle est adéquate pour *acacia* pour **accacias* ou *quidam* pour **quiddams*. Il serait peut-être judicieux de ne déclencher cette règle que si les voyelles entourant la ou les consonnes sont identiques, accents compris. Cependant, le corpus manque encore de données pour juger de l'applicabilité d'une telle règle.

En fin de compte, cette méthode intervient dans un bon nombre de cas, mais bien que les propositions soient nombreuses, elles sont nettement plus rarement retenues. C'est pourquoi cette méthode peut être jugée comme moyennement efficace.

6.4.2.1.3 Alpha-code élargi Passons maintenant à la méthode de l'alpha-code élargi. Elle se retrouve dans près de quinze pourcent des propositions correctes. Dans près de onze pourcent des cas, elle est seule à intervenir. Par contre, près de la moitié des propositions globales sont trouvées par cette méthode, ce qui en fait la méthode la plus prolifique ; ceci s'explique par le nombre élevé d'alpha-codes générés par cette méthode par rapport aux autres méthodes d'alpha-code et aux clés de recherche phonétique. Enfin, la moyenne de distance lexicographique par rapport au seuil est relativement

27. La proposition correcte est *éducatif*.

élevée.

Les erreurs d'omission sont très fréquentes pour cette méthode. En effet, une lettre est ajoutée à l'alpha-code du mot inconnu. On trouve des erreurs comme **mangons* pour *mangeons*, **remarqable* pour *remarquable* ou **enfans* pour *enfants*. Les autres types d'erreur sont beaucoup moins fréquents, à l'exception des erreurs phonétiques, qui sont un type fréquent mais qui ne peuvent être corrélées avec la méthode ; en effet, si l'omission d'une lettre aboutit à une erreur phonétique, ce n'est pas une relation de cause à effet.

Examinons maintenant les détails des résultats. Il est fréquent qu'une lettre soit manquante, comme dans **origin*, *exploration* pour **exporation* et *Thème* pour **Tem* [28]. *Connaître* pour **connetre* est à la limite du seuil $\mathcal{D}$ et seulement en treizième position. Comme une lettre est ajoutée, il est également fréquent que cette méthode retrouve des mots au pluriel comme *aéroports* pour **aeroport*. Fortuitement, la méthode trouve aussi les solutions *viendras* [29] et *vendrais* [30], à une distance égale au seuil bas pour **veniras*. Pour **plutot*, on trouve *plus tôt* qui se trouvait en tant que locution dans le lexique.

Un trop grand éloignement de la chaîne d'origine conduit, ici aussi, à des résultats non désirés. Ainsi de nombreux résultats atteignent ou dépassent $\mathcal{D}$: *fourrières* ou *fourbes* pour **fourures*, *pour que* pour **poque*, *titres* et *tirets* pour **trés*. Des mots sont absents du lexique et conduisent à des corrections absurdes. *Inca* donne notamment *Incita* et *Incisa*. *Caetera* donne notamment *citera*, *capter* et *carters*. De plus, *t-shirt* est absent du lexique : seuls *T-shirt* et *tee-shirt* [31] sont acceptés.

Certains mots donnent déjà lieu à des corrections assez surprenantes bien que d'une distance proche de la chaîne d'origine. Ainsi, pour **prefere*, le correcteur propose *profère* et *pluviers* et *plumiers* pour **plusiers* ; quant aux propositions *préfet* et *perforé*, elles sont au dessus du seuil $\mathcal{D}$. Enfin, signalons que la règle de la première lettre identique conduit parfois à l'absence de résultat correct : **poque* est une erreur pour *époque*.

Pour terminer nous pouvons faire les mêmes constatations que pour la méthode d'alpha-code restreint et juger cette méthode comme d'efficacité moyenne.

28. La proposition est à un ratio de distance de 91,3% du seul $\mathcal{D}\prime$, en troisième position.
29. Nous analyserons ce cas en détail au paragraphe 6.4.2.1.5.
30. Cette dernière solution est beaucoup moins heureuse que la première.
31. Cette proposition atteint presque le seuil $\mathcal{D}\prime$ et est donc sélectionnée *in extremis*, puisqu'une proposition atteignant ce seuil serait rejetée.

6.4.2.1.4 Méthode phonétique La méthode phonétique intervient pour plus de quarante pourcent des propositions correctes, mais elle est seule à intervenir dans seulement cinq pourcent des cas, ce qui en fait une méthode complémentaire aux méthodes d'alpha-code. En revanche, pour les résultats globaux de toutes les propositions, elle a un score moyen d'environ 13,5% et est seule à fournir une proposition dans plus de la moitié des cas. L'ordre moyen des propositions correctes est également bon, avec une moyenne de 1,7.

La méthode phonétique permet avant tout de retrouver les erreurs de diacritiques ; les erreurs de ce type font toujours intervenir une méthode d'alpha-code, ce qui explique en partie le fait que la méthode est efficace surtout en combinaison avec d'autres méthodes. Parmi ces cas, citons **economique*, **problem*, **ètudiants*, etc. En seconde position viennent les erreurs phonogrammatiques, qui marquent les erreurs débouchant sur une prononciation correcte, comme **Cependent*, **example*, **famme* ou **respet*. Les erreurs d'insertion et d'omission débouchent sont également bien représentées dans les erreurs trouvées par la méthode phonétique, si les lettres omises ou insérées ne changement pas ou peu la prononciation, comme dans **amitiée*, **noctambulles*, **sussurrés*, **nouriture*, **heurese* ou **seron*. Enfin, les erreurs phonétiques débouchent, elles, sur une véritable erreur de prononciation, comme dans **Chére*, **hautage* ou **ein* pour *un*.

Voyons à présent les détails des résultats de la méthode phonétique. Dans certains cas, elle est seule à découvrir la correction d'une erreur : pour **loix*, **cosmopolytes*, **jamaïquain*, **hautage*, **tros*, etc., les méthodes d'alpha-code ne peuvent remplacer la ou les lettres substituées ; par contre, pour **arjan*, **ensiennes*, etc., on a le cas d'une écriture complètement phonétique. Lorsque la méthode trouve une proposition conjointement aux méthodes d'alpha-code, les chaînes erronées sont souvent très proches des solutions, comme dans **energies*, **bien-tôt* ou **sentance*.

Pour **trés*, on obtient *très*, *trait*, *traie(s)*, *trais*, *traits* et *traient*. Le calcul du score pousse judicieusement en tête la proposition *très*, qui se perdait auparavant au milieu d'une multitude de propositions ; en effet, cette erreur donne lieu à 49 propositions, dont beaucoup ont un score identique. En additionnant les scores des propositions, on arrive à un score de 11, qui est plus petit que les scores obtenus par d'autres propositions qui sont des noms, comme l'analyse de *Fips* l'a établi par erreur pour la chaîne inconnue, ce qui donne des points supplémentaires. En favorisant les propositions à distance très faible de la chaîne inconnue, le problème est résolu.

Parfois, les substitutions de phonèmes peuvent donner lieu à des effets

non désirés. Par exemple, pour **onts*, la méthode donne *ans* en seconde position [32], tandis que la solution correcte *ont* vient en cinquième position ; le mot inconnu est considéré par *Fips* comme un nom, ce qui contribue à faire ressortir *ans*, *on* et *an* qui ont un score de 15, tandis que le bon choix a un score de 14. On peut également regretter *fessée* pour **faissaient*. Ici, la substitution [e / ɛ] n'est sans doute pas judicieuse, car la terminaison *-aient* est une indication que l'apprenant a une certaine maîtrise des terminaisons verbales. En revanche, l'élargissement des propositions donne de bons résultats pour **voe*, phonétisé comme [və/œ] et par conséquent [vɛ] et [vø].

Les emprunts à une langue étrangère sont parfois plutôt correctement phonétisés ; *sea* donne *ses*, *ces*, *sait* et *sais*. *Day* donne *Des*, *Dé(s)* et *Dès*. Par contre, *Last* donne *Lasse*, *Lacent*.

La méthode est aussi adéquate pour détecter des homophones et quasi-homophones lorsqu'un mot n'est pas correctement orthographié. Ainsi, pour **hotel*, on retrouve *hôtel(s)* et *autel(s)*.

Le déterminisme de la phonétisation empêche une plus grande efficacité de la méthode. Ainsi, **courrent* est phonétisé [kuʁɑ̃], alors que [kuʁ] est tout aussi vraisemblable. De même, des règles d'erreur pourraient encore ajouter la phonétisation [kuʁɑ̃t]. Le même phénomène touche **dancent* et **pouvent*. On peut citer aussi **fase* phonétisé [faz] alors que [fas] était aussi possible et souhaitable. Pour **temp*, on a [tɑ̃p], ce qui est discutable, et pas [tɑ̃], alors que pour **pens* on a [pɑ̃] et pas [pɑ̃s]. Par contre, pour **siseaux*, on obtient par chance la solution [sɪzo], alors que [sɪso] devrait être également trouvé par règle d'erreur, d'autant plus pour un correcteur orthographique. Une erreur est même à déplorer pour **dancer*, phonétisé [dɑ̃sɛʁ] et trouve *danseur* par substitution du son [œ] [33]. Pour **pay*, on trouve *paye* ([pɛj]), mais il serait sans doute également judicieux de phonétiser [peɪ]. Le mot **connettre* est phonétisé [kɔnɛtʁ], mais pourrait aussi donner [kɔnetʁ] et même [kɔnətʁ] avec une erreur de prononciation. Le mot **travai* donne [tʁave], mais devrait donner également [tʁavɛ] et, par une règle d'erreur, trouver la proposition correcte avec [tʁavaj]. Les règles d'erreur devraient intégrer les règles de phonétisation d'autres langues : pour **clear*, on phonétise [klɛʁ], mais on devrait également avoir [klɪʁ]

32. La première position est occupée par la proposition *ont s*, grâce au poids de la méthode de séparation de mots. Nous y reviendrons.
33. Par contre, pour **protecter*, au lieu de *protéger*, cette phonétisation permet de suggérer *protecteur*, conjointement avec l'alpha-code élargi, ce qui est sans doute la meilleure proposition possible. Il est en effet délicat de définir une règle morphologique pour éviter cette dérivation suite à emprunt à l'anglais *to protect*.

Concluons en remarquant que les futurs travaux d'amélioration de *FipsOrtho* pourraient porter prioritairement dans cette direction. En effet, empiriquement, chacun peut remarquer que les erreurs phonétiques sont plutôt bien corrigées par les correcteurs orthographiques disponibles sur le marché. Si l'on comparait les performances de notre correcteurs avec d'autres logiciels – ce que nous n'avons malheureusement pas eu le loisir de faire – *FipsOrtho* ne se distinguerait pas, voire, probablement, aurait des performances inférieures. Il est donc essentiel de corriger les faiblesses soulignées ici. Cependant, les résultats de cette méthode peuvent tout de même être jugés comme satisfaisants.

6.4.2.1.5 Méthode *ad hoc* La méthode *ad hoc* n'est intervenue que dans un peu plus d'un pourcent des cas pour retrouver la proposition correcte, dont deux cas sur cinq en combinaison avec une autre méthode. Comme la méthode est très sélective, elle a trouvé la bonne proposition dans tous les cas où elle est intervenue. Si l'on tient compte de toutes les propositions, elle intervient donc dans 0,7 pour mille des cas. Ce faible score est dû à la faible couverture des cas et pourrait augmenter si l'on ajoute de nouvelles règles. Enfin, soulignons que l'ordre moyen de la proposition sélectionnée est de 2,75, à cause du calcul des scores et de l'ordre des propositions.

Examinons maintenant les types d'erreurs trouvés par la méthode *ad hoc*. Bien évidemment, on trouve en tête le type morphologique. Il s'agit d'erreurs de conjugaison, **veniras* pour *viendras* [34], **devé* pour *dû* et **journals* pour *journaux*.

Passons pour finir aux lacunes de la méthode. Par exemple, **éducationnel* n'a qu'une proposition erronée, mais la solution *éducatif* pourrait être corrigé par une règle *tionnel#* → *tif*. Vu la bonne qualité des performances de la méthode, il serait sans doute adéquat d'en augmenter le score et de placer les propositions trouvées par cette méthodes avant les autres en cas de score identique, même si la distance lexicographique est plus grande. Globalement, cette méthode fonctionne correctement mais ne couvre pas suffisamment de cas.

6.4.2.1.6 Apostrophe manquante La méthode d'apostrophe manquante n'intervient que dans quatre cas, soit moins d'un pourcent, et ne

34. Nous avons comptabilisé une erreur de mode pour une des deux occurrences de cette erreur, car le subjonctif était nécessaire pour "*je suis très contente que tu viennes* [...]". Cette erreur est aussi corrigée par la méthode alpha-wide.

trouve jamais la solution correcte conjointement avec une autre méthode. Elle est toujours classée première des propositions. Si l'on prend en compte la totalité des propositions, la méthode intervient dans un peu plus de 2 pour mille des cas.

Sans surprise, toutes les erreurs trouvées par cette méthode sont des erreurs de séparation. Nous pouvons citer **sinstaller*, **leau* et **lécole*. Quant à la proposition **J'a* pour **Ja*, elle a été sélectionnée bien que incorrecte. Il s'agit d'une première partie de correction, qui devrait être complétée par un correcteur grammatical pour l'erreur d'accord, si l'apprenant ne se rend pas compte de l'erreur par lui-même.

Parmi les propositions incorrectes, on peut citer *L'aura* pour le prénom *Laura*, inconnu du lexique, *C'a* pour *Ca*, qui est considéré comme une erreur, bien que les diacritiques soient optionnels pour les mots en majuscules [35], et **d'i* pour *di*. L'abréviation *JA* pour le dollar jamaïquain est également corrigée *J'A* : il serait sans doute adéquat de prévoir une option pour ignorer les mots tout en majuscules.

En conclusion, soulignons que nous manquons de données pour évaluer la méthode et pour esquisser des pistes d'améliorations. Cependant, les résultats de cette méthode nous ont tout de même parus assez satisfaisants.

6.4.2.1.7 Séparation de mots La méthode de séparation de mots intervient dans cinq propositions correctes, dont trois en combinaison de méthode, soit un peu plus de un pourcent. Au niveau global des propositions, la proportion est *grosso modo* la même. La proposition correcte trouvée par cette méthode est toujours au premier rang.

La méthode retrouve, bien évidemment, les erreurs de séparation par un espace. Les erreurs trouvées sont **fauxsauniers* (*faux sauniers*), **unpoisson* (*un poisson*), **Haha* (*Ha ha*), **Autravers* (*Au travers*) [36], *plustôt* (*plus tôt*) et **weekend* (*week-end*). Comme les trois dernières erreurs portent sur des mots contenus dans le lexique, elles sont également trouvées par la méthode d'alpha-code.

Parmi les propositions non retenues, citons *A part* pour **Apart*, alors que *À part* est retenu et trouvé par les méthodes alpha-code et phonétique. On

35. Même si l'utilisation de *ça* appartient au registre parlé, il serait judicieux de pouvoir relâcher un comportement trop normatif de *Fips*.
36. Comme l'erreur commence par une majuscule, le système n'a pas détecté que *au travers* se trouve dans le lexique et compte deux propositions distinctes.

peut également citer *heure se* pour **heurese* et *ex ample* pour **example*, qui pourraient s'avérer correctes dans un contexte donné.

En revanche, d'autres propositions non retenues montre un fonctionnement inadéquat de la méthode. Par exemple, **onts* donne *ont s*, **Garden* donne *Gard en* et *Garde n*. Il est donc indispensable de modifier la méthode et de d'examiner que des segments de mots de deux caractères au minimum.

Pour terminer, les mots trouvés par cette méthode ont toujours une distance de 0, ce qui les favorise systématiquement. De plus, elle dispose déjà d'un poids élevé. Le score supplémentaire accordé aux faibles distances n'est donc pas justifié si cette méthode est seule à fournir la proposition.

6.4.2.1.8 Majuscule Étonnamment, on ne trouve que sept propositions correctes pour la méthode de majuscule, toutes sans le concours d'une autre méthode. Par ailleurs, le score global de la méthode pour toutes les propositions est de quatre pour mille.

La grande majorité des erreurs trouvées par cette méthode sont des erreur de casse. Cependant, les erreurs de découpage des phrases de *Fips* entraînent de nombreuses fausses détections. Le problème est particulièrement aigu avec des majuscules demandées après les signes de ponctuation ":" et ";". De même, les phrases agrammaticales donnent du fil à retordre au système : **Voyez-vous bientôt*, traduit littéralement de *see you later!*, est analysé comme deux phrases, avec une proposition sans verbe.

Au vu de ce qui précède, le fonctionnement de la méthode est insuffisant et doit être absolument amélioré. Néanmoins, le corpus ne contient pas assez d'exemples pour pouvoir décider s'il est judicieux de conserver cette méthode en mettant en place des heuristiques ou s'il faut simplement l'abandonner.

6.4.2.2 Erreurs problématiques

Dans cette section, nous discutons des cas problématiques où le correcteur est tenu en échec. Nous traitons successivement des non-erreurs, des erreurs corrigées manuellement et des erreurs non détectées.

6.4.2.2.1 Non-erreurs Les non-erreurs sont des mots inconnus qui ont été laissés tels quels et considérés comme corrects. Ils n'ont donc pas été rem-

placés par une proposition, même si parfois des corrections ont été proposées. Il s'agit essentiellement de noms propres comme *Champlain*, *Nouvelle-France*, ou de mots inconnus du lexique comme *acadiennes*, *ferry*, *moucharabiehs*. Mentionnons aussi des abréviations de mots courantes dans le langage parlé, comme *dicos* ou *sympas*, ou les sigles et acronymes comme *TP* (ou *tp*), *NTIC* ou la somme d'argent en dollars jamaïquains notée *$JA2650*. Dans cette catégorie, nous rangeons aussi les nombreux emprunts aux langues étrangères, comme le titre de film *The Day After Tomorrow* ou les noms de restaurant comme *Guilt Trip*. Les ordinaux en chiffre romains comme *XVIe* sont également considérés comme inconnus. Enfin, on trouve les fausses détections, marquées par la catégorie bruit ; il s'agit essentiellement d'erreurs de fonctionnement de la méthode majuscules qui veut insérer une lettre majuscule de façon erronée.

Penchons-nous maintenant sur les erreurs causées par le segmenteur de phrases de *Fips* : lorsque un mot composé ou un mot en plusieurs parties est entré dans le lexique, *Fips* cherche à construire le mot complet. Or si le mot complet n'est pas trouvé, *Fips* considère hélas le mot suivant comme faisant partie d'un même mot inconnu en deux parties, même si le second mot est parfaitement connu.

(45) *Au fil des années, des campagnes lancées par Amnesty sont devenues mondiale[s]* [...]

L'organisation *Amnesty International* fait l'objet d'une entrée du lexique. Cependant, dans la phrase (45), l'apprenant a utilisé l'abréviation courante du nom de cette organisation. *Fips* considère cependant que *Amnesty* et *sont* forment un unique mot inconnu.

Pour conclure cette partie, remarquons que les non-erreurs sont courantes dans le domaine de la correction orthographique. Mais pour notre applications, il est important de s'attacher à réduire les fausses détections évitables, qui peuvent déstabiliser les apprenants.

6.4.2.2.2 Erreurs corrigées manuellement Les erreurs détectées mais corrigées manuellement forment plus de 20% des cas. Voici les différents cas de figure, par ordre de fréquence décroissant :

– erreurs (parfois multiples) rendant impossible la correction automatique. Nous en avons relevé 39 occurrences. Dans certains cas, le sys-

tème fournit des propositions, dans d'autres pas. En voici quelques exemples :

  - **fauxsaniers* pour *faux sauniers* : ici, l'erreur d'insertion d'espace est doublée d'une erreur d'omission du *u* ;
  - **définait* pour *a défini* : ici, une erreur de morphologie pour l'imparfait a été doublée d'une erreur de temps ; autres exemple : **soyent* pour *soient* ou **Cettes* pour *Ces* ;
  - **comfortable* pour *confortable* : ici, l'apprenant a fait un emprunt à l'anglais, la phonétisation a échoué et la lettre *n* n'est pas présente dans l'alpha-code du mot erroné. On peut citer aussi les emprunts suivants à la langue de l'apprenant, qui sont trop éloignés de la correction ;
  - première(s) lettre(s) absente(s) : **poque* pour *époque*, **vaient* pour *avaient*, **uvre* pour *œuvre* ou *usqu* pour *jusqu'*. La règle qui élimine les propositions dont la première lettre n'est pas identique à celle du mot erroné devrait être assouplie ;
  - autres exemples pêle-mêle : **rendable* pour *rentable*, **pime* pour *première*, **mosseau* pour *morceau*, **déçai* pour *d'essai*, **jétaît* pour *j'étais*, **chalaourouse* pour *chaleureux* et , **di* pour *de*.

- erreur détectée et corrigée, mais une autre correction doit être introduite en même temps (33 occurrences) : **faime* donne *faim*, **éspere* donne *espère que*, **veniras* donne *viennes*, etc.
- erreur classique doublée d'une erreur lexicale (25 occurrences) : **préténdée* donne quelques propositions dérivées du verbe *prétendre* mais doit être corrigée en *faisaient semblant* [37], **ecoute* devient *apprendre*, *excitee* est corrigée en *je me réjouis* pour éviter une rupture de construction entre deux adjectifs (*Je suis très *heurese et *excitee*), etc.
- emprunts à une langue étrangère (22 occurrences) : *trade* pour *commerce*, **cuarante* pour *quarante*, **emergences* pour *urgences*, etc.
- mot inconnu du lexique, doublé d'erreurs (17 occurrences) : **Téophile* pour *Théophile*, **aquillon* pour *aquilon*, *colluseum* pour *Collisée*, etc.
- mot inconnu correct ou partiellement correct, et devant être corrigé (12 occurrences) : *Inca inca*, *Louvain la Neuve* en *Louvain-la-Neuve*, etc.
- erreur du correcteur avec correction d'une partie (7 occurrences) : *Amnesty entreprennent* pour *Amnesty entreprend*, **est-ce pa*, pour *n'est-ce pas?*, etc.
- erreur de ponctuation ou autre signe (6 cas) : **chez-moi*, **weekend-là* pour *week-end-là*, etc.

37. Ici, l'apprenant a voulu utiliser le verbe *prétendre* en pensant au sens du verbe anglais *to pretend*, qui est un faux ami.

- mot superflu (4 cas) : *vous* dans *Voyez-vous bientôt* corrigé en *A bientôt*, etc.
- erreur de *Fips* : *Ecris-moi* n'est pas accepté ;
- une erreur s'ajoute à une fausse détection : une erreur de casse est faussement détectée pour *ou*, mais il y a une erreur de confusion lexical avec l'adverbe *où*.

Pour améliorer un peu ces résultats, il faudrait améliorer la couverture des erreurs phonétiques et morphologiques, ainsi de certains comportements de l'analyseur lexical de *Fips* et de nos algorithmes de sélection des propositions. Par contre, d'autres erreurs peuvent difficilement être corrigées, comme les erreurs lexicales.

6.4.2.2.3 Erreurs non détectées Dans la plupart des cas, les erreurs non détectées sont des erreurs indétectables par un correcteur orthographique, à savoir des erreurs d'accord, de ponctuation, morphologiques ou lexicales. Nous pouvons aussi citer les mots manquants ou superflus, qui sont en quelque sorte des erreurs lexicales. Considérons la phrase suivante :

(46) a. *Ils sont trouvent des baleines aux Terres-Neuves.

b. ?Il se trouve des baleines à Terre-Neuve.

	*Ils	sont	trouvent	des baleines	aux	Terres-Neuves.
c.	LEX	SUP MOD	INS AGR		AGR	INS AGR LEX
	On		trouve	des baleines	à	Terre-Neuve.

La phrase (46a) ne contient qu'un seul mot inconnu, **Terres-Neuves*, qui est corrigé correctement car le mot est dans le lexique [38]. L'apprenant voulait probablement utiliser la locution *il se trouve*, dont nous n'avons trouvé qu'une attestation dans la locution *il se trouve que*, mais pas dans un sens impersonnel comme en (46b). Ainsi la version corrigée en (46c) est nettement préférable. Sur sept mots, cinq sont balisés comme incorrects. Avec cet exemple, nous illustrons la complexité de la tâche ; nous montrons également qu'un traitement des mots inconnus uniquement aurait été insuffisant pour notre corpus, où de nombreuses erreurs n'auraient pas été corrigées. Un autre exemple ne contient que des mots connus :

(47) a. Après la cérémonie, Antoine et Amélie *a *pris une photo avec tous les invités. C'était la première image *mariée.

38. Le substantif *terre-neuve*, désignant une race de chien, est considéré comme invariable par les dictionnaires que nous avons consultés.

b. Après la cérémonie, Antoine et Amélie ont posé pour une photo avec tous les invités. C'était la première image d'eux en tant que mariés.

La phrase (47a) contient des erreurs d'accord et des erreurs lexicales, indétectables par un correcteur orthographique, comme le montre la proposition de correction (47b). Parfois, des erreurs orthographiques peuvent être indétectables comme **c'est à dire* [39]. Dans d'autres phrases, pour **haut de gamme* ou **de haute gamme*, nous devons compter deux erreurs de séparation pour la correction *haut-de-gamme*. Le même cas s'applique pour la correction de **Para Militaires* en *paramilitaires*. Nous reviendrons sur quelques autres exemples au cours de la section suivante.

6.4.2.3 Types d'erreurs

Pour terminer notre examen systématique du corpus, nous examinons les résultats sous l'angle des différents types d'erreur. Le tableau (6.13) p. 235 récapitule les résultats par type d'erreur.

6.4.2.3.1 Erreurs typographiques Les fautes classiques d'insertion, d'omission, de substitution et d'inversion sont relativement fréquentes et concernent globalement 390 fautes, soit 15,8%.

Les erreurs d'insertion et d'omission sont les plus fréquentes. Elles sont le plus souvent corrigées automatiquement par notre système :

- confusion entre consonne double et simple. Citons les erreurs **méditerrannéens*, **faissaient* ou **sussurrés* pour l'insertion et **fourures*, **languissament* et **masacre* pour l'omission.
- omission d'une voyelle, comme dans **heurese* ou **coleur* ou en insère comme dans **alizée* ou **amitiée*.
- insertion ou omission de lettres déjà présentes dans le mot, comme *p* dans **photopgraphie* ou *n* dans **recontrerai*.

Remarquons aussi que les erreurs d'insertion et d'omission détectées automatiquement impliquent toujours une des méthodes d'alpha-code, accom-

39. Nous avons dû marquer des erreurs sur quatre mots pour cette unique erreur. Il fallait en effet effacer trois mots et noter la correction *c'est-à-dire* pour le mot qui reste.

Nom	Nb err	1 type /err	> 1 err	Auto	Expert
Erreurs typographiques: 390 (15,8%)					
INS	121 (4,9%)	29 (23,97%)	92 (76,03%)	90 (74,38%)	31 (25,62%)
OMI	175 (7,09%)	48 (27,43%)	127 (72,57%)	130 (74,29%)	45 (25,71%)
SUB	83 (3,36%)	4 (4,82%)	79 (95,18%)	69 (83,13%)	14 (16,87%)
INV	11 (0,45%)	2 (18,18%)	9 (81,82%)	10 (90,91%)	1 (9,09%)
Erreurs lexicales: 916 (37,12%)					
LEX	549 (22,24%)	285 (51,91%)	264 (48,09%)	66 (12,02%)	483 (87,98%)
NPR	165 (6,69%)	142 (86,06%)	23 (13,94%)	161 (97,58%)	4 (2,42%)
INC	60 (2,43%)	29 (48,33%)	31 (51,67%)	59 (98,33%)	1 (1.67%)
EMP	142 (5,75%)	18 (12,68%)	124 (87,32%)	95 (66,9%)	47 (33,1%)
Erreurs phonétiques: 643 (26,05%)					
PHG	87 (3,53%)	5 (5,75%)	82 (94,25%)	85 (97,7%)	2 (2,3%)
PHO	199 (8,06%)	10 (5,03%)	189 (94,97%)	133 (66,83%)	66 (33,17%)
HPO	45 (1,82%)	3 (6,67%)	42 (93,33%)	0 (0%)	45 (100%)
LNF	6 (0,24%)	0 (0%)	6 (100%)	6 (100%)	0 (0%)
DIA	306 (12,4%)	140 (45,75%)	166 (54,25%)	249 (81,37%)	57 (18,63%)
MOR	89 (3,61%)	25 (28,09%)	64 (71,91%)	32 (35,96%)	57 (64,04%)
AGR	454 (18,4%)	319 (70,26%)	135 (29,74%)	51 (11,23%)	403 (88,77%)
CPL	161 (6,52%)	96 (59,63%)	65 (40,37%)	2 (1,24%)	159 (98,76%)
Erreurs verbales: 144 (5,83%)					
AUX	19 (0,77%)	14 (73,68%)	5 (26,32%)	0 (0%)	19 (100%)
TPS	80 (3,24%)	42 (52,5%)	38 (47,5%)	4 (5%)	76 (95%)
MOD	45 (1,82%)	16 (35,56%)	29 (64,44%)	3 (6,67%)	42 (93,33%)
Erreurs de mots: 338 (13,7%)					
SUP	189 (7,66%)	157 (83,07%)	32 (16,93%)	8 (4,23%)	181 (95,77%)
MAN	149 (6,04%)	78 (52,35%)	71 (47,65%)	16 (10,74%)	133 (89,26%)
Erreurs de signe: 251 (10,17%)					
SEP	87 (3,53%)	56 (64,37%)	31 (35,63%)	23 (26,44%)	64 (73,56%)
SPC	22 (0,89%)	10 (45,45%)	12 (54,55%)	12 (54,55%)	10 (45,45%)
CAS	76 (3,08%)	41 (53,95%)	35 (46,05%)	22 (28,95%)	54 (71,05%)
PNC	66 (2,67%)	48 (72,73%)	18 (27,27%)	10 (15,15%)	56 (84,85%)
ORD	44 (1,78%)	26 (59,09%)	18 (40,91%)	4 (9,09%)	40 (90,91%)
BRU	29 (1,18%)	24 (82,76%)	5 (17,24%)	29 (100%)	0 (0%)

La troisième colonne compte les mots où un seul type d'erreur est marqué. La quatrième compte les mots marqués par plusieurs types d'erreur. La cinquième marque les erreurs détectées automatiquement et la dernière les erreurs détectées par l'expert.

TAB. 6.13 – *Corpus: résultat par type d'erreur*

pagnée parfois de la méthode phonétique. Le plus souvent, le système retrouve la proposition correcte, mais parfois, la solution doit être corrigée manuellement :

- si le mot correct n'est pas dans le lexique, comme **mastif* pour *mastiff* ;
- si d'autres erreurs interviennent, par exemple une erreur d'accord pour **certainq* corrigé par *certaines*, d'espacement avec **fauxsaniers* pour *faux sauniers*, ou lexicale avec **despuis* corrigé par *il y a*.

Quant aux erreurs détectées par l'expert, il s'agit de mots existants comme *trouvent* pour *trouve*, *est* pour *et*, *irions* pour *irons*, etc.

Par ailleurs, les erreurs d'inversion sont naturellement corrélées à la méthode d'alpha-code simple, car l'inversion de lettres renvoie le même alpha-code. Citons les erreurs **receuille*, **souvenis*, **intrevenir* et **parcitipent*, où les syllabes sont inversées. L'erreur **receuillis* est corrigée en *recueilli*, avec une erreur d'inversion et d'insertion dans le même mot, qui sont également considérées respectivement comme une erreur phonétique et une erreur d'accord.

Enfin, on remarque deux causes principales pour les erreurs de substitution :

- confusion graphème-phonème, comme dans **function*, **cependent* ;
- erreurs morphologiques comme **lieus* ou **loix*.

Là encore, les méthodes alpha-code et phonétique interviennent dans la plupart des cas. Pour terminer, signalons que les erreurs de substitution non détectées concernent, comme dans les autres cas, des erreurs débouchant sur des mots connus comme *pas de tout* pour *pas du tout*, *sons* pour *sont*, *tous* pour *tout* ou *suivent* pour *suivant*.

6.4.2.3.2 Erreurs lexicales Les erreurs lexicales, de mots inconnus, de nom propres et d'emprunts sont de loin les plus nombreuses car elles représentent 916 erreurs, soit 37,12%. Commençons par les erreurs lexicales. La plupart d'entre elles ne sont pas détectées automatiquement. Lorsqu'une erreur lexicale est signalée par le système, il s'agit de mots mal orthographiés doublés d'un mauvais choix lexical. Considérons les extraits suivants :

(48) a. *Ils préténdée que la chaise étaient un train de voyageurs [...].
→ Ils faisaient semblant que la chaise était un train de voyageurs.

b. *J'etais très jolie à ecoute que tu vas venir en Australie!
→ J'étais très heureuse d'apprendre que tu vas venir en Australie!

En (48a), l'apprenant a écrit **préténdée* par erreur pour *prétendaient*. Cependant, il a commis une erreur lexicale en utilisant le verbe *prétendre* à la place de son faux ami *to pretend*. Dans le même cas, citons **prevenir* pour *empêcher* par confusion lexicale avec *to prevent* ou **emergences* pour *urgence*. En (48b), l'erreur **ecoute* pour *écoute* est finalement corrigée en *apprendre* ; signalons aussi l'erreur lexicale non signalée de *jolie* corrigée en *heureuse*, qui entraîne une correction du complément *à* en *de*.

Dans d'autres cas, l'apprenant insère des mots de sa langue comme *much more* pour *bien d'autres choses*. Enfin, citons l'erreur lexicale *heurs* corrigée en *heures* [40].

Pour les erreurs non détectées, nous constatons également de nombreuses erreurs de choix lexicaux comme *débarcadère* à la place de *débarquement*, *tout* pour *tous*, etc. Mentionnons pour terminer l'erreur d'usage de *chambre* au lieu de *Chambre*, lorsque le mot est utilisé pour un organe parlementaire, en l'occurrence la Chambre des représentants de Belgique.

Passons maintenant aux erreurs de noms propres. La grande majorité de celles-ci est détectée automatiquement. Il s'agit avant tout de lacunes du lexique, comme *Samuel*, *Inca*, *Plymouth*, etc. Parfois cependant, ces noms propres contiennent une véritable erreur d'orthographe, comme **Terres-Neuves*, **trevi* pour *Trevi* ou **farenheit* pour *Fahrenheit*. Nous avons également compté comme nom propre les adjectifs ethniques comme *dravidien*, ou des sigles techniques comme *XML*, *DTD*, etc., qui ne sont pas dans le lexique. Quant aux erreurs non détectées automatiquement, citons *George*, incorrectement orthographié pour *Georges Clémenceau*.

Examinons maintenant les mots inconnus. Il s'agit essentiellement de mots absents du lexique comme *ferry*, *acadiennes* ou *caetera*, mais aussi des emprunts comme *vegemeats*, *patty* etc, que l'apprenant substitue aux mots français qu'il ne connaît pas [41]. De plus, nous avons compté comme mots inconnus les chiffres romains comme *XVIe* et les abréviations comme *km2* pour km^2. Signalons enfin que la catégorie des mots inconnus est également utilisée pour les véritables erreurs d'orthographe qui n'ont pas pu être corrigées pour cause de lacune lexicale, comme **tuyas* pour *thuyas*.

40. Le mot *heurs* existe dans l'expression *heurs et malheurs*, mais il est absent du lexique
41. Ici, il s'agit de spécialités culinaires jamaïcaines, dont il n'existe probablement pas de traduction française.

Terminons par les erreurs d'emprunts, qui désignent toute erreur due à une contamination de la langue première de l'apprenant. Il peut s'agir d'une substitution complète (**origin*, **trade* pour *commerce*), mais aussi d'une francisation d'un mot comme *historie* (pour *histoire*, incorrectement adapté de l'anglais *history*) ou *actualment* pour *actuellement* (adapté de l'espagnol *actualmente*). Nous avons également déjà commenté l'erreur *préténdra* au paragraphe 6.4.2.2.2. Parfois, l'apprenant cite un titre de film dans sa langue, *The Day After Tomorrow*, dans un exercice de critique de film [42]. Quant aux erreurs non détectées automatiquement, elles sont causées par les mêmes phénomènes, avec *moment*, traduit directement du substantif anglais *while*, alors qu'il faut employer la conjonction avec le sens de *tandis que*.

6.4.2.3.3 Erreurs phonétiques Les erreurs phonétiques, phonogrammatiques, d'homophonie, de diacritiques et de lettres non fonctionnelles représentent 643 erreurs, soit 26,05%. Les erreurs phonétiques et phonogrammatiques sont essentiellement repérées automatiquement, tandis que les erreurs d'homophonie ne sont généralement pas repérées par notre système.

Commençons par les erreurs phonogrammatiques, qui concernent les mauvaises transcription d'une prononciation correcte. Dans de très nombreux cas, les erreurs de ce type portent également sur les diacritiques, comme **aeroport* ou **specialite*. Les techniques en jeu pour trouver la proposition correcte sont les méthodes d'alpha-code ou la méthode phonétique, souvent combinées. Ce phénomène est facilement explicable car les chaînes d'alpha-code sont quasi-identiques et obtiennent un score élevé. Lorsqu'aucune méthode ne trouve de proposition, c'est à cause d'une lacune du lexique, comme dans **antant* pour *antan* ou de lacune du système de phonétisation comme dans **soyent* pour *soient*. Quant aux erreurs non détectées, elles sont combinées avec d'autres erreurs. Ainsi, *joué* est selon nous une erreur pour *jouaient*, mais est doublée d'une erreur de temps et doit être corrigée en *a joué*. L'autre erreur porte sur la confusion entre *s'* et *c'*.

Quant aux lettres non fonctionnelles, qui concernent les lettres qu'on ne prononce pas, elles sont une catégorie marginale qui se confond avec les erreurs phonogrammatiques [43]. Cette catégorie est évidemment détectée à l'aide des méthodes d'alpha-code, parfois secondées par la méthode phonologique. Mentionnons les erreurs **badaus* et **clochars*.

42. Il serait difficile de compter ceci comme une faute, car il est malaisé de traduire un titre de film. Le titre officiel de ce film en français est *Le jour d'après*, ce qui ne correspond pas à la traduction littérale de *surlendemain*.
43. Peu facile à isoler, cette catégorie n'a sans doute pas été utilisée avec toute la rigueur et la cohérence voulue. C'est pourquoi elle pourrait facilement être éliminée de notre typologie.

Les erreurs phonétiques, qui concernent des erreurs de prononciation débouchant sur une mauvaise transcription ou, parfois, sur un mot connu dont la prononciation est différente. Lorsqu'elles sont détectées automatiquement, là aussi, c'est grâce aux méthodes phonétique et d'alpha-codes. On trouve des erreurs comme **chaleureaux*, **cheres*, ou **plussieurs*, qui ont le même alpha-code que la chaîne correcte. La même remarque vaut pour les erreurs d'accord ou de morphologie et même de substitution (**acceuiller*, **evetement*). Parfois, l'erreur provient d'un emprunt à la langue première de l'apprenant, comme **vegetarian*, **function*, **cuarante* ou **intérior*. Lorsque le mot correct n'est pas trouvé, il s'agit souvent d'une lacune du lexique, comme pour **tuyas* (*thuyas*), ou de limites et mauvais fonctionnement de la méthode phonétique. Quant aux erreurs non détectées, il s'agit de mots existants qui sont mal prononcés, comme *conduisant* (*conduisaient*), *suivent* (*suivante*), *peut* (*pour*), *ferras* (*faudra*) ou *paye* (*pays*).

Les erreurs de diacritiques sont souvent combinées avec des erreurs phonogrammatiques ou phonétiques. Tantôt. les diacritique sont absents (**problemes* pour *problèmes*), tantôt superflus (**préstigieux* pour *prestigieux*), inadéquats (**trés* pour *très*) ou une combinaison de ces cas (**éspere* pour *espère*). Le même constat vaut pour les erreurs non détectées qui débouchent sur des mots existants comme *passes* pour *passés*, *dure* pour *duré*, etc.

Les erreurs d'homophonie ou quasi-homophonie sont exclusivement détectées manuellement. Il s'agit de mots dont la prononciation est identique, ou quasi-identique. Ainsi, *tout* est homophone de *tous* dans le cas de **tout les restaurants*. On peut également citer *oubli* pour *oublie* [44], mais aussi les paires minimales *prés / près*, *et / est* ou *été / était*, que nous avons choisi de tolérer comme quasi-homophone plutôt que comme erreur phonétique.

Signalons pour terminer qu'il a souvent été ardu de distinguer entre ces trois catégories. Il est donc probable que le marquage de ces catégories souffre d'incohérences.

6.4.2.3.4 Erreurs morphologiques Les erreurs morphologiques sont relativement fréquentes dans notre corpus (89 erreurs, soit 3,61%). Seul un tiers environ de ces erreurs sont détectées automatiquement. Elles dénotent une maîtrise insuffisante des règles morphologiques de la langue.

Commençons notre examen par les erreurs détectées automatiquement. La méthode *ad hoc* n'intervient que dans cinq cas, que nous avons déjà

44. Ces deux mots se prononcent de la même manière en français standard, mais pas en Suisse romande, avec [ublɪjə]. Même remarque pour *montée* ([mõtej]) pour *monter*.

analysés au paragraphe 6.4.2.1.5. Dans les autres cas, on trouve les méthode d'alpha-code, parfois en association avec la méthode phonétique. Il s'agit d'erreurs suffisamment proches du mot correct, comme **tristent*, **onts* ou **acceuiller*. Enfin, dans cinq cas, aucune méthode ne propose de solution correcte : **cettes* (à deux reprises), **soyent*, **protecter* et **réligionale*.

Les erreurs repérées par l'expert portent sur l'emploi erroné d'un mot connu. On peut les séparer en deux catégories :

- erreurs vraiment morphologiques, qui débouchent sur un mot existant : *trouves* pour *trouvent*, *fonds* pour *fondé*, *tues* pour *tuer*, *demande* pour *demander*, etc.
- erreurs phonétiques, basées sur un son approchant : **fréquente* pour *fréquentée*, *utilisée* pour *utilisaient*, *suivent* pour *suivante*, *étaient* pour *été*, etc.

Ces dernières erreurs sont classées tout de même comme morphologiques, car elles sont basées sur une forme incorrecte d'un même mot, notamment d'un verbe, ou de la même racine.

6.4.2.3.5 Erreurs d'accord Les erreurs d'accord sont très présentes dans notre corpus avec 454 erreurs, soit 18,4%. Moins de 11% de ces erreurs sont signalées par le correcteur, car elles s'ajoutent à d'autres erreurs. Ainsi, **Terres-Neuves* est compté comme erreur d'accord pour le mot invariable *Terre-Neuve* et *Inca* en tant qu'adjectif comme erreur pour *incas*, **educationnelle* pour *éducatif*, **réfuse* pour *refusent*, **Ja* pour *J'ai*, etc.

Le gros des erreurs sont repérées manuellement sur les verbes (*trouvent* pour *trouve*, *fréquente* pour *fréquentée*), les prépositions (*aux* pour *à*, *du* pour *de*) les adjectifs (*hollandaise* pour *hollandais*), les déterminants (*la* pour *le*), des noms (*cloche* pour *cloches*). Parfois, l'apprenant combine une erreur d'accord avec d'autres erreurs, comme une erreur lexicale (*sont* pour *dure*), morphologique (*serez* pour *serait*), d'homophonie (*cette* pour *cet*), d'élément manquant (*priés* pour *avons prié*), etc.

Notons encore pour terminer que les erreurs d'insertion ou d'omission du *e* ou du *s* débouchant sur une erreur d'accord ne sont la plupart du temps balisées que comme une erreur d'accord. Ainsi, par exemple, *femme* pour *femmes* et *écrites* pour *écrits* n'ont qu'une seule marque d'erreur, ce qui facilite le travail de l'expert.

6.4.2.3.6 Erreurs de complémentation Les erreurs de complémentation représentent 161 erreurs, soit 6,52%. Deux cas sont détectés automatiquement car une deuxième erreur a causé un mot inconnu :

- l'absence de la conjonction *que* s'ajoute à l'erreur de diacritique **éspere* pour *espère* ;
- **Autravers de plusieurs manifestations* est corrigé par *à travers plusieurs manifestations.*

Quant aux erreurs marquées manuellement, elles sont de plusieurs types :

- complément du nom ou du groupe nominal : *le contact *les pêcheurs* pour *le contact des pêcheurs*, *le shopping à Sidney n'est pas le même *à Paris* pour *le même qu'à Paris* ;
- complément de verbe ou locution verbale : **La function de Québec à cette poque est le débarcadère* corrigé par *la fonction de Québec à cette époque est d'être le débarcadère* [45], **nés à l'Angleterre* pour *nés en Angleterre* ;
- complément d'adjectif : *pertinents *aux régions* au lieu de *pertinent pour le régions* ;
- complément d'adverbe : *beaucoup *des* pour *beaucoup de* ;
- complément de présentatif : *il y a *autre *lieus* pour *il y a d'autres lieux* ;
- complément de locution conjonctives : *jusqu'*a tu *arriveras* pour *jusqu'à ce que tu arrives.*

Parfois, nous avons étiqueté des erreurs complexes comme erreur de complémentation, comme la deuxième phrase dans l'exemple (47) p. 233, où le simple adjectif *mariée* doit être corrigé par une périphrase, ou encore les phrases suivantes :

(49) a. Antoine et Amélie *sont dansé *par la musique des violons.
→ Antoine et Amélie ont dansé au son de la musique des violons.

b. La clientèle *dans le restaurant *est *le bourgeoises...
→ La clientèle du restaurant est constituée par des bourgeois

45. Ici, l'erreur est probablement causée par une tournure empruntée à l'anglais, où, d'après nous, une tournure comme *Quebec's function is a wharf* serait acceptable.

Dans la phrase (49a), la préposition *par* est corrigé par *au son de*. Pour la phrase (49b), nous avons estimé qu'il y a une erreur lexicale sur le verbe *est*, accompagnée d'une erreur de complémentation entraînée par la correction de ce verbe [46]. Enfin des ajouts ou compléments non obligatoires peuvent aussi connaître des erreurs de complémentation :

(50) a. *Mais cette proposition a connu des obstacles *des* plusieurs partis.
→ Mais cette proposition a connu des obstacles *de la part de* plusieurs partis.
b. ... augmentation des accidents autoroutes
→ augmentation des accidents sur les autoroutes.

Dans la phrase (50a), l'ajout *des plusieurs partis* n'est pas obligatoire. Toutefois, la prépositions *des* n'est pas adéquate, d'une part parce qu'elle n'est pas correcte devant *beaucoup* (erreur d'accord), et d'autre part du point de vue sémantique pour exprimer la provenance pour *obstacles*. Quant à l'extrait (50b), on peut imaginer un oubli de l'apprenant, ou éventuellement une erreur lexicale pour l'adjectif *autoroutier*.

Pour conclure, cette catégorie couvre de nombreux cas et nécessiterait d'être affinée en plusieurs catégories si le corpus était orienté vers les erreurs syntaxiques. Toutefois, comme nous avons choisi de ne pas restreindre notre examen du corpus aux simples erreurs d'orthographe, les erreurs de complémentation s'avèrent particulièrement nombreuses, ce qui confirme les résultats du corpus FRIDA (*French Interlanguage Database*, §4.1, Granger, 2003) dont nous avons repris certaines données.

6.4.2.3.7 Erreurs verbales Les erreurs verbales (auxiliaire, temps et mode) représentent 144 erreurs, soit 5,83%. Ces erreurs sont en majorité détectées manuellement. Les erreurs d'auxiliaire marquent la confusion entre les auxiliaires *être* et *avoir*, comme dans *sont dansé* ou *il y a été* pour *c'était*.

Quant aux erreurs de temps, elles sont un peu plus fréquentes. On y trouve fréquemment des erreurs sur les participes passés, comme dans *commence* pour *commencé* ou *visite* pour *visité*, qui sont également considérées comme des erreurs de diacritique. Dans le cas de *dansé* pour *ont dansé*, nous avons renoncé à l'explication d'une erreur phonétique pour l'imparfait *dansaient*. Au demeurant, le choix du passé composé est plus adéquat pour une narration d'un événement ponctuel. Quant aux erreurs détectées automatiquement, elles s'expliquent par un mot inconnu dont la correction la plus

46. Remarquons aussi l'erreur de complément du nom *clientèle*.

probable déboucherait sur une erreur de temps : ainsi, l'erreur **définait* peut être corrigée en *définit* ou éventuellement *définissait* [47], mais les temps de ces propositions ne sont pas adéquats dans ce contexte, où le passé composé est meilleur. La phrase suivante illustre un autre cas d'erreur de temps :

(51) a. *La salle de bain tu *à partoger* avec moi.
b. La salle de bain, tu la partageras avec moi.
c. La salle de bain est à partager avec moi.
d. La salle de bain, tu vas la partager avec moi.
...

La phrase (51a) est difficile à corriger. Le contexte est une lettre d'un étudiant australien à son correspondant francophone qui vient lui rendre visite. Dès lors, la phrase peut être corrigée par exemple par les propositions (51b), (51c) ou (51d). Notre choix portera sur (51b) : nous corrigeons dans un premier temps **partoger* par *partager*, mais vu le contexte, nous devons ajouter une erreur de temps.

Enfin, les erreurs de mode, qui concernent plusieurs domaines :

- verbes pronominaux ou constructions pronominales : **ils sont trouvent* pour *il se trouve*, *occupe* pour *s'occupe* ou *se concerne* pour *concerne* ;
- confusion entre subjonctif, indicatif, infinitif, conditionnel ou impératif : *soutienne* pour *soutient*, *me dire* pour *dis-moi*, *irions* pour *irons* ou *deviendrait* pour *devienne*.

Quant aux erreurs détectées automatiquement, elles comprennent d'autres erreurs qui débouchent sur un mot inconnu. Ainsi, l'erreur morphologique **veniras* débouche sur *viendras*, qui est inadéquat après la conjonction *que* qui demande le subjonctif. **Recomendera* est corrigé en *recommenderai* grâce à la méthode d'alpha-code élargi, mais dans le contexte, le conditionnel *recommanderais* est meilleur. Enfin, considérons l'exemple suivant :

(52) Je crois qu'il faut en parler, s'exprimer, discuter pour arriver à un accord qui *plait le *mieulleur possible à tous les participants.
→ Je crois qu'il faut en parler, s'exprimer, discuter pour arriver à un accord qui plaise le mieux possible à tous les participants.

47. Cette correction n'est pas retenue par le correcteur car elle est trop éloignée de la chaîne originale.

Ici, *plait* devrait être corrigé en *plaît*, mais dans le contexte, le subjonctif *plaise* apporte une nuance nécessaire au sens de la phrase.

Signalons pour terminer que parfois, il arrive que les erreurs verbales portent sur plusieurs types à la fois. Ainsi le verbe *soit* doit être corrigé en *a été* et porte donc les marques d'erreur de mode et de temps.

6.4.2.3.8 Erreurs de mots Les erreurs de mots manquants ou superflus représentent 338 erreurs, soit 13,7%. Là encore, ces erreurs ne peuvent pas être détectées automatiquement, et quelques erreurs détectées automatiquement comprennent en plus des mots superflus ou manquants.

Commençons par les mots manquants. Toutes les catégories lexicales peuvent manquer, comme un verbe ([*a*] *joué*, *se* [*trouve*]), une préposition ([*de*] *tabac*), un déterminant ([*les*] *attaques* , [*une*] *grande*), un pronom ([*t'*]*apprendre*, *est*[*-ce*]), une particule négative (*je ne me souviens* [*pas*]), etc.

Dans certains cas, les corrections sont plus complexes :

(53) a. *Les problèmes des passagers sont doivent se contenter de nourriture froide et de couchages détrempés.
→ Les problèmes des passagers sont [qu'ils] doivent se contenter de nourriture froide et de couchages détrempés.
b. *Là-bas magnifique surtout le soir.
→ Là-bas [l'ambiance est] magnifique surtout le soir.
c. *une au pair
→ une [jeune fille] au pair.

Quant aux mots superflus, le principe est similaire :

(54) a. *L'origin du commerce des peaux et des fourures est le contact les pêcheurs entrent avec les Amérindiens.
→ L'*origine* du commerce des peaux et des fourrures est le contact *des* pêcheurs ~~entre~~ avec les Amérindiens.
b. après 3 décennies ~~de plus~~

Pour la phrase (54a), nous avons interprété le verbe *entrent* comme une erreur d'homophonie et lexicale à corriger par la préposition *entre*, qui devient superflue dans ce contexte. Dans le cas (54b), *de plus* est redondant.

Par contre, nous avons beaucoup utilisé cette catégorie pour des corrections complexes. Dans le cas de **Para Militaires*, on compte une faute d'espace et un mot superflu pour *paramilitaires*. Les mots *de haute gamme* comptent comme une erreur lexicale pour *haut-de-gamme*, où *haute* porte une erreur lexicale, et les deux autres mots comme des mots superflus. Dans le cas de *d'améliorer moi-même*, nous corrigeons en *de m'améliorer* et comptons une erreur de mode et de mot manquant sur *améliorer* et supprimons *moi-même* en le considérant comme superflu. Parfois une correction complexe est marquée à la fois par des mots manquants et superflus.

(55) a. Absolument a voir.
→ ~~Absolument~~ *À* voir [absolument].
b. hier semaine
→ ~~hier~~ [la] semaine [passée]
c. L'entrée a été soupe de poulet
→ L'entrée *était* ~~été~~ [une] soupe de poulet.

Pour conclure, on peut remarquer qu'à cause du phénomène des corrections complexes, le score de la catégorie des mots superflus est un peu gonflé et devrait être considéré à part.

6.4.2.3.9 Erreurs de signe Les erreurs de casse, de ponctuation, de séparation et d'espacement représentent 251 erreurs, soit 10,17%. Dans près de 30% des cas, les erreurs de casse sont détectées automatiquement. La méthode d'ajout de majuscules intervient cinq fois, comme nous avons déjà eu l'occasion de le signaler au paragraphe 6.4.2.1.8. Dans cinq cas, on trouve la méthode d'alpha-code : **t-shirt* est corrigé en *T-shirts*, **pyrennes Pyrénées*, **Regionales* en *régionales*, **amerique* en *Amérique*. Pour *President*, la méthode d'alpha-code trouve *Présidentes*, mais nous devons corriger manuellement en *présidentes*. Dans la même veine, pour **jamaïquain*, la méthode phonétique trouve *jamaïcain*, mais nous devons corriger par *Jamaïque*. Pour la méthode alpha-code élargie, on trouve *Thème* pour **Tem*, qui doit être corrigé en *thème*. Enfin, la méthode alpha-code restreinte trouve *Maroc* pour **maroce*.

Parmi les mots inconnus sans proposition correcte dans la catégorie des erreurs de casse, nous trouvons des mots existants sans entrée dans le lexique comme *Inca* pour *inca*, ou des mots mal orthographiés qui comportent en outre une erreur de casse comme **China* pour *chinoise* ou **farenheit* pour *Fahrenheit*. Quant aux cas non détectés, on peut distinguer plusieurs cas. Tantôt une majuscule de début de phrase n'a pas pu être détectée à cause

d'une analyse partielle de la phrase et/ou d'autres erreurs, comme dans **grande* pour *Une grande* ou **mais* pour *Mais*. Dans d'autres cas, il s'agit d'une confusion entre un adjectif et un nom ethnique, comme **indien* pour *Indien*, *espagnole* pour *Espagnole* ou entre le substantif ethnique et le substantif désignant une langue, comme *Anglais* pour *anglais* et *Français* pour *français*. Parfois aussi, des majuscules sont insérées dans des titres, ce qui ne correspond pas à l'usage de la langue française. Signalons encore des erreurs dans des noms propres ou des noms d'institutions, comme **père Noël* pour *Père Noël*, **mosquée bleue* pour la *Mosquée Bleue* d'Istamboul, ou les variantes d'erreurs autour des Chambres du Parlement belge, la *Chambre des Représentants* [48] et le *Sénat*.

Les erreurs de ponctuation sont des erreurs d'emploi de la virgule et des points d'interrogation et d'exclamation, mais aussi des points, deux-points etc. Signalons également les difficultés pour l'emploi des abréviations comme **S.A* pour *SA*.

Nous avons déjà vu les résultats de la méthode d'espacement au paragraphe 6.4.2.1.7. Signalons que cette méthode ne permet pas de détecter **fauxsaniers*, où un *u* est omis en même temps que l'erreur d'espacement. Dans d'autres cas, l'erreur est détectée par d'autres méthodes, comme **Apart* pour *à part* ou **parce-que* pour *parce que* (méthodes alpha-code et phonétique). Enfin, quelques espaces manquants après la ponctuation de fin de phrase sont détectés, mais aucune correction n'est proposée. Quant aux erreurs marquées manuellement, il s'agit d'espaces incorrectement insérés et donnant deux mots valides séparément, comme dans **Il s* pour *Ils* ou **tout jour* pour *toujours*. Parfois également, le segmenteur de phrase est mis en difficulté par l'absence d'espace, par exemple pour le cas de "*soir.Cette*" où un seul mot est compté.

Enfin, plus d'un quart des erreurs de séparation sont détectées automatiquement. En premier lieu, ces erreurs sont détectées par la méthode d'alpha-code, comme **parce-que*, **bien-tôt* ou **weekend*. De même, nous avons déjà eu l'occasion d'analyser les résultats de la méthode d'apostrophe manquante au paragraphe 6.4.2.1.6. Par contre, d'autres erreurs empêchent de trouver la proposition correcte dans des cas comme **déçai* pour *d'essai* ou **jétaît* pour *j'étais*. Remarquons pur ce cas qu'il est difficile de faire des propositions si deux mots sont collés, de même que pour l'erreur **chez-moi*.

En ce qui concerne les erreurs de séparation non détectées, signalons l'abondance d'erreurs d'élision comme *je* pour *j'*, *me* pour *m'*, etc. En second lieu, on remarquera l'abondance d'erreurs de séparation par le tiret, comme

48. Cette institution est parfois aussi nommée simplement la *Chambre*.

**peut être* pour *peut-être* ou **mille neuf cent quatre-vingt quatorze* pour *mille neuf cent quatre-vingt-quatorze.*

6.4.2.3.10 Erreurs d'ordre des mots Les erreurs d'ordre des mots représentent 44 erreurs, soit 1,78%. Comme il s'agit d'une erreur de syntaxe, nous la repérons essentiellement manuellement. Comme dans le corpus FRIDA (§4, Granger, 2003), nous repérons de nombreuses erreurs d'ordre des adverbes (*que *vraiement elle mérite*) et des adjectifs (*mes *favories plats*). Mais il y a aussi des erreurs concernant d'autres catégories, comme dans la phrase suivante :

(56) [...] je ne me souvenis, dans lequel avion arrives-*tu?
→ je ne me souviens pas dans quel avion tu arrives.

Nous ne nous attarderons pas plus sur cette catégorie, qui n'intéresse que peu le but de notre corpus, mais que nous avons toutefois jugée utile afin de pouvoir proposer une correction complète de notre corpus.

6.4.2.3.11 Fausses erreurs ou bruit Les fausses erreurs ou bruit représentent 1,18% des erreurs. Comme il s'agit d'un mauvais fonctionnement du correcteur, cette catégorie est repérée exclusivement automatiquement. Nous avons déjà parlé du mauvais fonctionnement du correcteur aux paragraphes 6.4.2.1.8 et 6.4.2.2.1. Nous pouvons également remarquer les erreurs de mise en majuscule d'*Etant*, corrigé en *Étant*, ce qui devrait être une suggestion spécifiquement marquée comme non obligatoire. Enfin, l'erreur **mnombrables* pour *innombrables* a été marquée comme bruit, car nous suspectons un logiciel de reconnaissance de caractères d'avoir introduit cette erreur, car ce type de logiciel voit fréquemment un seul caractère là où il y en a deux.

6.4.3 Remarques finales sur les évaluations du correcteur

Dans cette courte section, nous terminons l'évaluation de notre correcteur par quelques remarques finales. Dans le cadre de la liste de mot, nous nous trouvons devant des erreurs entièrement artificielles, tandis que le corpus est majoritairement constitué de phrases authentiques. La liste de mots ne nous permet de comparer les résultats uniquement sur les principales méthodes, les alpha-codes et la méthode phonologique.

Dans la liste de mot, les mots sont répartis de manières relativement homogène. En effet, des variantes orthographiques ont été introduites dans certains mots pour tester l'efficacité des différentes méthodes, par exemple en doublant certaines consonnes ou au contraire en enlevant certaines consonnes doubles, ou encore en écrivant un mot phonétiquement. Enfin, les mots sans proposition correcte sont dans une proportion bien moindre que pour le corpus.

Par contre, le corpus confirme et accentue encore la suprématie de la méthode alpha-code simple et de la méthode phonétique constatés dans la liste de mots. Cela corrobore les observations mentionnées à la section 3.2.3.1 sur la fréquences des erreurs très proches de la chaîne correcte. En revanche, comme nous traitons un corpus d'apprenants, la fréquence des erreurs phonétiques est probablement plus élevée que dans un corpus constitué d'erreurs de locuteurs natifs. Enfin, les méthodes d'alpha-code restreint et élargi obtiennent une proportion relativement proche dans les deux corpus.

6.5 Discussion

Nous concluons ce chapitre par une discussion sur les performances de notre correcteur et sur les améliorations qui peuvent y être apportées. Nous commençons par une discussion sur les méthodes utilisées et sur l'ordre des propositions (§6.5.1). Nous poursuivons avec une évaluation de l'interface et sur les pistes d'amélioration de celle-ci (§6.5.2). Enfin, nous concluons notre chapitre en discutant de la qualité et de l'utilité de notre corpus (§6.5.3).

6.5.1 Méthodes et ordre des propositions

Parmi les méthodes que nous avons présentées ici, les méthodes d'alpha-code sont les plus efficaces, notamment parce qu'elles sont relativement simples à implémenter et parce qu'elles couvrent des erreurs très fréquentes, comme les inversions, substitution, omissions et insertion. Nous nous bornerons à suggérer de supprimer notre règle qui élimine les propositions dont la première lettre n'est pas la même que celle du mot inconnu. En effet, dans de rares cas, cette restriction a empêché la correction de l'erreur. En outre, la distance lexicographique semble être un filtre efficace pour prévenir les erreurs trop éloignées de la chaîne originale.

Comme nous l'avons signalé, il serait judicieux de revoir l'algorithme de

la méthode phonétique pour autoriser à phonétiser plusieurs propositions à partir de la même chaîne. Dans une première phase, seules des règles valides pourraient être mises en œuvre. Puis, si aucune proposition n'a été trouvée, des règles de substitution de phonèmes et / ou des règles d'erreurs pourraient intervenir.

Une étude approfondie serait nécessaire pour affiner les règles de phonétisation et les règles d'erreur. D'autres règles de substitutions pourraient être suggérées, notamment la substitution des sons [s] et [z]. Par exemple, pour **faissaient*, le son [ɛ] est déduit de la terminaison -*aient*. On peut en déduire que l'apprenant maîtrisait cette terminaison verbale complexe et n'a pas confondu ce phonème avec [e], ce qui exclura la proposition *fessée*.

Il serait sans doute judicieux de mettre en place une distance phonétique, dont nous esquissons ici l'algorithme. Chaque proposition serait évaluée par rapport à la phonétisation du mot erroné [49]. Chaque substitution serait alors plus ou moins pénalisée selon les cas. Au cas où plusieurs phonétisations sont possibles, il suffirait probablement de faire une moyenne des distances. Pour que ce système fonctionne correctement, il faudrait disposer d'un lexique où toutes les entrées sont phonétisées et validées par un expert. Bien évidemment, l'algorithme devrait être évalué par un test approfondi sur un corpus représentatif des erreurs phonétiques.

Enfin, à l'évidence, la méthode *ad hoc* n'est qu'une ébauche. Idéalement, comme nous l'avons déjà souligné, il faudrait un analyseur morphologique capable de décomposer un mot en racines et affixes. A défaut, on pourrait ajouter des règles supplémentaires pour traiter par exemple **devenira*, **voyera*, **notres*, **cettes*, **leures* ou **leure*. Toutefois, un grand nombre de règles pourrait rendre le système ingérable et risquerait de ralentir considérablement le fonctionnement du correcteur.

Passons maintenant aux améliorations de nos algorithmes de classement et de filtrage des propositions. En premier lieu, il serait judicieux de bloquer le nombre de propositions à cinq. En effet, la position moyenne de la proposition correcte est bien au dessous de ce seuil, quelle que soit la méthode utilisée. Cette restriction est souhaitable, du moins dans une première phase [50]. En outre, il serait judicieux de favoriser les propositions trouvées par plusieurs méthodes. En effet, la plupart des propositions ne sont trouvées que par une seule méthode. En revanche, près de trente pourcent des propositions correctes sont trouvées par deux méthodes ou plus, mais l'addition

49. Dans ce cas, bien évidemment, il ne faudrait utiliser aucune substitution ni règle d'erreur.

50. Nous reviendrons sur la correction en plusieurs temps au paragraphe 6.5.2.

des scores de chaque méthode semble parfois insuffisant pour faire ressortir la proposition correcte. De même, les propositions trouvées par certaines méthodes plus rares, *ad hoc* ou autre, pourraient être placées systématiquement dans les premières places. Il nous a manqué du temps et des ressources pour étudier l'adéquation des différents scores.

Pour les distances, à score égal, nous classons les propositions d'après la distance. Or il serait plus judicieux de classer les propositions d'après le pourcentage de cette distance par rapport au seuil : pour **souvenis*, *souviens* vient en septième position à cause d'un score de 10 et d'une distance élevée de 0,125 ; *souviennes* a le même score mais a une distance plus faible, bien que plus proche du seuil $\mathcal{D}\prime$. En outre, la faible pénalité frappant les différences entre consonne unique et doublée devrait peut-être n'intervenir que si cette consonne est entourée par des voyelles, voire même exactement les mêmes voyelles. De plus, l'insertion ou l'omission d'un espace devrait être pénalisée. Enfin, rappelons qu'il pourrait être utile d'utiliser une distance phonétique, y compris pour les propositions trouvées par d'autres méthodes que la méthode phonétique.

Pour les filtres, il serait envisageable de mettre en place des heuristiques. En premier lieu, il serait souhaitable de s'aider davantage de la syntaxe pour éliminer les propositions en fonction des mots environnants : nous avons déjà signalé (§6.4.2.1.2) le problème d'*ingénie*, qui est un verbe pronominal proposé pour **ingénière* ; sans pronom à sa gauche, cette proposition pourrait être éliminée. Par ailleurs, les fréquences d'occurrence des mots devrait être prises en compte pour pénaliser les mots rares, voire les éliminer de la liste des propositions. De plus, il conviendrait de ne pas proposer un mot en majuscule si le mot en minuscule existe et que le mot erroné est en minuscules : *Société* est proposé, à côté de *société* pour **societé*, ce qui est inutile. Enfin, le correcteur ne devrait probablement pas être déclenché pour les mots en une seule lettre.

Relevons à présent les différents problèmes du lexique. Certains non-mots y figurent : **oportune* apparaît au lieu d'*opportune*. Par contre, certains mots y manquent, comme *caetera*, *heurs*, ou les prénoms courants *Georges*, *Michelle*, *Liam* ou *Pete* [51].

Pour éviter le bruit, il serait peut-être judicieux d'enlever les mots trop rares ; nous avons relevé l'absence de *heurs* dans le lexique, qui, dans le cas

51. Plus délicat, le mot *Van*, préposition faisant partie d'un nom propre néerlandais, n'est pas dans le lexique. Pourtant, *van* est un substantif qui figure dans certains dictionnaires mais pas dans notre lexique, ce qui pose la question de la couverture du lexique et de ce qu'on souhaite y trouver.

d'espèce, a permis de relever une erreur lexicale, car ce mot était employé à la place de *heures*. En revanche, le substantif rare *adjective* devrait être banni [52].

Nous devrions aussi améliorer notre système de classement des propositions. En effet, à partir de **Autravers*, la méthode alpha-code trouve *au travers*, dont la première lettre est mise en majuscule. Or l'insertion de majuscules permet aussi de trouver cette proposition, mais notre système ne fusionne pas les propositions, et deux chaînes identiques apparaissent.

Enfin, le découpage des mots par *Fips* pose parfois problème. Ainsi, *eme* attaché à un nombre n'est pas considéré comme une marque d'ordinal. De même, *Etant* est considéré comme une erreur, alors que l'accent est facultatif en début de phrase, si ce n'est interdit. La gestion des parenthèses et de la ponctuation laisse à désirer. Ainsi, dans la chaîne *Le parlementaire P.H. du parti socialiste*, les initiales *P. H.* sont simplement effacées. De même, les signes comme l'étoile (*) posent problème. Enfin, signalons quelques problèmes avec des caractères spéciaux comme le *oe* ligaturé (*œ*) ou des signes typographiques insérés par des traitements de texte.

6.5.2 Interfaces

Le prototype de correcteur et corpus étaient avant tout des projets de recherche en linguistique informatique, dans le but premier d'évaluer des techniques de recherche de propositions. De plus, lors de la phase de test, il s'est rapidement avéré qu'il serait très difficile de trouver des apprenants pour tester le correcteur et récolter le corpus. Il y a donc de grandes possibilités d'améliorations, tant du point de vue technique que pédagogique.

Commençons par l'interface pour les apprenants, qui pourrait être améliorée grâce aux derniers développement des interfaces *web*, qui permettent une plus grande interactivité. Ainsi, l'apprenant pourrait plus ou moins retrouver l'interface familière des correcteurs orthographiques des traitement de texte. Le mot inconnu pourrait être souligné et les propositions apparaîtraient si le curseur de la souris survole le mot.

Le fonctionnement en soi du correcteur pourrait aussi être amélioré et paramétré. Ainsi, un petit nombre de proposition devrait être présenté dans un premier temps, les cinq premières par exemple ; le cas échéant, une icône

52. Dans l'état actuel du correcteur, aucune méthode ne permettrait de trouver *adjectif* à partir de *adjective*, mais une règle morphologique ou *ad hoc* pourrait faire l'affaire, à moins qu'elle ne produise trop de faux positifs.

ou une autre forme d'avertissement devrait signaler à l'apprenant qu'il existe d'autres propositions. De même, le seuil de distance pourrait varier. Dans un premier temps, seuls les mots à distance inférieure ou égale à $\mathcal{D}$ devraient être proposées, à mois que le nombre de propositions soit inférieur à cinq. Ensuite, si l'apprenant le souhaite, les propositions entre $\mathcal{D}$ et $\mathcal{D}\prime$ pourraient être affichées. Par ailleurs, des options courantes dans les correcteurs orthographiques usuels devraient être offertes : un dictionnaire personnel devrait pouvoir être constitué ; si la même erreur apparaît plusieurs fois dans une session de travail ou du moins dans la même phrase, elle devrait être corrigée automatiquement ou semi-automatiquement (après confirmation) ; enfin, les mots tout en majuscules comme *ARTE* devraient pouvoir être ignorés automatiquement.

Pour son auto-correction, l'apprenant devrait avoir à disposition des outils comme des dictionnaires bilingues et monolingues, un conjugueur, etc. Un reconnaisseur de langue (§3.1.1.3) pourrait signaler, au fur et à mesure de la frappe, les mots potentiellement étrangers. L'apprenant devrait aussi pouvoir entendre la prononciation ou les prononciations possibles des mots inconnus. Si l'apprenant ne corrige pas un mot, il devrait pouvoir indiquer s'il connaît le mot, si c'est un emprunt à sa langue pour pallier ses lacunes ou s'il n'a pas pu choisir parmi les propositions. Ainsi, il serait plus aisé de connaitre le degré de confiance de l'apprenant en ses propres capacités.

Pour chaque soumission, un score indicatif pourrait être indiqué, par exemple en fonction de la proportion de mots inconnus par rapport au total des mots. Ensuite, après une correction manuelle par un expert – l'enseignant habituel de l'apprenant, par exemple – la phrase corrigée devrait aussi être affichée avec le diagnostic des erreurs, afin de permettre à l'apprenant d'améliorer sa production.

Pour la gestion des apprenants, un profil des erreurs typiques pourrait être dégagé à l'aide du marquage des erreurs. Il serait également utile de montrer l'évolution des connaissances au cours de l'utilisation du système.

Passons maintenant à l'interface de l'expert. Il faudrait également une interface plus moderne, qui permette une validation et un diagnostic plus rapide. Il devrait être possible d'affiner le diagnostic en groupant plusieurs mots dans une même erreur, voire en déplaçant des mots. Comme pour *Fips* et FreeText (§5.3.1), on devrait pouvoir lier deux mots pour expliquer en quoi le mot est erroné (comme avec la balise <PARTERROR>), particulièrement dans le cas des erreurs d'accord. Mais un cas particulier mérite réflexion :

(57) *J'éspere vous aimez le bifteck parce-que c'est le cuisine prefere d'Aus-

traliens.
→ J'espère que vous aimez le bifteck parce que c'est le plat préféré des Australiens.

Ici, nous avons marqué une erreur d'erreur sur *le* par rapport à *cuisine* ; or ce dernier mot n'est pas approprié (erreur lexicale) et doit être corrigé en *plat*. Il faudrait probablement pouvoir indiquer optionnellement que l'erreur n'est valable qu'en lien avec le mot lié original, et non avec sa correction.

Enfin, signalons que, globalement, l'interface de consultation du corpus est satisfaisante. Il faudrait néanmoins proposer plus d'outils de recherche et de navigation dans le corpus. De plus, les utilisateurs du corpus devraient pouvoir signaler les erreurs ou faire d'autres propositions de correction.

6.5.3 Corpus

Nous n'avons pu récolter qu'un corpus de taille moyenne, qui permet de dégager certaines tendances. Nous nous sommes efforcés de varier les niveaux et les différentes provenance des apprenants et des autres textes, sans pouvoir distinguer l'âge, le niveau et le sexe des apprenants ; bien que nous avons pu noter quelque fois la provenance et le niveau, ces indications ne sont pas suffisamment fiables et précises pour pouvoir être exploitées. Il faudrait considérablement augmenter la taille du corpus, en variant les provenances et les niveaux, afin d'affiner nos observations et d'augmenter les performances des méthodes phonétique et *ad hoc*, en essayant de couvrir un maximum d'erreurs.

Pour les aspects positifs, le corpus nous donne une mesure assez précise des résultats du correcteur :

- taux de rappel des erreurs (erreurs détectables par rapport aux autres erreurs) ;
- efficacité des méthodes de recouvrement des propositions [53] ;
- efficacité de l'algorithme de classement des propositions ;
- erreurs qui devraient être détectées.

53. Et inversement, une étude détaillée d'un corpus devrait permettre d'améliorer les performances de ces méthodes.

Du point de vue pédagogique, le corpus nous permet d'évaluer :

- la manière d'écrire des apprenants ;
- les catégories d'erreurs commises et leur fréquence ;
- la manière d'utiliser les outils d'aide à l'apprentissage ;
- les choix parmi les propositions ;
- si les données étaient suffisantes, l'influence de la langue maternelle, de la langue du pays de provenance, de l'âge, du sexe et du niveau.

Notre corpus répond donc bien aux définitions (§3.1.7) d'un corpus d'apprenant et de son utilisation. Malheureusement, il nous a manqué du temps et des ressources pour analyser et exploiter tout le potentiel de ce corpus, malgré sa taille plutôt moyenne.

Enfin, soulignons qu'il manque une évaluation de notre outil par les apprenants eux-mêmes et par des enseignants. Comme nous l'avons souligné (§2.5), il est utile de connaître l'appréciation subjective des apprenants sur la facilité d'utilisation de l'interface, et même sur le fonctionnement des différents outils, notamment du correcteur lui-même. Le dispositif de test à envisager comprendrait un pré- et post-test afin de mesurer le gain d'apprentissage. Les apprenants ne devraient avoir à disposition que des outils autorisés, à savoir le correcteur lui-même et divers outils d'aides, comme un conjugueur ou un dictionnaire (électronique ou papier). On pourrait aussi séparer les apprenants en divers groupes, dont les aides à l'apprentissage ne seraient pas les mêmes. Enfin, on pourrait faire trois groupes d'apprenants, les uns travaillant seuls, les autres en tandem de niveau de français comparable et les derniers en tandems de niveau divergent. La tâche à réaliser devrait forcer les apprenants à collaborer et les entrées de chacun devraient être différenciées. Mais un tel dispositif dépassait hélas de beaucoup le temps et les ressources dont nous disposions et nous avons dû renoncer à ce test.

Chapitre 7

Outils “sémantiques” pour l’ALAO

La syntaxe s’avère parfois insuffisante pour évaluer les réponses des apprenants. Considérons les exemples suivants :

(58) a. – As-tu vu la voiture rouge?
  i. – Oui je l’ai vu.
  ii. – Oui, je l’ai vue.
 b. – De quelle couleur est le chien?
  i. – Le chien est blanc et noir.
  ii. – Il est noir.
  iii. – Le caniche est noir et blanc.
 c. – Qu’est-ce que Jean a acheté?
  i. – Jean a acheté un perroquet.
  ii. – Jean a acheté un oiseau.
 d. – Qu’est-ce que Paul a fait?
  i. – Il a lu ses livres.
  ii. – Il a lu ces livres.
 e. – Quelle langue est parlée au Québec?
  i. – On parle français au Québec.
  ii. – Le français est parlé au Québec.
  iii. – On parle le français au Québec.
  iv. – Le français se parle au Québec.

Pour la phrase (58a), la réponse (i) est incorrecte dans ce contexte, bien que parfaitement grammaticale ; c’est la réponse (ii) qui est correcte, avec

le participe passé accordé au féminin singulier et le clitique *la* qui est élidé. Il est impossible de détecter l'erreur et d'en poser un diagnostic avec un simple vérificateur grammatical. En (58b), on peut imaginer un exercice de description d'une image d'un chien noir et blanc : la réponse (i) est correcte et la réponse elliptique (ii) peut être jugée correcte ou incomplète. Pour la réponse (58b iii), deux problèmes se posent : tout d'abord, il est nécessaire de disposer d'un lexique qui encode qu'un caniche est une race de de chien, autrement dit un hyponyme [1] ; par contre, si l'apprenant se trompe de race de chien, sa description est fausse, mais aucun outil informatique actuel à notre connaissance ne serait capable d'opérer une telle distinction [2].

En (58c), si (i) est la réponse attendue, (ii) illustre l'utilisation d'un hyperonyme. Il serait judicieux de prévoir un diagnostic particulier pour les hyponymes et les hyperonymes. Pour la phrase (58d), on trouve la confusion fréquente entre *ces* et *ses*, qui touche également les locuteurs natifs. Enfin, en (58e), on trouve différentes variations de voix : si la phrase attendue est (i), (ii) pourra être considérée comme strictement équivalente ; en (iii), on pourra signaler l'insertion d'un déterminant ; enfin, (iv) est plus problématique car il est à peu près équivalent aux autres mais ne répond pas tout à fait à la question ; en effet, cette tournure correspondrait plutôt à la question *Où parle-t-on français?* et à la réponse *Le français se parle au Québec, en France, en Suisse, en Belgique, au Luxembourg etc.*, mais notre système ne permettrait pas de faire cette distinction.

Pour tous ces exemples, la correction intelligente d'un exercice va au-delà de la syntaxe. Dans des cas simples, il serait sans doute envisageable de comparer l'analyse syntaxique de deux réponses et de lister les différences. Avec une analyse riche telle que celle de *Fips* (§5), cette tâche serait possible. Cependant, il peut arriver que des structures soient trop différentes pour être comparées syntaxiquement. Dans le cadre d'exercices à réponse prévisible, il s'agit donc de comparer une réponse de référence à celle de l'apprenant et d'en évaluer la pertinence sémantique, autant que faire se peut. Par ailleurs, même si ce type d'exercice est peu courant, nous avons élaboré quelques réflexions autour d'un outil de manipulation du sens qui permette d'observer l'effet du changement de certains paramètres sur la structure des phrases.

Dans ce chapitre, nous présentons deux outils à composante sémantique développés en marge du projet FreeText (§4), l'un expérimental et l'autre resté à une description théorique. Nous avons repris des structures de représentation sémantique de phrases (§3.4) utilisées pour un système de

1. *WordNet* (§3.5.3) peut être un outil adéquat pour ce type d'application.
2. Par contre, si la réponse attendue par le système concerne une race de chien et que l'apprenant répond avec le nom générique *chien*, le diagnostic est possible.

génération (§3.1.5). Nous commençons par décrire les *Structures Pseudo-Sémantiques*, le formalisme que nous avons utilisé pour élaborer nos outils. (§7.1). Nous continuons par une description de la comparaison de phrases (§7.2). Puis nous poursuivons par une application théorique, la reformulation de phrases (§7.3). Enfin nous terminons par une discussion (§7.4).

7.1 Les structures pseudo-sémantiques (PSS)

Dans cette partie, nous décrivons les structures sémantiques que nous utilisons pour la comparaison sémantique de phrases (§7.2) et la reformulation de phrase (§7.3). Nous expliquons brièvement notre modélisation informatique de ces structures. Les structures pseudo-sémantiques (*Pseudo-Semantic Structures*, PSS, Clark, 1993; Wehrli et Clark, 1995; Etchegoyhen, 1997; Etchegoyhen *et al.*, 1999) sont des structures hybrides qui combinent des informations sémantiques abstraites avec des informations lexicales portant sur les classes lexicales ouvertes (noms, verbes, adverbes et adjectifs). Elles sont nommées *pseudo*-sémantiques du fait qu'elles ne donnent pas une interprétation complète du sens des propositions, en restant à un niveau proche de la syntaxe. Les PSS offrent une couverture étendue de la langue. Conçues pour la génération automatique de phrases (§3.1.5), elles permettent de s'affranchir des structures et des données d'une langue particulière, tout en ne nécessitant pas la création d'un formalisme sémantique complet. Ainsi, pour la traduction automatique, le passage d'une langue à une autre est théoriquement facilité, car il n'est nécessaire de traduire que les éléments lexicaux ; les différences de structures sont ensuite gérées par le processus de génération (Laenzlinger *et al.*, 2000; L'haire *et al.*, 2000).

Extraites à partir d'une analyse produite par *Fips* (§5) grâce à un parcours descendant et récursif de la structure de la phrase, les PSS sont utilisées comme entrées du générateur *GBGen* (Etchegoyhen, 1997; Etchegoyhen et Wehrle, 1998; Etchegoyhen et Wehrli, 1998; Etchegoyhen *et al.*, 1999), qui, comme *Fips*, est basé sur la théorie du Gouvernement & Liage (§3.3.5). Lors du processus de génération, les différents éléments des PSS sont projetés dans une structure profonde par une procédure déterministe ; puis au moyen de règles de mouvement, on déplace les différents éléments dans leur position finale, notamment pour les structures passive ou interrogatives ; enfin un module morphologique procède aux flexions, élisions et contractions pour produire la forme finale. Ces opérations s'avèrent efficaces et d'une complexité computationnelle faible (Etchegoyhen *et al.*, 1999).

Les PSS sont des structures hiérarchiques, qui comportent des descen-

dants contenus dans une liste de *satellites*, qui sont eux-mêmes des PSS. Elles peuvent être de type générique ou appartenir aux sous-types que nous décrivons maintenant.

Tout d'abord, les *structures-DP* (*Determiner Phrase Structures* ou *DP-Structures*, DPS) représentent les groupes nominaux et contiennent bien évidemment les noms comme élément lexical ; dans le cas des clitiques, l'élément lexical est vide et représenté par la valeur Δ. Les éléments abstraits sont les suivants : l'*opérateur* sémantique définit la détermination de l'élément lexical et représente les déterminants (indéfinis, définis, démonstratifs, etc.), les quantifieurs (*tous, chaque, quelques, aucun*, etc.), des valeurs numériques définies (*trois pommes*) ou des ordres de grandeur (*au plus n, moins de n, à peu près, entre n et m*, etc.). Dans certains cas, l'opérateur peut être vide et reçoit la valeur Δ. Les DPS contiennent aussi des traits lexicaux de genre, de nombre et de personne [3]. Par ailleurs, ces structures contiennent aussi un indice référentiel unique, qui permet de coréférencer les éléments dans la phrase, comme nous le verrons plus loin.

Les *structures de caractéristiques* (*Characteristic Structures*, CHS) représentent les modifieurs des syntagmes et contiennent les adverbes et les adjectifs. Les informations abstraites sont la portée de l'adverbe (phrase ou prédicat) et le degré de l'adjectif (positif, comparatif, superlatif).

Quant aux *structures de clause* (*Clause Structures*, CLS), elles représentent les événements et les états et correspondent au prédicat. Elles ont pour noyau les verbes (à l'infinitif) et parfois les adjectifs quand elles représentent un état (avec le verbe *être*). Survolons maintenant les différents éléments abstraits qui composent ces CLS.

Le type énonciatif permet de faire la distinction entre phrases déclaratives, exclamatives, interrogatives globales (questions oui/non) et interrogatives partielles. Le mode de jugement sert à distinguer les constructions impersonnelles (*thétiques*) des constructions classiques avec sujet (*catégoriques*). Le mode distingue les phrases non tensées et tensées. Ces dernières sont divisées en phrases au mode réel (indicatif), irréel (subjonctif et conditionnel) et impératif. Quant à la modalité, elle peut être de type possibilité, obligation, recommandation, permission ou indéfinie [4]. On trouve aussi un trait pour la voix (active ou passive), pour la négation (nié, non nié) et pour les phrases causatives comme *Jean fait se laver les dents aux enfants*.

3. Malheureusement, lorsqu'un élément peut avoir les deux genres, seul le masculin est conservé.

4. Cette distinction est surtout utile pour les modaux en anglais et allemand, respectivement *must* / *müssen*, *should* ou *ought* / *sollen*, *can* / *können*, *may* / *dürfen*.

Pour l'aspect, qui ne concerne que les temps finis, on peut définir le caractère progressif ou non et perfectif ou non. Pour le temps, un système inspiré du système de Reichenbach (Reichenbach, 1947) distingue le point de l'événement (E) et celui de l'énonciation (S) [5] et utilise des relations de précédence (<) et de simultanéité. La table (7.1) résume la représentation des temps dans *GBGen*.

E <S			E=S			S <E		
indéf.	perf.	prog.	indéf.	perf.	prog.	indéf.	perf.	prog.
passé simp., +subj. imp.	PQP, +subj. pqp	imp.	pres, subj. pres	p.c, +subj passé	pres. prog.	fut, +cond	fut. ant., +cond passé	fut. prog.

La deuxième ligne définit l'aspect. Le signe + indique le mode irréel

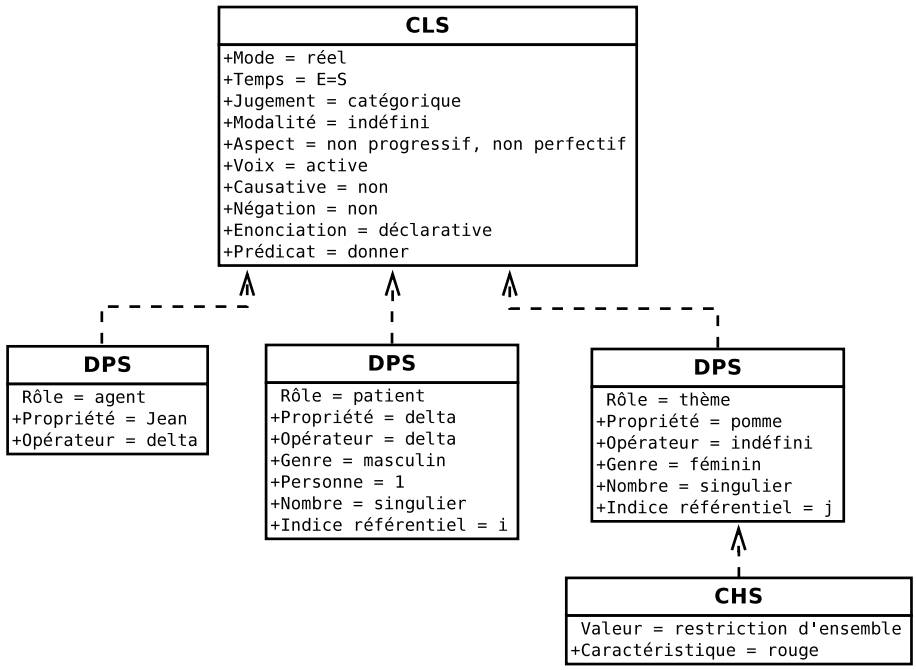
TAB. 7.1 – GBGen*: représentation des temps dans les CLS*

Notons que les éléments lexicaux sont en forme normale (infinitif pour les verbes, singulier pour les noms et masculin singulier pour les adjectifs) sauf dans les CHS. Pour terminer, signalons que dans le cas des coordonnées, les PSS n'appartiennent à aucun sous-type.

Poursuivons maintenant par la combinaison des PSS dans une structure hiérarchique. Comme nous l'avons déjà signalé, les PSS contiennent une liste d'autres PSS qui en dépendent et sont appelées *satellites*. Les satellites possèdent aussi des valeurs et de traits que nous allons présenter maintenant. Les traits permettent de repérer les topicalisations (*moi, j'ai faim*) et les focalisations ou clivées (*c'est Jean qui t'a donné la balle*). On attribue aussi parfois un rôle thématique (v. p. 119) à certains satellites qui représentent des arguments. En outre, les satellites peuvent prendre de nombreuses valeurs qui représentent la relation entre le satellite et la PSS qui le contient ; ces valeurs peuvent être réalisées par exemple par des prépositions, des complémenteurs ou un adverbe. Ces valeurs peuvent être spatiales (lieu, direction), temporelles (périodes, antériorité, simultanéité etc.), ou générales (cause, manière, conséquence, etc.).

Examinons maintenant la figure (7.1). Le prédicat contient trois satellites, qui représentent les trois arguments du verbe *donner*. L'adjectif épithète *rouge* est noté comme satellite de *pomme*, avec une valeur sémantique de restriction d'ensemble. Pour représenter la phrase passive "*Une pomme rouge m'est donnée par Jean*", il suffit de changer uniquement le trait de voix de la CLS. Pour la phrase clivée "*C'est une pomme rouge que Jean me donne*",

5. Reichenbach utilise aussi le point de référence (R). Voir Etchegoyhen (1997) pour la justification de cette simplification.



La flèche dénote la relation “est satellite de”. Les traits précédés de + concernent les sous-types de PSS et les autres concernent les PSS elles-mêmes.

FIG. 7.1 – *PSS pour* Jean me donne une pomme rouge.

le trait *focus* devra être ajouté à *pomme*.

À la figure (7.2), la relative est un satellite qui n'a pas de rôle thématique mais une valeur de restriction d'ensemble. On peut noter que le terme *voiture* repris par le pronom relatif figure deux fois dans la structure et est coréférencé par l'indice.

Pour le complément du nom, Etchegoyhen (1998) propose quatre cas différents, représentés par les phrases suivantes :

(59) a. la correction des examens par Eric ;
b. le fait que les Suisses ont interdit les minarets a provoqué une belle pagaille ;
c. la thèse de Sébastien / sa thèse ;
d. le bord du lac ; la photo des enfants.

En (59a), les déverbaux sont des noms dérivés d'un verbe. A ce titre, ils partagent la même grille thématique que le verbe dont ils sont dérivés. Les compléments seront donc des satellites : pour notre phrase, *examens* aura le

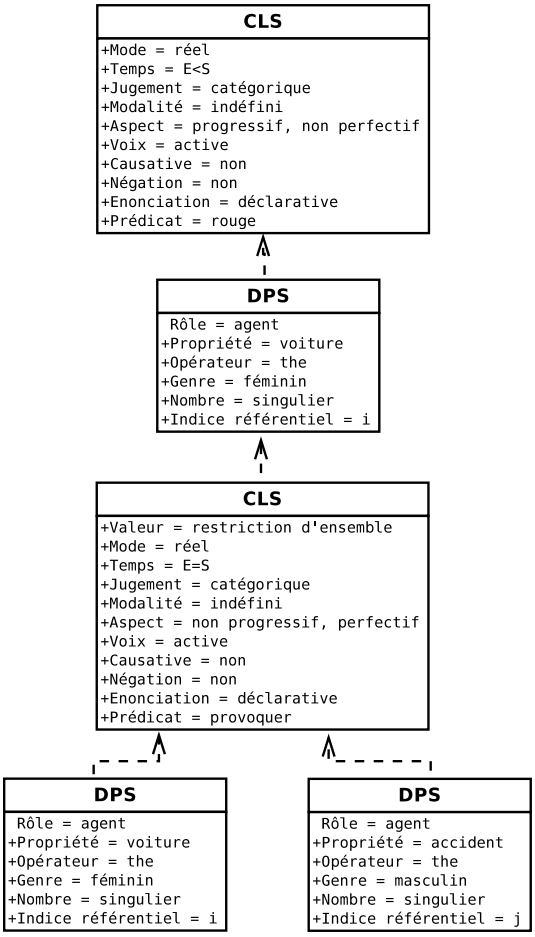


FIG. 7.2 – *PSS pour* La voiture qui a provoqué l'accident était rouge.

rôle de thème, et *Eric* d'agent. En (59b), le nom principal *fait* assigne aussi un rôle thématique de *prédication*, comme l'illustre la figure (7.3).

En (59c), nous illustrons le traitement des possessifs. Dans ces deux cas, le possesseur sera représenté par un satellite qui porte la valeur de possession, qui est attaché au possédé. Pour le déterminant possessif, le satellite sera une DPS avec une propriété et un opérateur vides. Enfin, en (59d) se retrouvent les nombreux cas où le complément ne dénote pas la possession. Dans le premier cas, il est simple de déterminer qu'un complément inanimé ne peut avoir de valeur possessive. En revanche, dans le second exemple, nous illustrons l'interprétation possible de *des enfants* comme exprimant le sujet – au sens non grammatical – de la photo. Ici, le traitement d'une telle interprétation va au delà de la sémantique. C'est pourquoi le traitement des compléments génitifs en *de* est rudimentaire et parfois imprécis : les complé-

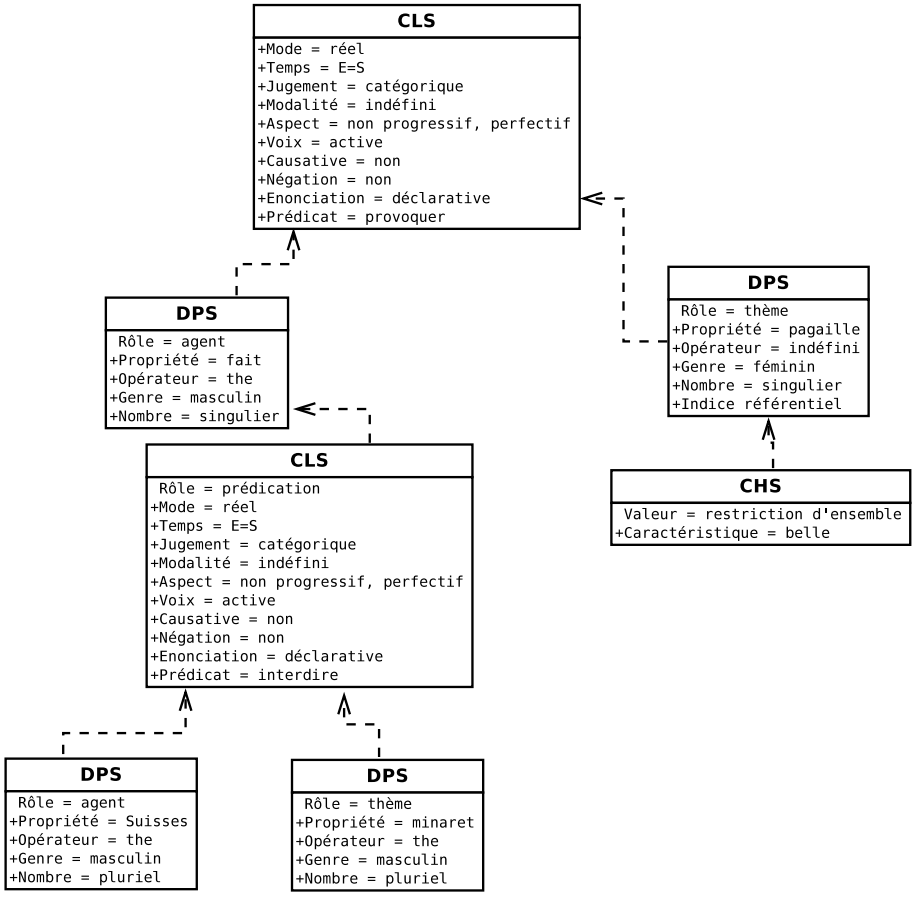


FIG. 7.3 – *PSS pour* Le fait que les Suisses ont interdit les minarets a provoqué une belle pagaille. *(ex. 59b)*

ments inanimés reçoivent la valeur de *relation* et les compléments la valeur de *possession*.

Poursuivons ce survol par le cas des verbes à montée, comme *sembler*, *paraître*, ou *s'avérer* :

(60) a. il semble que Jean dort ;

b. plusieurs étudiants semblent dormir en classe.

En (60a), la position du sujet est remplie par un *il* explétif, tandis qu'en (60b), le sujet est réalisé. Dans les deux cas, la structure sera la même, comme illustré à la figure (7.4). Le verbe *sembler* n'a qu'un seul argument, qui a le rôle de prédication.

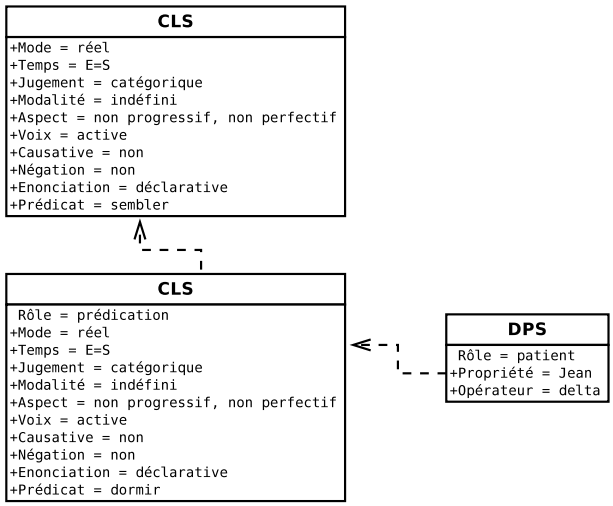


FIG. 7.4 – *PSS pour* Il semble que Jean dort. *(ex. 60a)*

Enfin, en (7.5), notre dernier exemple aborde les question globales et les hypothétiques, qui sont marquées par quelques traits spécifiques. Pour les questions partielles, le pronom interrogatif est marqué par un opérateur "WH" pour les pronoms en "qu" [6] ou par les adverbes interrogatifs.

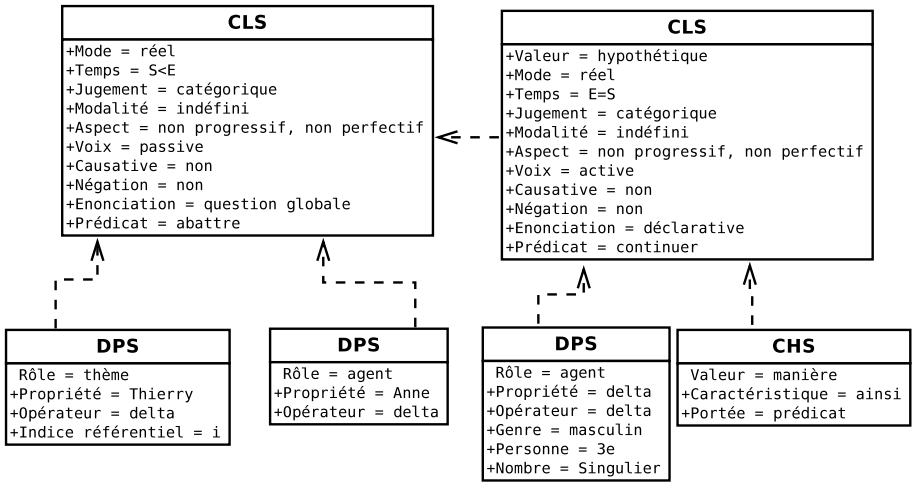


FIG. 7.5 – *PSS pour* Est-ce qu'Anne abattra Thierry s'il continue ainsi?

Ainsi, les PSS sont des structures simples qui peuvent facilement être parcourues, que ce soit pour générer des phrases ou pour comparer les PSS

6. Les pronoms interrogatifs sont parfois appelés WH en référence aux pronoms anglais *who*, *which*, *whom*, *where*, etc.

de plusieurs phrases. Les PSS se parcourent de la manière suivante :

i. On lit l'élément de tête ;
ii. On appelle récursivement la procédure de parcours de PSS sur chacun des satellites.

Les derniers développements du générateur *GBGen* et de l'extraction de PSS ont eu lieu en l'an 2000. Au moment où nous avons commencé nos travaux en 2002, l'analyseur *Fips* (§5) avait connu de grands changements structurels, notamment dus à une remodélisation en programmation orientée objet (Meyer, 2008). Nous avons donc élaboré un modèle orienté-objet pour les PSS en réécrivant l'extraction des PSS pour la langue française à partir de l'analyse de *Fips* et en ignorant la partie génération et l'extraction d'autres langues qui n'étaient plus directement utiles [7]. Un objet PSS contient les éléments communs à toutes les PSS, rôle, valeur, traits et liste de satellites. L'objet PSS peut être étendu en objets CHS, CLS et DPS, qui contiennent les propriétés propres à chaque sous-type. L'analyse sélectionnée par *Fips* est ensuite parcourue pour en extraire les informations [8]. Au final, nous avons dû nous contenter de réécrire le code sans toutefois réussir à bâtir toutes les constructions qu'extrayait l'ancien système, par manque de temps, de connaissances linguistiques et de personnes de référence pour élaborer ou valider des stratégie d'extraction des structures. Toutefois, la couverture était suffisante pour pouvoir extraire des structures significatives et élaborer notre stratégie de comparaison de phrases.

Poursuivons cette section par un bilan. Les PSS sont des structures relativement simples mais qui permettent de représenter une grande variété de phénomènes linguistiques et couvrent une large étendue de la langue. Par contre, leur extraction et leur précision repose sur un lexique riche et consistant en ce qui concerne les traits sémantiques, ce qui n'est hélas pas toujours le cas du lexique dont nous disposons. En outre, l'ordre des constituants, qui est parfois significatif, ne peut être préservé. Enfin, ces structures ne permettent pas de tenir compte des collocations (v. p. 66) comme *break a record* $\leftrightarrow$ *battre un record*. C'est une des raisons principales pour lesquelles cette

7. En l'absence des principaux auteurs du système, nous avons heureusement encore pu tester l'ancienne version du générateur qui fonctionnait encore sur un mini-ordinateur DEC tournant sous système d'exploitation *OpenVMS*.

8. Nous avons dû fortement adapter les algorithmes d'extraction aux nouvelles structures de *Fips*, dont les structures d'analyse ont parfois été modifiées par rapport à celles qui prévalaient lors du premier développement de *GBGen*. De plus, certaines parties des spécifications de *GBGen* n'avaient pas été implémentées dans le code, d'autres étaient codées de manière peu lisible sans commentaires ou encore contenaient des traitements *ad hoc* peu génériques ; cette mauvaise pratique du codage informatique est connue sous l'anglicisme de *hack*.

approche a été abandonnée en ce qui concerne la traduction automatique.

Il nous reste encore à comparer les PSS avec d'autres systèmes. Zock (1992, §B.4.33) présente un système de graphes conceptuels organisés entre eux de manière hiérarchique afin de représenter la phrase. Comme *GBGen*, les graphes mélangent traits abstraits (mode, voix, etc.) et valeurs lexicales. De même, le verbe est relié à différents actants avec des valeurs sémantiques (agent, objet, etc.). Quant aux LCS (§3.4.2), elles sont de véritables structures sémantiques, beaucoup plus abstraites puisque les mots sont représentés sous forme de combinaisons de primitives au lieu de formes lexicales. Vu la complexité du formalisme, nous nous attendons à une procédure d'extraction des LCS qui nécessite de nombreuses vérifications et un filtrage de structures erronées, par rapport à *GBGen* qui est beaucoup plus proche des structures syntaxiques. Par contre, comme *GBGen*, les LCS s'affranchissent de la syntaxe en utilisant des listes d'arguments et de modifieurs et en utilisant des traits pour la voix, etc. Notons encore que, pour les deux formalismes, il est relativement simple de comparer différentes structures entre elles pour relever les différences sémantiques.

Terminons par les QLF (§3.4.3) qui sont aussi une structure intermédiaire destinées à la traduction automatique. Basées sur des structures logiques, elles forment des combinaisons de formules composées de prédicats et d'arguments, reliées entre elles par des connecteurs. Ces structures sont plus éloignées des structures syntaxiques que les PSS. *GBGen* propose un modèle plus simple avec des valeurs qui caractérisent la dépendance hiérarchique du satellite par rapport à son parent. De plus, les rôles des différents éléments de la phrase sont plus explicitement mentionnés. Nous estimons tout de même qu'il demeure possible de comparer les QLF entre elles afin de déterminer les différences entre deux phrases.

7.2 Comparaison “sémantique” de phrases

Après cette présentation des PSS dans les grandes lignes, cette section aborde la technique de comparaison “sémantique” de phrases et ses résultats préliminaires. Nous avons expliqué la motivation de cette technique en début de chapitre. Ici, nous esquissons la technique de comparaison proprement dite. Nous poursuivons par une réflexion sur le diagnostic et terminons par une comparaison avec d'autres techniques existantes.

Pour comparer des phrases, les PSS présentent l'avantage d'une plus grande abstraction que les structures syntaxiques, avec des éléments lexi-

caux qui se limitent à des classes ouvertes et des structures de dépendance simples. Tous ces éléments rendent plus aisées les comparaisons. Nous avons successivement développé deux techniques de comparaison sémantique. Au cours même du projet FreetText (§4), nous avons esquissé un premier algorithme de diagnostic, qui comparait deux structures de PSS en se basant sur des listes des différents types de PSS. Nous essayions d'intégrer le diagnostic dans la sortie *XML* du système de diagnostic (5.3.1), qui représente une structure syntaxique, en ajoutant des balises d'erreurs. Cependant, après réexamen au moment de rédiger ce chapitre, cette technique nous a paru insuffisante. Il fallait traiter plus efficacement les constituants manquants et supplémentaires. L'ellipse posait également problème. Par conséquent, le formalisme utilisé pour le diagnostic d'erreurs syntaxiques nous a paru inadéquat pour notre diagnostic “sémantique”, bien que le diagnostic soit parfois proche de la syntaxe, comme dans l'exemple (58a) p. 255. Nous avons donc décidé de réécrire la partie de comparaison des phrases en corrigeant certaines erreurs dans l'extraction des PSS.

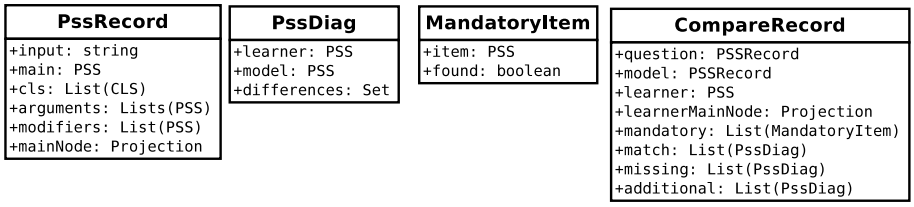


FIG. 7.6 – *Comparaison sémantique : structures utilisées*

La figure (7.6) montre les structures que nous utilisons pour comparer des phrases. Le concepteur d'exercice peut, de manière facultative, inclure dans la base de données la question de l'exercice, qui nous sert ensuite à déterminer les composants obligatoires de la question. Les structures des phrases sont stockées dans des objets *PssRecord* ; ceux-ci contiennent la PSS principale de la phrase, l'analyse de la phrase par *Fips* et des listes de PSS, une pour les CLS, une pour les arguments – les PSS qui ont un rôle thématique – et l'autre pour les modifieurs. Ces listes permettent de trouver plus rapidement des correspondances entre PSS. L'objet principal dont nous nous servons pour comparer les phrases est appelé *CompareRecord* : il contient deux objets *PssRecord* pour la question et la phrase modèle ; les PSS de la phrase de l'apprenant ne sont pas stockées dans un *PssRecord* puisqu'elles sont directement parcourues et comparées aux listes de la phrase modèle. L'objet *CompareRecord* contient quatre listes :

- *match* : liste des éléments appariés où l'on a trouvé une correspondance entre le modèle de réponse et celle de l'apprenant ;
- *missing* : liste des éléments manquants ne figurant pas dans la phrase

de l'apprenant [9] ;
- *additional* : liste des éléments additionnels qui ne figurent pas dans la phrase modèle ;
- *mandatory* : le cas échéant, liste des éléments obligatoires.

Quant aux objets *PssDiag*, ils contiennent les PSS des éléments correspondants et la liste des différences sous forme d'un ensemble. Enfin, les objets *MandatoryItem* permettent de stocker les éléments obligatoires de la phrase et de vérifier leur présence à la fin du processus de comparaison.

Notre stratégie de comparaison est résumée par la figure (7.7). Si l'apprenant a donné exactement la même réponse que la phrase modèle, le processus s'arrête. De même, si l'une des phrases contient une erreur, le système ne va pas plus loin car il est difficile d'extraire des PSS d'une phrase incorrecte.

Si la question a été fournie au système, une première phase cherche à déterminer les éléments obligatoires dans la réponse modèle. Il faut remplir deux conditions :

i. la phrase de la question doit être une question partielle ;
ii. les CLS principales des deux phrases doivent avoir le même élément lexical.

Si ces conditions sont remplies, on essaye de faire correspondre les arguments des deux phrases. En cas de correspondance de deux DPS, si celle de la question a un opérateur 'WH' qui marque les éléments interrogatifs, alors la DPS correspondante de la réponse modèle sera considérée comme un élément obligatoire.

Ensuite, on lance la comparaison proprement dite entre la phrase modèle et celle de l'apprenant. Dans une première phase, les PSS de la réponse de l'apprenant sont parcourues en établissant les correspondances les plus évidentes entre les phrases.

Les CLS sont comparées à d'autres CLS et aux CHS, car les CLS peuvent contenir des adjectifs. On recherche donc un élément qui contient la même tête lexicale dans les listes de CLS et de modifieurs du modèle. En cas de succès, l'élément retrouvé est retiré des listes du modèle et de la liste des éléments manquants de l'objet de comparaison. Les DPS ne sont comparées

9. Au début du processus, la liste contient toutes les PSS de la phrase modèle, puis cette liste est vidée petit à petit au fur et à mesure des correspondances.

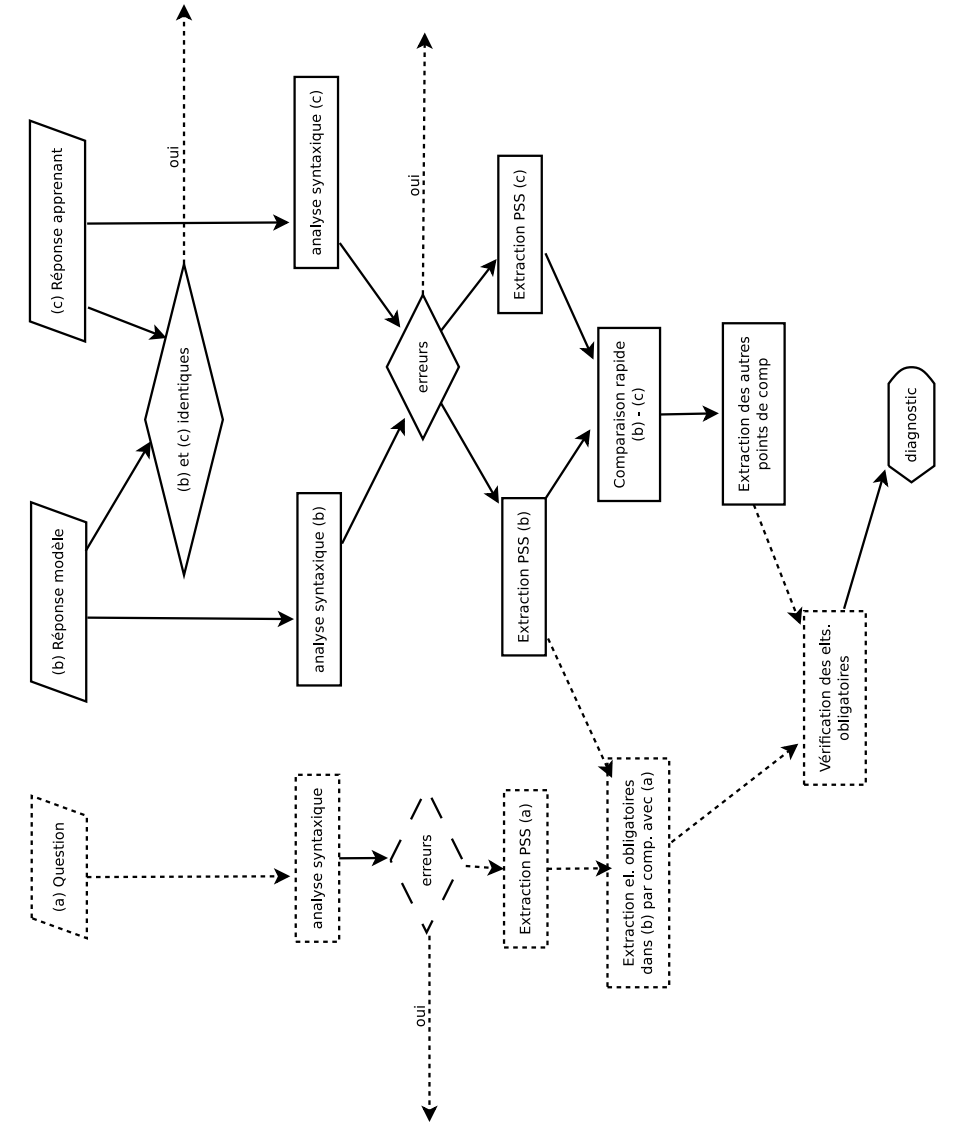


FIG. 7.7 – *Comparaison sémantique : processus*

qu'à d'autres DPS, et sont recherchées dans la liste des arguments (avec le même rôle thématique) et des modifieurs. Pour les CHS, c'est le même principe que pour les CLS. Si aucun élément correspondant n'est trouvé, la PSS de l'apprenant est ajoutée à la liste des éléments additionnels. Ainsi, après cette première phase de comparaison, les correspondances entre structures partageant la même tête lexicale sont retrouvées.

Puis nous tentons une seconde phase en parcourant la liste des éléments manquants, si cette liste n'est pas vide. Pour chaque élément de cette liste,

nous parcourons la liste des éléments additionnels. Si les types des deux éléments comparés sont compatibles entre eux – d'après les règles de la première phase – nous tentons deux types de comparaisons. Tout d'abord, nous parcourons la liste des éléments appariés pour trouver le parent de l'élément manquant. Si ce parent est également le parent de l'élément additionnel et si leurs rôles thématiques correspondent, nous les ajoutons aux éléments appariés. Si aucun élément n'est trouvé ou si la liste des éléments appariés est vide, on tente de retrouver une correspondance dans les éléments obligatoires.

Enfin, nous pouvons procéder à un diagnostic. Pour l'instant, nous nous contentons d'afficher les quatre listes que nous avons élaborées. Nous affichons en outre les différences entre les éléments appariés. Ce diagnostic est très sommaire et devrait être affiné sur le plan pédagogique. Par manque de temps et de compétences didactiques et pédagogiques, nous avons renoncé à développer plus loin nos travaux. Nous reviendrons plus tard sur diverses propositions dans ce domaine.

Pour évaluer grossièrement notre prototype, nous lui avons soumis quelques phrases de test. Le résultat détaillé est présenté à l'annexe I.

(61) a. Qui mange la souris grise?
b. Le chat noir mange la souris grise.
c. Le chat noir.

En (61, §I.1), nous illustrons un cas simple d'ellipse. La question (61a) fournie par le concepteur de l'exercice porte sur le sujet de la phrase, l'agent, qui est lexicalisé par un pronom interrogatif marqué par le trait WH. La réponse attendue (61b) est la plus complète possible. Par contre, la réponse de l'apprenant (61c) est elliptique, avec un simple groupe nominal, où figure l'élément obligatoire. Aucun élément supplémentaire n'est extrait de la phrase de l'apprenant. En revanche, des éléments manquants sont signalés. Le feedback pourrait soit valider la réponse, soit la signaler comme incomplète, par exemple selon les paramètres de l'exercice.

(62) a. Qu'est-ce que le chat mange?
b. Le chat mange une souris grise.
c. Une souris blanche.

La phrase (62, §I.2) est très similaire. La question porte sur l'objet (thème) de la phrase. La comparaison de phrases apparie les deux adjectifs *blanche* et *grise*. Ici, une erreur lexicale devrait être signalée, ainsi que

l'ellipse selon les exercices. Pour les adjectifs, en se basant sur un lexique contenant des traits sémantiques riches et consistants, notre algorithme d'appariement de phrases devrait idéalement n'accepter l'appariement que sur la base de traits indiquant qu'il s'agit d'un adjectif de couleur.

La phrase (§I.3) illustre l'exemple (58a) p. 255. La question est entrée dans le système, mais aucun élément obligatoire ne peut être extrait. Les deux réponses ne diffèrent que par le genre du clitique. Une différence de genre est détectée. Ici, le feedback pourrait facilement reprendre le diagnostic syntaxique comme celui présenté à la section 5.3.3.

(63) a. C'est Jean qui m'a donné cette belle pomme rouge.
b. La pomme m'a été donnée par Jean.

En (63, §I.4), l'exemple compare des structures assez différentes. La réponse attendue (63a) a un sujet focalisé. La réponse de l'apprenant (63b) est une structure passive qui reprend la plupart des éléments de la réponse attendue, mais dans une structure syntaxique très différente. Les éléments manquants sont bien repérés et les éléments appariés sont corrects [10].

En (§I.5), on traite l'exemple (58c) p. 255. Ici, nous avons une différence lexicale importante entre *perroquet* et *oiseau*. L'algorithme d'appariement devrait idéalement disposer d'un dictionnaire de synonymes et de relations sémantiques comme présentés au chapitre 3.5, afin de valider le rapprochement entre les deux termes.

En (§I.6), on traite l'exemple (58d) p. 255. La question n'est pas entrée dans le système. La différence entre *ses* et *ces* devrait faire l'objet d'un diagnostic particulier. En effet, la représentation des déterminants possessifs est plus complexe que les démonstratifs. Le système détecte donc deux différences, une sur l'opérateur et l'autre sur un élément additionnel.

En (§I.7), on traite l'exemple (58e) p. 255. Pour la question, nous repérons la présence d'un élément WH qui a une tête lexicalisée, *langue*, alors que les exemples précédents avaient une tête vide. La réponse attendue est une phrase passive. La première réponse a une structure active avec le pronom impersonnel *on* [11]. On repère l'ajout d'un agent alors que celui-ci a été

10. Une erreur de l'extraction des PSS attribue un trait féminin au clitique, ce qui amène notre système à détecter une différence de genre qui n'a pas lieu d'être, surtout si l'on considère que la DPS d'un élément au genre ambigu ne devrait pas être marqué uniquement comme masculin.
11. Une erreur dans l'extraction des PSS attribue le rôle de thème à l'agent.

omis dans la réponse attendue. En outre, la DPS *français* a une différence d'opérateur. Pour la seconde réponse, les structures sont pratiquement identiques, hormis la présence d'un agent. La réponse 3, elliptique, est également validée grâce à la présence de l'élément obligatoire. Quant à la dernière réponse, elle utilise une voix pronominale, avec une erreur d'extraction des PSS qui compte le pronom *se* comme thème.

(64) a. Est-ce que Jean a lu ce livre?
b. Jean a-t-il lu ce livre?

L'exemple (64, §I.8) traite de constructions strictement équivalentes. Aucune différence n'est donc détectée entre les deux phrases. Si le concepteur d'exercice veut obligatoirement cette construction particulière, l'interface du logiciel devra prévoir un feedback particulier pour cette situation.

(65) a. la sœur du capitaine qui a été tuée.
b. la sœur du capitaine qui a été tué.

La phrase (65, §I.9) présente un cas proche du cas (58a, §I.3) : nous avons un cas d'accord du participe passé avec l'antécédent. Cependant, ici, il s'agit aussi d'identifier l'antécédent du pronom relatif. Ce pronom représente le thème du verbe *tuer*, qui est un satellite qui représente une valeur de restriction d'ensemble ; ce satellite est rattaché dans un cas à *sœur* et dans l'autre à *capitaine*. Notre algorithme repère des différences d'arguments du verbe *tuer*. Une autre tactique serait aussi de repérer plutôt que le verbe n'est pas rattaché au même élément. A ce stade, nous manquons de données pour déterminer quelle stratégie est la plus efficace et la moins sujette aux erreurs.

(66) a. Avec ses jumelles, Jean a regardé cet homme.
b. Jean a regardé cet homme avec ses jumelles.

L'exemple (66, §I.10) illustre les structures focalisées. Ici, *jumelles* porte le trait de focalisation. Dans certains cas, on pourra juger les phrases comme équivalentes. Dans d'autres types d'exercices, ce trait pourrait être signalé comme obligatoire.

(67) a. Jean a observé cet homme avec ses jumelles.

b. Jean a observé cet homme.

Pour l'exemple (67, §I.11), nous détectons le manque d'un ajout rattaché à la phrase. Celui-ci figure dans la liste des constituants manquants. Un feedback fin sera nécessaire pour les constituants manquants. Ici, nous avons un DPS qui est analysé comme modifieur du prédicat et porte le trait de *withManner*. Ces relations sémantiques permettent de fournir un feedback plus riche et complet que les techniques usuelles de reconnaissance de patrons (§3.1.1.1), en montrant la fonction des éléments.

(68) a. Le chien est noir.
b. Il est noir.
c. Noir.

En (68, §I.12), nous illustrons le cas de l'ellipse. Pour la réponse attendue (68a), le prédicat CLS contient un adjectif. Une réponse possible est (68b), pour autant qu'on accepte les pronoms. Enfin, (68c) est une réponse complètement elliptique. Dans ce cas simple, notre système ne demande qu'une seule réponse type, là où un exercice de logiciel avec *pattern matching* demanderait l'élaboration d'une formule.

(69) a. Les enfants que tu as vus hier dorment.
b. Les gamins mangent une pomme.
c. Les gamins dorment.

Enfin, en (69, §I.13), nous avons introduit des réponses (69b) et (69c) bien différentes de la réponse attendue, afin de tester l'efficacité du système. Dans le premier cas, aucun élément n'a pu être associé. Il n'y a aucune structure commune entre les deux phrases. En revanche, dans le second cas, le verbe *dormir* permet de trouver une structure commune. Dans ce cas, un lexique sémantique serait bien utile. Dans le premier cas, *dormir* et *manger* n'ont aucune relation, bien que *gamins* et *enfants* soient synonymes. Dans le deuxième cas, cette synonymie permettrait de confirmer l'appariement entre les deux DPS patientes du verbe *dormir*.

Pour finir, la table (7.2) récapitule les constructions que notre technique pourrait traiter et ébauche une proposition de feedback.

Ces exemples illustrent bien le potentiel de notre technique. Nous avons manqué de temps et de moyens pour affiner notre algorithme, tant pour

Construction	**Réponse modèle**	**Réponse apprenant**	**Rétroaction**
Accord du participe	Je les ai vues.	Je les ai vus	Erreur de genre sur le pronom.
Erreur de signification	La grosse souris est grise.	La grosse souris est blanche	Erreur de signification sur l'adjectif
Question globale (oui/non)	Jean a-t-il lu ce livre?	Est-ce que Jean a lu ce livre?	Construction équivalente
Question partielle	Quel livre est-ce que Jean lit?	Quel livre Jean lit-il?	Construction équivalente
Erreur de voix	Le chat poursuit la souris.	La souris est poursuivie par le chat.	Erreur de voix sur le verbe
Propositions relatives	La sœur du capitaine qui a été tuée.	La sœur du capitaine qui a été tué.	Erreur d'accord pronom relatif-verbe
Constructions clivées	C'est Pierre que Jean a vu.	Jean a vu Pierre	Erreur de focalisation.
Topicalisation	Avec ses jumelles, Jean a regardé cet homme.	Jean a regardé cet homme avec ses jumelles.	Erreur de focalisation
Ajout	Jean a observé cet homme avec ses jumelles.	Jean a observé cet homme.	Constituant manquant.
Montée du sujet	Jean semble dormir.	Il semble que Jean dort.	Construction équivalente.
Pronominalisation	Le chien est blanc et noir.	Il est blanc et noir	Avertissement sur le pronom
Ellipse	Le chien est blanc et noir	Blanc et noir	Avertissement : verbe manquant

TAB. 7.2 – *Constructions traitées par la comparaison de phrase.*

l'extraction des PSS que pour la technique de comparaison proprement dite. Un projet de recherche pourrait sans doute, sans moyens démesurés, arriver à un prototype testable et exploitable. De plus en plus de ressources linguistiques peuvent être librement exploitées à des fins de recherche et nous pourrions procéder à des vérifications sémantiques au moins pour une couverture significative de la langue.

Dans le cadre du projet FreeText (§4), nous avons évalué les possibilités d'exploitation de notre technique dans le cadre d'un tutoriel. Certaines question du contexte de production pourraient être évaluées, pour autant que les réponses ne soient pas trop complexes, comme dans l'exemple :

(70) a. Qui a écrit ce document?
b. Il a été écrit par le docteur Varonnier et la Société Suisse d'Aérobiologie (SSA).

En revanche, les exercices de manipulation sont tout à fait traitables par notre technique, avec des transformations de l'actif au passif, transformation de phrases complètes en phrase nominale et inversement, impérative en

complétive, phrases impératives vers le futur simple, création de phrases hypothétiques à partir de deux mots etc. Finalement, l'on peut constater que la comparaison sémantique de phrases est plutôt rarement compatible avec les exercices et l'approche pédagogique de FreeText, qui est plus communicative que grammaticale. L'outil serait plus ciblé pour des apprenants un peu moins avancés, afin de disposer d'exercices de transformation et de compréhension où les réponses attendues sont assez précises et d'une complexité grammaticale relativement simple [12].

Pour terminer, comparons notre technique avec une stratégie similaire. Nous avons déjà évoqué la technique de *MILT* (§B.3.3), basée sur les LCS (§3.4.2), où la réponse de l'apprenant est également comparée à une réponse stockée. Nous avons constaté que les LCS étaient beaucoup plus abstraites que les PSS, qui sont plus simples et plus orientées vers la syntaxe (§7.1). La couverture de la langue des PSS est vraisemblablement plus large, puisqu'il ne faut pas représenter le sens des phrases. Avec les LCS, on peut s'attendre à ce qu'une petite variation syntaxique entraîne de grandes différences de structures, ce qui est moins le cas avec les PSS. Enfin, pédagogiquement, les buts des deux approches sont différentes : MILT porte sur un champ plutôt restreint de la langue, où il s'agit de diriger un agent dans un micromonde, alors que notre approche vise à corriger des phrases dans un contexte beaucoup plus large.

Murray (1995) décrit un formalisme conceptuel indépendant de la langue. Le but est de fournir des indications à un micromonde dans le cadre du projet *Athena* (§B.3.1). Les informations sont extraites des structures fournies par un analyseur GB (§3.3.5) et sont proches des LCS (§3.4.2). Pour agir sur un micromonde, Mulford (1989) utilise également un formalisme sémantique (qui n'est pas décrit dans l'article) qui sert aussi à générer une phrase correcte et à demander confirmation à l'apprenant.

DiBEx (Klenner, 2004, §B.4.15) compare la phrase de l'apprenant à la phrase stockée dans le système mais semble ne pas utiliser de système sémantique. *VINCI* (Levison et Lessard, 1996, §B.2.38) compare également une phrase attendue avec celle de l'apprenant, mais seulement sur la base de la distance entre les deux chaînes, sans syntaxe ni sémantique.

12. Après la présentation de nos premiers résultats, le consortium de FreeText a décidé de laisser de côté le développement de cette technique pour porter les priorités sur d'autres outils plus utiles et immédiatement utilisables. De plus, le temps nécessaire développement de l'outil ne laissait pas le temps d'élaborer des exercices pertinents afin de tester l'utilité pédagogique de l'outil.

7.3 Reformulation de phrases

Dans cette section, nous esquissons d'autres utilisations théoriquement possibles du générateur *GbGen* pour des applications d'ALAO. Le projet FreeText proposait l'idée d'un reformulateur de phrases, également basé sur les structures et la génération de phrase de *GbGen* (Vandeventer et Hamel, 2000). L'idée générale est de fournir un outil qui permette à l'apprenant d'agir sur certains paramètres d'une phrase, d'essayer de prédire les changements syntaxiques sur sa phrase et de comparer sa production avec le résultat produit par le générateur de phrase. Une autre variante possible serait de générer la phrase sans demander à l'apprenant de prédire les changements de structure. De même, la phrase de départ pourrait être fournie par un concepteur d'exercice ou par l'apprenant lui-même.

En soi, les PSS sont trop complexes pour être exploitées telles quelles. Cependant, en permettant de manipuler certains paramètres, il est possible d'exploiter la puissance du générateur. Certaines options seraient cachées et les notions des différentes structures devraient être adaptées au niveau de grammaire de l'apprenant. Enfin, certaines options par défaut devraient faciliter le travail des apprenants. Les transformations possibles sont :

i. Voix : active ou passive ;
ii. Négation : phrase négative ou non négative ;
iii. Type d'énoncé : affirmatif, question oui/non, question partielle, exclamatif ;
iv. Topicalisation : éléments topicalisés ou non dans le texte ;
v. Modes : réel, irréel, impératif ;
vi. Temps : présent, passé, futur,...

Transformation	**Phrase**
Phrase d'entrée	Les acariens envahissent la literie.
Voix	La literie est envahie par les acariens.
Voix et négation	La literie n'est pas envahie par les acariens.
Enoncé interrogatif (question oui/non)	Est-ce que les acariens envahissent la literie?
Focalisation obj. direct	C'est la literie que les acariens envahissent.

TAB. 7.3 – *Exemples de transformation de phrases*

Le tableau 7.3 donne un exemple d'une série de transformations. Il n'est pas nécessaire de commencer par une phrase simple. Il serait possible de

transformer une interrogative passive en phrase affirmative active. Pour que le système fonctionne, l'apprenant doit participer activement en faisant des choix, aidé de diverses aides telles que des dictionnaires en ligne, les tutoriels du logiciel, etc. Il devra notamment penser aux différentes unités de la phrase, ce qui peut l'aider à acquérir les structures grammaticales par une compréhension de la langue.

Vandeventer et Hamel (2000) identifient les avantages suivants pour cette approche :

- elle développe l'autonomie de l'apprenant en réclamant sa participation active ;
- elle concerne à la fois la production et la compréhension de la langue ;
- elle permet à l'apprenant d'identifier ses lacunes ;
- elle facilite les stratégies d'autoremédiation.

Esquissons maintenant les interfaces possibles. Selon Vandeventer et Hamel (2000), l'apprenant devrait être guidé par un dialogue qui lui simplifie la tâche. Certaines options comme la focalisation pourraient n'être activées qu'en fin de processus. Lors d'une première étape, l'apprenant devrait avoir le choix d'écrire la phrase cible qu'il a en tête. Cette phrase pourrait être vérifiée grâce à un visualiseur de phrase comme une grammaire en couleurs (§5.3.2). Dans une seconde étape, l'apprenant construirait l'aspect général de sa phrase. A ce stade, il devrait choisir le type de phrase (affirmative, interrogative, négative, impérative). A la troisième étape, il définirait le type de prédicat (verbe ou adjectif), le mode, le temps, etc. Ensuite, il faudrait définir le nombre de satellites en distinguant entre arguments et ajouts. Enfin, chaque satellite serait défini.

Selon Dillenbourg (communication personnelle, 1999), il est à craindre cependant que cette tâche soit trop compliquée pour des apprenants, qui ne maîtrisent souvent qu'imparfaitement les structures grammaticales leur propre langue et n'apprennent plus ou peu les règles de grammaire. Un tel outil ne serait pas adéquat pour développer les capacités de production et de compréhension écrite et orale. Il s'agirait donc davantage d'un outil de linguiste, difficile à intégrer dans une séquence pédagogique. Nous ne pouvons qu'aller dans son sens en craignant que la construction d'un tel dispositif de test ne donne que de très maigres et décourageants résultats.

Comme pour *Swim* (Zock, 1992, §B.4.33), il est probablement préférable de concevoir un système beaucoup plus guidé, basé sur un lexique. Un dialogue devrait permettre, à l'aide de notions grammaticales simples en lien

avec une grammaire de référence et divers dictionnaires, de construire un squelette. De plus, une interface graphique attractive pourrait servir à manipuler les éléments. Plusieurs niveaux de difficulté et plusieurs interfaces et mode de fonctionnement pourraient coexister, adaptés à l'âge ou au niveau de connaissance de l'apprenant, y compris en langue première. Comme scénario, imaginons une activité autour d'un texte sur la révolution française, avec des apprenants universitaires d'un bon niveau de langue. Une question de compréhension d'un texte demanderait pourquoi Louis XVI a déménagé à Paris. L'apprenant veut répondre que la foule a exigé le départ de Versailles du roi.

L'apprenant serait d'abord invité à choisir le prédicat de sa phrase :

(71) – Votre phrase principale est-elle :

i. un énénement ? (ex: le chien *mange* la souris)
ii. un état ? (ex: le ciel est *bleu*)

Après avoir choisi un événement, l'apprenant sélectionnerait le verbe *demander*, qui est un verbe connu, mais veut trouver un verbe plus fort. Grâce à un dictionnaire sémantique, il trouverait le verbe *exiger*. L'étape suivant est de sélectionner la bonne construction du verbe :

(72) – Le verbe *exiger* peut avoir plusieurs utilisations. Quel est son *objet* :

i. un nom ?
ii. une phrase ?

Si nécessaire, la notion d'objet serait expliquée par un glossaire et / ou une grammaire de référence. L'apprenant choisit la seconde solution. Ensuite, le système lui demande le sujet. Il sélectionne le nom *foule*. Comme l'apprenant a demandé un objet de type phrase, le système lui demande quel est le prédicat de la phrase objet. L'apprenant choisit alors le verbe *quitter*. Comme sujet, il sélectionne *roi*, puis comme objet il sélectionne *Versailles*. Alors, le système demande si d'autres éléments doivent modifier les phrases. Après une réponse négative, le générateur peut produire la phrase (73) :

(73) La foule exige que le roi quitte Versailles.

Ensuite, l'apprenant peut décider de mettre la phrase principale au passé. A travers des choix, il peut sélectionner les traits propres au passé composé. Le dialogue lui demande alors s'il veut encore renforcer un des composants de la phrase : il choisit de marquer *foule*. Enfin, après réflexion, il choisit encore d'ajouter un modifieur pour la phrase subordonnée. Parmi les modifieurs possibles, il choisit un modifieur de temps. Parmi les possibilités, il choisit d'utiliser un adverbe et peut obtenir une liste d'adverbes de temps fréquemment utilisés. Il choisit l'adverbe *immédiatement*. Au final, la phrase devient :

(74) C'est la foule qui a exigé que le roi quitte immédiatement Versailles.

Cette première ébauche de scénario est très sommaire et un projet de recherche devrait élaborer un système plus complet, élaborer divers scénarios d'utilisation et tester un prototype avec des apprenants. Dans l'idéal, il faudrait disposer de plusieurs publics cibles et varier les scénarios [13].

7.4 Discussion

Ce chapitre a présenté des outils beaucoup moins aboutis, mais aussi plus novateurs. La comparaison sémantique a pu faire l'objet d'un premier prototype qui a pu donner des premiers résultats. Ceux-ci, selon nous, confirment la faisabilité de l'outil. Sans moyens démesurés, il serait possible de développer un prototype de recherche et de procéder à un test approfondi pour valider cette approche.

En revanche, la seconde technique est encore très floue. Il est difficile d'imaginer des scénarios pédagogiques simples pour inciter les apprenants à construire des phrases. Il est aussi difficile de concevoir une interface facile à utiliser. Il serait envisageable de concevoir une application destinée à des étudiants en linguistique ou en didactique des langues, ou encore des apprenants universitaires de L2 avec un profil littéraire ou linguistique.

Il serait aussi envisageable d'utiliser le reformulateur comme outil de correction grammaticale (Vandeventer, 1998). Dans des cas précis où la détection d'erreur fonctionne correctement, une correction pourrait être proposée

13. Pour les mêmes raisons que la comparaison “sémantique” de phrases, le reformulateur de phrases n'a pas non plus pu être réalisé dans le cadre du projet FreeText. Ajoutons également qu'au moment de la décision du consortium, aucun développement n'avait pu être fait sur cette partie, qui nécessitait en plus le développement de la partie génération.

à l'apprenant. Dans d'autres cas, le générateur pourrait servir à générer des questions de désambiguïsation, afin de pouvoir choisir entre plusieurs corrections possibles et proposer une remédiation ou une correction. Enfin, ajoutons aussi que l'apprenant pourrait être amené à écrire une phrase dans sa langue première. Ensuite, un dialogue interactif pourrait l'aider à transposer la phrase dans la langue cible, de manière plus ou moins directive, que ce soit une simple traduction automatique passant par des PSS – avec une éventuelle désambiguïsation du sens – ou une simple aide à la rédaction.

Une autre technique serait d'utiliser le générateur comme aide à la rédaction pour compléter une phrase, comme dans *ILLICO* (Pasero et Sabatier, 1998, §B.5.8). Par ailleurs, ce système permet également de gérer un dialogue. *Robo Sensei* (Nagata, 2009, §B.2.22) génère les réponses possibles à un exercice à partir d'un schéma syntaxique de réponse et de la phrase de l'apprenant, pour ensuite comparer les réponses correctes à celle de l'apprenant et poser un diagnostic. La stratégie de triangulation didactique d'*ExoGen* (Blanchard *et al.*, 2009, §B.2.21) compare également une réponse attendue à celle de l'apprenant, mais en annotant les deux phrases à l'aide d'outils de base comme un étiqueteur. Bailey et Meurers (2008) utilisent une technique d'alignement sémantique entre deux phrases, en tenant compte de la question.

Dans le domaine peu exploré de la génération syntaxique, nous avons exploré plusieurs pistes d'utilisation d'un générateur syntaxique à large couverture. Le potentiel est intéressant, mais dans une niche pédagogique méconnue. Nous avons aussi rencontré la difficulté de devoir réécrire un outil complexe ; l'ampleur de la tâche dans un temps limité nous a empêché d'aller plus loin dans la réflexion pédagogique.

Chapitre 8

Conclusion

Dans cette thèse, nous avons tout d'abord survolé l'état de l'art de l'ALAO, sur le plan historique, de l'ingénierie du logiciel, de la pédagogie, de la méthodologie d'évaluation des logiciels, des méthodes d'évaluation automatique des connaissances des apprenants, de la typologie de l'ALAO et des technologies de réseau d'ordinateurs. Nous avons constaté que le domaine est vaste et plutôt prolixe. Cependant, les techniques d'évaluation traditionnelles telles que les Questionnaires à Choix Multiples ne sont ni satisfaisantes, ni suffisantes pour l'enseignement des langues.

Puis nous sommes passés au survol des technologies du TAL appliquées à l'ALAO, en examinant particulièrement la correction orthographique, l'analyse syntaxique et la détection d'erreur ainsi que les formalismes sémantiques et lexicaux. Là encore, de nombreux outils existent, avec une couverture plus ou moins grande de la langue et une fiabilité variable. Les résultats sont encourageants, bien que pas encore suffisamment probants aux yeux des enseignants de langues.

Dans la partie pratique, nous avons décrit nos travaux personnels. Nous avons commencé par une description générale du projet de recherche FreeText, qui a été à l'origine des outils présentés ici. Nous avons ensuite montré les applications d'analyse syntaxique et de détection d'erreurs et les interfaces d'aide à l'apprenant du logiciel FreeText (grammaire en couleurs, interface du diagnostic d'erreurs, analyse syntaxique arborescente).

Le chapitre suivant a décrit le correcteur orthographique *FipsOrtho* ; cet outil utilise des techniques variées (recherche de mots par alpha-code, recherche phonologique, correction morphologie *ad hoc*, etc.) destinées à corri-

ger les erreurs typiques des apprenants en langue ; des techniques de classements des résultats permettent ensuite de sélectionner les meilleures propositions. Nous avons également présenté deux évaluations de notre correcteur et le corpus qui a servi de base à l'une d'entre elles.

Enfin, le dernier chapitre nous a permis de montrer les premiers essais d'une technique de comparaison "sémantique" de phrases, qui permet d'évaluer les réponses à des exercices en tenant compte du contenu, et non seulement de la syntaxe ; en utilisant un formalisme pseudo-sémantique qui combine les éléments lexicaux des classes ouvertes à des informations abstraites, nous comparons une réponse type à celle de l'apprenant. Nous avons aussi esquissé un outil de reformulation de phrases, sans en réaliser l'implémentation.

Dans cette conclusion, nous commençons évaluons notre contribution au domaine (§8.1). Nous poursuivons par les perspectives de recherche (§8.2) et terminons par des remarques finales (§8.3).

8.1 Contributions au domaine

L'originalité du projet FreeText était surtout l'utilisation de nombreux outils de TAL (synthèse vocale, correction orthographique, conjugueur, analyseur syntaxique, détection d'erreurs syntaxiques). Vandeventer Faltin (2003) a déjà montré que l'objectif d'étendre un analyseur robuste à large couverture pour la détection d'erreurs a été atteint ; les limites de l'approche sont essentiellement le trop grand nombre d'analyses à filtrer et les erreurs de diagnostic. Notre contribution personnelle a consisté à tirer partie de la riche analyse syntaxique fournie par l'analyseur *Fips*. Des interfaces riches et variées permettent de visualiser les structures des phrases de plusieurs manières.

Pour la correction orthographique, nous avons construit un système fiable et robuste orienté vers la correction orthographique d'erreurs d'apprenants du français langue seconde. Ce champ de recherche est plutôt délaissé par la recherche. Reprenant la technique connue des alpha-codes, nous avons ensuite amélioré la technique de filtrage des propositions par distance lexicographique. Les résultats que nous avons obtenus montrent que cette méthode est plutôt efficace. Les autres méthodes sont moins innovatrices mais, globalement et en combinaison, offrent un outil performant. En outre, nous avons construit un corpus annoté d'une taille moyenne, qui est librement utilisable pour les chercheurs.

La comparaison sémantique de phrases est une technique plus originale. Nous comparons la phrase de l'apprenant à la réponse attendue par le concepteur de l'exercice. En outre, si elle est disponible, nous utilisons la question de l'exercice pour isoler les éléments obligatoires et facultatifs. Pour ces comparaisons, nous disposons de structures pseudo-sémantiques qui sont suffisamment abstraites mais donnent des informations lexicales qui évitent de lourds calculs sémantiques. Même si notre outil n'est qu'à l'état de prototype et n'a pas été testé en profondeur, nous traitons des structures variées qui couvrent un champ important de la langue.

Enfin, un outil de reformulation de phrase semble peu utilisable par les apprenants et sans application pédagogique évidente. Il est très probable que des apprenants de langue seconde soient vite rebutés par de telles applications, même en simplifiant l'interface au maximum. Nous avons toutefois esquissé quelques scénarios d'utilisation qui peuvent être testés.

8.2 Perspectives de recherche

Commençons par les perspectives de la détection d'erreurs et de l'analyse. En premier lieu, les techniques d'analyse de *Fips* ont considérablement évolué. Le travail sur les collocations pourrait donner naissance à un outil important pour l'aide à l'apprentissage et l'aide à l'acquisition de vocabulaire. Il serait d'ailleurs intéressant de construire un outil de recherche des collocations en contexte comme aide à l'apprentissage, notamment du vocabulaire. Une étude de corpus permettrait sans doute d'améliorer la couverture de la détection d'erreurs ; d'autres heuristiques pourraient également être extraites. Les techniques de diagnostic sont perfectibles et des diagnostics déroutants peuvent être évités. Enfin, grâce aux progrès techniques, les interfaces pédagogiques peuvent être améliorées et étendues.

Pour la correction orthographique, le filtrage des propositions du correcteur peut être amélioré, comme nous l'avons déjà souligné. Les travaux devraient davantage porter vers les méthodes phonétiques et *ad hoc*. Quant au corpus, il pourrait être étendu en faisant utiliser notre outil par les apprenants eux-mêmes. Ceci permettrait d'avoir enfin des indications précises sur les apprenants afin de déterminer s'il est possible de paramétrer le correcteur, par exemple en fonction de leur origine. Il reste également à faire un test d'utilisabilité du correcteur en soi.

En ce qui concerne la comparaison "sémantique" de phrases, les perspectives sont nombreuses. Tout d'abord, notre adaptation de l'extraction

des PSS est imparfaite et incomplète et il faudrait atteindre au minimum la couverture des structures atteinte par la première version du générateur *GBGen*. Bien entendu, la partie génération devrait aussi être implémentée, afin de pouvoir construire de nouveaux outils. Après ces améliorations techniques, il serait indispensable de créer des exercices adaptés aux possibilités de cet outil, afin de mener un test réel. Quant aux perspectives de la reformulation de phrases, elles sont plus floues. Il serait possible de développer plusieurs outils autour des scénarios que nous avons présentés, afin d'infirmer ou confirmer le scepticisme sur l'utilisabilité et l'utilité de tels outils. Il vaudrait peut-être mieux destiner de tels outils à des apprenants universitaires en linguistique ou en didactique du français. Il serait éventuellement possible d'utiliser *GBGen* comme traducteur à partir d'une phrase en langue première de l'apprenant, pour ensuite élaborer des exercices d'exploration de phrases, où l'apprenant peut manipuler certains paramètres. D'autres pistes d'utilisation du générateur méritent d'être explorées, comme la génération de questions de désambiguïsation ou l'aide à la rédaction.

8.3 Remarques finales

Comme nous l'avons passablement montré, l'application du TAL à l'ALAO suscite à la fois craintes et espoirs. D'un côté, les besoins d'exercices ouverts sont immenses et une correction automatique est nécessaire, d'une part parce que les apprenants ne disposent pas toujours d'un enseignant, d'autre part parce qu'il est préférable de disposer d'un feedback immédiat. En revanche, de l'autre côté, plus la phrase est complexe et / ou contient des fautes, plus les outils sont fragiles et peu fiables. Dans de nombreux cas, la connaissance de la langue par les apprenants est insuffisante [1] pour qu'il puisse détecter les erreurs de diagnostic.

A travers cet ouvrage, nous espérons avoir montré la diversité des techniques de TAL et leur apport potentiel pour l'ALAO. La reconnaissance vocale est déjà une technique imparfaite pour les locuteurs natifs. Toutefois, de nombreuses améliorations sont déjà possibles, notamment en augmentant la qualité pédagogique du feedback. Par contre, pour la détection d'erreurs, c'est au niveau technique que le potentiel est le plus fort. D'une part, les analyseurs profonds à large couverture améliorent leurs performances sur des textes corrects et les nouvelles avancées dans ce domaine peuvent être appliquées à l'ALAO. Vu que les méthodes statistiques produisent de bons résultats sur des erreurs très locales, l'avenir se situe sans doute dans les

1. Parfois, notamment chez les anglophones, les notions grammaticales de leur langue première par les apprenants est faible.

combinaisons de méthodes. Compte tenu de l'augmentation de puissance des ordinateurs personnels, même dans le bas de gamme, des techniques de traitement parallèle donneraient sans doute de bons résultats.

Depuis une dizaine d'années, l'emprise des ordinateurs et des réseaux de communication ne cesse d'augmenter. Les téléphones mobiles augmentent en capacités et en puissance et pourront de plus en plus embarquer des applications complexes. D'un autre côté, la connaissance des langues étrangères est de plus en plus importante, avec un enseignement scolaire désormais précoce. Aujourd'hui plus que jamais, il est important de multiplier les recherches en didactique des langues, en psycholinguistique et en TAL pour faire avancer la recherche. Certaines applications de TAL sont utilisables à large échelle ou en passe de l'être. Quant aux autres applications – notamment celles à plus large couverture ou utilisant des techniques complexes – il est important de poursuivre les recherches dans tous les domaines, afin de faire progresser les connaissances et, sans doute un jour, de vaincre les réticences et de trouver des applications intelligentes.

Bibliographie

Abeillé, Anne (1993). *Les nouvelles syntaxes: grammaires d'unification et analyse du français*. Linguistique. Paris: Armand Colin.

Abney, Steven Paul (1989). A Computational Model of Human Parsing. *Journal of Psycholinguistic Research*, 18(1):129–144.

Abney, Steven Paul (1997). Part-of-speech tagging and partial parsing. *In* Young, Steve et Bloothooft, Gerrit (Eds.). *Corpus-based methods in language and speech processing*, vol. 2 de *Text, speech and language technology*, pp. 118–136. Dordrecht: Kluwer.

Adriaens, Geert et Schreurs, Dirk (1992). From COGRAM to ALCOGRAM: Toward a Controlled English Grammar Checker. *In* Boitet, Christian (Ed.). *Proceedings of the fifteenth International Conference on Computational Linguistics COLING-92*, pp. 595–601, Nantes: International Comitee on Computational Linguistics: GETA (IMAG) and Association Champollion.

Aduriz, Itziar, Alegria, Iñaki, Artola, Xabier, Ezeiza, Nerea, Sarasola, Kepa et Urkia, Miriam (1991). A spelling corrector for Basque based on morphology. *Literary and Linguistic Computing*, 12(1).

Agirre, E., Alegria, I., Arregi, X., Artola, X., Diaz de Ilarraza, A., Maritxalar, M., Sarasola, K. et Urkia, M. (1992). XUXEN: A Spelling Checker / Corrector for Basque Based on Two-Level Morphology. *In Third Conference on Applied Natural Language Processing: Proceedings of the Conference*, pp. 119–125, Trento, Italy: Association for Computational Linguistics.

Agirre, Eneko, Gojenola, Koldo, Sarasola, Kepa et Voutilainen, Atro (1998). Towards a single proposal in spelling correction. *In Coling-ACL '98. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: Proceedings of the Conference*, vol. 1, pp. 22–28, Montréal, Canada: Université de Montréal.

Aho, A. V. et Johnson, S. C. (1974). LR Parsing. *ACM Computing Surveys*, 6(2):99–124.

Aist, Gregory (1999). Speech Recognition in Computer-Assisted Language Learning. *In* Cameron, Keith (Ed.). *Computer-Assisted Language Learning (CALL): Media, Design and Application*, pp. 165–181. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

Aldabe, Itziar, Arrieta, Bertol, Díaz de Ilarraza, Arantza, Maritxalar, Montse, Niebla, Ianire, Oronoz, Maite et Uria, Larraitz (2006). The use of NLP tools for Basque in a multiple user CALL environment and its feedback. *In* Mertens, Piet, Fairon, Cédrick, Dister, Anne et Watrin, Patrick (Eds.). *verbum ex machina. Actes de la 13e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 2, pp. 815–824, Leuven: Université Catholique de Louvain: UCL Presses.

Aldezabal, I., Alegria, I., Ansa, O., Arriola, J. M., Ezeiza, N., Aduriz, I. et Da Costa, A. (1999). Designing spelling correctors for inflected languages using lexical transducers. *In Proceedings of EACL'99: Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 265–266, Bergen, Norway: Association for Computational Linguistics.

Alegria, Iñaki, Arrieta, Bertol, Díaz de Ilarraza, Arantza, Izagirre, Eli et Maritxalar, Montse (2006). Using Machine Learning Techniques to Build a Comma Checker for Basque. *In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pp. 1–8, Sydney, Australia: Association for Computational Linguistics.

Allen, John Robin (1997). Ten Desiderata for Computer-Assisted Language Learning Programs: The Example of ELSE. *Computers and the Humanities*, 30:441–455.

Allodi, Alessandro, Dokter, Duco et Kuipers, Edwin (1998). WELLS: Web-Enhanced Language Learning. *In* Jager, Sake, Nerbonne, John et Van Essen, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 123–135. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

Alshawi, Hiyan (1990). Resolving Quasi Logical Forms. *Computational Linguistics*, 16(3):133–144.

Alshawi, Hiyan (Ed.) (1992). *The Core Language Engine*. Cambridge Mass.: MIT Press.

Alshawi, Hiyan et Carter, David (1994). Training and Scaling Preference Functions for Disambiguation. *Computational Linguistics*, 20(4):635–660.

Alshawi, Hiyan, Carter, David, Rayner, Manny et Gambäck, Björn (1991). Translation by Quasi Logical Form Transfer. *In 29th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 161–168, Berkeley: University of California, Berkeley: Association for Computational Linguistics.

Alshawi, Hiyan et van Eijck, Jan (1989). Logical Forms in the Core Language Engine. *In 27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 25–32, Vancouver,

Canada: University of British Columbia: Association for Computational Linguistics.

Altman, Joel et Polguère, Alain (2003). La BDéf : base de définitions dérivée du dictionnaire explicatif et combinatoire. *In Actes de la première conférence internationale de théorie Sens-Texte (MTT2003)*, pp. 43–54, Paris.

Amaral, Luiz et Meurers, Detmar (2008). From Recording Linguistic Competence to Supporting Inferences about Language Acquisition in Context : Extending the Conceptualization of Student Models for Intelligent Computer-Assisted Language Learning. *Computer Assisted Language Learning (CALL): An International Journal*, 21(4):323–338.

Amaral, Luiz A. et Meurers, Detmar (2006). Using Foreign Language Tutoring Systems for Grammatical Feedback. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 63, Granada: Eurocall.

Amsili, Pascal et Hathout, Nabil (1998). Systèmes de types pour la ($\lambda-$)DRT ascendante. *In TALN 1998, Cinquième conférence annuelle sur le Traitement Automatique des Langues Naturelles: Actes*, pp. 92–101. Paris: ATALA.

Angell, Richard C., Freund, George E. et Willett, Peter (1983). Automatic spelling correction using a trigram similarity measure. *Information Processing and Management*, 19(4):255–261.

Angelova, G., Boytcheva, S., Kalaydjiev, O., Trausan-Matu, S., Nakov, P. et Strupchanska, P. (2002). Adaptivity in Web-based CALL. *In Proc. of ECAI'02, the 15th European Conference on AI*, pp. 444–449. IOS Press.

Angelova, Galia, Strupchanska, Albena, Kalaydijev, Ognyan, Yankova, Milena, Boytcheva, Svetla, Vitanova, Irena et Nakov, Preslav (2004). Towards deeper understanding and personalisation in CALL. *In* Lemnitzer, Lothar, Meurers, Detmar et Hinrichs, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 45–52, Geneva, Switzerland: COLING.

Antoniadis, G., Echinard, S., Kraif, O., Lebarbé, T., Loiseau, M. et Ponton, C. (2004a). NLP-based scripting for CALL activities. *In* Lemnitzer, Lothar, Meurers, Detmar et Hinrichs, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 18–25, Geneva, Switzerland: COLING.

Antoniadis, Georges, Echinard, Sandra, Kraif, Olivier, Lebarbé, Thomas et Ponton, Claude (2004b). Modélisation de l'intégration de ressources TAL pour l'apprentissage des langues: la plateforme MIRTO.

In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes, pp. 57–70, Grenoble: LIDILEM: ATALA - XRCE.

Ariew, Robert (1982). A management system for foreign language tests. *Computers & Education*, 6:117–120.

Armitage, Nicholas et Bowerman, Chris (2005). The LOM Approach – A CALL for Concern? *Computer Assisted Language Learning (CALL): An International Journal*, 18(1-2):109–118.

Arneil, Stewart et Holmes, Martin (1999). Juggling Hot Potatoes: decisions and compromises in creating authoring tools for the Web. *ReCALL*, 11(2):12–19.

Asher, N., Aurnague, N., Bras, M., Sablayrolles, P. et Vieu, L. (1994). Computing the Spaciotemporal Structure of Discourse. *In* Bunt, H., Muskens, R. et Rentier, G. (Eds.). *Proceedings of IWCS' 94*, pp. 1–10, Tilburg, NL: International Workshop on Computational Semantics.

Asher, Nicholas (1993). *Reference to abstract objects in discourse*, vol. 50 de *Studies in linguistics and philosophy*. Dordrecht: Kluwer Academic Publ.

Asher, Nicholas, Denis, Pascal, Kuhn, Jonas, Larson, Erik, McCready, Eric, Palmer, Alexis, Reese, Brian et Wang, Linton (2004). Extracting and Using Discourse Structure to Resolve Anaphoric Dependencies: Combining Logico-Semantic and Statistical Approaches. *In Workshop SDRT TALN-04, Fès, 22 avril 2004*, p. sans pagination, Fès: ATALA.

Asher, Nicholas et Lascarides, Alex (2003). *Logics of conversation*. Studies in natural language processing. Cambridge, UK: Cambridge University Press.

Atwell, Eric et Elliott, Stephen (1987). Dealing with ill-formed English text. *In* Garside, Roger, Leech, Geoffrey et Sampson, Geoffrey (Eds.). *The computational analysis of English: a corpus-based approach*, pp. 120–138. London & New York: Longman.

Audras, Isabelle et Ganascia, Jean-Gabriel (2004). Analyses comparatives de production d'apprenants du français et de francophones, à l'aide d'outils d'extraction automatique du langage. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 37–46, Grenoble: LIDILEM: ATALA - XRCE.

Audras, Isabelle et Ganascia, Jean-Gabriel (2005). Analyses comparatives de productions écrites d'apprenants de français et de locuteurs francophones, à l'aide d'outils d'extraction automatique du langage. *Alsic*, 8:81–94.

Audras, Isabelle et Ganascia, Jean-Gabriel (2006). Le TALN au service de la didactique du français langue étrangère écrit. *In* Mertens, Piet, Fairon, Cédrick, Dister, Anne et Watrin, Patrick (Eds.). *verbum ex machina. Actes de la 13e Conférence sur le Traitement Automatique des*

Langues Naturelles (TALN 2006), vol. 2, pp. 825–834, Leuven: Université Catholique de Louvain: UCL Presses.

Austin, John Langshaw (1962). *How to do things with Words*. Oxford: Urmson.

Ayoun, Dalila (2007). The second language acquisition of grammatical gender and agreement. *In* Ayoun, Dalila (Ed.). *French Applied Linguistics*, vol. 16 de *Language Teaching & Language Learning*, pp. 130–170. Amsterdam, Philadelphia: John Benjamins.

Bailey, Stacey et Meurers, Detmar (2006). Using Foreign Language Tutoring Systems for Grammatical Feedback. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 67, Granada: Eurocall.

Bailey, Stacey et Meurers, Detmar (2008). Diagnosing meaning errors in short answers to reading comprehension questions. *In Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 107–115, Columbus, Ohio: Association for Computational Linguistics.

Bailin, Alan (1995). AI and Language Learning: Theory and Evaluations. *In* Holland, V. Melissa, Kaplan, Jonathan D. et Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 327–343. Mahwah, NJ: Lawrence Erlbaum Associates.

Bailin, Alan et Levin, Lori (1989). Introduction: Intelligent Computer-Assisted Language Instruction. *Computers and the Humanities*, 23:3–11.

Bailin, Alan et Thomson, Philip (1988). The Use of Natural Language Processing in Computer-Assisted Language Instruction. *Computers and the Humanities*, 22(2):99–110.

Banerjee, Satanjeev et Lavie, Alon (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan: Association for Computational Linguistics.

Bangs, Paul et Shield, Lesley (1999). Why change authors into programmers? *ReCALL*, 11(1):19–29.

Bänziger, T., Grandjean, D., Bernard, P.-J., Klasmeyer, G. et Scherer, K. R. (2001). Prosodie de l'émotion : étude de l'encodage et du décodage. *Cahiers de Linguistique Française*, 23:11–37.

Bar-Hillel, Yehoshua (1970). *Aspects of language : essays and lectures on philosophy of language, linguistic philosophy and methodology of linguistics*. Jerusalem, Amsterdam: The Magnes press, The Hebrew University, North-Holland.

Barchan, J., Woodmansee, B. et Yazdani, M. (1986). A Prolog-based Tool for French Grammar Analysis. *Instructional Science*, 15(1):21–48.

Bateman, John et Zock, Michael (2003). Natural Language Generation. *In* Mitkov, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 284–304. Oxford: Oxford Univ. Press.

Bates, Madeleine et Ingria, Robert (1981). Controlled Transformational Sentence Generation. *In 19th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Stanford, CA, USA: Stanford University: Association for Computational Linguistics.

Ben Othmane Zribi, Chiraz et Zribi, Adnane (1999). Algorithmes pour la correction des erreurs orthographiques en arabe. *In TALN 99. 6^{e} conférence annuelle sur le Traitement Automatique des Langues Naturelles: Actes*, pp. 223–232, Cargèse, Corse: ATALA.

Bender, Emily M., Flickinger, Dan, Oepen, Stephan, Walsh, Annemarie et Baldwin, Timothy (2004). Arboretum: Using a precision grammar for grammar checking in CALL. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Berghel, H. L. (1987). A logical framework for the correction of spelling errors in electronic documents. *Information Processing and Management*, 23(5):477–494.

Bernard, Pascale, Dendien, Jacques et Pierrel, Jean-Marie (2004). A computerized dictionary : Le trésor de la langue française informatisé (TLFi). *In* Zock, Michael (Ed.). *COLING 2004 Enhancing and using electronic dictionaries*, pp. 40–43, Geneva, Switzerland: COLING.

Bernth, Arendse (1997). EasyEnglish: A Tool for Improving Document Quality. *In Fifth Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 166–173, Washington Marriott Hotel, Washington, DC, USA: Association for Computational Linguistics.

Berthouzoz, Cathy (2000). Le Modèle Directionnel d'Interprétation du Discours. *Cahiers de Linguistique Française*, 22:101–146.

Biais, Maxime (2005). GRAC GRAmmar Checker. Rapport technique, SourceForge.

Bick, Eckhard (2004). Parsing and evaluating the French Europarl corpus. *In* Paroubek, Patrick, Robba, Isabelle et Vilnat, Anne (Eds.). *Méthodes et outils pour l'évaluation des analyseurs syntaxiques (Journée ATALA, May 15, 2004)*, pp. 4–9, Paris: ATALA.

Bick, Eckhard (2005). Grammar for Fun: IT-based Grammar Learning with VISL. *In* Henriksen, Peter Juel (Ed.). *CALL for the Nordic Languages*, Copenhagen Studies in Language, pp. 49–64. Copenhagen: Samfundslitteratur.

Bigert, Johnny (2004). Probabilistic Detection of Context-Sensitive Spelling Errors. *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, vol. 5, pp. 1633–1638, Lisbon, Portugal: ELRA - European Language Resources Association.

BLACHE, Philippe (2005). Combiner analyse superficielle et profonde : bilan et perspectives. *In Actes de TALN-RECITAL 2005*, pp. 93–102, Dourdan.

BLANCHARD, Alexia (2007). L'analyse morphologique des réponses d'apprenants. *In* BENAMARA, Farah, HATHOUT, Nabil, MULLER, Philippe et OZDOWSKA, Sylwia (Eds.). *Actes de la 14^e conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 1 de *Actes TALN et RECITAL*, pp. 437–446, Toulouse: Association pour le Traitement Automatique des Langues.

BLANCHARD, Alexia, KRAIF, Olivier et PONTON, Claude (2009). Mastering Overdetection and Underdetection in Learner-Answer Processing: Simple Techniques for Analysis and Diagnosis. *Calico Journal*, 26(3):592–610.

BLANCHE-BENVENISTE, Claire et CHERVEL, André (1978). *L'orthographe*. Paris: Maspéro, 3^e ed.

BLIN, Françoise (1999). CALL and the development of learner autonomy. *In* DEBSKI, Robert et LEVY, Mike (Eds.). *WORLDCALL: Global Perspectives on Computer-Assisted Language Learning*, pp. 133–148. Lisse, The Netherlands: Swets & Zeitlinger B. V.

BOITET, Christian (2000). Traduction assistée par ordinateur. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 271–291. Paris: Hermès Science Publications.

BOLT, Philip (1991). eL: A computer-based system for parsing and correcting written English. *Computer Assisted Language Learning (CALL): An International Journal*, 4(3):173–182.

BOLT, Philip et YAZDANI, Masoud (1998). The Evolution of a Grammar-Checking Program: LINGER to ISCA. *Computer Assisted Language Learning (CALL): An International Journal*, 11(1):55–112.

BORCHARDT, Frank L. (1987). Neural Network Computing and Natural Language Processing. *Calico Journal*, 5(4):63–75.

BORCHARDT, Frank L. (1995). Language and Computing at Duke University: or, Virtue Triumphant, for the Time Being. *Calico Journal*, 12 (4):57–83.

BORIN, Lars (2002). What have you done for me lately? The fickle alignement of NLP and CALL. Rapport technique 2, Uppsala Learning Lab, Uppsala.

BOS, Edwin (1994). Error Diagnosis in a Tutoring System for the Conjugation and Spelling of Dutch Verbs. *Computers in Human Behavior*, 10 (1):33–49.

BOUCHER, Paul, DANNA, Frédéric et SÉBILLOT, Pascale (1993). COMPOUNDS: an Inteligent Tutoring System for Learning to use Compounds in English. *Computer Assisted Language Learning (CALL): An International Journal*, 6(3):249–272.

BOUCHER, Paul et SÉBILLOT, Pascale (1993). Interprétation et génération automatique de noms composés anglais à l'aide de formes logiques. *Traitement automatique des langues*, 34(2):89–104.

BOUILLON, Pierrette, VANDOOREN, Françoise et LEHMANN, Sabine (Eds.) (1998). *Traitement automatique des langues naturelles*. Universités francophones. Champs linguistiques. Recueils. Bruxelles: Duculot.

BOURAOUI, Jean-Léon, BOISSIÈRE, Philippe, MOJAHID, Mustapha, VIGOUROUX, Nadine, LAGARRIGUE, Aurélie, VELLA, Frédéric et NESPOULOUS, Jean-Luc (2009). Problématique d'analyse et de modélisation des erreurs en production écrite. Approche interdisciplinaire. *In Actes de TALN-RECITAL 2009*, Senlis.

BOWERMAN, Chris (1990). ICALL: An Underview of the State of the Art in CALL. *Computer Assisted Language Learning (CALL): An International Journal*, 3:45–54.

BOWERMAN, Chris (1992). Writing and the computer: An intelligent tutoring systems solution. *Computers & Education*, 18(1–3):77–83.

BOYTCHEVA, Svetla, KALAYDJIEV, Ognian, NENKOVA, Ani et ANGELOVA, Galia (2000). Integration of Resources and Components in a Knowledge-Based Web-Environment for Terminology Learning. *In* CERRI, Stefano A. et DOCHEV, Danail (Eds.). *Artificial intelligence : methodology, systems, and applications : 9th international conference, AIMSA 2000, Varna, Bulgaria, September 20-23, 2000 : proceedings*, pp. 210–220, Berlin: Springer.

BOYTCHEVA, Svetla, KALAYDJIEV, Ognian, STRUPCHANSKA, Albena et ANGELOVA, Galia (2001). Between language correctness and domain knowledge in CALL. *In Proceedings of Euroconference RANLP 2001 (Recent Advances in NLP)*, Tzigov Chark, Bulgaria.

BOYTCHEVA, Svetla, VITANOVA, Irena, STRUPCHANSKA, Albena, YANKOVA, Milena et ANGELOVA, Galia (2004). Towards the assessment of free learner's utterances in CALL. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

BRANTS, Thorsten (2000). TnT – A Statistical Part-of-Speech Tagger. *In 6th Applied Natural Language Processing Conference & 1st Meeting of the North American Chapter of the Association for Computational Linguistics. Proceedings of the Conference*, pp. 224–231, Seattle, WA: Association for Computational Linguistics.

BREHONY, Tom et RYAN, Kevin (1994). Francophone Stylistic Grammar Checking (FSGC) Using Link Grammars. *Computer Assisted Language Learning (CALL): An International Journal*, 7(3):257–269.

BREIDT, Elisabeth et FELDWEG, Helmut (1997). Accessing Foreign Languages with COMPASS. *Machine Translation*, 12(1-2):153–174.

BREIDT, Elisabeth, SEGOND, Frédérique et VALETTO, Giuseppe (1996). Formal Description of Multi-Word Lexemes with the Finite-State For-

malism IDAREX. *In COLING 96: The 16th International Conference on Computational Linguistics. Proceedings of the Conference*, vol. 2, pp. 1036–1040, Copenhagen, Denmark: Center for Sprogteknologi.

Brett, David (2004). Drag'n'drop Exercises Made Easy. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 83, Vienna: EUROCALL.

Brett, David (2006). Conductor: a Flash-based tool for the creation of rich and varied CALL material. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 81, Granada: Eurocall.

Brill, Eric (1995). Transformation-Based-Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565.

Brock, Mark N. (1990). Customizing a Computerized Text Analyzer for ESL Writers: Cost Versus Gain. *Calico Journal*, 8(2):51–60.

Brockett, Chris, Dolan, William B. et Gamon, Michael (2006). Correcting ESL Errors Using Phrasal SMT Techniques. *In COLING/ACL 06 : Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 249–256, Sydney, Australia: Association for Computational Linguistics.

Brown, Charles Grant (2002). Inferring and Maintaining the Learner Model. *Computer Assisted Language Learning (CALL): An International Journal*, 15(4):343–355.

Brown, Charles Grant, Keim, Nathan, Brammer, Kevin et Flagel, Lorne (2004). The Incomplete Grammar approach to the development of a Strong-AI based ICALL system. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Brun, Caroline, Parmentier, Thibault, Sándor, Ágnes et Segond, Frédérique (2002). Les outils de TAL au service de la e-formation en langues. *In* Segond, Frédérique (Ed.). *Multilinguisme et traitement de l'information*, chap. 10, (pp. 223–250). Paris: Hermès Science.

Brunelle, Eric (2004). Antidote: correcteur, dictionnaire et plus. *BULAG*, 29:25–31.

Brunelle, Eric et Charest, Simon (2007). Présentation du logiciel Antidote RX. *In* Benamara, Farah, Hathout, Nabil, Muller, Philippe et Ozdowska, Sylwia (Eds.). *Actes de la 14^{e} conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^{e} Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 2 de *Actes TALN et RECITAL*, pp. 315–317, Toulouse: Association pour le Traitement Automatique des Langues.

Burstein, Jill et Chodorow, Martin (1999). Automated Essay Scoring for Nonnative English Speakers. *In* Broman Olsen, Mari (Ed.). *Computer Mediated Language Assessment and Evaluation in Natural Language Processing. Proceedings of a Symposium sponsored by the Association for Computational Linguistics and International Association of Language Learning Technologies*, pp. 68–75, College Park, Maryland USA: University of Maryland: Association for Computational Linguistics.

Burstein, Jill et Marcu, Daniel (2000). Towards Using Text Summarization for Essay-Based Feedback. *In Actes de TALN 2000*, pp. 51–59, Lausanne: ATALA.

Burston, Jack (1989). Towards Better Tutorial CALL: A Matter of Intelligent Control. *Calico Journal*, 6(4):75–89.

Burston, Jack (1993). Exploiting Available Technology. *Calico Journal*, 11(1):47–52.

Burston, Jack (1995/6). A comparative evaluation of French grammar checkers. *Calico Journal*, 13(2&3):104–111.

Burston, Jack (1998). Antidote 98. *Calico Journal*, 16(2):197–212.

Burston, Jack (2008). Review of BonPatron: An Online Spelling, Grammar, and Expression Checker. *Calico Journal*, 25(2).

Burston, Jack L. (1990). Maximizing Intelligent Use of Unintelligent Response Handling Devices. *Calico Journal*, 8(2):77–90.

Burston, Jack L. (1991). Using Système-D In A Classroom Environmnent. *Calico Journal*, 8(4):51–57.

Bush, Vannevar (1945). As we may think. *The Atlantic Monthly*, July.

Butcher, William, Galletly, John et Wong, Andrew (1990). Prolog and Language Analysis Intelligent Response to Comprehension Replies. *Computer Assisted Language Learning (CALL): An International Journal*, 3:27–36.

Buvet, Pierre-André et Issac, Fabrice (2006). TAEMA: Traitement Automatique de l'Écriture de Mots Affectifs. *In* Mertens, Piet, Fairon, Cédrick, Dister, Anne et Watrin, Patrick (Eds.). *verbum ex machina. Actes de la 13e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 2, pp. 856–867, Leuven: Université Catholique de Louvain: UCL Presses.

Campione, Estelle, Véronis, Jean et Deulofeu, José (2005). The French corpus. *In* Cresti, Emanuela et Moneglia, Massimo (Eds.). *C-ORAL-ROM, Integrated Reference Corpora for Spoken Romance Languages*, vol. 15 de *Studies in Corpus Linguistics*, pp. 111–133. Amsterdam: John Benjamins.

Candido, Arnaldo, Maziero, Erick, Specia, Lucia, Gasperin, Caroline, Pardo, Thiago et Aluisio, Sandra (2009). Supporting the Adaptation of Texts for Poor Literacy Readers: a Text Simplification Editor for Brazilian

Portuguese. *In Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 34–42, Boulder, Colorado: Association for Computational Linguistics.

Carbonell, Jaime G. et Hayes, Philip J. (1984). Coping with Extragrammaticality. *In 10^{th} International Conference on Computational Linguistics and 22^{nd} Annual Meeting of the Association for Computational Linguistics: Proceedings of Coling-ACL 84*, pp. 437–443, Stanford University, CA: Association for Computational Linguistics.

Carl, Michael, Schmidt-Wigger, Antje et Hong, Munpyo (1997). KURD - A Formalism for Shallow Post Morphological Processing. *In Proceeding of the Natural Language Processing Pacific Rim Symposium 1997 (NLPRS'97)*, Phuket, Thaïland.

Carlberger, Johan, Domeij, Rickard, Kann, Viggo et Knutsson, Ola (2002). A Swedish grammar checker. Submitted for Association for Computational Linguistics.

Carpenter, Bob et Penn, Gerald (2001). ALE: The Attribute Logic Engine: User's guide. Rapport technique, Department of Computer Science, University of Toronto, Toronto.

Carpenter Binkley, Susan (2002). The Bilingual Corrector (Ver 2.0). Calico Software Review Online.

Carpenter Binkley, Susan (2004). The Bilingual Corrector. *Calico Journal*, 21(2):431–440.

Carson-Berndsen, Julie (1998). Computational autosegmental phonology in pronunciation teaching. *In* Jager, Sake, Nerbonne, John et Van Essen, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 11–20. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

Cassart, Anne, Granger, Sylviane et Husquet, Cécile (2002). Validation du prototype FreeText par des professeurs de français. Rapport technique, Université Catholique de Louvain, Centre for English Corpus Linguistics.

Catach, N., Gruaz, C. et Duprez, D. (1986). *L'orthographe française. Traité théorique et pratique.* Paris: Nathan, 2^e ed.

Catach, Nina (1978). *L'orthographe.* Que sais-je? Paris: Presses Universitaires de France.

Catach, Nina (1989). *Les délires de l'orthographe.* Paris: Plon.

Catach, Nina, Duprez, Daniel et Legris, Michel (1980). *L'enseignement de l'orthographe : l'alphabet phonétique international, la typologie des fautes, la typologie des exercices.* Dossiers didactiques : formation initiale et continue. Paris: Nathan.

Catt, Mark et Hirst, Graeme (1990). An Intelligent CALI System for Grammatical Error Diagnosis. *Computer Assisted Language Learning (CALL): An International Journal*, 3:3–26.

CATT, Mark Edward (1988). Intelligent Diagnosis of Ungrammaticality in Computer-Assisted Language Instruction. Mémoire de D.E.A. / Master, Computer Systems Research Institute, University of Toronto, Toronto.

CAWS, Catherine (2005). Application de principes cognitivistes et constructivistes à l'enseignement de l'écrit assisté par ordinateur : perception des étudiants. *Alsic*, 8.

CAZADE, Alain (1999). De l'usage des courbes sonores et autres supports graphiques pour aider l'apprenant en langues. *Alsic*, 2(2):3–31.

CERRI, Stefano A. (1989). ALICE: Acquisition of Linguistic Items in the Context of Examples. *Instructional Science*, 18:63–92.

CERRI, Stefano A., CHELI, Elena et MCINTYRE, Angus (1992). Nobile: Object-based user model acquisition for second language learning. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 171–190. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

CHAMBERS, Angela et O'SULLIVAN, Ide (2004). Advanced learners' writing skills in French: the role of corpus consultation skills. *In Actes d'UNTELE 2004*, Compiègne.

CHAN, Tun-pei et LIOU, Hsien-Chin (2005). Effects of web-based concordancing instruction on EFL students' learning of verb-noun collocations. *Computer Assisted Language Learning (CALL): An International Journal*, 18(3):231–250.

CHANIER, Thierry (1996). Learning a second language for specific purposes within a hypermedia framework. *Computer Assisted Language Learning (CALL): An International Journal*, 9(1):3–43.

CHANIER, Thierry, DUQUETTE, Lise, LAURIER, Michel et POTHIER, Maguy (1997). Stratégies d'apprentissage et évaluation dans des environnements multimédia d'aide à l'apprentissage du français. *In Journées Scientifiques et Techniques du Réseau FRANCophone de l'Ingénierie de la Langue de l'Aupelf-Uref (JST'97)*, pp. 271–276, Avignon, France: Aupelf-Uref.

CHANIER, Thierry, PENGELLY, Michael, TWIDALE, Michael et SELF, John (1992). Conceptual Modelling in Error Analysis in Computer-Assisted Languge Learning Systems. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 125–150. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

CHANIER, Thierry et SELVA, Thierry (2004). The ALEXIA System: the Use of Visual Enhanced Vocabulary Learning. *Computer Assisted Language Learning (CALL): An International Journal*, 11(5):367–399.

CHANIER, Thierry et VETTER, Anna (2006). Multimodalité et expression en langue étrangère dans une plate-forme audio-synchrone. *Alsic*, 9.

Chapelle, Carol et Jamieson, Joan (1983). Recognition of Student Input in Computer-Assisted Language Learning. *Calico*, 1(3):7–10.

Chapelle, Carol A. (1998). Multimedia CALL: Lessons to be Learned from Research on Instructed SLA. *Language Learning & Technology*, 2 (1):22–34.

Chapelle, Carol A. (2001). *Computer Applications in Second Language Acquisition: Foundations for teaching, testing and research*. Applied Linguistics. Cambridge, UK: Cambridge University Press.

Chapelle, Carol A. (2003). *English Language Learning and Technology: Lectures on applied linguistics in the age of information and communication technology*, vol. 7 de *Language Learning and Language Teaching*. Amsterdam/Philadelphia: John Benjamins.

Charest, Simon, Brunelle, Éric, Fontaine, Jean et Pelletier, Bertrand (2007). Élaboration automatique d'un dictionnaire de cooccurrences grand public. *In* Benamara, Farah, Hathout, Nabil, Muller, Philippe et Ozdowska, Sylwia (Eds.). *Actes de la 14^e conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 1 de *Actes TALN et RECITAL*, pp. 283–292, Toulouse: Association pour le Traitement Automatique des Langues.

Charniak, Eugene (2000). A Maximum-entropy-inspired parser. *In 1st Meeting of the North American Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 132–139, Seattle, WA, USA: Association for Computational Linguistics.

Chen, Howard et Chen, C. T. (2004). A Report on Using Wireless Tablet PCs in a Language Classroom. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 95, Vienna: EUROCALL.

Chen, Lei, Zechner, Klaus et Xi, Xiaoming (2009). Improved pronunciation features for construct-driven assessment of non-native spontaneous speech. *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 442–449, Boulder, Colorado: Association for Computational Linguistics.

Chen, Li et Kurtz, Barry L. (1989). XTra-TE: Using Natural Language Processing Software to Develop an ITS for Language Learning. *In Proceedings of Artificial Intelligence and Education (AI&Ed.)*, pp. 54–63.

Chen, Liang, Tokuda, Naoyuki et Hou, Pingkui (2005). A Table Look-Up Parser in Online ILTS Applications. *Computer Assisted Language Learning (CALL): An International Journal*, 18(1-2):49–62.

Chen, Liang, Tokuda, Naoyuki et Xiao, Dahai (2002). A POST Parser-Based Learner Model for Template-Based ICALL for Japanese-English

Writing Skills. *Computer Assisted Language Learning (CALL): An International Journal*, 15(4):357–372.

Cherry, Lorinda L. et Macdonald, Nina H. (1983). The Unix Writer's Workbench Software. *BYTE: the small systems journal*, 8(10):241–248.

Chevalier, Monique, Dansereau, Jules et Poulin, Guy (1978). TAUM-METEO: description du système. Technical report, groupe de recherche en traduction automatique, groupe TAUM, Université de Montréal, Montréal.

Chinnery, George M. (2006). Going to the MALL: Mobile Assisted Language Learning. *Language Learning & Technology*, 10(1):9–16.

Chinnery, George M. (2008). You've Got some GALL: Google-Assisted Language Learning. *Language Learning & Technology*, 12(1):3–11.

Chomsky, Noam (1957). *Syntactic Structures*. The Hague: Mouton and Co.

Chomsky, Noam (1965). *Aspects of the Theory of Syntax*. Cambridge, Mass: MIT Press.

Chomsky, Noam (1981). *Lectures on Government and Binding*. Dordrecht: Foris Publications.

Chomsky, Noam (1995). *The Minimalist Programm*. Cambridge, Mass.: MIT Press.

Chomsky, Noam et Lasnik, Howard (1995). The Theory of Principles and Parameters. *In* Chomsky, Noam (Ed.). *The Minimalist Programm*, pp. 13–127. Cambridge, Mass.: MIT Press.

Chun, Dorothy M. (1998). Signal analysis software for teaching discourse intonation. *Language Learning & Technology*, 2(1):61–77.

Church, Kenneth W. et Gale, William A. (1991). Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103.

Clancy, P. (1973). A visit to the Plato computer teaching project for French. *Babel*, 9(2):13–17.

Clark, Robin (1993). Semantics for Computers. Rapport technique 9, Université de Genève, Laboratoire d'Analyse et de Technologie du Langage, Genève.

Claveau, Vincent et Sébillot, Pascale (2004). From efficiency to portability: acquisition of semantic relations by semi-supervised machine learning. *In Coling Geneva 2004: 20th International Conference on Computational Linguistics: Proceedings*, vol. 1, pp. 261–267, Geneva, Switzerland: COLING.

Claveau, Vincent, Sébillot, Pascale, Bouillon, Pierrette et Fabre, Cécile (2001). Acquérir des éléments du lexique génératif : quels résultats et à quel coût? *Traitement automatique des langues*, 42(3):729–753.

Clear, Jeremy (2000). Do you believe in Grammar? *In* Burnard, Lou et McEnery, Tony (Eds.). *Rethinking language pedagogy from a corpus*

perspective : Papers from the Third International Conference on Teaching and Language Corpora, vol. 2 de *Łódź Studies in Language*, pp. 19–30, Frankfurt: Peter Lang.

Clément, Lionel, Gerdes, Kim et Marlet, Renaud (2009). Grammaires d'erreur – correction grammaticale avec analyse profonde et proposition de corrections minimales. *In Actes de TALN-RECITAL 2009*, Senlis.

Cobb, Tom (1999). Breadth and depth of lexical acquisition with hands-on concordancing. *Computer Assisted Language Learning (CALL): An International Journal*, 12(4):345–360.

Coch, José et Morize, Geneviève (1990). Un analyseur conçu pour le traitement d'erreurs et ambiguïtés. *In Les systèmes experts et leurs applications : dixièmes journées internationales, Avignon, France, 28 mai - 1er juin 1990 : conférence spécialisée : le traitement du langage naturel et ses applications*, pp. 147–160. EC2.

Cohen, Rachel (1993). The use of voice synthesizer in the discovery of the written language by young children. *Computers & Education*, 21(1–2):25–30.

Coniam, D. (1997). A preliminary inquiry into using corpus word frequency data in the automatic generation of English language cloze tests. *Calico Journal*, 14(2–4):15–33.

Coniam, David (1998). The use of speech recognition software as an English language oral assessment instrument: an exploratory study. *Calico Journal*, 15(4):7–23.

Cook, Vivian (1988). Designing a BASIC parser for CALL. *Calico Journal*, 6(1):50–67.

Cordier-Gauthier, Corinne et Dion, Chantal (2003). La correction et la révision de l'écrit en français langue seconde : médiation humaine, médiation informatique. *Alsic*, 6(1):29–43.

Cornu, Etienne (1997). *Correction automatique des erreurs morphologiques et syntaxiques produites à l'écrit en langue seconde*. Thèse de doctorat, Université de Neuchâtel, Faculté des Lettres, Neuchâtel.

Cornu, Etienne, Kübler, Natalie, Bodmer, Franck, Grosjean, François, Grosjean, Lysiane, Léwy, Nicolas, Tschichold, Cornelia et Tschumi, Corinne (1996). Prototype of a second language writing tool for French speakers writing in English. *Natural Language Engineering*, 2(3):211–228.

Cosi, P., Delmonte, R., Biscetti, S., Cole, R. A., Pellom, B. et van Vuren, S. (2004). Italian Litteracy Tutor: tools and technologies for individuals with cognitive disabilities. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Couch, Pamela (2000). Review of Easy Writer. *Language Learning & Technology*, 4(2):37–42.

COURTIN, J., DUJARDIN, D., KOWARSKI, I., GENTHIAL, D. et DE LIMA, V. L. (1991). Towards a complete detection/correction system. *In International Conference on Current Issues in Computational Linguistics*, pp. 158–173, Penang, Malaysia.

COVINGTON, Michael A. (Ed.) (1994). *Natural Language Processing for Prolog Programmers*. Englewood Cliffs, NJ: Prentice-Hall.

CRISWELL, Eleanor, BYRNES, Heidi et PFISTER, Günter (1992). Intelligent Automated Strategies of Teaching Foreign Language in Context. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 307–319. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

CROSS, Jeremy (2002). 'Noticing' in SLA: Is it a valid concept? *TESL-EJ Teaching English as a Second or Foreign Language*, 6(3):1–9.

CRYSMANN, Berthold, BERTOMEU, Nuria, ADOLPHS, Peter, FLICKINGER, Daniel et KLÜWER, Tina (2008). Hybrid Processing for Grammar and Style Checking. *In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 153–160, Manchester, UK: Coling 2008 Organizing Committee.

CULICOVER, Peter W. et JACKENDOFF, Ray (2005). *Simpler Syntax*. Oxford University Press.

CULICOVER, Peter W. et JACKENDOFF, Ray (2006). The simpler syntax hypothesis. *Trends in Cognitive Sciences*, 10(9):413–18.

CUTTING, Doug, KUPIEC, Julian, PEDERSEN, Jan et SIBUN, Penelope (1992). A Practical Part-of-Speech Tagger. *In Third Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 133–140, Trento, Italy: Association for Computational Linguistics.

DAHLGREN, Kathleen (1988). *Naive Semantics for Natural Language Understanding*. Boston, Dordrecht: Kluwer Academic Publishers.

DALBY, Jonathan et KEWLEY-PORT, Diane (1999). Explicit Pronunciation Training Using Automatic Speech Recognition Technology. *Calico Journal*, 16(3):425–445.

DAMERAU, Fred J. (1964). A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the Association for Computing Machinery*, 7(3):171–176.

DANIELS, P. (2006). Mobile Media Projects for the Web. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 107, Granada: Eurocall.

DANLOS, Laurence et EL GHALI, Adil (2002). A complete integrated NLG system using AI and NLU tools. *In* TSENG, Shu-Chuan, CHEN, Tsuei-Er et LIU, Yi-Fen (Eds.). *COLING 2002: Proceedings of the 19th International Conference on Computational Linguistics*, vol. 1, pp. 211–217, Taipei, Taiwan: Howard International House.

DANLOS, Laurence, GAIFFE, Bertrand et ROUSSARIE, Laurent (2001). Document Structuring à la SDRT. *In Proceedings of the ACL 2001 Eight European Workshop on Natural Language Generation (EWNLG)*, Toulouse: Association for Computational Linguistics.

DANLOS, Laurence et ROUSSARIE, Laurent (2000). Génération automatique de textes. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 311–330. Paris: Hermès Science Publications.

DANNA, Frédéric (1997). *Modélisation de l'apprenant dans un logiciel d'Enseignement Intelligemment Assisté par Ordinateur. Application à un tutoriel intelligent dédié aux composés anglais.* Thèse de doctorat, Université de Rennes I.

DANNA, Frédéric et SÉBILLOT, Pascale (1997). A formalization of student modeling. *Computer Assisted Language Learning (CALL): An International Journal*, 10(2):121–147.

DANSUWAN, Suyada, NISHINA, Kikuko, AKAHORI, Kanji et SHIMIZU, Yasutaka (2001). Development and Evaluation of a Thai Learning System on the Web Using Natural Language Processing. *Calico Journal*, 19(1):67–88.

D'AOUST, Debbie (1990). Instructional Software for the Introductory Composition Classroom: One Teacher's Review and Recommandations. *Computers and Composition*, 7(3):109–127.

DAVIDSON, Natalia et ISENBERG, Florence (2005). Speech recognition software and University students of EFL. *In EUROCALL 2005 Conference. CALL, WELL and TELL: fostering autonomy*, p. 81, Kraków, Poland: Jagiellonian University.

DAVIES, Graham (2003). Computer Assisted Language Learning: Where are we now and where are we going? *futurelab.*

DAVIES, Graham (2006). Lessons from the past, lessons for the future: 20 years of CALL. Resource on the web.

DE FELICE, Rachel et PULMAN, Stephen (2009). Automatic detection of preposition errors in learner writing. *Calico Journal*, 26(3):512–528.

DE FELICE, Rachele et PULMAN, Stephen G. (2008). A Classifier-Based Approach to Preposition and Determiner Error Correction in L2 English. *In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 169–176, Manchester, UK: Coling 2008 Organizing Committee.

DE HAAN, Ab et OPPENHUIZEN, Tinus (1994). SPELLER: A Reflexive ITS to Support the Learning of Second Language Spelling. *Computers in Human Behavior*, 10(1):21–31.

DE HEER, T. (1982). The application of the concept of homeosemy to natural language information retrieval. *Information Processing and Management*, 18(5):229–236.

DE MARNEFFE, Marie-Catherine, MACCARTNEY, Bill et MANNING, Christopher D. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. *In* CALZOLARI, N. (Ed.). *Proceedings of the Language Resources and Evaluation LREC 2006*, Genoa: European Language Resources Association.

DEBSKI, Robert et LEVY, Mike (Eds.) (1999). *WORLDCALL: Global Perspectives on Computer-Assisted Language Learning*. Lisse, The Netherlands: Swets & Zeitlinger B. V.

DECROZANT, Lisa et VOSS, Clare R. (1999). Dual Use of Linguistic Resources: Evaluation of MT Systems and Language Learners. *In* BROMAN OLSEN, Mari (Ed.). *Computer Mediated Language Assessment and Evaluation in Natural Language Processing. Proceedings of a Symposium sponsored by the Association for Computational Linguistics and International Association of Language Learning Technologies*, pp. 32–37, College Park, Maryland USA: University of Maryland: Association for Computational Linguistics.

DEFAYS, Jean-Marc et DELTOUR, Sarah (2003). *Le français langue étrangère et seconde. Enseignement et apprentissage*. Liège: Mardaga.

DELCLOQUE, Philippe (1998). Design Methologies. *In* BLIN, Françoise, CHÉNIK, Nicole et THOMPSON, June (Eds.). *CALL Courseware Development: a Handbook*, pp. 1–13. Hull, UK: Eurocall.

DELMONTE, Rodolfo (2003). Linguistic Knowledge and Reasoning for Error Diagnosis and Feedback Generation. *Calico Journal*, 20(3):513–532.

DELMONTE, Rodolfo (2004a). Evaluating Students' Summaries with GETARUNS. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

DELMONTE, Rodolfo (2004b). Text Understanding with GETARUNS for Q/A and Summarization. *In* HIRST, Graeme et NIRENBURG, Sergei (Eds.). *ACL 2004: Second Workshop on Text Meaning and Interpretation*, pp. 97–104, Barcelona, Spain: Association for Computational Linguistics.

DELMONTE, Rodolfo et DIBATTISTA, Denise (2000). Parsing with GETARUNS. *In Actes de TALN 2000*, pp. 133–146, Lausanne: ATALA.

DEMAIZIÈRE, Françoise et DUBUISSON, Colette (1989). L'analyse des messages de l'apprenant en EAO. *Langue française*, 83:51–66.

DEMAIZIÈRE, Françoise et DUBUISSON, Colette (1992). *De l'EAO aux NTF: utiliser l'ordinateur pour la formation*. Paris: Ophrys.

DEMAIZIÈRE, Françoise (1982). An experiment in Computer Assisted Learning of English grammar at the University of Paris VII. *Computers & Education*, 6:121–125.

DENDIEN, Jacques et PIERREL, Jean-Marie (2003). Le Trésor de la Langue Française informatisé : un exemple d'informatisation d'un dictionnaire de langue de référence. *Traitement automatique des langues*, 44(2):11–37.

DESCLÈS, Jean-Pierre et MINEL, Jean-Luc (2000). Résumé automatique et filtrage sémantique des textes. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 253–270. Paris: Hermès Science Publications.

DESMARAIS, L. et BISAILLON, J. (1998). Apprentissage de l'écrit et ALAO. *Etudes de linguistique appliquée*, 110:193–202.

DESMEDT, William H. (1995). Herr Kommissar: An ICALL Conversation Simulator for Intermediate German. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 153–174. Mahwah, NJ: Lawrence Erlbaum Associates.

DESMET, Piet et HÉROGUEL, Armand (2005). Les enjeux de la création d'un environnement d'apprentissage électronique axé sur la compréhension orale à l'aide du système auteur IDIOMA-TIC. *Alsic*, 8.

DESMET, Piet, LOUWAGIE, Barbara et WYLIN, Bert (2004). Tous les exercices électroniques ne sont pas fermés et de niveau (faux) débutant! Bilan didactique et technique du système auteur IDIOMA-TIC. *In Actes d'UNTELE 2004*, Compiègne.

DESSUS, Philippe et LEMAIRE, Benoît (2002). Using Production to Assess Learning: An ILE That Fosters Self-Regulated Learning. *In* CERRI, Stefano A., GOUARDÈRES, Guy et PARAGUAÇU, Fábio (Eds.). *Intelligent Tutoring Systems: 6th International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain, June 2002. Proceedings*, pp. 772–781, Berlin: Springer.

DEVILLE, Guy et DUMORTIER, Laurence (2004). Evaluation d'un outil en ligne d'aide à la lecture de textes en langue étrangère. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 93–102, Grenoble: LIDILEM: ATALA - XRCE.

DÍAZ DE ILARRAZA, Arantza, GOJENOLA, Koldo et ORONOZ, Maite (2005). Design and Development of a System for the Detection of Agreement Errors in Basque. *In* GELBUKH, Alexander (Ed.). *Computational Linguistics and Intelligent Text Processing: 6th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2005 Mexico City, Mexico, February 13-19, 2005 Proceedings*, vol. 3406 de *Lecture Notes in Computer Science*, pp. 793–803, Berlin etc.: Springer Verlag.

DÍAZ DE ILARRAZA, Arantza, MARITXALAR, Montse et ORONOZ, Maite (1998). An Implemented Interlanguage Model for Learners of Basque. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 149–166. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

DICKINSON, Markus et HERRING, Joshua (2008). Developing Online ICALL Resources for Russian. *In Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 1–9, Columbus, Ohio: Association for Computational Linguistics.

DILLENBOURG, P., BAKER, M., BLAYE, A. et O'MALLEY, C. (1996). The evolution of Research on Collaborative Learning. *In* REIMANN, Peter et SPADA, Hans (Eds.). *Learning in Humans and Machines: Towards an interdisciplinary learning science*, p.? Oxford, Tarrytown: Elesevier Science Pergamon.

DILLENBOURG, Pierre (1989). The Design of a Self-Improving Tutor: PROTO-TEG. *Instructional Science*, 18(3):193–216.

DILLENBOURG, Pierre (1993). Multimédia et Formation : toujours les mêmes erreurs. *CBT Forum*, 2.

DILLENBOURG, Pierre (1994). Evolution épistémologique en EIAO. *Sciences et techniques éducatives*, 1(1):39–52.

DILLENBOURG, Pierre et SCHNEIDER, Daniel (1995). Mediating the mechanisms which make collaborative learning sometimes effective. *International Journal of Educational Telecommunications*, 1(2–3):131–146.

DIMITROVA, Vania, SELF, John et BRNA, Paul (2000). Maintaining a Jointly Constructed Student Model. *In* CERRI, Stefano A. et DOCHEV, Danail (Eds.). *Artificial intelligence : methodology, systems, and applications : 9th international conference, AIMSA 2000, Varna, Bulgaria, September 20-23, 2000 : proceedings*, pp. 221–231, Berlin: Springer.

DINA, L. et MALNATI, G. (1993). Weak constraints and preference rules. *In* BENNETT, P. et PAGGIO, P. (Eds.). *Preference in EUROTRA*. Luxembourg: European Commission.

DINNEMATIN, Seymour, SANZ, Didier et BONNET, Alain (1990). Sept correcteurs pour l'orthographe et la grammaire. *Science et Vie Micro*, (78):118–130.

DOBRIN, David N. (1990). A New Grammar Checker. *Computers and the Humanities*, 24(1–2):67–80.

DODIGOVIC, Marina (2005). *Artificial Intelligence in Second Language Learning: Raising Error Awareness*. N^o 13 *in* Second Language Acquisition. Clevedon UK: Multilingual Matters Ltd.

DOKTER, Duco et NERBONNE, John (1998). A Session with Glosser-RuG. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 88–94. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

DOLL, Frédéric et COULOMBE, Claude (2004). L'avenir des correcteurs grammaticaux: un point de vue industriel. *BULAG*, 29:33–50.

DONALDSON, Randall P. et KÖTTER, Markus (1999). Language Learning in Cyberspace: Teleporting the Classroom into the Target Culture. *Calico Journal*, 16(4):531–557.

DORR, Bonnie, HENDLER, James, BLANKSTEEN, Scott et MIGDALOFF, Barrie (1995). On Beyond Syntax: Use of Lexical Conceptual Structure for Intelligent Tutoring. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et

Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 289–309. Mahwah, NJ: Lawrence Erlbaum Associates.

Dorr, Bonnie J. (1997). Large-Scale Acquisition of LCS-Based Lexicons for Foreign Language Tutoring. *In Proceedings of the ACL Fifth Applied Natural Language Processing Conference*, pp. 139–146, Washington, DC.

Dorr, Bonnie Jean (1990). *Lexical Conceptual Structures and Machine Translation*. Thèse de doctorat, Massachussets Institute of Technology.

Dorr, Bonnie Jean (1994). Machine Translation Divergences: A Formal Description and Proposed Solution. *Computational Linguistics*, 20(4): 597–633.

Douglas, Sarah A. (1995). LingWorlds: An Intelligent Object-Oriented Environment for Second Language Tutoring. *In* Holland, V. Melissa, Kaplan, Jonathan D. et Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 201–220. Mahwah, NJ: Lawrence Erlbaum Associates.

Druard, Sophie (1993). Les correcteurs d'orthographe et de syntaxe. Mémoire de D.E.A. / Master, Université Libre de Bruxelles, Philosophie et Lettres, Bruxelles.

Duchiron, Emanuelle (2003). Atouts, limites et exploitations potentielles du choix fourni dans les logiciels de langue. *Alsic*, 8(3):5–17.

Dumais, Susan T., Furnas, George W., Landauer, Thomas K., Deerwester, Scott et Harshman, Richard (1988). Using Latent Semantic Analysis to improve access to textual information. *In Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington DC)*, pp. 281–285, New-York: ACM.

Durel, Patrick (2006). Utilisation de l'assistant grammatical Antidote dans le cadre d'activités de révision - Analyse exploratoire de protocoles d'observation. *Alsic*, 9.

Dutoit, Thierry, Couvreur, Laurent, Malfrère, Fabrice, Pagel, Vincent et Ris, Christophe (2002). Synthèse vocale et reconnaisance de la parole : Droites gauches et mondes parallèles. *In Actes du 6^e Congrès Français d'Acoustique*, Lille.

Dutoit, Thierry et Stylianou, Yannis (2003). Text-To-Speech synthesis. *In* Mitkov, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 323–338. Oxford: Oxford Univ. Press.

Earley, Jay (1970). An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 13(2):94–102.

Egan, Kathleen B. (1999). Speaking: a critical skill and a challenge. *Calico Journal*, 16(3):277–293.

Ehsani, Farzad et Knodt, Eva (1998). Speech Technology in Computer-Aided Language Learning: Strengths and Limitations of A New CALL Paradigm. *Language Learning & Technology*, 2(1):45–60.

ELMI, Mohammad Ali (1994). *A Natural Language Parser with Interleaved Spelling Correction Supporting Lexical Functional Grammar and Ill-Formed Input*. Thèse de doctorat, Illinois Institute of Technology.

ELMI, Mohammad Ali et EVENS, Martha (1998). Spelling Correction Using Context. *In Coling-ACL '98. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: Proceedings of the Conference*, vol. 1, pp. 360–364, Montréal, Canada: Université de Montréal.

EMIRKANIAN, Louisette et BOUCHARD, Lorne H. (1988). Knowledge integration in a robust and efficient morpho-syntactic analyzer for French. *In Coling Budapest: proceedings of the 12th International Conference on Computational Linguistics, 22-27 August 1988*, vol. 2, pp. 166–171, Budapest: J. von Neumann Society for Computing Science.

EMIRKANIAN, Louisette et BOUCHARD, Lorne H. (1989). La correction des erreurs d'orthographe d'usage dans un analyseur morpho-syntaxique du français. *Langue française*, 83:106–122.

ENGUEHARD, Chantal et MBODJ, Chérif (2004). Des correcteurs orthographiques pour les langues africaines. *BULAG*, 29:51–68.

ESKENAZI, Maxine (1999a). Using a Computer in Foreign Language Pronunciation Training: What Advantages? *Calico Journal*, 16(3):447–469.

ESKENAZI, Maxine (1999b). Using automatic speech processing for foreign language pronunciation tutoring : some issues and a prototype. *Language Learning & Technology*, 2(2):62–76.

ESLING, John H. (1992). Speech Technology Systems in Applied Linguistics Instruction. *In* PENNINGTON, Martha C. et STEVENS, Vance (Eds.). *Computers in applied linguistics: an international perspective*, vol. 75 de *Multilingual Matters*, pp. 244–272. Multilingual Matters.

ETCHEGOYHEN, Thierry (1997). Génération automatique de phrases : le système GBGen. Mémoire de diplôme d'etudes supérieures, Université de Genève, Faculté des Lettres, Genève.

ETCHEGOYHEN, Thierry (1998). Traitement des compléments du nom dans GBGen. Rapport technique 2, Université de Genève, Laboratoire d'Analyse et de Technologie du Langage, Genève.

ETCHEGOYHEN, Thierry, MENGON, Juri, VANDEVENTER, Anne et WEHRLE, Thomas (1999). Une approche efficace à la génération syntaxique mutlilingue : le système GBGen. *In Actes de GAT99: Génération Automatique de Textes, Deuxième Colloque Francophone*, pp. 135–144, Grenoble: Université Stendhal-Grenoble 3.

ETCHEGOYHEN, Thierry et WEHRLE, Thomas (1998). Overview of GBGen: A Large-Scale, Domain Independent Syntactic Generator. *In Proceedings of the 9th International Workshop on Natural Language Generation*, pp. 288–291, Niagara-on-the-Lake: Association for Computational Linguistics.

ETCHEGOYHEN, Thierry et WEHRLI, Eric (1998). Traduction automatique et structures d'interface. *In TALN 1998, Cinquième conférence annuelle sur le Traitement Automatique des Langues Naturelles: Actes*, pp. 2–11. Paris: ATALA.

EVENS, Martha W., CHANG, Ru-Charn, LEE, Yoon Hee, SHIM, Leem Seop, WOO, Chong Woo et ZBANG, Yuemei (1997). CIRCSIM-Tutor: An Intelligent Tutoring System Using Natural Language Dialogue. *In Fifth Conference on Applied Natural Language Processing: Descriptions of System Demonstrations and Videos*, pp. 13–14, Washington, DC, USA: Washington Marriott Hotel: Association for Computational Linguistics.

FAIRFIELD, John (1999). Speech comparison in The Rosetta Stone. *In* BROMAN OLSEN, Mari (Ed.). *Computer Mediated Language Assessment and Evaluation in Natural Language Processing. Proceedings of a Symposium sponsored by the Association for Computational Linguistics and International Association of Language Learning Technologies*, pp. 12–15, College Park, Maryland USA: University of Maryland: Association for Computational Linguistics.

FARRINGTON, Brian (1982). Computer based exercises for language learning at University level. *Computers & Education*, 6:113–116.

FARRINGTON, Brian (1994). Bon Accord revisited. *In* THOMPSON, June et CHESTERS, Graham (Eds.). *Emancipation through learning technology. Selected papers from the EUROCALL'93 conference, University of Hull, UK, 15-17 September 1993. A special double issue of Computers and Education, Vol 23, No. 1/2, 1994*, pp. 21–26, Oxford: EUROCALL: Pergamon.

FAULK, R. D. (1964). An inductive approach to language translation. *Communications of the ACM*, 7:647–653.

FELLBAUM, Christiane (Ed.) (1998). *WordNet, an electronic lexical database*. Cambridge, Mass.: The MIT Press.

FELLBAUM, Christiane et MILLER, George A. (2003). Morphosemantic Links in WordNet. *Traitement automatique des langues*, 44(2):69–80.

FELSHIN, Sue (1995). The Athena Language Learning Project NLP System: A Multilingual System for Conversation-Based Language Learning. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 257–272. Mahwah, NJ: Lawrence Erlbaum Associates.

FENG, Cheng, OGATA, Hiroaki et YANO, Yoneo (2000). Mark-up Based Writing Error Analysis Model in an On-line Classroom. *Computer Assisted Language Learning (CALL): An International Journal*, 13(1):79–97.

FEUERMAN, Ken, MARSHALL, Catherine, NEWMAN, David et RYPA, Marikka (1987). The CALLE Project. *Calico Journal*, 4:25–34.

FISCHER, William B. (1986). Master(')s Voice: The Victor 9000 and High-Fidelity Voice Reproduction for CALI. *Calico Journal*, 3(4):21–31.

FLOWERDEW, John (1996). Concordancing In Language Learning. *In* PENNINGTON, Martha C. (Ed.). *The Power of CALL*, pp. 97–113. Houston: Athelstan.

FLUHR, Christian (2000). Indexation et recherche d'information textuelle. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 235–251. Paris: Hermès Science Publications.

FONTENELLE, Thierry (2006). Les nouveaux outils de correction linguistique de Microsoft. *In* MERTENS, Piet, FAIRON, Cédrick, DISTER, Anne et WATRIN, Patrick (Eds.). *verbum ex machina. Actes de la 13^{e} Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 1, pp. 3–19, Leuven: Université Catholique de Louvain: UCL Presses.

FORBES-RILEY, Kate et LITMAN, Diane (2005). Correlating Student Acoustic-Prosodic Profiles with Student Learning in Spoken Tutoring Dialogues. *In Proceedings 9th European Conference on Speech Communication and Technology (Interspeech-2005/Eurospeech)*, Lisbon, Portugal.

FORBES-RILEY, Kate et LITMAN, Diane (2006). Modelling User Satisfaction and Student Learning in a Spoken Dialogue Tutoring System with Generic, Tutoring, and User Affect Parameters. *In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference (HTL-NAACL)*, pp. 264–271, New York City, USA: Association for Computational Linguistics.

FORBES-RILEY, Kate, LITMAN, Diane, SILLIMAN, Scott et TETREAULT, Joel (2006). Comparing Synthesized versus Pre-recorded Tutor Speech in an Intelligent Tutoring Spoken Dialogue System. *In Proceeding 19th International FLAIRS (Florida Artificial Intelligence Research Society) Conference*, Melbourne Beach, FL.

FOSTER, Jennifer (2004). Parsing Ungrammatical Input: An Evaluation Procedure. *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, vol. 6, pp. 2039–2042, Lisbon, Portugal: ELRA - European Language Resources Association.

FOTH, Kilian, MENZEL, Wolfgang, POP, Horia F. et SCHRÖDER, Ingo (2000). An Experiment On Incremental Analysis Using Robust Parsing Techniques. *In The 18th International Conference on Computational Linguistics: Coling in Europe. Proceedings of the Conference*, vol. 2, pp. 1026–1030, Saarbrücken, Germany: Universität des Saarlandes.

FOTH, Kilian A. et MENZEL, Wolfgang (2006). Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. *In COLING/ACL 06 : Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 321–328, Sydney, Australia: Association for Computational Linguistics.

FOUCOU, Pierre-Yves et KÜBLER, Natalie (1999). A web-based language learning environment : general architecture. *In* SCHULZE, Mathias, HA-

MEL, Marie-Josée et THOMPSON, June (Eds.). *Language Processing in CALL*, pp. 31–39, CCL UMIST, Manchester and EUROCALL: RECALL.

FOUCOU, Pierre-Yves et KÜBLER, Natalie (2000). A web-based environment for teaching technical English. *In* BURNARD, Lou et MCENERY, Tony (Eds.). *Rethinking language pedagogy from a corpus perspective : Papers from the Third International Conference on Teaching and Language Corpora*, vol. 2 de *Łódź Studies in Language*, pp. 65–73, Frankfurt: Peter Lang.

FREDERIKSEN, Carl H., DONIN, Janet et DÉCARY, Michel (1995). A Discourse Processing Approach to Computer-Assisted Language Learning. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 99–120. Mahwah, NJ: Lawrence Erlbaum Associates.

FREDERIKSEN, Carl H., DONIN, Janet, DÉCARY, Michel et HOOVER, Michael (1992). Semantic Discourse Processing and Tutoring Systems for Second Language Learning. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 103–121. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

FUCHS, Catherine, DANLOS, Laurence, LACHERET-DUJOUR, Anne, LUZZATI, Daniel et VICTORRI, Bernard (1993). *Linguistique et Traitement Automatique des Langues*. Hachette Université : Langue Linguistique Communication. Paris: Hachette Supérieur.

FUM, Danilo, PANI, Bruno et TASSO, Carlo (1992). Naive vs. Formal Grammars: A Case foir Integration in the Design of a Foreign Langage Tutor. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 51–64. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

GAGNON, Michel et DA SYLVA, Lyne (2005). Text Summarization by Sentence Extraction and Syntactic Pruning. *In Proceedings of CliNE'05: 3rd Workshop on Computational Linguistics in the North East*, Gatineau: Université du Québec en Outaouais.

GAIFFE, Bertrand (2006). Références. *In* SABAH, Gérard (Ed.). *Compréhension des langues et interaction*, Cognition et traitement de l'information, pp. 171–193. Paris: Hermes Science, Lavoisier.

GAMON, Michael, LEACOCK, Claudia, BROCKETT, Chris, DOLAN, William B., GAO, Jianfeng, BELENKO, Dmitriy et KLEMENTIEV, Alexandre (2009). Using Statistical Techniques and Web Search to Correct ESL Errors. *Calico Journal*, 26(3):491–511.

GANASCIA, Jean-Gabriel (2001). Extraction automatique de motifs syntaxiques. *In Actes de TALN 2001 et Récital 2001, Tours du 2 au 5 juillet 2001*, vol. 1, pp. 193–202, Tours: ATALA.

GARMAN, Joe, MARTIN, Jeffery, MERLO, Paola et WEINBERG, Amy (1993). A Principle-based Parser for Foreign Language Training in German and Arabic. *In Third International Workshop on Parsing Technologies. IWPT3*, pp. 73–87, Tilburg, NL & Durbuy BE: Association for Computational Linguistics Special Interest Group on Parsing.

GARRETT, Nina (1995). ICALL and Second Language Acquisition. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 345–358. Mahwah, NJ: Lawrence Erlbaum Associates.

GARSIDE, Roger (1987). The CLAWS word-tagging system. *In* GARSIDE, Roger, LEECH, Geoffrey et SAMPSON, Geoffrey (Eds.). *The computational analysis of English: a corpus-based approach*, pp. 30–41. London & New York: Longman.

GAUDINAT, Arnaud et GOLDMAN, Jean-Philippe (1998). Le système de synthèse FipsVox : syntaxe, phonétisation et prosodie. *In Actes des XXII^e Journées d'études sur la parole*, pp. 139–142, Martigny: IDIAP.

GAUDINAT, Arnaud et WEHRLI, Eric (1997). Analyse syntaxique et synthèse de la parole : le projet *FipsVox*. *Traitement automatique des langues*, 38 (1):121–134.

GAZDAR, Gerald, KLEIN, E., PULLUM, G. et SAG, Ivan (1985). *Generalized Phrase Structure Grammar*. Oxford: Blackwell.

GAZDAR, Gerald et MELLISH, Christopher S. (1989). *Natural language processing in LISP : an introduction to computational linguistics*. Workingham, etc.: Addison-Wesley.

GENTHIAL, Damien, COURTIN, Jacques et MÉNÉZO, Jacques (1994). Towards a More User-friendly Correction. *In Proceedings of the 15^th International Conference on Computational Linguistics (COLING '94)*, pp. 1083–1088, Kyoto, Japan.

GIBBON, F. E., HARDCASTLE, W. J. et SUZUKI, H. (1991). An electropalatographic study of the /r/, /l/ distinction for Japanese learners of English. *Computer Assisted Language Learning (CALL): An International Journal*, 4(3):153–171.

GIRARD, Marie-Christine et VOCE, Julie (2003). Recommandations pour l'implémentation de feedbacks dans le système de diagnostic d'erreurs de UGEN. Rapport technique, UMIST, Manchester.

GLENCROSS, Michael (1993). Grammar and CALL: a review of current practice in some French language programs. *ReCALL*, 8:21–26.

GLENCROSS, Michael (1995). Using the Robert Electronique as a language learning resource. *ReCALL*, 7(1):54–58.

GODWIN-JONES, Robert (2003). Blogs and Wikis: Environments for On-line collaboration. *Language Learning & Technology*, 7(2):12–16.

GODWIN-JONES, Robert (2004a). Language in Action: From Webquests to Virtual Realities. *Language Learning & Technology*, 8(3):9–14.

GODWIN-JONES, Robert (2004b). Learning Objects: Scorn or SCORM? *Language Learning & Technology*, 8(2):7–12.

GODWIN-JONES, Robert (2005). Skype and Podcasting: Disruptive Technologies for Language Learning. *Language Learning & Technology*, 9(3):9–12.

GOLDING, Andrew R. et SCHABES, Yves (1996). Combining Trigram-based and Feature-based Methods for Context-Sensitive Spelling Correction. *In 34th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 71–78, Santa Cruz, CA: University of California, Santa Cruz: Association for Computational Linguistics.

GOLDMAN, Jean-Philippe (2001). De l'analyse syntaxique à la synthèse de la parole dans le système FipsVox : phonétisation et génération de la prosodie. *Cahiers de Linguistique Française*, 23:211–234.

GOLDMAN, Jean-Philippe, GAUDINAT, Arnaud et WEHRLI, Eric (2000). Utilisation de l'analyse syntaxique pour la synthèse de la parole, l'enseignement des langues et l'étiquetage grammatical. *In Actes de TALN 2000*, pp. 427–430, Lausanne: ATALA.

GOODFELLOW, Robin, LAMY, Marie-Noëlle et JONES, Glyn (2001). Assessing learners' writing using lexical frequency. *ReCALL*, 14(1):133–145.

GRANGER, Richard H. (1983). The NOMAD System: Expectations-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Text. *American Journal of Computational Linguistics*, 9(3–4):188–196.

GRANGER, Sylviane (2003). Error-tagged Learner Corpora and CALL: A Promising Synergy. *Calico Journal*, 20(3):465–480.

GRANGER, Sylviane, VANDEVENTER, Anne et HAMEL, Marie-Josée (2001). Analyse de corpus d'apprenants pour l'ELAO basée sur le TAL. *Traitement automatique des langues*, 42(2):609–621.

GREENE, Cara, KEOGH, Katrina, KOLLER, Thomas, WAGNER, Joachim, WARD, Monica et VAN GENABITH, Josef (2004). Using NLP Technology in CALL. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

GROBET, Anne et SIMON, Anne-Catherine (2001). Différents critères de définition des unités prosodiques maximales. *Cahiers de Linguistique Française*, 23:143–163.

GROSS, Maurice (1975). *Méthodes en syntaxe*. Paris: Hermann.

GROVER, Claire, MATHESON, Colin, MIKHEEV, Andrei et MOENS, Marc (2000). LT TTT - A Flexible Tokenisation Tool. *In* GAVRILIDOU, Maria, CARAYANNIS, George, MARKANTONATOU, Stella, PIPERIDIS, Stelios

et STAINHAOUER, Gregory (Eds.). *Second International Conference on Language Resources and Evaluation. Proceedings of LREC 2000*, Athens: ELRA - European Language Resources Association.

GUREVICH, Olga et DEANE, Paul (2007). Document similarity measures to distinguish native vs. non-native essay writers. *In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pp. 49–52, Rochester, New York: Association for Computational Linguistics.

GUYOMARD, Marc, SIROUX, Jacques, PERNICI, Dominique et ROYER, Christophe (1997). A General Public Application of Pedagogic and Linguistic Vocations of Speech Synthesis: Ordictée. *In Proceedings of the 10th Research on Computational Linguistics International Conference ROCLING97*, pp. 316–324.

HABERT, Benoît (2006). TAL sur corpus: histoire, acquis, défis. *In* SABAH, Gérard (Ed.). *Compréhension des langues et interaction*, Cognition et traitement de l'information, pp. 249–275. Paris: Hermes Science, Lavoisier.

HACKENBERG, Robert G. (1985). Generate : A Natural Language Generator. *Calico*, 2(2):5–8.

HAGEN, L. Kirk (1995). Unification-based parsing applications for intelligent foreign language tutoring systems. *Calico Journal*, 12(2&3):5–31.

HALLIDAY, Michael Alexander Kirkwood (1985). *An Introduction to Functional Grammar*. London: Edward Arnold.

HAMBUGER, Henry, SCHOELLES, Michael et REEDER, Florence (1999). More Intelligent CALL. *In* CAMERON, Keith (Ed.). *Computer-Assisted Language Learning (CALL): Media, Design and Application*, pp. 183–202. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

HAMBURGER, Henry (1995). Tutorial Tools for Language Learning by Two-Medium Dialogue. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 183–199. Mahwah, NJ: Lawrence Erlbaum Associates.

HAMBURGER, Henry et HASHIM, Raza (1992). Foreign Language Tutoring and Learning Environment. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 201–218. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

HAMEL, Marie-Josée (1996). NLP Tools in CALL for error diagnosis. *Computer Assisted Language Learning (CALL): An International Journal*, 18 (2):125–142.

HAMEL, Marie-Josée (1998). Les outils de TALN dans SAFRAN. *ReCALL*, 10(1):79–85.

HAMEL, Marie-Josée et GIRARD, Marie-Christine (2004). Une exploitation pédagogique de la typologie des textes en ALAO. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 103–112, Grenoble: LIDILEM: ATALA - XRCE.

HAMEL, Marie-Josée, NKWENTI-AZEH, Blaise et ZÄHNER, Christoph (1995). The Conceptual Dictionary in CALL. *In* GIMENO, Ana (Ed.). *Proceedings of EUROCALL 95': Technology Enhanced Language Learning: Focus On Integration*, pp. 329–349, Valencia: Universidad Politécnica de Valencia.

HAMEL, Marie-Josée et VANDEVENTER, Anne (1998). FipsGram: un analyseur syntaxique dans un contexte d'ELAO. *In Le traitement automatique du langage et les applications industrielles - accent sur l'enseignement de la langue - Acte de TAL + AI 98*, pp. 18–24, Moncton, Canada: GRÉTAL: Groupe d'étude sur le traitement automatique des langues.

HAMEL, Marie-Josée et VANDEVENTER, Anne (2000). Adapter un analyseur syntaxique et l'intégrer dans un système d'ELAO : le cas *FipsGram* dans SAFRAN. *In* DUQUETTE, Lise et LAURIER, Michel (Eds.). *Apprendre une langue dans un environnement multimédia*, Théories et pratiques dans l'enseignement, pp. 117–136. Outremont: Ed. Logiques.

HAMEL, Marie-Josée et WEHRLI, Eric (1997). Outils de TALN en EIAO: le projet SAFRAN. *In Journées Scientifiques et Techniques du Réseau FRANCophone de l'Ingénierie de la Langue de l'Aupelf-Uref (JST'97)*, pp. 277–282, Avignon, France: Aupelf-Uref.

HAMMING, Richard W. (1950). Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2):147–160.

HAN, Na-Rae, CHODOROW, Martin et LEACOCK, Claudia (2004). Detecting Errors in English Article Usage with a Maximum Entropy Classifier Trained on a Large, Diverse Corpus. *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, vol. 5, pp. 1625–1628, Lisbon, Portugal: ELRA - European Language Resources Association.

HANA, Jirka, FELDMAN, Anna, AMARAL, Luiz et BREW, Chris (2006). Tagging Portuguese with a Spanish Tagger. *In* INKPEN, Diana, STRAPPARAVA, Carlo et AGIRRE, Eneko (Eds.). *Cross-Language Knowledge Induction Workshop, International Workshop held as part of EACL 2006, The 11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 33–40, Trento, Italy: Association for Computational Linguistics.

HANDKE, Jürgen (1992). WIZDOM: A Multiple-Purpose Language Tutoring System Based on AI Techniques. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced

Institute Series, pp. 293–305. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

Handley, Zöe et Hamel, Marie-Josée (2005). Establishing a methodology for benchmarking speech synthesis for Computer-Assisted Language Learning (CALL). *Language Learning & Technology*, 9(3):99–119.

Hannahs, S. J. (2007). French phonology and L2 acquisition. *In* Ayoun, Dalila (Ed.). *French Applied Linguistics*, vol. 16 de *Language Teaching & Language Learning*, pp. 50–74. Amsterdam, Philadelphia: John Benjamins.

Hanrieder, G. (1994). MORPH - Ein modulares und robustes Morphologieprogramm für das Deutsche in Common Lisp. *LDV-Forum. Zeitschrift für Computerlinguistik und Sprachtechnologie. Journal for Computational Linguistics and Language Technology*, 11(1):30–38.

Harashima, Hideto D. (2006). Integrating Text-To-Speech Conversion Technology with Moodle. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 128, Granada: Eurocall.

Harless, William G., Zier, Marcia A. et Duncan, Robert C. (1999). Virtual dialogues with native speakers: the evaluation of an interactive multimedia method. *Calico Journal*, 16(3):313–337.

Harroff, Susan (1986). Die Sprachmaschine: A Microworld for Language Experimentation. *Calico Journal*, 3:32–34.

Hart, R. S. (1995). The Illinois PLATO foreign language project. *Calico Journal*, 12(4):15–37.

Harvey, Brian (1986). *Computer science Logo style.* Cambridge, Mass: MIT Press.

Hayes, Philip J. et Mouradian, George V. (1981). Flexible Parsing. *American Journal of Computational Linguistics*, 7(4):232–242.

Heidorn, G. E. (1972). *Natural language inputs to a simulation programming system.* Ph.d. diss., Yale University.

Heidorn, G. E., Jensen, K., Miller, L. A., Bird, R. J. et Chodorow, M. S. (1982). The EPISTLE text-critiquing system. *IBM Systems Journal*, 21(3):305–326.

Heidorn, George E. (1975). Augmented Phrase Structure Grammars. *In* Schank, Roger et Nash-Webber, Bonnie (Eds.). *Theoretical Issues in Natural Language Processing: An Interdisciplinary Workshop in Computational Lingustics, Psychology, Linguistics, Artifical Intelligence*, pp. 1–5, Cambridge, Mass.

Heidorn, George E. (2000). Intelligent Writing Assistance. *In* Dale, Robert, Moisl, Hermann et Somers, Harold (Eds.). *Handbook of Natural Language Processing*, pp. 181–207. New York, Basel: Marcel Dekker, Inc.

Heift, Trude (2001). Error-specific and individualised feedback in a Web-based language tutoring system: Do they read it? *ReCALL*, 13(1):99–109.

HEIFT, Trude (2002). Learner Control and Error Correction in ICALL: Browsers, Peekers, and Adamants. *Calico Journal*, 19(2):295–313.

HEIFT, Trude (2003). Multiple Learners Errors and Meaningful Feedback: A Challenge for ICALL Systems. *Calico Journal*, 20(3):513–532.

HEIFT, Trude (2004). Inspectable Learner Models for Web-based Instruction. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 124, Vienna: EUROCALL.

HEIFT, Trude et MCFETRIDGE, Paul (1999). Exploiting the student model to emphasize language pedagogy in natural language processing. *In* BROMAN OLSEN, Mari (Ed.). *Computer Mediated Language Assessment and Evaluation in Natural Language Processing. Proceedings of a Symposium sponsored by the Association for Computational Linguistics and International Association of Language Learning Technologies*, pp. 55–61, College Park, Maryland USA: University of Maryland: Association for Computational Linguistics.

HEIFT, Trude et NICHOLSON, Devlan (2001). Web Delivery of Adaptive and Interactive Language Tutoring. *International Journal of Artificial Intelligence in Education*, 12:310–324.

HEIFT, Trude et SCHULZE, Mathias (2005). Intelligent CALL – Back to the Future. *In EUROCALL 2005 Conference. CALL, WELL and TELL: fostering autonomy*, p. 110, Kraków, Poland: Jagiellonian University.

HEIFT, Trude et SCHULZE, Mathias (2007). *Errors and Intelligence in Computer-Assisted Language Learning. Parsers and Pedagogues.* New York, Oxnon: Routledge.

HEIFT, Trude, TOOLE, Janine, MCFETRIDGE, Paul, POPOWICH, Fred et TSIPLAKOU, Stavroula (2000). Learning Greek with an Adaptive and Intelligent Hypermedia System. *IMEj: Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 2(2).

HEILMAN, Michael, COLLINS-THOMPSON, Kevyn, CALLAN, Jamie et ESKENAZI, Maxine (2007). Combining lexical and grammatical features to improve readability measures for first and second language texts. *In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 460–467, Rochester, New York: Association for Computational Linguistics.

HEINECKE, Johannes, KUNZE, Jürgen, MENZEL, Wolfgang et SCHRÖDER, Ingo (1998). Eliminative Parsing with Graded Constraints. *In Coling-ACL '98. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: Proceedings of the Conference*, vol. 1, pp. 526–530, Montréal, Canada: Université de Montréal.

HELM, Carl E. et MCIVER, Charlotte (1974). Automated Sentence Analysis for Language Instruction. *Computers and the Humanities*, 8:239–2454.

HENDRICKS, Harold, BENNION, Junius L. et LARSON, Jerry (1983). Technology and Language Learning at BYU. *Calico*, 1(3):23–30.

HENDRIX, Gary G., SACERDOTI, Earl D., SAGALOWICZ, Daniel et SLOCUM, Jonathan (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, 3(2):105–147.

HERMET, Matthieu et ALAIN, Désilets (2009). Using first and second language models to correct preposition errors in second language authoring. *In Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 64–72, Boulder, Colorado: Association for Computational Linguistics.

HERMET, Matthieu, SZPAKOWICZ, Stan et DUQUETTE, Lise (2006). Automated Analysis of Students' Free-text Answers for Computer-Assisted Assessment. *In* MERTENS, Piet, FAIRON, Cédrick, DISTER, Anne et WATRIN, Patrick (Eds.). *verbum ex machina. Actes de la 13e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 2, pp. 835–845, Leuven: Université Catholique de Louvain: UCL Presses.

HILLER, Steven, ROONEY, Edmund, VAUGHAN, Rebecca, ECKERT, Miriam, LAVER, John et JACK, Mervyn (1994). An automated system for computer-aided pronunciation learning. *Computer Assisted Language Learning (CALL): An International Journal*, 7(1):51–63.

HINCKS, Rebecca (2003). Speech technologies for pronunciation feedback and evaluation. *ReCALL*, 15(1):3–20.

HIRST, Graeme (1987). *Semantic Interpretation and the resolution of ambiguity*. Studies in Natural Language Processing. Cambridge: Cambridge University Press.

HIRST, Graeme et BUDANITSKY, Alexander (2005). Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111.

HIRST, Graeme et ST-ONGE, David (1998). Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. *In* FELLBAUM, Christiane (Ed.). *WordNet, an electronic lexical database*, pp. 305–332. Cambridge, Mass.: The MIT Press.

HOLAN, Tomáš, KUBOŇ, Vladislav et PLÁTEK, Martin (1997). A Prototype of a Grammar Checker for Czech. *In Fifth Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 147–154, Washington Marriott Hotel, Washington, DC, USA: Association for Computational Linguistics.

HOLDICH, C. E., CHUNG, P. W. H. et HOLDICH, R. G. (2004). Improving children's written grammar and style: revising anf editing with HARRY. *Computers & Education*, 42(1):1–23.

HOLLAND, V. Melissa (1994). Lessons Learned in Designing Intelligent CALL: Managing Communication Across Disciplines. *Computer Assisted Language Learning (CALL): An International Journal*, 7(3):227–256.

HOLLAND, V. Melissa, KAPLAN, J. D. et SABOL, M. A. (1999). Preliminary tests of language learning in a speech-interactive graphics microworld. *Calico Journal*, 16(3):339–360.

HOLLAND, V. Melissa, MAISANO, Richard, ALDERKS, Cathie et MARTIN, Jeffery (1993). Parsers in Tutors: What Are They Good For? *Calico Journal*, 11(1):28–46.

HOLMES, Glyn et KIDD, Marilyn (1980). The Evolving Case for Computers in the Study of Modern Languages. *ALLC Journal*, 1:7–10.

HORTON, Jane, ELLIS, David et BLACK, Philip (1990). The design and development of an Intelligent Tutoring System for adult literacy students. *Computer Assisted Language Learning (CALL): An International Journal*, 2:69–81.

HOUSER, C. (2006). How to make educational web sites for mobile phones. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 133, Granada: Eurocall.

HOUSER, C. et THORNTON, P. (2006). Using Mobile Phones for Foreign Language Education. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 134, Granada: Eurocall.

HOVY, Eduard (2003). Text summarization. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 583–598. Oxford: Oxford Univ. Press.

HOVY, Eduard H. (1988). Planning coherent multisentential texts. *In 26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 179–186, Buffalo, NY: State University of New York at Buffalo: Association for Computational Linguistics.

HU, Qian, HOPKINS, Jeff et PHINEY, Marianne (1998). Native English Writing Assistant – A CALL Product fot English Reading and Writing. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 95–100. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

HUANG, Xuedong, ALLEVA, Fileno, HON, Hsiao-Wuen, HWANG, Mei-Yuh et ROSENFELD, Ronald (1993). The SPHINX-II speech recognition system: an overview. *Computer Speech and Language*, 7(2):137–148.

HUBBARD, Philip (2004). Using Google as a tool for writing instruction. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 133, Vienna: EUROCALL.

HUBBARD, Philip et LEVY, Mike (Eds.) (2006). *Teacher Education in CALL*. N^o 14 *in* Language Learning & Language Teaching. Amsterdam / Philadelphia: John Benjamins.

HUTCHINS, John (2003). Machine Translation: general overview. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 501–511. Oxford: Oxford Univ. Press.

IMLAH, G. et DU BOULAY, J. B. H. (1985). Robust natural language parsing in computer-assisted language instruction. *System: an international journal of educational technology and applied linguistics*, 13(2):137–147.

INGRAHAM, Bruce, CHANIER, Thierry et EMERY, Chris (1994). Language training for various purposes in several languages on a common hypermedia framework. *In* THOMPSON, June et CHESTERS, Graham (Eds.). *Emancipation through learning technology. Selected papers from the EUROCALL'93 conference, University of Hull, UK, 15-17 September 1993. A special double issue of Computers and Education, Vol 23, No. 1/2, 1994*, pp. 107–115, Oxford: EUROCALL: Pergamon.

ISHIOKA, Tsunenori et KAMEDA, Masayuki (2006). Automated japanese essay scoring system based on articles written by experts. *In COLING/ACL 06 : Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 233–240, Sydney, Australia: Association for Computational Linguistics.

ISSAC, Fabrice (1997). *Analyse syntaxique et apprentissage des langues*. Thèse de doctorat, Paris Nord, Paris.

ISSAC, Fabrice et HÛ, Olivier (2002). Formalism for Evaluation: Feedback on Learner Knowledge Representation. *Computer Assisted Language Learning (CALL): An International Journal*, 15(2):183–199.

IZUMI, Emi, UCHIMOTO, Kiyotaka, SAIGA, Toyomi, SUPNITHI, Thepchai et ISAHARA, Hitoshi (2003). Automatic error detection in the Japanese learners' English spoken data. *In ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pp. 145–148, Morristown, NJ, USA: Association for Computational Linguistics.

JACKENDOFF, Ray (1975). A System of Semantic Primitives. *In Theoretical Issues in Natural Language Processing*, pp. 24–29. Association for Computational Linguistics.

JACKENDOFF, Ray S. (1983). *Semantics and Cognition*. Cambridge MA, USA: MIT Press.

JACKENDOFF, Ray S. (1990). *Semantic Structures*. Cambridge MA, USA: MIT Press.

JAFFRÉ, Jean-Pierre (1992). *Didactiques de l'orthographe*. Pédagogies pour demain. Didactiques 1er degré. Paris: INRP Hachette.

JAGER, Sake (2001). From gap-filling to filling the gap: A re-assessment of Natural Language Processing in CALL. *In* CHAMBERS, Angela et DAVIES, Graham (Eds.). *ICT and language learning : a European perspective*, Language learning and language technology, pp. 101–110. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

JAMES, Carl (1998). *Errors in Language Learning and Use: Exploring Error Analysis*. London and New York: Longman.

JAMIESON, Joan, CHAPELLE, Carol A. et PREISS, Sherry (2004). Putting principles into practice. *ReCALL*, 16(2):396–415.

JEHLE, Fred (1987). A Free-Form Dialog Program in Spanish. *Calico Journal*, 5(2):11–22.

JEKER, Florence (1992). An Error-Detection System for Grammar Checkers. Manuscript, Institut d'Informatique et d'Intelligence Artificielle, Université de Neuchâtel.

JENSEN, Karen, HEIDORN, George, MILLER, Lance et RAVIN, Yael (1993). Prose fitting and prose fixing. *In* JENSEN, Karen, HEIDORN, George E. et RICHARDSON, Stephen D. (Eds.). *Natural language processing: the PLNLP approach*, pp. 53–64. Dordrecht: Kluwer ed.

JENSEN, Karen, HEIDORN, George E., MILLER, Lance A. et RAVIN, Yael (1983). Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness. *American Journal of Computational Linguistics*, 9(3–4):147–160.

JI, Hyungsuk et PLOUX, Sabine (2003). Automatic Contexonym Organizing Model (ACOM). *In Proceedings of the 25th Annual Conference of the Cognitive Science Society*, pp. 622–627.

JI, Hyungsuk, PLOUX, Sabine et WEHRLI, Eric (2003). Lexical Knowledge Representation with Contexonyms. *In Proceedings of the 9th Machine Translation Summit*, pp. 194–201.

JONES, Michael P. et MARTIN, James H. (1997). Contextual Spelling Correction Using Latent Semantic Analysis. *In Fifth Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 166–173, Washington Marriott Hotel, Washington, DC, USA: Association for Computational Linguistics.

JONES, Randall L. (1995). TICCIT and CLIPS: The Early Years. *Calico Journal*, 12(4):84–96.

JOSCELYNE, Andrew (1994). La Correcteur. *Language Industry Monitor*.

JURAFSKY, Daniel et MARTIN, James H. (2000). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall series in artificial intelligence. Upper Saddle River N.J: Prentice Hall, cop.

KAHANE, Sylvain (2001). Grammaires de dépendance formelles et théorie sens-texte. *In Actes de TALN 2001 et Récital 2001, Tours du 2 au 5 juillet 2001*, vol. 2, pp. 17–76, Tours: ATALA.

KAMP, Hans (1981). Evénements, représentations discursive et référence temporelle. *Langages*, 64:34–64.

KANG, Y. S. et MACIEJEWSKI, A. A. (2000). A student model of technical Japanese reading proficiency for an intelligent tutoring system. *Calico Journal*, 18(1):9–40.

KAPLAN, Jonathan D. et HOLLAND, V. Melissa (1995). Application of Learning Principles to the Design of a Second Language Tutor. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 273–287. Mahwah, NJ: Lawrence Erlbaum Associates.

KAPLAN, Jonathan D., SABOL, Mark A., WISHER, Robert A. et SEIDEL, Robert J. (1998). The Military Language Tutor (MILT) Program: A Advanced Authoring System. *Computer Assisted Language Learning (CALL): An International Journal*, 11(3):267–287.

KAPLAN, Ronald M. (2003). Syntax. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 70–90. Oxford: Oxford Univ. Press.

KAPLAN, Ronald M. et BRESNAN, Joan (1982). LFG: a formal system for grammatical representation. *In* BRESNAN, Joan (Ed.). *The Mental Representation of grammatical relations*. Cambridge MA, USA: MIT Press.

KARLSSON, Fred (1990). Constraint Grammar as a framework for parsing running text. *In* KARLGREN, Hans (Ed.). *Coling-90: papers presented to the 13th International Conference on Computational Linguistics, on the occasion of the 25th anniversary of Coling and the 350th anniversary of Helsinki University*, pp. 168–173, Helsinki: Universitas Helsigiensis.

KAY, Martin (1984). Functional Unification Grammar: A Formalism for Machine Translation. *In 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics: Proceedings of Coling-ACL 84*, pp. 75–78, Stanford University, CA: Association for Computational Linguistics.

KAY, Martin (1985). Parsing in functional unification grammar. *In* DOWTY, David R., KARTTUNEN, Lauri et ZWICKY, Arnold M. (Eds.). *Natural Language Parsing: Psychological, computational, and theoretical perspectives*, Studies in Natural Language Processing, pp. 251–278. Cambridge: Cambridge University Press.

KAY, Martin (1992). Unification. *In* ROSNER, Michael et JOHNSON, Roderick (Eds.). *Computational linguistics and formal semantics*, pp. 1–29. Cambridge, UK: Cambridge University Press.

KELLER, Frank (2000). *Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality*. Thèse de doctorat, University of Edinburgh.

KEMPEN, Gerard (1992). Language Technology and Language Instruction: Computational Diagnosis of Word Level Errors. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 191–198. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

KEMPEN, Gerard (1999). Visual Grammar: Multimedia for Grammar and Spelling Instruction in Primary Education. *In* CAMERON, Keith (Ed.). *Computer-Assisted Language Learning (CALL): Media, Design and Application*, pp. 223–238. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

KEMPEN, Gerard (2004). Interactive visualization of syntactic structure assembly for grammar-intensive first- and second-language instruction. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

KEOGH, Katrina, KOLLER, Thomas, WARD, Monica, UÍ DHONNCHADHA, Elaine et VAN GENABITH, Josef (2004). CL for CALL in the Primary School. *In* LEMNITZER, Lothar, MEURERS, Detmar et HINRICHS, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 53–60, Geneva, Switzerland: COLING.

KIRSCHNING, Ingrid (2004). CSLU Toolkit-based Vocabulary Tutors for the Jean Piaget Special Education School. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

KLEIN, Philipp (1998). El Corrector (Version 1.5). *Calico Journal*, 16 (1):64–75.

KLENNER, Manfred (2004). Tutorial Dialogue in DiBEx. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

KNILL, K. et YOUNG, S. (1997). Hidden Markov Models in Speech and Language Processing. *In* YOUNG, Steve et BLOOTHOOFT, Gerrit (Eds.). *Corpus-based methods in language and speech processing*, vol. 2 de *Text, speech and language technology*, pp. 27–68. Dordrecht: Kluwer.

KNUTH, D. E. (1965). On the translation of languages from left to right. *Information and Control*, 8:607–639.

KNUTSSON, Ola, BIGERT, Johnny et KANN, Viggo (2003a). A robust shallow parser for Swedish. *In Proceedings of the 14th Nordic Conference on Computational Linguistics (NoDaLiDa)*, Reykjavik, Iceland.

KNUTSSON, Ola, CERRATO PARGMAN, Teresa et SEVERINSON EKLUNDH, Kerstin (2003b). Transforming Grammar Checking Technology into a Learning Environment for Second Language Writing. *In* BURSTEIN, Jill et LEACOCK, Claudia (Eds.). *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pp. 38–45.

KNUTSSON, Ola, CERRATTO PARGMAN, Teresa, SEVERINSON EKLUNDH, Kerstin et WESTLUND, Stefan (2007). Designing and developing a lan-

guage environment for second language writers. *Computers & Education*, 49(4):1122–1146.

KNUTSSON, Ola, PARGMAN, Teresa Cerratto et EKLUNDH, Kerstin Severinson (2002). Computer support for second language learners' free text production – Initial studies. *European Journal of Open, Distance and E-Learning (EURODL)*.

KNUTSSON, Ola, PARGMAN, Teresa Cerratto et EKLUNDH, Kerstin Severinson (2003c). Transforming grammar checking technology into a learning environment for second language writing. *In Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*, pp. 38–45, Morristown, NJ, USA: Association for Computational Linguistics.

KOLLER, Thomas (2003). Knowledge-based intelligent error feedback in a Spanish ICALL system. *In Proceedings of The 14th Irish Conference on Artificial Intelligence & Cognitive Science*, pp. 117–121, Dublin: Trinity College.

KOLLER, Thomas (2004). Creation and evaluation of animated grammars. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 153, Vienna: EUROCALL.

KOLLER, Thomas (2005). Web-based plurilingual learning software for French. *In EUROCALL 2005 Conference. CALL, WELL and TELL: fostering autonomy*, p. 143, Kraków, Poland: Jagiellonian University.

KOSKENNIEMI, Kimmo (1994). A General Computational Model for Word-form Recognition and Production. *In Proceedings of the 15th International Conference on Computational Linguistics (COLING '94)*, pp. 178–181, Kyoto, Japan.

KRAIF, Olivier, ANTONIADIS, Georges, ECHINARD, Sandra, LOISEAU, Mathieu, LEBARBÉ, Thomas et PONTON, Claude (2004). NLP Tools for CALL: the Simpler the Better. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

KRAIF, Olivier et PONTON, Claude (2007). Du bruit, du silence et des ambiguïtés : que faire du TAL pour l'apprentissage des langues ? *In* BENAMARA, Farah, HATHOUT, Nabil, MULLER, Philippe et OZDOWSKA, Sylwia (Eds.). *Actes de la 14^{e} conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^{e} Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 2 de *Actes TALN et RECITAL*, pp. 143–152, Toulouse: Association pour le Traitement Automatique des Langues.

KRAMSCH, Claire, MORGENSTERN, Douglas et MURRAY, Janet H. (1985). An Overview of the MIT Athena Language Learning Project. *Calico*, 2 (4):31–34.

KREYER, Steeve et CRISWELL, Eleanor (1995). Instructor as Author in an Adaptive, Multimedia, Foreign Language Tutor. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 45–54. Mahwah, NJ: Lawrence Erlbaum Associates.

KRÜGER, Anja et HAMILTON, Simon (1997). RECALL: individual language tutoring through intelligent error diagnosis. *ReCALL*, 9(2):51–58.

KÜBLER, Natalie (1995). *L'automatisation de la correction d'erreurs syntaxiques: application aux verbes de transfert en anglais pour francophones.* Thèse de doctorat, Paris VII - Denis Diderot, Paris.

KÜBLER, Natalie et CORNU, Etienne (1992). How can errors be detected and corrected? *In* GROSJEAN, François et BLACHE, Philippe (Eds.). *Workshop on Natural language processing: First and second language correction of written texts. Proceedings of SGAICO Annual Meeting*, Neuchâtel: Swiss Group of Artificial Intelligence and Cognitive Science: Université de Neuchâtel.

KUKICH, Karen (1992). Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4):377–439.

KWASNY, Stan C. et SONDHEIMER, Norman K. (1981). Relaxation Techniques for Parsing Gramatically Ill-Formed Input in Natural Language Understanding Systems. *American Journal of Computational Linguistics*, 7(2):99–108.

LA TORRE, María Dolores (1999). A web-based resource to improve translation skills. *ReCALL*, 11(3):41–49.

LABOUR, Michel (2001). Social constructivism and CALL: evaluating some interactive features of network-based authoring tools. *ReCALL*, 13(1):32–46.

LAENZLINGER, Christopher (1992). Les correcteurs grammaticaux: linguistiquement satifaisants? Rapport technique, Université de Genève, Laboratoire d'Analyse et de Technologie du Langage, Genève.

LAENZLINGER, Christopher (2003). *Initiation à la syntaxe formelle du français: le modèle "Principes et paramètres" de la grammaire générative transformationnelle*, vol. 70 de *Sciences pour la communication.* Berne: Peter Lang.

LAENZLINGER, Christopher, L'HAIRE, Sébastien et MENGON, Juri (2000). Generic Pasing and Hybrid Transfer in Automatic Translation. *In* CHRISTODOULAKIS, Dimitris N. (Ed.). *Natural Language Processing – NLP 2000: Second International Conference Patras, Greece, June 2–4, 2000. Proceedings*, pp. 304–314, Berlin: Springer.

LAENZLINGER, Christopher et WEHRLI, Eric (1991). Un analyseur interactif pour le français. *T.A. Informations*, 32(2):35–49.

LAMEL, Lori et GAUVAIN, Jean-Luc (2003). Speech recognition. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 305–322. Oxford: Oxford Univ. Press.

LAMY, Marie-Noëlle et GOODFELLOW, Robin (1998). "Conversations réflexives" dans la classe de langues virtuelle par conférence asynchrone. *Alsic*, 1(2):81–101.

LAMY, Marie-Noëlle, KARLSKOV MORTENSEN, Hans Jørgen et DAVIES, Graham (2005). ICT4LT Module 2.4: Using concordance programs in the Modern Foreign Languages classroom. Resource on the web.

LAPALME, Guy et MACKLOVITCH, Elliott (2006). Compréhension et multilinguisme. *In* SABAH, Gérard (Ed.). *Compréhension des langues et interaction*, Cognition et traitement de l'information, pp. 357–374. Paris: Hermes Science, Lavoisier.

LAROCCA, Stephen A., MORGAN, John J. et BELLINGER, Sherri M. (1999). On the path to 2x learning: exploring the possibilities of advanced speech recognition. *Calico Journal*, 16(3):295–310.

LASCARIDES, Alex et ASHER, Nicholas (1991). Discourse Relations and Defeasible Knowledge. *In 29th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 55–62, Berkeley: University of California, Berkeley: Association for Computational Linguistics.

LAURENT, Dominique (1999). CORDIAL 6 Universités. *In Acte de la Journée ATALA "Outils pour le Traitement Automatique des Langues"*, Paris: ATALA, Association pour le Traitement Automatique des Langues.

LAURENT, Dominique, NÈGRE, Sophie et SÉGUÉLA, Patrick (2007). Logiciel Cordial. *In* BENAMARA, Farah, HATHOUT, Nabil, MULLER, Philippe et OZDOWSKA, Sylwia (Eds.). *Actes de la 14^{e} conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^{e} Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 2 de *Actes TALN et RECITAL*, pp. 319–321, Toulouse: Association pour le Traitement Automatique des Langues.

LAURENT, Dominique, NÈGRE, Sophie et SÉGUÉLA, Patrick (2009a). Apport des cooccurrences à la correction et à l'analyse syntaxique. *In Actes de TALN-RECITAL 2009*, Senlis.

LAURENT, Dominique, NÈGRE, Sophie et SÉGUÉLA, Patrick (2009b). L'analyseur syntaxique Cordial dans Passage. *In Actes de TALN-RECITAL 2009*, Senlis.

LAURIER, M. (2000). Can computerised testing be authentic? *ReCALL*, 12 (1):93–104.

LEACOCK, Claudia, GAMON, Michael et BROCKETT, Chris (2009). User Input and Interactions on Microsoft Research ESL Assistant. *In Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 73–81, Boulder, Colorado: Association for Computational Linguistics.

LEBARBÉ, Thomas (2004). CALL: from current problems to NLP solutions. MIRTO: a user-oriented NLP-based teaching platform. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, pp. 158–159, Vienna: EUROCALL.

LECHELT, Myriam (2005). Analyse et conception d'un correcteur grammatical libre pour le français. Rapport de stage de master, Université Stendhal-Grenoble 3, Grenoble.

LEDGERWOOD, Mikle D. (2000). Système-D: Writing Assistant for French. Calico Software Review Online.

LELOUCHE, Ruddy (1991). Using multiple knowledge bases to help teach some pragmatic aspects of French. *Computer Assisted Language Learning (CALL): An International Journal*, 4(1):29–40.

LEMNITZER, L. et KUNZE, C. (2002). Adapting GermaNet for the Web. *In Proceedings of the First Global Wordnet Conference*, pp. 174–181, Mysore, India: Central Institute of Indian Languages.

LESSARD, Greg et LEVISON, Michael (2007). Lexical creativity in L2 French and cultural language generation. *In* AYOUN, Dalila (Ed.). *French Applied Linguistics*, vol. 16 de *Language Teaching & Language Learning*, pp. 299–333. Amsterdam, Philadelphia: John Benjamins.

LEVENSHTEIN, Vladimir I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

LEVI, Tatiana, STOKOWSKI, Steve, KOSTER, Nikolaus et RYCSHKA, Andreas (2004). Voice Recognition Feature of the German Express Courseware: Conceptualization, Specification and Prototyping – Model Elaboration through Phases. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

LEVIN, Lori S. et EVANS, David A. (1995). ALICE-chan: A Case Study in ICALL Theory and Practice. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 77–97. Mahwah, NJ: Lawrence Erlbaum Associates.

LEVISON, M. et LESSARD, G. (1996). Using a language generation system for second language learning. *Computer Assisted Language Learning (CALL): An International Journal*, 9(2–3):181–189.

LEVISON, Michael et LESSARD, Greg (2004). Generated Narratives for Computer-aided Language Teaching. *In* LEMNITZER, Lothar, MEURERS,

Detmar et HINRICHS, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 26–31, Geneva, Switzerland: COLING.

LEVISON, Michael, LESSARD, Greg et WALKER, Derek (2000). A Multi-Level Approach to the Detection of Second Language Learner Errors. *Literary and Linguistic Computing*, 15(3):313–322.

LEVY, Michael (1997). *Computer-assisted language learning : context and conceptualization*. Oxford: Clarendon Press.

LEVY, Mike (1999). Design Process in CALL: Integrating Theory, Research and Evaluation. *In* CAMERON, Keith (Ed.). *Computer-Assisted Language Learning (CALL): Media, Design and Application*, pp. 83–107. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

L'HAIRE, Sébastien (2000). L'Enseignement Assisté par Ordinateur et le Traitement Automatique du Langage Naturel. Mémoire de DES, Université de Genève, Faculté des Lettres, Genève.

L'HAIRE, Sébastien (2004). Vers un feedback plus intelligent. Les enseignements du projet Freetext. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL). 22 octobre 2004, Grenoble. Actes*, pp. 1–12, Grenoble: ATALA.

L'HAIRE, Sébastien, MENGON, Juri et LAENZLINGER, Christopher (2000). Outils génériques et transfert hybride pour la traduction automatique sur Internet. *In Actes de TALN 2000*, pp. 253–262, Lausanne: ATALA.

L'HAIRE, Sébastien et VANDEVENTER FALTIN, Anne (2003a). Diagnostic d'erreurs dans le projet FreeText. *Alsic*, 6(2):21–37.

L'HAIRE, Sébastien et VANDEVENTER FALTIN, Anne (2003b). Error Diagnosis in the FreeText Project. *Calico Journal*, 20(3):481–495.

LIAN, Andrew (1992). Intelligence in Computer-Aided Language Learning. *In* PENNINGTON, Martha C. et STEVENS, Vance (Eds.). *Computers in applied linguistics: an international perspective*, pp. 66–76. Clevedon, Philadelphia: Multilingual Matters.

LIOU, Hsien-Chin (1996). Development of intelligent tutoring systems: an exemple for English writing-revision. *Computer Assisted Language Learning (CALL): An International Journal*, 9(1):85–94.

LIOU, Hsien-Chin (2004). Corpora-based NLP technologies for an online EFL learning environment. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 163, Vienna: EUROCALL.

LISTER, Roy et RANTA, Leila (1997). Corrective Feedback and Learner Uptake. *Studies in Second Language Acquisition*, 20:37–66.

LITMAN, Diane et FORBES-RILEY, Kate (2006). Correlations between dialogue acts and learning in spoken tutoring dialogues. *Natural Language Engineering*, 12(2):161–176.

LITMAN, Diane J. et SILLIMAN, Scott (2004). Itspoke: An intelligent tutoring spoken dialogue system. *In* SUSAN DUMAIS, Daniel Marcu et ROUKOS, Salim (Eds.). *HLT-NAACL 2004: Demonstration Papers*, pp. 5–8, Boston, Massachusetts, USA: Association for Computational Linguistics.

LIU, Anne Li-E, WIBLE, David et TSAO, Nai-Lung (2009). Automated suggestions for miscollocations. *In Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 47–50, Boulder, Colorado: Association for Computational Linguistics.

LIU, Chao-Lin, WANG, Chun-Hung, GAO, Zhao-Ming et HUANG, Shang-Ming (2005). Applications of lexical information for algorithmically composing multiple-choice cloze items. *In Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pp. 1–8, Ann Arbor, Michigan: Association for Computational Linguistics.

LOISEAU, Mathieu (2004). La description de ressources pédagogiques: état de l'art et application aux ressources textuelles pour l'enseignement des langues. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 23–35, Grenoble: LIDILEM: ATALA - XRCE.

LORITZ, Donald (1987). An Introductory Lisp Parser. *Calico Journal*, 4 (4):51–70.

LORITZ, Donald (1992). Generalized Transition Network Parsing for Language Study : The GPARS System for English, Russian, Japanese and Chinese. *Calico Journal*, 10(1):5–22.

LORITZ, Donald (1995). GPARS: A Suite of Grammar Assesment Systems. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 121–133. Mahwah, NJ: Lawrence Erlbaum Associates.

LOTHERINGTON, Heather et XU, Yejun (2004). How to chat in English and Chinese: Emerging digital language conventions. *ReCALL*, 16(2):308–329.

LOWRANCE, Roy et WAGNER, Robert A. (1975). An Extension of the String-to-String Correction Problem. *Journal of the Association for Computing Machinery*, 22(2):177–183.

MALONE, Stuart et FELSHIN, Sue (1991). GLR parsing for erroneous input. *In* TOMITA, Masaru (Ed.). *Generalized LR parsing*, pp. 129–139. Boston, Dordrecht: Kluwer.

MANGENOT, François (1998a). Classification des apports d'Internet à l'apprentissage des langues. *Alsic*, 1(2):133–146.

MANGENOT, François (1998b). Réseau Internet et apprentissage du français. *Etudes de linguistique appliquée*, 110:205–214.

MANGEOT, Mathieu et THEVENIN, David (2004). Online Generic Editing of Heterogeneous Dictionary Entries in Papillon Project. *In Coling Geneva 2004: 20th International Conference on Computational Linguistics: Proceedings*, vol. 1, pp. 1029–1035, Geneva, Switzerland: COLING.

MANGU, Lidia et BRILL, Eric (1997). Automatic rule acquisition for spelling correction. *In Proceedings of 14 th International Conference on Machine Learning*, pp. 187–194. Morgan Kaufmann.

MANN, William C. et THOMPSON, Sandra A. (1987). Rhetorical Structure Theory: description and construction of text structures. *In* KEMPEN, Gerard (Ed.). *Natural Language Generation: new results in Artificial Intelligence, Psychology and Linguistics*, NATO Advanced Science Institute Series, pp. 85–95. Dordrecht, NL: Martinus Nijhoff.

MANNING, Christopher D. et SCHÜTZE, Hinrich (2000). *Foundations of Statistical Natural Language Processing*. Cambridge MA and London: MIT Press, 3^{rd} ed.

MARCUS, Mitchell P. (1985). *A Theory of Syntactic Recognition for Natural Language*. MIT Press Series in Artifical Inteligence. Cambridge, Mass.: MIT Press, 3^{rd} ed.

MARKOSIAN, Lawrence Z. et AGER, Tryg A. (1984). Application of parsing theory to Computer-Assisted Instruction. *In* WYATT, David H. (Ed.). *Computer-assisted language instruction*, Language teaching methodology, pp. 65–77. Oxford: Pergamon Press.

MARQUET, Pascal (2004). *Informatique et enseignement: progrès ou évolution?* Sprimont: Mardaga.

MARSHALL, Ian (1987). Tag selection using probabilistic methods. *In* GARSIDE, Roger, LEECH, Geoffrey et SAMPSON, Geoffrey (Eds.). *The computational analysis of English: a corpus-based approach*, pp. 42–56. London & New York: Longman.

MARTIN, Philippe (2004a). WinPitch Corpus: A text to speech Alignment Tool for Multimodal Corpora. *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, vol. 2, pp. 537–540, Lisbon, Portugal: ELRA - European Language Resources Association.

MARTIN, Philippe (2004b). WinPitch LTL II, a Multimodal Pronunciation Software. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

MARTIN, Philippe (2004c). WinPitch LTL, un logiciel d'enseignement de la prosodie multimédia. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 71–82, Grenoble: LIDILEM: ATALA - XRCE.

MATTHEWS, Clive (1992). Going AI. Foundations of ICALL. *Computer Assisted Language Learning (CALL): An International Journal*, 5(1-2): 13–32.

MATTHIESEN, Christian et THOMPSON, Sandra A. (1988). The structure of discourse and 'subordination'. *In* HAIMAN, John et THOMPSON, Sandra A. (Eds.). *Clause Combining in Grammar and Discourse*, pp. 275–329. Amsterdam: John Benjamins.

McEnery, Tony (2003). Corpus linguistics. *In* Mitkov, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 448–463. Oxford: Oxford Univ. Press.

McEnery, Tony et Wilson, Andrew (1993). The role of corpora in Computer-Assisted Language Learning. *Computer Assisted Language Learning (CALL): An International Journal*, 6(3):233–248.

McEnery, Tony et Wilson, Andrew (1997). Teaching and language corpora (TALC). *ReCALL*, 9(1):5–14.

McEnery, Tony et Wilson, Andrew (2005). ICT4LT Module 3.4: Corpus linguistics. Resource on the web.

McEnery, Tony, Wilson, Andrew et Barker, Paul (1997). Teaching grammar again after twenty years: corpus-based help for teaching grammar. *ReCALL*, 9(2):8–16.

McIlroy, M. Douglas (1982). Development of a spelling list. *IEEE Transactions on Communications*, COM-30(1):91–99.

Mellish, Chris S. (1989). Some chart-based techniques for parsing ill-formed input. *In 27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 102–109, Vancouver, Canada: University of British Columbia: Association for Computational Linguistics.

Mel'čuk, I. A., Arbatchewsky-Jumarie, N., Dagenais, L., Elnitsky, L., Iordanskaja, L., Lefebvre, M.-N. et Mantha, S. (1988). *Dictionnaire explicatif et combinatoire du français contemporain : recherches lexico-sémantique*, vol. II. Montréal: Les Presses de l'Université de Montréal.

Mel'čuk, I. A., Arbatchewsky-Jumarie, N., Elnitsky, L., Iordanskaja, L. et Lessard, A. (1984). *Dictionnaire explicatif et combinatoire du français contemporain : recherches lexico-sémantique*, vol. I. Montréal: Les Presses de l'Université de Montréal.

Mel'čuk, I. A., Arbatchewsky-Jumarie, N., Iordanskaja, L. et Mantha, S. (1992). *Dictionnaire explicatif et combinatoire du français contemporain : recherches lexico-sémantique*, vol. III. Montréal: Les Presses de l'Université de Montréal.

Mel'čuk, I. A., Arbatchewsky-Jumarie, N., Iordanskaja, L., Mantha, S. et Polguère, A. (1999). *Dictionnaire explicatif et combinatoire du français contemporain : recherches lexico-sémantique*, vol. IV. Montréal: Les Presses de l'Université de Montréal.

Mel'čuk, Igor Alexandrovich (1997). Vers une linguistique Sens-Texte. Collège de France, chaire internationale, discours inaugural.

Mel'čuk, Igor Alexandrovich (2001). *Communicative organization in natural language : the semantic-communicative structure of sentences*. Amsterdam: J. Benjamins, cop.

MENDELSOHN, Patrick et JERMANN, Patrick (1997). *Les technologies de l'information appliquées à la formation : rapport de tendances*. Bern, Aarau: Direction du Programme National de Recherche PNR33 et Centre suisse de Coordination pour la Recherche en Education (CSRE).

MENÉZO, Jacques, GENTHIAL, Damien et COURTIN, Jacques (1996). Reconnaissances pluri-lexicales dans CELINE, un système multi-agents de détection et corrections des erreurs. *In Proceedings of the International Conference NLP-IA/TAL-AI 96*, pp. 174–180, Moncton, Canada.

MENÉZO, Jacques, GENTHIAL, Damien, COURTIN, Jacques et DUJARDIN, Danièle (1998). Modèles Humains dans un Système Multi-Agents orienté Apprentissage et Détection-Correction des Erreurs. *In Le traitement automatique du langage et les applications industrielles - accent sur l'enseignement de la langue - Acte de TAL + AI 98*, pp. 38–44, Moncton, Canada: GRÉTAL: Groupe d'étude sur le traitement automatique des langues.

MENZEL, Wolfgang (1988). Error diagnosis and selection in a training system for second language learning. *In Coling Budapest: proceedings of the 12th International Conference on Computational Linguistics, 22-27 August 1988*, vol. 2, pp. 414–419, Budapest: J. von Neumann Society for Computing Science.

MENZEL, Wolfgang (1990). Anticipation-free Diagnosis of Structural Faults. *In* KARLGREN, Hans (Ed.). *Coling-90: papers presented to the 13th International Conference on Computational Linguistics, on the occasion of the 25th anniversary of Coling and the 350th anniversary of Helsinki University*, pp. 422–424, Helsinki: Universitas Helsigiensis.

MENZEL, Wolfgang (2004). Errors, Intentions, and Explanations: Feedback Generation for Language Tutoring Systems. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

MENZEL, Wolfgang, HERRON, Daniel, MORTON, Rachel, PEZZOTTA, Dario, BONAVENTURA, Patrizia et HOWARTH, Peter (2001). Interactive pronunciation training. *ReCALL*, 13(1):67–78.

MENZEL, Wolfgang et SCHRÖDER, Ingo (1998a). Constraint-based diagnosis for Intelligent Language Tutoring Systems. *In Proceedings IT & KNOWS, XV. IFIP World Computer Congress*, pp. 484–497, Vienna, Budapest.

MENZEL, Wolfgang et SCHRÖDER, Ingo (1998b). Error Diagnosis for Language Learning Systems. *In Le traitement automatique du langage et les applications industrielles - accent sur l'enseignement de la langue - Acte de TAL + AI 98*, pp. 45–51, Moncton, Canada: GRÉTAL: Groupe d'étude sur le traitement automatique des langues.

MENZEL, Wolfgang et SCHRÖDER, Ingo (1999). Error diagnosis for language learning systems. *In* SCHULZE, Mathias, HAMEL, Marie-Josée et THOMPSON, June (Eds.). *Language Processing in CALL*, pp. 20–30, CCL UMIST, Manchester and EUROCALL: RECALL.

MERTENS, Piet, AUCHLIN, Antoine, GOLDMAN, Jean-Philippe, GROBET, Anne et GAUDINAT, Arnaud (2001). Intonation du discours et synthèse de la parole : premiers résultats d'une approche par balises. *Cahiers de Linguistique Française*, 23:189–209.

METCALF, Vanessa et MEURERS, Detmar (2006). Generating Web-based English Preposition Exercises from Real-World Texts. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 174, Granada: Eurocall.

MEYER, Bertrand (2008). *Conception et programmation orientées objet*. Paris: Eyrolles, 3 ed.

MICARELLI, Alessandro et BOYLAN, Patrick (1997). Conversation Rebuilding : from the foreign language classroom to implementation in an Intelligent Tutoring System. *Computers & Education*, 29(4):163–180.

MICHAUD, Lisa N., MCCOY, Kathleen et STARK, Litza A. (2001). Modeling the Acquisition of English: an Intelligent CALL Approach. *In Proceedings of the 8th International Conference on User Modeling*, pp. 14–23, Sonthofen, Germany: Springer.

MICHAUD, Lisa N. et MCCOY, Kathleen F. (1999). Modeling User Language Proficiency in a Writing Tutor for Deaf Learners of English. *In* BROMAN OLSEN, Mari (Ed.). *Computer Mediated Language Assessment and Evaluation in Natural Language Processing. Proceedings of a Symposium sponsored by the Association for Computational Linguistics and International Association of Language Learning Technologies*, pp. 47–54, College Park, Maryland USA: University of Maryland: Association for Computational Linguistics.

MICHEL, Johan et LEHUEN, Jérôme (2002). Conception of a Language Learning Environment Based on the Communicative and Actional Approaches. *In* CERRI, Stefano A., GOUARDÈRES, Guy et PARAGUAÇU, Fábio (Eds.). *Intelligent Tutoring Systems: 6th International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain, June 2002. Proceedings*, pp. 651–660, Berlin: Springer.

MICHEL, Johan et LEHUEN, Jérôme (2004). Un analyseur hypotético-déductif non déterministe pour l'apprentissage et la pratique d'une langue. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 13–22, Grenoble: LIDILEM: ATALA - XRCE.

MIKHEEV, Andrei (2003). Text segmentation. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 201–218. Oxford: Oxford Univ. Press.

MILIĆEVIĆ, Jasmina et HAMEL, Marie-Josée (2005). Un dictionnaire de reformulation pour les apprenants du français langue seconde. *In Actes du 29^{e} colloque annuel de l'Association de Linguistique des Provinces Atlantiques (ALPA)*, Moncton, NB.

MILLER, George A. et FELLBAUM, Christiane (1992). WordNet and the Organization of Lexical Memory. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 89–102. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

MILLER, Lance A., HEIDORN, George E. et JENSEN, Karen (1981). Text-critiquing with the EPISTLE system: an author's aid to better syntax. *In* ORDEN, Alex, EVENS, Martha et HAWKES, Albert K. (Eds.). *AFIPS Conference Proceedings: 1981 National Computer Conference*, vol. 50 de *AFIPS Conference Proceedings*, pp. 649–655, Chicago: American Federation of Information Processing Societies: AFIPS Press.

MIRZAIEAN, Vahid et RAMSAY, Allan (2005). Content-based support for Persian learners of English. *ReCALL*, 17(1):139–154.

MITAMURA, Teruko, BAKER, Kathryn, NYBERG, Eric et SVOBODA, David (2003). Diagnostics for Interactive Controlled Language Checking. *In Proceedings of the Joint Conference of the 8^{th} International Workshop of the European Association for Machine Translation and the 4^{th} Controlled Language Applications Workshop*. Dublin City University.

MITTON, Roger (1987). Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information Processing and Management*, 23 (5):495–505.

MITTON, Roger (1996). Spellchecking by computers. *Journal of the Simplified Spelling Society*, 20(1):4–11.

MOGILEVSKI, Eugene (1998). Le Correcteur 101 (A Comparative Evaluation of Version 2.2 and Version 3.5 Pro). *Calico Journal*, 16(2):183–196.

MOLLA, Steven R., SANDERS, Alton F. et SANDERS, Ruth H. (1989). Artificial Intelligence in a German Adventure Game: Spion in PROLOG. *Calico Journal*, 6(1):9–23.

MONSON, Christian, LEVIN, Lori, VEGA, Rodolfo, BROWN, Ralf, FONT LLITJOS, Ariandna, LAVIE, Alon, CARBONELL, Jaime, CAÑULEF, Eliseo et HUISCA, Rosendo (2004). Data Collection and Analysis of Mapundungun Morphology for Spelling Correction. *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, vol. 5, pp. 1629–1632, Lisbon, Portugal: ELRA - European Language Resources Association.

MOORTGAT, Michael (1990). La Grammaire catégorielle généralisée: le calcul de Lambek-Gentzen. *In* MILLER, Philip et TORRIS, Thérèse (Eds.). *Formalismes syntaxiques pour le traitement automatique du langage naturel*, Langue, raisonnement, calcul, chap. 3, (pp. 127–182). Paris: Hermès.

MORGAN, John J. (2004). Making a Speech Recognizer Tolerate Non-native Speech through Gaussian Mixture Merging. *In Proceedings of InSTIL/*

ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems, Venice.

MORRIS, Robert et CHERRY, Lorinda L. (1975). Computer detection of typographical errors. *IEEE Transactions on Professional Communication*, PC-18(1):54–64.

MORTON, Hazel et JACK, Mervyn A. (2005). Scenario-Based Spoken Interaction with Virtual Agents. *Computer Assisted Language Learning (CALL): An International Journal*, 18(3):171–191.

MOSTOW, Jack, AIST, Greg, BECK, Joseph, CHALASANI, Raghuvee, CUNEO, Andrew, JIA, Peng et KADARU, Krishna (2002). A La Recherche du Temps Perdu, or As Time Goes By: Where Does the Time Go in a Reading Tutor That Listens? *In* CERRI, Stefano A., GOUARDÈRES, Guy et PARAGUAÇU, Fábio (Eds.). *Intelligent Tutoring Systems: 6th International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain, June 2002. Proceedings*, pp. 300–329, Berlin: Springer.

MOSTOW, Jack et AIST, Gregory (1999). Giving Help and Praise in a Reading Tutor with Imperfect Listening - Because Automated Speech Recognition Means Never Being Able to Say You're Certain. *Calico Journal*, 16 (3):407–424.

MOTE, Nicolaus, JOHNSON, Lewis, SETHY, Abhinav, SILVA, Jorge et NARAYANAN, Shrikanth (2004). Tactical Language Detection and Modeling of Learner Speech Errors: The case of Arabic tactical language training for American English speakers. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

MULFORD, George W. (1989). Semantic Processing for Communicative Exercises in Foreign-Language Learning. *Computers and the Humanities*, 23:31–44.

MURPHY, Maureen, KRÜGER, Anja et GRIESZEL, Andrea (1998). RECALL - Providing an Individualized CALL Environment. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 62–73. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

MURPHY, Maureen et MCTEAR, Michael (1997). Learner modelling for intelligent CALL. *In User Modeling: Proceedings of the Sixth International Conference, UM97*, pp. 301–312. Vienna, New York: Springer Wien New York.

MURPHY-JUDY, Kathryn (2002). Sans-Faute Writing Environment. Calico Software Review Online.

MURRAY, Janet H. (1995). Lessons Learned from the Athena Language Learning Project: Using Natural Language Processing, Graphics, Speech Processing, and Interactive Video for Communication-Based Language

Learning. *In* Holland, V. Melissa, Kaplan, Jonathan D. et Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 243–256. Mahwah, NJ: Lawrence Erlbaum Associates.

Murray, Tom (1999). Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. *International Journal of Artificial Intelligence in Education*, 10:98–129.

Mydlarski, Donna (1999). CorText. *Calico Journal*, 16(4):584–595.

Naber, Daniel (2003). A Rule-Based Style and Grammar Checker. Diplomarbeit, Technische Fakultät, Universität Bielefeld, Bielefeld.

Nadasdi, Terry et Sinclair, Stéfan (2008). LePatron : correcteur pédagogique pour le Français Langue Étrangère. *In* Foucher, Anne-Laure, Pothier, Maguy, Rodrigues, Christine et Quanquin, Véronique (Eds.). *TICE et Didactique des Langues Étrangères et Maternelles : la problématique des aides à l'apprentissage*, pp. 419–427, Presses Blaise-Pascal: Cahiers du Laboratoire de Recherche sur le Langage n^{o}2.

Nadasdi, Terry et Sinclair, Stéfan (2009). Anything I can do, CPU can do better: a comparison of human and computer grammar correction for L2 writing using BonPatron.com. *Journal of French Language Studies*, (19):406–409.

Nagata, Noriko (1995). An Effective Application of Natural Language Processing in Second Language Instruction. *Calico Journal*, 13(1):47–67.

Nagata, Noriko (1997). An experimental comparison of deductive and inductive feedback generated by a simple parser. *System: an international journal of educational technology and applied linguistics*, 25(4):515–534.

Nagata, Noriko (2002). BANZAI: An Application of Natural Language Processing to Web-based Language Learning. *Calico Journal*, 19(3):583–599.

Nagata, Noriko (2009). Robo-Sensei's NLP-Based Error Detection and Feedback Generation. *Calico Journal*, 26(3):562–579.

Nagata, Ryo, Kawai, Atsuo, Morihiro, Koichiro et Isu, Naoki (2006). A Feedback-Augmented Method for Detecting Errors in the Writing of Learners of English. *In COLING/ACL 06 : Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 241–248, Sydney, Australia: Association for Computational Linguistics.

Namer, Fiametta, Bouillon, Pierrette et Jacquey, Evelyne (2007). Un Lexique Génératif de référence pour le français. *In* Benamara, Farah, Hathout, Nabil, Muller, Philippe et Ozdowska, Sylwia (Eds.). *Actes de la 14^e conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 2 de *Actes TALN et RECITAL*, pp. 233–242, Toulouse: Association pour le Traitement Automatique des Langues.

NAPOLITANO, Diane M. et STENT, Amanda (2009). TechWriter: An Evolving System for Writing Assistance for Advanced Learners of English. *Calico Journal*, 26(3):611–625.

NAVARRO, Gonzalo (2001). A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 33(1):31–88.

NAZARENKO, Adeline (2006). Le point sur l'état actuel des connaissances en traitement automatique des langues (TAL). *In* SABAH, Gérard (Ed.). *Compréhension des langues et interaction*, Cognition et traitement de l'information, pp. 31–70. Paris: Hermes Science, Lavoisier.

NDIAYE, Mar et VANDEVENTER FALTIN, Anne (2003). A Spell Checker Tailored to Language Users. *Computer Assisted Language Learning (CALL): An International Journal*, 16(2–3):213–232.

NDIAYE, Mar et VANDEVENTER FALTIN, Anne (2004). Correcteur orthographique adapté à l'apprentissage du français. *BULAG*, 29:117–134.

NEEDLEMAN, S. et WUNSCH, C. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:444–453.

NELSON, Theodor Holm (1965). Complex information processing: a file structure for the complex, the changing and the indeterminate. *In ACM '65: Proceedings of the 1965 20th national conference*, pp. 84–100, New York, NY, USA: Association of Computing Machinery (ACM).

NERBONNE, John (2003). Natural Language Processing in Computer-Assisted Language Learning. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 670–698. Oxford: Oxford Univ. Press.

NERBONNE, John et DOKTER, Duco (1999). An Intelligent Word-Based Language Learning Assistant. *Traitement automatique des langues*, 40 (1):125–142.

NERBONNE, John, DOKTER, Duco et SMIT, Petra (2002). Morphological Processing and Computer-Assisted Language Learning. *Computer Assisted Language Learning (CALL): An International Journal*, 15(5):543–559.

NERIMA, Luka, SERETAN, Violeta et WEHRLI, Eric (2003). Creating a Multilingual Collocation Dictionary from Large Text Corpora. *In* 10^{th} *Conference of the European Chapter of the Association for Computational Linguistics EACL: Proceedings of the Conference*, pp. 131–134, Budapest, Hungary.

NERIMA, Luka, SERETAN, Violeta et WEHRLI, Eric (2006). Le problème des collocations en TAL. *Nouveaux cahiers de linguistique française*, 27:95–115.

NEUWIRTH, Christine M. (1989). Intelligent Tutoring Systems: Exploring Issues in Learning and Teaching Writing. *Computers and the Humanities*, 23(1):45–57.

New, E. (1999). Computer-aided writing in French as a foreign language: a qualitative and quantitative look at the process of revision. *The Modern Language Journal*, 83(1):80–97.

Nicholas, Nick, Debski, Robert et Lagerberg, Robert (2004). Skryba: An Online Orthography Teaching Tool for Learners from Bilingual Backgrounds. *Computer Assisted Language Learning (CALL): An International Journal*, 17(3–4):441–458.

Niño, Ana (2005). Using machine translation output to enhance proficiency in foreign language written production. *In EUROCALL 2005 Conference. CALL, WELL and TELL: fostering autonomy*, p. 169, Kraków, Poland: Jagiellonian University.

Odell, M. K. et Russell, R. C. (1918, 1922). Patent Numbers 1,261,167 (1918) and 1,435,663 (1922). U.S. Patent Number, U.S. Patent 0ffice.

Oflazer, Kemal (1996). Error-tolerant Finite-state Recognition with Application to Morphological Analysis and Spelling Correction. *Computational Linguistics*, 22(1):73–89.

Oflazer, Kemal (2003). Lenient morphological analysis. *Natural Language Engineering*, 9(1):87–99.

Oflazer, Kemal et Güzey, Cemaleddin (1994). Spelling Correction in Agglutinative Languages. *In Fourth Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 194–195, Stuttgart, Germany: ACL.

Oliva, Karel (1997). Techniques for Accelerating a Grammar-Checker. *In Fifth Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 155–158, Washington Marriott Hotel, Washington, DC, USA: Association for Computational Linguistics.

Oliver, Jonathan J. (1993). Decision graphs - an extension of decision trees. *In Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pp. 343–350.

Ott, Heike, Rodríguez Prados, Francisco J., Ludwig, Bernd et Harriehausen-Mühlbauer, Bettina (2005). Language e-Learning with role-based mobile phone games. *In LIT Leipzig 05 proceedings*, Leipzig.

Oxford, Rebecca L. (1995). Linking Theories of Learning with Intelligent Computer-Assisted Language Learning (ICALL). *In* Holland, V. Melissa, Kaplan, Jonathan D. et Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 359–369. Mahwah, NJ: Lawrence Erlbaum Associates.

Pain, Helen (1981). A computer aid for spelling error classification in language teaching. *In* Lewis, Bob et Tagg, Donovan (Eds.). *Computers in education : proceedings of the IFIP TC-3 : 3rd World Conference on Computers in Education-WCCE 81*, vol. 1, pp. 297–302, Lausanne: IFIP - International Federation for Information Processing: North Holland.

PANKHURST, James (2005). LIPSTIC: A Limited Intelligence Parser Seeking Typical Interference Constructions. *The EUROCALL Review*, 8.

PAPERT, Seymour (1970). Teaching children thinking. Rapport technique 247 (Logo Memo 2), MIT Artificial Intelligence Laboratory Memo, Cambridge, MA.

PAPINENI, Kishore, ROUKOS, Salim, WARD, Todd et ZHU, Wei-Jing (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. *In Association for Computational Linguistics, 40th Annual Meeting and 10th Conference of the European Chapter: Proceedings of the Conference*, pp. 311–318, Philadelphia.

PARAMSKAS, D. M. (1999). The Shape of Computer-Mediated Communication. *In* CAMERON, Keith (Ed.). *Computer-Assisted Language Learning (CALL): Media, Design and Application*, pp. 13–34. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

PARAMSKAS, Dana M. (1993). ELAO: genèse et avenir. *In* LIDDELL, P. (Ed.). *CALL: Theory and Applications*, pp. 65–75. Victoria: Victoria University Press.

PARK, Jong C., PALMER, Martha et WASHBURN, Clay (1997). An English Grammar Checker as a Writing Aid for Students of English as a Second Language. *In Fifth Conference on Applied Natural Language Processing. Descriptions of System Demonstrations and Videos*, p. 24, Washington: Association for Computational Linguistics.

PAROUBEK, Patrick et RAJMAN, Martin (2000). Étiquetage morpho-syntaxique. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 131–150. Paris: Hermès Science Publications.

PASERO, Robert et SABATIER, Paul (1998). Linguistic Games for Language Learning: A Special Use of the ILLICO Library. *Computer Assisted Language Learning (CALL): An International Journal*, 11(5):561–585.

PASKALEVA, E. et MIHOV, S. (1998). Second Language Acquisition from aligned corpora. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 43–52. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

PAYETTE, Julie (1990). Intelligent Computer-Assisted Instruction in Syntactic Style. Mémoire de D.E.A. / Master, Computer Systems Research Institute, University of Toronto, Toronto.

PEDLER, Jennifer (2001). Computer spellcheckers and dyslexics – a performance survey. *British Journal of Educational Technology*, 32(1):23–37.

PENNINGTON, Martha C. et ESLING, John H. (1996). Computer-Assisted Development of Spoken Language Skills. *In* PENNINGTON, Martha C. (Ed.). *The Power of CALL*, pp. 153–189. Houston: Athelstan.

PEREIRA, Fernando C. N. et SHIEBER, Stuart M. (1987). *Prolog and Natural-Language Analysis*. N^o 10 *in* CSLI Lecture Notes. Stanford: CSLI Publications.

PETERSON, James L. (1980). Computer Programs for Detecting and Correcting Spelling Errors. *Communications of the Association for Computing Machinery*, 23(12):676–687.

PHILIPS, Lawrence (2000). The Double Metaphone Search Algorithm. *C/C++ Users Journal*, (6).

PHINNEY, Marianne (1996). Exploring the Virtual World: Computers in the Second Language Writing Classroom. *In* PENNINGTON, Martha C. (Ed.). *The Power of CALL*, pp. 137–152. Houston: Athelstan.

PIAGET, Jean (1969). *Psychologie et pédagogie*. coll. Médiations. Paris: Gonthiers Denoël.

PIERREL, Jean-Marie et ROMARY, Laurent (2000). Dialogue homme-machine. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 331–349. Paris: Hermès Science Publications.

PIJLS, Fieny, DAELEMANS, Walter et KEMPEN, Gerard (1987). Artificial intelligence tools for grammar and spelling instruction. *Instructional Science*, 16(4):319–336.

PLOUX, Sabine et JI, Hyungsuk (2003). A Model for Matching Semantic Maps between Languages (French / English, English / French). *Computational Linguistics*, 29(2):155–178.

POLGUÈRE, Alain (1998). La théorie sens-texte. *Dialangue*, 8-9:9–30.

POLGUÈRE, Alain (2003). Etiquetage sémantiques des lexies dans la base de données DiCo. *Traitement automatique des langues*, 44(2):39–68.

POLLARD, Carl et SAG, Ivan (1994). *Head-driven Phrase Structure Grammar*. CSLI Series. Chicago: University of Chicago Press.

POLLOCK, Joseph L. et ZAMORA, Antonio (1984). Computer Programs for Detecting and Correcting Spelling Errors. *Communications of the Association for Computing Machinery*, 27(4):358–368.

POTTER, Keith R. (2002). Taming overgeneration in a constraint-relaxed parser. Communication to 13^{th} meeting of Computational Linguistics in the Netherlands.

PRÉVOT, Laurent, MULLER, Philippe, DENIS, Pascal et VIEU, Laure (2002). Une approche sémantique et rhétorique du dialogue. un cas d'étude: l'explication d'un itinéraire. *Traitement automatique des langues*, 43(2):43–70.

PROBST, Katharina, KE, Yan et ESKENAZI, Maxine (2002). Enhancing foreign language tutors — In search of the golden speaker. *Speech Communication*, 37:161–173.

PROST, Jean-Philippe (2008). *Modelling Syntactic Gradience with Loose Constraint-based Parsing*. Thèse de doctorat, Macquarie University & Université de Provence, Sidney, Australia & Aix-en-Provence, France.

PUJOLÀ, Joan-Tomàs (2001). Did CALL feedback feed back? Researching learners' use of feedback. *ReCALL*, 13(1):79–98.

PUJOLÀ, Joan-Tomás (2002). CALLing for help: researching language learning strategies using help facilities in a web-based multimedia program. *ReCALL*, 14(2):235–262.

PULMAN, S. G. (1984). Limited Domain Systems for Language Teaching. *In 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics: Proceedings of Coling-ACL 84*, pp. 84–87, Stanford University, CA: Association for Computational Linguistics.

PUSACK, James P. (1984). Answer-processing and error correction in foreign language CAI. *In* WYATT, David H. (Ed.). *Computer-assisted language instruction*, Language teaching methodology, pp. 53–64. Oxford: Pergamon Press.

PUSTEJOVSKY, James (1995). *The Generative Lexicon*. Cambridge MA, USA: MIT Press.

PUSTEJOVSKY, James et BOGURAEV, Branimir (1993). Lexical knowledge representation and natural language processing. *Artificial Intelligence*, 63:193–223.

QIAO, Hong Liang et SUSSEX, Roland (1996). Using the Longman Mini Concordancer on tagged and parsed corpora with special reference to their use as an aid to grammar learning. *System: an international journal of educational technology and applied linguistics*, 24(1):41–64.

QUIXAL, Martí, BADIA, Toni, BENAVENT, Francesc, BOULLOSA, Jose R., DOMINGO, Judith, GRAU, Bernat, MASSÓ, Guillem et VALENTÍN, Oriol (2008). User-Centred Design of Error Correction Tools. *In* CALZOLARI, Nicoletta, CHOUKRI, Khalid, MAEGAARD, Bente, MARIANI, Joseph, ODJIK, Jan, PIPERIDIS, Stelios et TAPIAS, Daniel (Eds.). *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pp. 1985–1989, Marrakech, Morocco: European Language Resources Association (ELRA).

RAINERO, Roger (2004). Débats télévisés en direct du Sénat du Canada: correction automatique des sous-titres en français. *BULAG*, 29:135–152.

RAMBOW, Owen et KORELSKY, Tanya (1992). Applied Text Generation. *In Third Conference on Applied Natural Language Processing, Proceedings of the Conference*, pp. 40–47, Trento, Italy: Association for Computational Linguistics.

RAMÍREZ BUSTAMANTE, Flora, DECLERCK, Thierry et SANCHÉZ LEÓN, Fernando (2000). Towards a Theory of Textual Errors. *In Proceedings of the 3rd International Workshop on Controlled Language Applications (CLAW'00), April 29-30*, Seattle, USA.

RAMÍREZ BUSTAMANTE, Flora et SÁNCHEZ LEÓN, Fernando (1996). GramCheck: A Grammar and Style Checker. *In COLING 96: The 16th International Conference on Computational Linguistics. Proceedings of the*

Conference, vol. 1, pp. 175–181, Copenhagen, Denmark: Center for Sprogteknologi.

RAMLUCKUN, Mira et WEHRLI, Eric (1993). ITS-2 : an interactive personal translation system. *In Sixth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 476–477, Utrecht, The Netherlands: Association for Computational Linguistics.

RAMSAY, Allan (2005). *The Parasite Manual*. University of Manchester School of Informatics, Manchester, UK.

RAUX, Antoine et ESKENAZI, Maxine (2004). Using Task-Oriented Spoken Dialogue Systems for Language Learning: Potential, Practical Applications and Challenges. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

RAVIN, Yael (1993). Grammar Errors and Style Weaknesses in a Text-Critiquing System. *In* JENSEN, Karen, HEIDORN, George E. et RICHARDSON, Stephen D. (Eds.). *Natural Language Processing: The PNLP Approach*, pp. 65–76. Dordrecht: Kluwer.

REEDER, Florence, HAMBURGER, Henry et SCHOELLES, Michael (1999). Real Talk: Authentic Dialogue Practice. *In* DEBSKI, Robert et LEVY, Mike (Eds.). *WORLDCALL: Global Perspectives on Computer-Assisted Language Learning*, pp. 319–337. Lisse, The Netherlands: Swets & Zeitlinger B. V.

REICHENBACH, Hans (1947). *Elements of Symbolic Logic*. New-York: Free Press.

REID, Joy, LINDSTROM, Peggy, MCCAFFREY, Maggie et LARSON, Doug (1983). Computer-assisted Text-analysis for ESL Students. *Calico*, 1 (3):40–42.

RENIÉ, Delphine et CHANIER, Thierry (1995). Collaboration and computer-assisted acquisition of a second language. *Computer Assisted Language Learning (CALL): An International Journal*, 8(1):3–29.

RENIÉ, Delphine et CHANIER, Thierry (1996). ÉLÉONORE: quelle place pour la collaboration dans un environnement d'apprentissage du français langue seconde? *Sciences et Techniques Educatives*, 3(3):353–380.

REUER, Veit (1999). Dialogue processing in a CALL-system. *In Proceedings of EACL'99: Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 253–256, Bergen, Norway: Association for Computational Linguistics.

REUER, Veit (2000). Error Recognition and Parsing of Syntactically mildly ill-formed Natural Language. *In* BUTT, Miriam et HOLLOWAY KING, Tracy (Eds.). *Proceedings of the LFG00 Conference, University of California, Berkeley*, Stanford, CA, USA: CSLI Publications.

REUER, Veit (2003). Error Recognition and Feedback with Lexical Functional Grammar. *Calico Journal*, 20(3):497–512.

Revuz, Dominique (1991). *Dictionnaires et Lexiques, Méthodes et Algorithmes*. Thèse de doctorat, Université Paris VII.

Reynaert, Martin (2004). Text Induced Spelling Correction. *In Coling Geneva 2004: 20th International Conference on Computational Linguistics: Proceedings*, vol. 1, pp. 834–840, Geneva, Switzerland: COLING.

Rézeau, Joseph (1994). Integrating telematics data into CALL packages. *In* Thompson, June et Chesters, Graham (Eds.). *Emancipation through learning technology. Selected papers from the EUROCALL'93 conference, University of Hull, UK, 15-17 September 1993. A special double issue of Computers and Education, Vol 23, No. 1/2, 1994*, pp. 159–164, Oxford: EUROCALL: Pergamon.

Richardson, Stephen et Braden-Harder, Lisa (1993). The Experience of Developing a Large-Scale Natural Language Processing System: Critique. *In* Jensen, Karen, Heidorn, George E. et Richardson, Stephen D. (Eds.). *Natural Language Processing: The PNLP Approach*, pp. 77–89. Dordrecht: Kluwer.

Richardson, Stephen D. et Braden-Harder, Lisa C. (1988). The Experience of Developing A Large-Scale Natural Language Text Processing System: CRITIQUE. *In Second Conference on Applied Natural Language Processing: Proceedings of the Conference*, pp. 195–202, Austin, Texas, USA.

Rimrott, Anne (2003). SANTY: A Spell Checking Algorithm for Treating Predictable Verb Inflection Mistakes Made by Non-Native Writers of German. Term Paper for LING 807 Computational Linguistics, Simon Fraser University.

Rimrott, Anne et Heift, Trude (2005). Language Learners and Generic Spell Checkers in CALL. *Calico Journal*, 23(1):17–48.

Rimrott, Anne et Heift, Trude (2008). Evaluating Automatic Detection of Misspellings in German. *Language Learning & Technology*, 12(3):73–92.

Riseman, E. M. et Hanson, A. R. (1974). A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computers*, C-23(5):480–493.

Rizzo, Paola, Shaw, Erin et Johnson, W. Lewis (2002). An Agent That Helps Children to Author Rhetorically-Structured Digital Puppet Presentations. *In* Cerri, Stefano A., Gouardères, Guy et Paragaçu, Fábio (Eds.). *Intelligent Tutoring Systems: 6th International Conference ITS 2002, Biarritz, France and San Sebastian, Spain, June 2-7, 2002. Proceedings*, pp. 903–912, Berlin: Springer.

Robertson, Judy et Wiemer-Hastings, Peter (2002). Feedback on Children's Stories via Multiple Interface Agents. *In* Cerri, Stefano A., Gouardères, Guy et Paragaçu, Fábio (Eds.). *Intelligent Tutoring Systems: 6th International Conference ITS 2002, Biarritz, France and San*

Sebastian, Spain, June 2-7, 2002. Proceedings, pp. 923–932, Berlin: Springer.

Roosmaa, Tiit et Prószéky, Gabor (1998). GLOSSER - using language technology tools for reading texts in a foreign language. *In* Jager, Sake, Nerbonne, John et Van Essen, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 101–107. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

Rosé, Carolyn P., Bhembe, Dumisizwe, Roque, Antonio, Siler, Stephanie, Srivastava, Ramesh et VanLehn, Kurt (2002). A hybrid language understanding approach for robust selection of tutoring goals. *In* Cerri, Stefano A., Gouardères, Guy et Paraguaçu, Fábio (Eds.). *Intelligent Tutoring Systems: 6^{th} International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain, June 2002. Proceedings*, pp. 552–561, Berlin: Springer.

Roussarie, Laurent (1998). Le problème de la structuration et de la représentation du discours vu sous l'angle de la génération automatique. *Traitement automatique des langues*, 39(2):35–55.

Roux, Claude (2004). Annoter les documents XML avec un outil d'analyse syntaxique. *In* Bel, Bernard et Marlien, Isabelle (Eds.). *Actes de TALN 2004: XI^{e} Conférence sur le Traitement Automatique des Langues Naturelles*, Fès, Maroc: LPL - Université Sidi Mohammed Ben Abdellah - ENS Fès: ATALA.

Rüschoff, Bernd (1998). Evaluating and Testing. *In* Blin, Françoise, Chénik, Nicole et Thompson, June (Eds.). *CALL Courseware Development: a Handbook*, pp. 15–24. Hull, UK: Eurocall.

Rypa, Marikka et Feuerman, Ken (1995). CALLE: An Exploratory Environment for Foreign Language Learning. *In* Holland, V. Melissa, Kaplan, Jonathan D. et Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 55–76. Mahwah, NJ: Lawrence Erlbaum Associates.

Rypa, Marikka Elizabeth et Price, Patti (1999). VILTS: A Tale of Two Technologies. *Calico Journal*, 16(3):385–404.

Sabah, Gérard (2000). Sens et traitement automatique des langues. *In* Pierrel, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 77–108. Paris: Hermès Science Publications.

Sabah, Gérard et Grau, Brigitte (2000). Compréhension automatique de textes. *In* Pierrel, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 293–310. Paris: Hermès Science Publications.

Sams, Michelle R. (1995). Advanced Technologies for Language Learning: The BRIDGE Project Within the ARI Language Tutor Program. *In* Holland, V. Melissa, Kaplan, Jonathan D. et Sams, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 7–21. Mahwah, NJ: Lawrence Erlbaum Associates.

SANDERS, Alton et SANDERS, Ruth (1987). Designing and Implementing a Syntactic Parser. *Calico Journal*, 5(1):77–86.

SANDERS, Ruth (1991). Error Analysis in Purely Syntactic Parsing of Free Input. The Example of German. *Calico Journal*, 9(1):72–89.

SANDERS, Ruth H. et SANDERS, Alton F. (1995a). History of an AI Spy Game : Spion. *Calico Journal*, 12(4):114–127.

SANDERS, Ruth H. et SANDERS, Alton F. (1995b). History of an AI Spy Game: Spion. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 141–151. Mahwah, NJ: Lawrence Erlbaum Associates.

SANZ, Didier (1992). Grammaire: quatre ténors à l'épreuve. *Science et Vie Micro*, (90):100–108.

SAXENA, Anju et BORIN, Lars (2002). Teaching grammar with a treebank and a parser. *In Eurocall 2002: Preconference workshop: NLP in CALL, No Longer Pertinent or New Light Penetrates? 3rd Eurocall workshop organised by the SIG in Language Processing*, Jyväskylä, Finland.

SCHMID, Helmut (1994). Part-of-speech Tagging with Neural Networks. *In Proceedings of the 15th International Conference on Computational Linguistics (COLING '94)*, pp. 172–176, Kyoto, Japan.

SCHMIDT, Helmut (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. *In Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

SCHMIDT, Paul, GARNIER, Sandrine, SHARWOOD, Mike, BADIA, Toni, DÍAZ, Lourdes, QUIXAL, Martí, RUGGIA, Ana, VALDERRABANOS, Antonio S., CRUZ, Alberto J., TORREJON, Enrique, RICO, Celia et JIMENEZ, Jorge (2004). ALLES: Controlled language tools and information extraction tools for CALL Applications. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

SCHNEIDER, David et MCCOY, Kathleen F. (1998). Recognizing Syntactic Errors in the Writing of Second Language Learners. *In Coling-ACL '98. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: Proceedings of the Conference*, vol. 2, pp. 1198–1204, Montréal, Canada: Université de Montréal.

SCHOELLES, Michael et HAMBURGER, Henry (1996). Cognitive tools for language pedagogy. *Computer Assisted Language Learning (CALL): An International Journal*, 9(2–3):213–234.

SCHRÖDER, Ingo, MENZEL, Wolfgang, FOTH, Kilian et SCHULZ, Michael (2000). Modeling dependency grammar with restricted constraints. *Traitement automatique des langues*, 41(1):113–144.

SCHULZE, Mathias (1995). Textana: Text Reproduction In A Hypertext Environment. *In* GIMENO, Ana (Ed.). *Proceedings of EUROCALL 95':*

Technology Enhanced Language Learning: Focus On Integration, pp. 415–425, Valencia: Universidad Politécnica de Valencia.

Schulze, Mathias (1997). TEXTANA – Text Production in A Hypertext Environment. *Computer Assisted Language Learning (CALL): An International Journal*, 10(1):71–82.

Schulze, Mathias (1998). Checking Grammar – Teaching Grammar. *Computer Assisted Language Learning (CALL): An International Journal*, 11 (2):215–227.

Schulze, Mathias (1999). From the developer to the learner: describing grammar - learning grammar. *ReCALL*, 11(1):117–124.

Schulze, Mathias (2004). WatPAL - Learning Designs, Learning Objects, and Tablet PCs. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 214, Vienna: EUROCALL.

Schulze, Mathias (2008). Modeling SLA processes using NLP. *In* Chapelle, Carol A., Chung, Y. R. et Xu, J. (Eds.). *Towards adaptive CALL: Natural language processing for diagnostic language assessment*, pp. 149–166. Ames: Iowa State University.

Schulze, Mathias et Hamel, Marie-Josée (2000). Towards authentic tasks and experiences: The example of parser-based CALL. *Canadian Journal of Applied Linguistics*, 3(1–2):79–90.

Schulze, Mathias et Penner, Nikolai (2008). Construction grammar in icall. *Computer Assisted Language Learning (CALL): An International Journal*, 21(5):427–440.

Schuster, Ethel (1986). The Role of Native Grammars in Correcting Errors in Second Language Learning. *Computational Intelligence*, 2(2):93–98.

Schütz, Jörg (1996). The ALEP Formalism in a Nutshell. Rapport technique, IAI, Saarbrücken.

Schwartz, Lee, Aikawa, Takako et Pahud, Michel (2004). Dynamic Language Learning Tools. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Schwind, Camilla B. (1986). Overview of an Intelligent Language Tutoring System. *In CIIAM 86: Actes du 2^{e} Colloque International d'Intelligence Artificielle / Proceeding of the 2nd International Conference on Artificial Intelligence*, pp. 389–407, Marseille: Hermes.

Schwind, Camilla B. (1988). Sensitive Parsing: Error Analysis and Explanation in an Intelligent Language Tutoring System. *In Coling Budapest: proceedings of the 12th International Conference on Computational Linguistics, 22-27 August 1988*, vol. 2, pp. 608–613, Budapest: J. von Neumann Society for Computing Science.

Schwind, Camilla B. (1995). Error Analysis and Explanation in Knowledge Based Language Tutoring. *Computer Assisted Language Learning (CALL): An International Journal*, 8(4):295–324.

Scinicariello, S. (2006). Podcasts in the Language Curriculum: Integrating Language, Content and Technology. *In Eurocall 2006 Granada. Integrating CALL into study programmes*, p. 222, Granada: Eurocall.

Searle, John Rogers (1969). *Speech acts : an essay in the philosophy of language*. Cambridge, London: Cambridge University Press.

Segond, Frédérique et Parmentier, Thibault (2004). NLP serving the cause of language learning. *In* Lemnitzer, Lothar, Meurers, Detmathr et Hinrichs, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 11–17, Geneva, Switzerland: COLING.

Segond, Frédérique, Parmentier, Thibault, Stock, Roberta, Rosner, Ran et Usteran Muela, Mariola (2005). Situational language training for hotel receptionists. *In Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pp. 85–92, Ann Arbor, Michigan: Association for Computational Linguistics.

Selva, Thierry et Chanier, Thierry (2000). Génération automatique d'activités lexicales dans le système ALEXIA. *Sciences et Techniques Educatives (STE)*, 7(2):385–412.

Selva, Thierry, Verlinde, Serge et Binon, Jean (2003). Vers une deuxième génération de dictionnaires électroniques. *Traitement automatique des langues*, 44(2):177–197.

Seneff, Stephanie, Wang, Chao et Chao, Chihyu (2007). Spoken dialogue systems for language learning. *In Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pp. 13–14, Rochester, New York, USA: Association for Computational Linguistics.

Sentance, Sue (1997). A rule network for English article usage within an Intelligent Language Tutoring System. *Computer Assisted Language Learning (CALL): An International Journal*, 10(2):173–200.

Seretan, Violeta (2008). *Collocation extraction based on syntactic parsing*. Thèse de doctorat, University of Geneva.

Seretan, Violeta (2009). Extraction de collocations et leurs équivalents de traduction à partir de corpus parallèles. *Traitement automatique des langues*, 50(1):305–332.

Seretan, Violeta et Wehrli, Eric (2006). Accurate collocation extraction using a multilingual parser. *In COLING/ACL 06 : Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 953–960, Sydney, Australia: Association for Computational Linguistics.

Seretan, Violeta et Wehrli, Eric (2007). Collocation translation based on sentence alignment and parsing. *In* Benamara, Farah, Hathout, Nabil,

MULLER, Philippe et OZDOWSKA, Sylwia (Eds.). *Actes de la 14^{e} conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^{e} Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 1 de *Actes TALN et RECITAL*, pp. 401–410, Toulouse: Association pour le Traitement Automatique des Langues.

SHAALAN, Khaled F. (2005). An Intelligent Computer Assisted Language Learning (CALL): An International Journal System for Arabic Learners. *Computer Assisted Language Learning (CALL): An International Journal*, 18(1-2):81–108.

SHIEBER, Stuart M. (1983). Sentence Disambiguation by a Shift-Reduce Parsing Technique. *In 21st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 113–118, Cambridge, MA: Massachusetts Institute of Technology: Association for Computational Linguistics.

SI-QING, Chen et LUOMAI, Xu (1990). Grammar-Debugger : A Parser for Chinese EFL Learners. *Calico Journal*, 8(2):63–75.

SILBERZTEIN, Max et TUTIN, Agnès (2004). NooJ: un outil de TAL de corpus pour l'enseignement des langues et de la linguistique. Une application à l'étude des impersonnels. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 47–56, Grenoble: LIDILEM: ATALA - XRCE.

SILBERZTEIN, Max D. (1994). INTEX: a Corpus Processing System. *In Proceedings of the 15th International Conference on Computational Linguistics (COLING '94)*, pp. 579–583, Kyoto, Japan.

SIMARD, Michel et DESLAURIERS, Alexandre (2001). Real-time automatic insertion of accents in French text. *Natural Language Engineering*, 7(2): 143–165.

SIMON, Anne-Catherine (2001). Le rôle de la prosodie dans le repérage des unités textuelles minimales. *Cahiers de Linguistique Française*, 23:99–125.

SINGLETON, James, KEANE, John et NKWENTI-AZEH, Blaise (1998). CALL meets software engineering: towards a multimedia conceptual dictionary. *ReCALL*, 10(2):33–43.

SITBON, Laurianne, BELLOT, Patrice et BLACHE, Philippe (2007). Traitements phrastiques phonétiques pour la réécriture de phrases dysorthographiées. *In* BENAMARA, Farah, HATHOUT, Nabil, MULLER, Philippe et OZDOWSKA, Sylwia (Eds.). *Actes de la 14^{e} conférence sur le Traitement Automatique des Langues Naturelles (communications orales) et Actes de la 11^{e} Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (communications orales)*, vol. 2 de *Actes TALN et RECITAL*, pp. 263–272, Toulouse: Association pour le Traitement Automatique des Langues.

SKINNER, B. F. (1954). The Science of Learning and the Art of Teaching. *Harvard Educational Review*, 24:86–97.

SKRELIN, P. et VOLSKAJA, N. (1998). The application of new technologies in the development of education programs. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 21–24. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

SLEATOR, et TEMPERLEY, (1991). Parsing English with a link grammar. Rapport technique CMU-CS-91-196, Carnegie-Mellon University.

SMRŽ, Pavel (2004). Integrating Natural Language Processing into E-Learning - A Case of Czech. *In* LEMNITZER, Lothar, MEURERS, Detmar et HINRICHS, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 1–10, Geneva, Switzerland: COLING.

SOFTISSIMO, (2002). Correcteur Orthographique Hugo. Rapport technique, Softissimo, Paris.

SOMERS, Harold (2003). Machine Translation: latest developments. *In* MITKOV, Ruslan (Ed.). *The Oxford handbook of computational linguistics*, pp. 512–528. Oxford: Oxford Univ. Press.

SOMERS, Harold (2004). Does machine translation have a role in language learning? *In Actes d'UNTELE 2004*, Compiègne.

SORIA, Jesús (1997). Expert CALL: data-based versus knowledge-based interaction and feedback. *ReCALL*, 9(2):43–50.

SPORTICHE, Dominique (1996). Clitic constructions. *In* ROORYCK, Johan et ZARING, Laurie (Eds.). *Phrase structure and the lexicon*, Studies in natural language and linguistic theory. Dordrecht: Kluwer.

SUN, Guihua, LIU, Xiaohua, CONG, Gao, ZHOU, Ming, XIONG, Zhongyang, LEE, John et LIN, Chin-Yew (2007). Detecting erroneous sentences using automatically mined sequential patterns. *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 81–88, Prague, Czech Republic: Association for Computational Linguistics.

SURI, Linda Z. (1994). Designing computer assisted language learning tools and appropriate tests for deaf writers. *In* KAPLAN, R. M. et BURSTEIN, J. C. (Eds.). *Educational testing service conference on natural language processing techniques and techology in assessment and education.* Princeton: ETS.

SVENCONIS, D. J. et KERST, S. (1995). Investigating the teaching of second-language vocabulary through semantic mapping in a hypertext environment. *Calico Journal*, 12(2&3):33–57.

SWARTZ, Merryanna L. (1992). Issues for Tutoring Knowledge in Foreign Language Intelligent Tutoring Systems. *In* SWARTZ, Merryanna L.

et Yazdani, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 219–233. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

Swartz, Merryanna L. et Yazdani, Masoud (1992). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*. NATO Advanced Institute Series. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

Tanaka, Eiichi et Kojima, Yurie (1987). A High Speed String Correction Method Using a Hierarchical File. *IEEE transactions on pattern analysis and Machine Intelligence*, 9(6):806–815.

Tasso, Carlo, Fum, Danilo et Giangrandi, Paolo (1992). The Use of Explanation-Based Learning for Modelling Student Behavior in Foreign Language Tutoring. *In* Swartz, Merryanna L. et Yazdani, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 151–170. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

Teixeira Martins, Ronaldo, Hasegawa, Ricardo, Volpe Nunes, Maria das Graças, Montilha, Gisele et Novais de Oliveira Jr., Osvaldo (1998). Linguistic issues in the development of ReGra: A grammar checker for Brazilian Portuguese. *Natural Language Engineering*, 4(4):287–307.

Teixeira Martins, Ronaldo, Machado Rino, Lúcia Helena, Graças Volpe Nunes, Mariadas et Novais Oliveira Jr, Osvaldo (2002). The UNL distinctive features: inferences from a NL-UNL enconverting task. *In Proc. of the First International Workshop on UNL, other Interlinguas and their Applications. LREC 2002*, pp. CD–Rom, Las Palmas, Spain: ELRA - European Language Resources Association.

Tellier, Christine (1995). *Eléments de syntaxe du français: méthodes d'analyse en grammaire générative*. Montréal: Presses de l'Université de Montréal.

ten Hacken, Pius et Tschichold, Cornelia (2001). Word Manager and CALL: structured access to the lexicon as a tool for enriching learners' vocabulary. *ReCALL*, 13(1):99–109.

Tesnières, Lucien (1959). *Éléments de syntaxe structurale*. Paris: Klincksieck.

Thomas, Craig, Levison, Michael et Lessard, Greg (2004a). Experiments in Prosody for the Generation of Oral French. *In Proceedings of InSTIL/ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Thomas, Pete, Haley, Debra, deRoeck, Anne et Petre, Marian (2004b). E-Assessment using Latent Semantic Analysis in the Computer Science Domain: A Pilot Study. *In* Lemnitzer, Lothar, Meurers, Detmar et

HINRICHS, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 38–44, Geneva, Switzerland: COLING.

THURMAIR, Gregor (1990). Parsing for Grammar and Style Checking. *In* KARLGREN, Hans (Ed.). *Coling-90: papers presented to the 13th International Conference on Computational Linguistics, on the occasion of the 25th anniversary of Coling and the 350th anniversary of Helsinki University*, pp. 365–370, Helsinki: Universitas Helsigiensis.

TOKUDA, Naoyuki et CHEN, Liang (2004). A New KE-Free Online ICALL System Featuring Error Contingent Feedback. *Computer Assisted Language Learning (CALL): An International Journal*, 17(2):177–201.

TOMITA, Masaru (1986). *Efficient parsing for natural language: a fast algorithm for practical systems*. Boston, Dordrecht: Kluwer.

TOMLIN, Russel S. (1995). Modeling Individual Tutorial Interactions: Theoretical and Empirical Bases of ICALL. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Michelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 201–220. Mahwah, NJ: Lawrence Erlbaum Associates.

TOOLE, Janine et HEIFT, Trude (2002). The Tutor Assistant: An Authoring Tool for an Intelligent Language Tutoring System. *Computer Assisted Language Learning (CALL): An International Journal*, 15(4):373–386.

TORLAKOVIC, Edina, MARTIN, Joel et FERGUSON, Darrell (2004). Teaching grammar using intelligent feedback and a virtually unlimited corpus of authentic examples. *In EUROCALL 2004 Conference. TELL and CALL in the Third Millennium: Pedagogical Approaches in a Growing EU-Community*, p. 230, Vienna: EUROCALL.

TSCHICHOLD, Cornelia (1999a). Grammar Checking for CALL: Strategies for Improving Foreign Language Grammar Checkers. *In* CAMERON, Keith (Ed.). *Computer-Assisted Language Learning (CALL): Media, Design and Application*, pp. 203–221. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

TSCHICHOLD, Cornelia (1999b). Intelligent grammar checking for CALL. *In* SCHULZE, Mathias, HAMEL, Marie-Josée et THOMPSON, June (Eds.). *Language Processing in CALL*, pp. 5–11, CCL UMIST, Manchester and EUROCALL: RECALL.

TSCHICHOLD, Cornelia (2003). Lexically Driven Error Detection and Correction. *Calico Journal*, 20(3):549–559.

TSCHICHOLD, Cornelia (2006). Intelligent CALL: The magnitude of the task. *In* MERTENS, Piet, FAIRON, Cédrick, DISTER, Anne et WATRIN, Patrick (Eds.). *verbum ex machina. Actes de la 13^{e} Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 2, pp. 806–814, Leuven: Université Catholique de Louvain: UCL Presses.

TSCHICHOLD, Cornelia, BODMER, Franck, CORNU, Etienne, GROSJEAN, François, GROSJEAN, Lysiane, KÜBLER, Nathalie et TSCHUMI, Corinne (1994). Detecting and Correcting Errors in Second Language Texts. *Computer Assisted Language Learning (CALL): An International Journal*, 7 (2):151–160.

TSCHICHOLD, Cornelia et TEN HACKEN, Pius (1998). English Phraseology in Word Manager. *In Proceedings of the 3rd International Symposium on Phraseology*, pp. 219–225, Stuttgart.

TUFIS, Dan et MASON, Oliver (1998). Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. *In Proceedings of the First International Conference on Language Resources & Evaluation (LREC)*, pp. 589–596, Granada (Spain).

TURCATO, Davide, NICHOLSON, Devlan, HEIFT, Trude, TOOLE, Janine et TSIPLAKOU, Stavroula (2000). A Parsing Methodology for Error Detection. *In IWPT 2000: Sixth International Workshop on Parsing Technologies*, Trento, Italy.

UMIST CENTRE FOR COMPUTATIONAL LINGUISTICS, (2002). Standards for the evaluation of speech synthesis for CALL : an evaluation of FIPSvox in FreeText. Rapport technique, UMIST, Manchester.

VALLI, André et VÉRONIS, Jean (1999). Etiquetage grammatical des corpus de parole: problèmes et perspectives. *Revue Française de Linguistique Appliquée*, 4(2):113–133.

VAN BERKELT, Brigitte et DE SMEDT, Koenraad (1988). Triphone Analysis: A Combined Method for the Correction of Orthographical and Typographical Errors. *In Second Conference on Applied Natural Language Processing: Proceedings of the Conference*, pp. 77–83, Austin, Texas, USA.

VAN EIJCK, J. et ALSHAWI, H. (1992). Logical Forms. *In* ALSHAWI, H. (Ed.). *The Core Language Engine*. Cambridge MA: MIT Press.

VAN HEUVEN, Vincent J. (1998). COOL/CALP: Computer-Assisted Learning to Parse in Dutch. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 74–81. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

VAN MARCKE, Kris (1987). KRS: An Object Oriented Representation Language. *Revue d'intelligence artificielle*, 1(4):43–68.

VAN MARCKE, Kris (1990). A Generic Tutoring Environment. *In* CARLUCCI AIELLO, Luigia, SANDEWALL, E., HAGERT, G. et GUSTAVSSON, B. (Eds.). *ECAI 90 : proceedings of the 9th European Conference on Artificial Intelligence, August 6-10, 1990, Stockholm, Sweden*, pp. 655–660, London: European Coordinating Committee for Artificial Intelligence (ECCAI) & Swedish Artificial Intelligence Society (SAIS): Pitman Publishing.

VANDEVENTER, Anne (1998). An Automatic System for Error Diagnosis in CALL. *In Le traitement automatique du langage et les applications*

industrielles - accent sur l'enseignement de la langue - Acte de TAL + AI 98, pp. 77–83, Moncton, Canada: GRÉTAL: Groupe d'étude sur le traitement automatique des langues.

VANDEVENTER, Anne (2000). Diagnostic d'erreurs grammaticales par relâchement de contraintes dans le cadre de l'ELAO. *In Actes de TALN 2000*, pp. 357–366, Lausanne: ATALA.

VANDEVENTER, Anne (2001). Creating a grammar checker for CALL by constraint relaxation : a feasibility study? *ReCALL*, 13(1):110–120.

VANDEVENTER, Anne et HAMEL, Marie-Josée (2000). Reusing a syntactic generator for CALL purposes. *ReCALL*, 12(1):79–91.

VANDEVENTER, Anne et NDIAYE, Mar (2002). A spell checker tailored to language learners. *In* COLPAERT, Jozef, DECOO, Wilfried, SIMONS, Mathea et VAN BUEREN, Saskia (Eds.). *Proceedings of CALL 2002: CALL Professionals and the Future of CALL Research*, pp. 315–329, Antwerp: University of Antwerp.

VANDEVENTER FALTIN, Anne (2003). *Syntactic Error Diagnosis in the context of Computer Assisted Language Learning*. Thèse de doctorat, Université de Genève, Faculté des Lettres, Genève.

VANLEHN, Kurt, JORDAN, Pamela W., ROSÉ, Carolyn P., BHEMBE, Dumisizwe, BÖTTNER, Michael, GAYDOS, Andy, MAKTCHEV, Maxim, PAPPUSWAMY, Umarami, RINGENBERG, Michael, ROQUE, Antonio, SILER, Stephanie et SRIVASTAVA, Ramesh (2002). The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing. *In* CERRI, Stefano A., GOUARDÈRES, Guy et PARAGAÇU, Fábio (Eds.). *Intelligent Tutoring Systems: 6th International Conference ITS 2002, Biarritz, France and San Sebastian, Spain, June 2-7, 2002. Proceedings*, pp. 158–162, Berlin: Springer.

VERLINDE, Serge, SELVA, Thierry et BINON, Jean (2003). Alfalex : un environnement d'apprentissage du vocabulaire français en ligne, interactif et automatisé. *Romaneske*, 1.

VÉRONIS, Jean (1988a). Computerized Correction of Phonographic Errors. *Computers and the Humanities*, 22:43–56.

VÉRONIS, Jean (1988b). Morphosyntactic correction in natural language interfaces. *In Coling Budapest: proceedings of the 12th International Conference on Computational Linguistics, 22-27 August 1988*, vol. 2, pp. 708–713, Budapest: J. von Neumann Society for Computing Science.

VÉRONIS, Jean (2000). Annotation automatique de corpus: panorama et état de la technique. *In* PIERREL, Jean-Marie (Ed.). *Ingéniérie des langues*, pp. 111–129. Paris: Hermès Science Publications.

VÉRONIS, Jean (2004). Inf l11: Langage et informatique 1. cours 10: correction orthographique. Course syllabus on the web.

VÉRONIS, Jean et GUIMIER DE NEEF, Emilie (2006). Le traitement des nouvelles formes de communication écrite. *In* SABAH, Gérard (Ed.). *Com-*

préhension des langues et interaction, Cognition et traitement de l'information, pp. 227–248. Paris: Hermes Science, Lavoisier.

Virvou, Maria, Maras, Dimitris et Tsiriga, Victoria (2000). Student Modelling in an Intelligent Tutoring System for the Passive Voice of English Language. *Educational Technology & Society*, 3(4):139–150.

Virvou, Maria et Tsiriga, Victoria (2001). Web Passive Voice Tutor: an Intelligent Computer Assisted Language Learning System over the WWW. *In Proceedings of IEEE International Conference on Advanced Learning Technologies (ICALT'01)*, pp. 131–134, Madison, USA: University of Wisconsin: IEEE.

Visser, Henriëtte (1999). CALLex: a CALL game to study lexical relationships based on a semantic database. *In* Schulze, Mathias, Hamel, Marie-Josée et Thompson, June (Eds.). *Language Processing in CALL*, pp. 50–56, CCL UMIST, Manchester and EUROCALL: RECALL.

Vitanova, Irena (2004). Evaluating integrated NLP in foreign language learning: technology meets pedagogy. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Viterbi, Andrew (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, pp. 260–269.

Vlugter, Peter, Knott, Alistair et Weatherall, Victoria (2004). A human-machine dialogue system for CALL. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Vosse, Theo (1992). Detecting and Correcting Morpho-syntactic Errors in Real Texts. *In Third Conference on Applied Natural Language Processing: Proceedings of the Conference*, pp. 111–118, Trento, Italy: Association for Computational Linguistics.

Vygotsky, L. H. (1962). *Thought and Language*. Cambridge MA: MIT Press.

Wachowicz, Krystina A. et Scott, Brian (1999). Software that listens: it's not a question of whether, it's a question of how. *Calico Journal*, 16 (3):253–276.

Wagner, Joachim (2004). A false friends exercise with authentic material retrieved from a corpus. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Wagner, Robert A. et Fischer, Michael J. (1974). The String-to-String Correction Problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

Walther Green, Catherine (2002). The Reference Grammar. Rapport technique, Université de Genève, Département de Linguisitique, Genève.

WALTHER GREEN, Catherine (2004). FreeText: A CALL system for French featuring NLP tools for a smart treatment of authentic documents and free production exercises. *In* KELLEHER, M., HALDANE, A. et KRUIZINGA, E. (Eds.). *Researching Technology for Tomorrow's Learning: Insights from the European research community*. Bilthoven: CIBIT Consultants/Educators.

WANG, Chao et SENEFF, Stephanie (2004). High-quality Speech Translation for Language Learning. *In Proceedings of InSTIL/ ICALL2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

WANG, Chao et SENEFF, Stephanie (2007). Automatic assessment of student translations for foreign language tutoring. *In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 468–475, Rochester, New York: Association for Computational Linguistics.

WARD, Arthur et LITMAN, Diane (2005). Predicting learning in tutoring with the landscape model of memory. *In Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pp. 21–24, Ann Arbor, Michigan: Association for Computational Linguistics.

WARD, R. D., FOOT, R. et ROSTRON, A. B. (1999). Natural language processing in CALL: language with a purpose. *In* SCHULZE, Mathias, HAMEL, Marie-Josée et THOMPSON, June (Eds.). *Language Processing in CALL*, pp. 40–49, CCL UMIST, Manchester and EUROCALL: RECALL.

WARD, Robert (1996). Formes Cachées: a computer game with a French natural language interface. *ReCALL*, 8(2):35–40.

WARE, Paige et O'DOWD, Robert (2008). Peer Feedback on Language Form in Telecollaboration. *Language Learning & Technology*, 12(1):43–63.

WARGA, Muriel (2007). Interlanguage pragmatics in L2 French. *In* AYOUN, Dalila (Ed.). *French Applied Linguistics*, vol. 16 de *Language Teaching & Language Learning*, pp. 171–207. Amsterdam, Philadelphia: John Benjamins.

WARSCHAUER, M. (1996). Computer-Assisted Language Learning: An Introduction. *In* FOTOS, S. (Ed.). *Multimedia Language Teaching*, pp. 3–20. Tokyo: Logos International.

WATERS, Richard C. (1994). The Audio Interactive Tutor. Technical Report MERL-TR-94-04, Mitsubishi Electric Research Laboratories, Cambridge, MA.

WATERS, Richard C. (1995). The Audio Interactive Tutor. *Computer Assisted Language Learning (CALL): An International Journal*, 8(4):325–354.

WEBER, Jean-Jacques (2001). A concordance- and genre-informed approach to ESP essay writing. *ELT Journal*, 55(1):14–20.

WEHRLI, Eric (1997). *L'analyse syntaxique des langues naturelles: problèmes et méthodes*. Paris: Masson.

WEHRLI, Eric (1998). Translating Idioms. *In Coling-ACL '98. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: Proceedings of the Conference*, vol. 2, pp. 1388–1392, Montréal, Canada: Université de Montréal.

WEHRLI, Eric (2000). Parsing and Collocations. *In* CHRISTODOULAKIS, Dimitris N. (Ed.). *Natural Language Processing – NLP 2000: Second International Conference Patras, Greece, June 2–4, 2000. Proceedings*, pp. 272–282, Berlin: Springer.

WEHRLI, Eric (2004a). Traduction, traduction de mots, traduction de phrases. *In* BEL, Bernard et MARLIEN, Isabelle (Eds.). *Actes de TALN 2004: XI^{e} Conférence sur le Traitement Automatique des Langues Naturelles*, pp. 483–491, Fès, Maroc: LPL - Université Sidi Mohammed Ben Abdellah - ENS Fès: ATALA.

WEHRLI, Eric (2004b). Un modèle multilingue d'analyse syntaxique. *In* AUCHLIN, Antoine, BURGER, Marcel, FILLIETTAZ, Laurent, GROBET, Anne, MOESCHLER, Jacques, PERRIN, Laurent, ROSSARI, Corinne et DE SAUSSURE, Louis (Eds.). *Structures et discours: mélanges offerts à Eddy Roulet*, Langues et pratiques discursives, pp. 311–329. Québec: Nota Bene.

WEHRLI, Eric (2006). TwicPen: Hand-held Scanner and Translation Software for non-Native Readers. *In Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 61–64, Sydney, Australia: Association for Computational Linguistics.

WEHRLI, Eric (2007). Fips, a "Deep" Linguistic Multilingual Parser. *In* BALDWIN, Timothy, DRAS, Mark, HOCKENMAIER, Julia, KING, Tracy Holloway et VAN NOORD, Gertjan (Eds.). *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, pp. 120–127, Prague: Association for Computational Linguistics.

WEHRLI, Eric et CLARK, Robin (1995). Natural Language Processing, Lexicon and Semantics. *Methods of Information in Medicine*, 34(1/2):68–74.

WEHRLI, Eric et NERIMA, Luka (2009). L'analyseur syntaxique Fips. *In* VILLEMONTE DE LA CLERGERIE, Eric et PAROUBEK, Patrick (Eds.). *Journée ATALA "Quels analyseurs syntaxiques pour le français?"*, Paris: ATALA.

WEI, Yu Hong et DAVIES, Graham (1996). Do grammar checkers work? A report on research into the effectiveness of Grammatik V based on samples of authentic essays by EFL students. *In* KOHN, János, RÜSCHOFF, Bernd et WOLFF, Dieter (Eds.). *New Horizons in CALL: Proceedings of EUROCALL 96*, pp. 169–188, Szombathely, Hungary: Berzsenyi Dániel College.

WEINBERG, Amy, GARMAN, Joe, MARTIN, Jeffery et MERLO, Paola (1995). A Principle-Based Parser for Foreign Language Tutoring in German and Arabic. *In* HOLLAND, V. Melissa, KAPLAN, Jonathan D. et SAMS, Mi-

chelle R. (Eds.). *Intelligent Language Tutors: Theory Shaping Technology*, pp. 23–44. Mahwah, NJ: Lawrence Erlbaum Associates.

WEISCHEDEL, Ralph M. et BLACK, John E. (1980). Responding Intelligently to Unparsable Inputs. *American Journal of Computational Linguistics*, 6 (2):97–109.

WEISCHEDEL, Ralph M. et SONDHEIMER, Norman K. (1983). Meta-rules as a Basis for Processing Ill-Formed Input. *American Journal of Computational Linguistics*, 9(3–4):161–177.

WEISCHEDEL, Ralph M., VOGE, Wilfried M. et JAMES, Mark (1978). An Artificial Intelligence Approach to Language Instruction. *Artificial Intelligence*, 10:225–240.

WEIZENBAUM, Joseph (1966). ELIZA–A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9(1):36–45.

WENGER, E. (1987). *Artificial intelligence and tutoring systems: Computational and Cognitive Approaches to the Communication of Knowledge.* Los Altos, CA: Morgan Kaufmann Publishers.

WHISTLE, Jeremy (1999). Concordancing with students 'off-the-Web' corpus. *ReCALL*, 11(2):74–80.

WIBLE, David, KUO, Chin-Hwa, CHEN, Meng-Chang, TSAO, Nai-Lung et HUNG, Tsung-Fu (2006). A Computational Approach to the Discovery and Representation of Lexical Chunks. *In* MERTENS, Piet, FAIRON, Cédrick, DISTER, Anne et WATRIN, Patrick (Eds.). *verbum ex machina. Actes de la 13e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 2, pp. 868–875, Leuven: Université Catholique de Louvain: UCL Presses.

WILKS, Yorick (1975). Preference Semantics. *In* KEENAN, Edward L. (Ed.). *Formal Semantics of Natural Language : Papers from a colloquium sponsored by the King's College Research Centre, Cambridge*, pp. 329–348. Cambridge University Press.

WILKS, Yorick et FARWELL, David (1992). Building an Intelligent Tutoring System from Whatever Bits You Happen to Have Lying Around. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 263–274. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

WILKS, Yorick et FASS, Dan (1992). Preference semantics. *In* SHAPIRO, Stuart C. (Ed.). *Encyclopedia of Artifical Intelligence*, vol. 2, pp. 1182–1194. New-York: John Wiley & Sons, Inc., 2^{nd} ed.

WILSON, Eve (1994). Using corpora as a resource in language teaching. *In* THOMPSON, June et CHESTERS, Graham (Eds.). *Emancipation through learning technology. Selected papers from the EUROCALL'93 conference,*

University of Hull, UK, 15-17 September 1993. A special double issue of Computers and Education, Vol 23, No. 1/2, 1994, pp. 41–51, Oxford: EUROCALL: Pergamon.

WILSON, Eve (1997). The automatic generation of CALL exercises from general corpora. *In* WICHMANN, Anne, FLIGELSTONE, Steven, MCENERY, Tony et KNOWLES, Gerry (Eds.). *Teaching and language corpora*, pp. 116–130. London, New York: Longman.

WILSON, Kirk (1986). ILIAD: An Example of AI in Language Instruction. *Tesol newsletter: Teachers of English to Speakers of Other Languages*, 20 (1):22–23.

WINIWARTER, Werner (2004). PETRA - the Personal Embedded Translation and Reading Assistant. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

WINOGRAD, Terry (1972). *Understanding natural language*. Edinburgh: Edinburgh University Press.

WINOGRAD, Terry (1983). *Language as a Cognitive Process*. Reading, MA: Addison-Wesley.

WIRTH, Niklaus (1987). *Algorithmes et structures de données*. Paris: Eyrolles.

WITT, Silke et YOUNG, Steeve (1998). Computer-Assisted Pronunciation Teaching Based on Automatic Speech Recognition. *In* JAGER, Sake, NERBONNE, John et VAN ESSEN, Arthur (Eds.). *Language Teaching and Language Technology*, pp. 25–35. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

WOODIN, Jane (1997). Email tandem learning and the communicative curriculum. *ReCALL*, 9(1):22–33.

WOODS, William A. (1980). Cascaded ATN Grammars. *American Journal of Computational Linguistics*, 6(1):1–12.

XUEREB, Anne et CAELEN, Jean (2004). Un modèle d'interprétation pragmatique en dialogue homme-machine basé sur la SDRT. *In Workshop SDRT TALN-04, Fès, 22 avril 2004*, p. sans pagination, Fès: ATALA.

YANG, Jie Chi et AKAHORI, Kanji (1997). Error Analysis in Japanese Writing and Its Implementation in a Computer Assisted Language Learning System on the World Wide Web. *Calico Journal*, 15(1–3):47–66.

YANG, Jie Chi et AKAHORI, Kanji (1999). An Evaluation of Japanese CALL Sytems on the WWW Comparing a Freely Input Approach with Multiple Selection. *Computer Assisted Language Learning (CALL): An International Journal*, 12(1):59–79.

YANNAKOUDAKIS, E. J. et FAWTHROP, D. (1983a). An intelligent spelling error corrector. *Information Processing and Management*, 19(2):101–108.

YANNAKOUDAKIS, E. J. et FAWTHROP, D. (1983b). The rules of spelling errors. *Information Processing and Management*, 19(2):87–99.

YAROWSKY, David (1994). Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. *In 32nd Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pp. 88–95, Las Cruces, New Mexico: New Mexico State University: Association for Computational Linguistics.

YASUDA, Keiji, SUGAYA, Fumiaki, SUMITA, Eiichiro, TAKEZAWA, Toshiyuki, KIKUI, Genichiro et YAMAMOTO, Seiichi (2004). Automatic Measuring of English Language Proficiency using MT Evaluation Technology. *In* LEMNITZER, Lothar, MEURERS, Detmar et HINRICHS, Erhard (Eds.). *COLING'04 - The 20th International Conference on Computational Linguistics. Workshop eLearning for Computational Linguistics and Computational Linguistics for eLearning: proceedings*, pp. 53–60, Geneva, Switzerland: COLING.

YAZDANI, Masoud (1991). The LINGER Project: An Artificial Intelligence approach to Second-Language Tutoring. *Computer Assisted Language Learning (CALL): An International Journal*, 4(2):107–116.

YAZDANI, Masoud et UREN, J. (1988). Generalising language-tutoring systems: A French/Spanish case study, using LINGER. *Instructional Science*, 17:179–188.

YENCKEN, Lars et BALDWIN, Timothy (2008). Measuring and Predicting Orthographic Associations: Modelling the Similarity of Japanese Kanji. *In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 1041–1048, Manchester, UK: Coling 2008 Organizing Committee.

YI, Xing, GAO, Jianfeng et DOLAN, William B. (2008). A Web-based English Proofing System for English as a Second Language Users. *In Proceedings of the Third International Joint Conference on Natural Language Processing*, vol. 2, pp. 619–624, Hyderabad, India: Asian Federation of Natural Language Processing.

YOUNG, Steve, EVERMANN, Gunnar, GALES, Mark, HAIN, Thomas, KERSHAW, Dan, LIU, Xunying (Andrew), MOORE, Gareth, ODELL, Julian, OLLASON, Dave, POVEYA, Dan, VALTCHEV, Valtcho et WOODLAND, Phil (2006). *The HTK Book (for HTK Version 3.4)*. Cambridge University Engineering Department, Cambridge.

ZAMORANO MANSILLA, Juan Rafael (2004). Text generators, error analysis and feedback. *In Proceedings of InSTIL/ ICALL 2004: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

ZAMPA, Virginie (2004). Utilisation de l'analyse sémantique latente pour tenter d'optimiser l'acquisition d'une langue étrangère de spécialité. *In Journée d'étude de l'ATALA "TAL & Apprentissage des langues" (TAL&AL): Actes*, pp. 83–92, Grenoble: LIDILEM: ATALA - XRCE.

ZAMPA, Virginie (2005). Utilisation de l'analyse sémantique latente pour tenter d'optimiser l'acquisition par exposition à une langue étrangère de spécialité. *Alsic*, 8:135–146.

ZAMPA, Virginie et LEMAIRE, Benoît (2002). Latent Semantic Analysis for User Modeling. *Journal of Intelligent Information Systems*, 18(1):15–30.

ZANELLA, Paolo et LIGIER, Yves (1989). *Architecture et technologie des ordinateurs*. Paris: Dunod.

ZOCK, Michael (1992). SWIM or Sink: The Problem of Communicating Thought. *In* SWARTZ, Merryanna L. et YAZDANI, Masoud (Eds.). *Intelligent Tutoring Systems for Foreign Language Learning: The Bridge to International Communication*, NATO Advanced Institute Series, pp. 235–247. Berlin: Springer-Verlag, NATO Scientific Affairs Division.

ZOCK, Michael (2002). Sorry, but what was your name again, or, how to overcome the tip of the tongue problem with the help of a computer? *In COLING-02: SEMANET: Building and Using Semantic Networks. Proceedings*, Taipei, Taiwan.

ZOCK, Michael (2006). Capitalisation d'une ressource en or: le dictionnaire. *In* MERTENS, Piet, FAIRON, Cédrick, DISTER, Anne et WATRIN, Patrick (Eds.). *verbum ex machina. Actes de la 13e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, vol. 2, pp. 846–855, Leuven: Université Catholique de Louvain: UCL Presses.

Annexe A

Principales abréviations

AI Artificial Intelligence
ALAO Apprentissage des Langues Assisté par Ordinateur
ALIAO Apprentissage des Langues Intelligemment Assisté par Ordinateur
CALL Computer-Assisted Language Learning
CHS Characteristic Structure
CLS Clause Structure
CMS Content Management System
DCG Definite Clause Grammar
DPS Determiner Phrase Structure
HMM Hidden Markov Model
HPSG Head-driven Phrase Structure Grammar
HTML Hypertext Markup Language
IA Intelligence Artificielle
IRC Internet Relay Chat
LFG Lexical Functional Grammar
LSA Latent Semantic Analysis
LMS Learning Management System
LOM Learning Objects Model ou Learning Objects Model
NLP Natural Language Processing
PSS Pseudo-Semantic Structure
RSS Really Simple Syndication
SCORM Sharable Content Object Reference Model
TAL Traitement Automatique des Langues
URL Uniform Resource Locator
XML Extended Markup Language

Annexe B

Descriptions de logiciels d'ALAO

B.1 Logiciels auteurs

B.1.1 CALIS & WinCALIS

CALIS (*Computer Assisted Language Instruction System*, Borchardt, 1995) est un des logiciels pionniers de l'ALAO, né vers la fin des années 1970 à l'Université Duke à Durham, Caroline du Nord (USA) et utilisé jusqu'à la fin des années 1990 au moins. Ce logiciel était à l'origine écrit en langage *BASIC* (§2.7.1) et tournait sur des mini-ordinateurs équipés de terminaux. Les versions successives ont fait appel au langage C et au système PC. La version sous *Windows* est appelée *WinCALIS*. Le logiciel permet de créer facilement des exercices et peut être considéré comme un des premiers logiciels-auteur (§2.7.2).

Une des principales qualités de *CALIS* est d'avoir dès le début affronté le problème du multilinguisme et des caractères non standards, avec de nombreuses langues et alphabets différents. Les évaluation d'exercices se basent principalement sur techniques de reconnaissances de formes (§3.1.1.1) permettent d'évaluer les réponses complètes, avec des variantes qui doivent être prévues par les concepteurs de logiciel.

L'originalité de *CALIS* est de recourir aux réseaux de neurones (§3.3.3.3) pour détecter les erreurs intentionnelles d'apprenants, qu'ils commettent

pour faire apparaître immédiatement la bonne réponse, afin de pouvoir revenir en arrière et refaire ensuite un parcours sans faute. Les réseaux ont été entraînés sur des centaines de phrases authentiques, tantôt avec des erreurs intentionnelles, tantôt sans. Le système ne pratique pas de fausse détection d'erreurs et laisse passer quelques erreurs intentionnelles.

B.1.2 Système de Helm et McIver

Helm et McIver (1974) présentent un système d'apprentissage de l'anglais à faible couverture grammaticale. Un enseignant doit entrer une grammaire indépendante du contexte (§3.3.2.1) et des listes de mots destinés à construire une phrase. L'apprenant doit alors construire une phrase uniquement à partir des mots proposés. Dans certains cas, plusieurs phrases peuvent être reconnues. Le système analyse les phrases une à une au fur et à mesure qu'elles sont entrées par l'apprenant. Le diagnostic est limité à l'acceptation de la (des) phrase(s) par la grammaire et à quelques simples règles de ponctuation.

B.1.3 IDIOMA-TIC

IDIOMA-TIC (Desmet et Héroguel, 2005), système auteur axé sur la compréhension orale, présente des exercices de compréhension orale comportant des exercices semi-ouverts corrigés par une technique évoluée de reconnaissance de patrons, nommée *approximate string matching* (§3.1.1.1). Le concepteur d'exercices doit fournir une série de réponses possibles et le logiciel essaie de déterminer quelle est la réponse la plus proche de celle de l'apprenant. Le logiciel marque l'emplacement des mots erronés, des mots manquants et des mots superflus. Progressivement, l'apprenant pourra corriger sa réponse jusqu'à l'obtention d'une réponse complète. La phrase (75) donne un exemple de patron utilisé par *IDIOMA-TIC*.

(75) [Le soir, ma sœur Marie regarde [souvent / fréquemment] la [télé / télévision] française/ Ma sœur Marie regarde [souvent / fréquemment] la [télé / télévision] française le soir].

B.2 Didacticiels

B.2.1 $\mathcal{A}$lexi$\mathcal{A}$

$\mathcal{A}$lexi$\mathcal{A}$ est un prototype d'environnement informatique d'aide à l'apprentissage du français langue seconde (Issac, 1997; Chanier et Selva, 2004; Selva et Chanier, 2000). Il est doté de modules de corpus (§3.1.7), d'un dictionnaire général, d'un dictionnaire personnel, d'activités lexicales, d'un modèle de l'apprenant (§2.7.4.2), d'un lemmatiseur (§3.1.1.4) et d'un étiqueteur (§3.1.2).

Le corpus de 400 textes a été constitué autour du thème du travail, de l'emploi et du chômage, en variant les sources et le niveau de langue. Les textes sont étiquetés automatiquement et indexés par mots-clés. Les ambiguïtés sémantiques sont levées manuellement par des experts.

Deux cents mots (vocables) ont été sélectionnés pour le dictionnaire d'après leur fréquence d'apparition dans le corpus, qui représentent près de 400 sens (lexies) différents. Ces mots ont été indexés et les relations de synonymie, hypo/hyperonymie etc. ont été encodées. Le dictionnaire personnel permet aux apprenants de constituer leurs propres listes et d'organiser les mots de manière hiérarchique. L'apprenant peut également annoter les mots des textes de manière simple en ajoutant par exemple une traduction ; cependant, aucune relation entre les mots ne peut être ajoutée.

Le générateur d'exercices prépare des exercices à trous de différents types (avec ou sans liste de propositions, mots fléchis ou canoniques) où l'apprenant doit recontextualiser des vocables qu'il/elle a déjà rencontrés dans les textes.

B.2.2 BASIC Parser

BAP (*BASIC Parser*) est un programme d'apprentissage de l'anglais écrit en *BASIC* (Cook, 1988, §2.7.1) et adaptable à plusieurs plate-formes d'enseignement. Ce petit analyseur fonctionne grâce à un langage existant sur les plate-formes les plus répandues dans les salles de classe, contrairement aux langages *Lisp* (Gazdar et Mellish, 1989; Loritz, 1987) et *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2), mieux adaptés au traitement du langage mais moins répandus à l'époque des micro-ordinateurs. *BAP* est capable, à l'aide d'un lexique restreint, de traiter les phrases de débutants.

Les constructions couvertes sont les interrogatives, les impératives et les déclaratives. Les erreurs sont signalées par un *feedback* rudimentaire.

B.2.3 Système de Butcher et al.

Butcher *et al.* (1990) décrivent un prototype d'ALAO d'enseignement de l'anglais qui comporte un analyseur basé sur les *DCG Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Le système ne permet pas d'analyser les erreurs syntaxiques. En revanche, un module basé sur des automates à états finis (§3.3.4.1) gère l'ordre des adjectifs dans un groupe nominal.

B.2.4 CAMILLE

CAMILLE (*Computer-Aided Multimedia Interactive Langage Learning*, Ingraham *et al.*, 1994; Chanier, 1996; Chanier *et al.*, 1997) est un consortium comportant des partenaires de Grande-Bretagne, France, Espagne et Pays-Bas. Le projet vise à construire du matériel de niveau débutants d'espagnol et néerlandais et de niveau avancé de français pour affaires. Le projet est issu du logiciel *France InterActive*. Il vise à promouvoir la communication et à aider à réaliser des tâches quotidiennes comme acheter de la nourriture, demander des informations etc. Le *cédérom* est accompagné d'un cahier d'activités et comporte une grammaire, un dictionnaire avec enregistrements vocaux, des enregistrements audios et vidéos et un livre sur la culture de la langue cible.

B.2.5 Circsim-Tutor

Circsim-Tutor est un système intelligent de dialogue pour l'enseignement de physiologie cardio-vasculaire (Elmi, 1994; Evens *et al.*, 1997; Elmi et Evens, 1998). Le système de dialogue utilise une grammaire LFG (§3.3.5) et un petit lexique de 700 mots. Le système corrige des erreurs orthographiques, d'abréviation et de segmentation en essayant de trouver le candidat le plus plausible dans son lexique.

B.2.6 Didalect

Le prototype *Didalect* (Hermet *et al.*, 2006) fournit une aide à la lecture du français pour apprenants de niveau intermédiaire à avancé. Il est basé sur un corpus de texte pour lequel il existe des exercices de compréhension. Le système utilise l'analyseur *Xerox Incremental Parser* (XIP) (Roux, 2004), qui procède à une analyse par morceaux (§3.3.4.3), ainsi qu'un dictionnaire de synonymes et un dictionnaire relationnel. Pour évaluer la réponse de l'apprenant, le système utilise une à trois phrases sélectionnées dans le texte par le concepteur de la question comme réponse ; il bâtit trois listes de mots, (i) celle des mots communs à la réponse de l'apprenant et à la phrase modèle, (ii) celle des mots présents uniquement dans la réponse de l'apprenant et (iii) celle des mots présents uniquement dans la phrase modèle. Les synonymes, hyperonymes et hyponymes sont traités grâce au dictionnaire *ad hoc*. La réponse est évaluée sur le plan sémantique et syntaxique, en vérifiant la grammaticalité de la phrase et la présence de constituants obligatoires.

B.2.7 Easy Writer

EasyWriter (Couch, 2000) [1] est un logiciel d'apprentissage destiné à corriger des erreurs de grammaire. Les apprenants doivent choisir un point de grammaire pour les exercices d'écriture, ainsi qu'un degré de difficulté. Les textes présentés ont des thèmes et des registres variés. Les apprenants peuvent également ajouter leurs propres productions. Le logiciel est aussi doté d'une grammaire de référence. Aucune précision n'est donnée sur les techniques utilisées.

B.2.8 Exills

Exills (*Edutainment for Internet Language Learning Solution*) [2] est une plateforme d'apprentissage qui utilise différents outils développés dans le cadre de la recherche chez *Xerox* pour améliorer les performances des apprenants plutôt que leurs compétences (Brun *et al.*, 2002; Segond et Parmentier, 2004; Segond *et al.*, 2005). *Exills* offre des outils de communication synchrone et asynchrone avec des robots et des avatars d'autres apprenants dans un environnement ludique de réalité virtuelle. L'apprenant doit accomplir une mission à travers des tâches de communication variées faisant

1. `http://www.softwareforstudents.com`, consulté le 31 mai 2005.
2. `http://www.exills.com/`, dernier accès le 23 juillet 2006.

appel à des technologies collaboratives (§2.3). Il doit essentiellement communiquer à travers un *chat* (§2.7.5) avec des tuteurs, des robots et d'autres apprenants.

Différents outils de TAL sont intégrés à l'environnement. Un identificateur de langue vérifie que l'apprenant communique bien dans la langue requise (§3.1.1.3). Si l'apprenant utilise la mauvaise langue, sa phrase ne sera pas lue par les autres participants et robots. Un analyseur morphologique (§3.1.1.4) permet de construire automatiquement les tableaux de conjugaison. Un dictionnaire intelligent propose la traduction des mots en contexte en se basant sur la désambiguïsation syntaxique et la recherche d'expressions idiomatiques (v. p. 66). Un correcteur orthographique (§3.2) permet de corriger les mots et offre une correction phonétique et une réaccentuation des mots en plus des corrections traditionnelles. L'étiqueteur *Xelda* aide à désambiguïser les mots et offre leur analyse morphologique (§3.1.2). Enfin, un moteur de recherche multilingue avec requête en langue naturelle permet de personnaliser les recherches d'après le profil de l'apprenant : si un apprenant est intéressé par un sujet particulier et a des difficultés avec certaines expressions idiomatiques, le moteur lui fournira des textes concernant son domaine d'intérêt qui contiennent des expressions idiomatiques.

B.2.9 Generate

Generate (Hackenberg, 1985) est un logiciel conçu pour des micro-ordinateurs *Radio Schack* et *Commodore 64*. Il est écrit en langage *BASIC* (§2.7.1) et l'auteur annonce prévoir une adaptation en *Lisp* (Gazdar et Mellish, 1989; Loritz, 1987) et Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Il utilise un générateur (§3.1.5) basé sur une grammaire hors contexte et un petit lexique de vingt verbes et vingt mots. Les phrases sont ensuite prononcées à l'aide d'un synthétiseur SAM (*Software Activated Mouth*, §3.1.4). Les phrases sont générées et prononcées aléatoirement. Les traits sémantiques empêchent la génération de phrases incohérentes. Le logiciel est destiné aux étudiants débutants de linguistique pour les familiariser avec les grammaires génératives transformationnelles. Toutes les étapes du processus sont affichées afin de familiariser les apprenants avec les règles de grammaire et le processus de transformation.

B.2.10 GETARUNS

Le système *GETARUNS* (*General Text and Reference Understanding System*, Delmonte et Dibattista, 2000) est un système complet de compréhension de textes qui peut être utilisé par des apprenants de linguistique pour tester des phrases et des hypothèses. Sa grammaire LFG (§3.3.5) utilise un analyseur écrit en DCG Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2), aidé par un désambiguïsateur syntaxique qui utilise un automate (§3.3.4.1) pour un prétraitement de la phrase. *GETARUNS* construit un modèle du discours d'un texte en tenant compte de la syntaxe (Delmonte, 2004a,b). Il est également doté d'un système de raisonnement. *GETARUNS* peut être utilisé à des fins pédagogiques : Delmonte (2003) décrit des exercices d'allemand pour italophones à partir d'une version simplifiée de *GETARUNS*. Des exercices de résumé de texte à l'aide de la version complète sont décrits dans Delmonte (2004a).

B.2.11 Glosser-RuG

Glosser-R*u*G (Dokter et Nerbonne, 1998; Roosmaa et Prószéky, 1998; Nerbonne et Dokter, 1999; Nerbonne *et al.*, 2002) est un logiciel d'apprentissage de vocabulaire basé sur l'étude de textes, qui permet la désambiguïsation de partie du discours (§3.1.2). Le logiciel de démonstration traite du français pour apprenants néerlandophones, mais il existe des versions pour l'anglais pour apprenants estoniens, bulgares ou hongrois. Chaque mot est analysé et accompagné de sa traduction dans la langue maternelle de l'apprenant. Si le désambiguïsateur morphologique se trompe, l'apprenant peut aussi accéder aux autres analyses. Le logiciel dispose aussi d'un aligneur de textes bilingues, de l'accès à un dictionnaire bilingue et d'un concordancier (§3.1.7). Le logiciel est accessible sur *Internet* [3].

B.2.12 GPARS & Ms. Pluralbelle

GPARS est un projet, qui a débuté en 1978, destiné aux apprenants qui ont un retard de langage. Dès les années 1980, ces systèmes ont été enrichis d'outils de TAL (Loritz, 1992, 1995). Une première version, appelée *Ms. Fidditch*, tournait déjà sur un micro-ordinateur *Apple II* au début de la décennie 1980. L'apparition du PC a permis le portage du système dans cet environnement, avec le logiciel *Ms. Grundy* écrit en *Lisp* (Gazdar et Mellish,

3. `http://www.let.rug.nl/~glosser/`, dernier accès le 23 juillet 2006.

1989). En 1988, le système reçoit des fonds pour le développement d'un environnement d'apprentissage d'anglais langue seconde pour malentendants et devient *Ms. Pluralbelle*.

B.2.13 Grim

Grim est un environnement d'apprentissage du suédois écrit qui prévoit des outils d'exploration de la langue tels qu'un correcteur grammatical, un analyseur syntaxique (§3.3), des outils morphologiques (§3.1.1.4), un dictionnaire et un concordancier (§3.1.7). L'environnement est développé en *Java* (§2.7.5).

Le vérificateur grammatical *Granska* (Carlberger *et al.*, 2002; Knutsson *et al.*, 2002, 2003c, 2007) utilise des méthodes probabilistes et des règles grammaticales. Le cœur du système est un étiqueteur statistique (§3.1.2) basé sur les Modèles de Markov cachés (v. p. 49). Les phrases étiquetées sont ensuite analysées par des règles d'erreurs, des règles d'aide et des règles d'exception, qui analysent des combinaisons de bigrammes.

L'analyse grammaticale est effectuée par l'analyseur superficiel *Granska Text Analyzer* (GTA, Knutsson *et al.*, 2003a, §3.3.4.3), qui applique des règles de grammaire à un texte étiqueté. L'analyseur compte environ 200 règles.

B.2.14 ImPRESSions

ImPRESSions (Pujolà, 2001, 2002) est un système d'apprentissage de la lecture et l'écoute de l'anglais sur le *web*. Il offre plusieurs feedbacks différents adaptés aux différents styles d'apprentissage.

B.2.15 Systèmes de Kempen

Kempen (1992) présente trois logiciels d'aide à l'apprentissage du néerlandais. Le premier est un outil de conjugaison (§3.1.1.4) et de correction orthographique (§3.2) des verbes, qui se base sur une analyse morphologique et orthographique et donne un *feedback* immédiat lorsque l'apprenant commet une erreur. Le second système est un logiciel d'apprentissage de la grammaire, qui est doté d'un analyseur basé sur l'*Incremental Procedural*

Grammar. L'analyseur détecte les erreurs par des règles d'erreurs et par le relâchement de contraintes. Enfin, le dernier système est une aide à la rédaction, dotée d'un analyseur morphologique qui décompose les mots composés et d'un correcteur orthographique.

Pijls *et al.* (1987) et Kempen (1999) décrivent des systèmes originaux destinés à enseigner des notions de grammaire à des apprenants qui n'en ont pas. Ces outils simples et ludiques, nommés *BOUWSTEEN*, *COGO* et *TDTDT*, sont destinés à des enfants ou adolescents. Le système expert allie un générateur de phrases (§3.1.5) et un analyseur. Un composant didactique analyse les connaissances des apprenants et dispose d'une liste d'erreurs fréquentes. Un module d'enseignement gère les séquences pédagogiques et un générateur d'exercices est à disposition. Le formalisme grammatical utilisé est une version de l'*Incremental Procedural Grammar*. Les analyses sont représentées sous formes d'arbres syntaxiques. Ces arbres peuvent être manipulés par les étudiants.

La première interface présente les mots sous forme de fantômes. Certains fantômes possèdent des bras dont les extrémités possèdent des formes différentes. Les fantômes et leurs bras ont aussi des couleurs différentes. Ces formes correspondent à des formes de têtes d'autres fantômes, ce qui permet de n'autoriser que certaines combinaisons syntaxiques lorsque les pièces peuvent s'emboîter à la manière d'un puzzle. Le système peut afficher des termes grammaticaux tels que les catégories lexicales et certaines relations syntaxiques. Une deuxième interface, probablement destinée à des apprenants plus âgés, permet de construire un temple antique à l'aide d'un système de colonnes et de chapiteaux qui s'emboîtent pour représenter les relations syntaxiques.

Un dernier système ludique se base sur le jeu bien connu *Tetris* qui consiste à empiler le plus rapidement des blocs de taille et de forme différentes, qui tombent de plus en plus rapidement, afin de remplir des lignes complètes. Dans ce jeu, une phrase est affichée mot à mot en bas de la ligne. Lorsqu'un bloc apparaît, une catégorie lexicale (ou syntaxique selon le mode de jeu) est affichée et l'apprenant doit faire tomber le bloc dans un segment de ligne où apparaît un mot correspondant à la catégorie lexicale affichée.

Le *Performance Grammar Workbench* et le *Performance Grammar Trainer* sont décrits par Kempen (2004). Les deux systèmes, écrits en *Java* (§2.7.5), permettent la manipulation d'arbres. Le premier système est doté d'outils de TALN. Les arbres ne sont combinés que si l'unification est possible au nœud demandé. Quant au second, il compare des arbres statiques créés par les tuteurs et stockés dans le système à ceux créés par les appre-

nants.

B.2.16 LADDER

LADDER (Weischedel et Sondheimer, 1983) est un système d'enseignement de l'allemand écrit en LISP (Gazdar et Mellish, 1989; Loritz, 1987). La grammaire est implémentée grâce à des automates (§3.3.4.1). En cas d'erreur, des méta-règles désignent les actions et conditions permettant de franchir l'arc et de marquer l'erreur.

B.2.17 McGill Language Learning Environment

Le *McGill Language Learning Environment* (Frederiksen *et al.*, 1995) est un logiciel d'ALIAO utilisable en complément à l'enseignement classique, pour entraîner différentes fonctions communicatives, basé sur des modèles psycholinguistiques. Il entraîne les fonctions du discours dans diverses situations. Il est composé d'une base de données multimédia de documents structurés. L'apprenant choisit un texte en fonction des buts du discours. L'environnement offre des aides pour la phonologie, morphologie, syntaxe, sémantique et discours ; le système évalue l'apprenant lors de tâches simples et de productions limitées. Sinon les apprenants doivent procéder à une auto-évaluation guidée par le système, en comparant leur production à celle fournie par le système. L'enseignant peut interagir avec les apprenants. Les productions des apprenants sont évaluées par un système de reconnaissance de patrons (*pattern matching*, §3.1.1.1) basés sur la notation BNF (*Backus-Naur Form*, Wirth, 1987).

B.2.18 Système de Markosian et Ager

Markosian et Ager (1984) présentent un logiciel d'ALIAO pour l'arménien et l'arabe. Le système est inspiré d'un logiciel d'enseignement de la logique. La couverture lexicale et syntaxique est restreinte. Il dispose d'un générateur d'exercices de drill et d'un synthétiseur vocal (§3.1.4). Il tourne sur un système *DEC* et est écrit en *Lisp* (Gazdar et Mellish, 1989; Loritz, 1987). L'analyseur est basé sur une grammaire LR-1 (Knuth, 1965). Les erreurs de dérivation lexicales sont autorisées mais ne sont pas systématiquement corrigées. La description du système ne donne pas beaucoup de détails sur les techniques utilisées.

B.2.19 Système de Micarelli & Boylan

Micarelli et Boylan (1997) présentent un tutoriel intelligent destiné aux apprenants italophones de l'anglais. Le logiciel est basé sur une technique communicative de reconstruction de conversation (*Conversation rebuilding*), où le système guide l'apprenant pas à pas pour construire une phrase correcte. L'algorithme d'analyse est écrit en Lisp (Gazdar et Mellish, 1989) et la grammaire compte 124 règles.

B.2.20 Systèmes de Microsoft

Schwartz *et al.* (2004) présentent une suite de logiciels développés pour l'apprentissage des langues par *Microsoft*, construits autour du système de TAL *NLPWin* (§C.24). *NLPWin* groupe des outils de traitement lexical (*tokenizer*, segmentation de mots, analyse morphologique de mots, identification de mots composés, recherche lexicale, §3.1.1), traitement syntaxique (§3.3), extraction de forme logique, module de génération (§3.1.5) et alignement de textes bilingues (§3.1.7).

Le *Content Question Generator* se base sur l'extracteur de formes logiques, à partir desquelles des questions sont produites automatiquement. L'outil est capable de créer automatiquement des questions de compréhension d'un texte, en utilisant l'outil de résumé de textes (§3.1.8). Le *Grammar Exercise Generator* permet de générer plusieurs variations d'un même énoncé (passivation, exercices de conjugaison, etc.).

Brockett *et al.* (2006) décrivent l'utilisation d'une technique de traduction automatique statistique de syntagmes pour la correction d'erreurs d'apprenants sinophones de l'anglais (§3.1.6).

Gamon *et al.* (2009) et Leacock *et al.* (2009) décrivent le *Microsoft Research (MSR) ESL Assistant*, un prototype de recherche ciblé sur les erreurs typiques d'apprenants des pays d'extrême-orient. Basé sur l'étude de corpus, il utilise un étiqueteur (§3.1.2) et met en œuvre différents modules statistiques ciblés sur les erreurs typiques des apprenants. Puis les résultats sont filtrés par un module statistique basé sur les heptagrammes (3.3.3.3). Avec la proposition de correction, l'apprenant voit le résultat d'une recherche sur *Internet* du terme erroné de sa phrase et de la proposition de correction.

B.2.21 MIRTO et ExoGen

Lancée en 2003, *MIRTO* est une plateforme de création d'activités pédagogiques d'apprentissage des langues basée sur des outils de TAL (Lebarbé, 2004; Kraif *et al.*, 2004; Antoniadis *et al.*, 2004a,b). Elle a pour but de fournir aux enseignants la possibilité de concevoir des scénarios pédagogiques ouverts grâce à une base de textes indexée pédagogiquement (Loiseau, 2004). Des exercices adaptés aux apprenants pourront être générés automatiquement. Les réponses seront évaluées qualitativement. Les scénarios créés pourront être non linéaires. *MIRTO* est conçu sur quatre niveaux, associés à la base de données (ou éventuellement à un dictionnaire pour certains types d'exercice de conjugaison ou lexicaux), qui permettent de concevoir des scénarios :

i. le *niveau fonction*, invisible pour les utilisateurs normaux, concerne les applications de TAL, comme la segmentation du texte (§3.1.1.2), la reconnaissance de la langue (§3.1.1.3), l'analyse morphologique (§3.1.1.4) et l'étiquetage (§3.1.2) ;
ii. le *niveau script* permet d'appliquer les fonctions pour la didactique des langues. Générer des exercices lacunaires à partir d'un texte nécessitent d'identifier la langue, de segmenter le texte, d'analyse morphologique, d'étiquetage et enfin de sélection des éléments en fonction des paramètres fixés par l'utilisateur. Il existe aussi des scripts de collocations (p. 66), d'appariement lexical etc. ;
iii. le *niveau activité* consiste en la réalisation d'une activité pédagogique minimale à travers un exercice. Il s'agit de travailler une notion, de réviser une conjugaison etc. A ce niveau sont aussi définies les aides à l'apprentissage (dictionnaire, fiches pédagogique...) et les consignes et l'évaluation de l'exercice ;
iv. le *niveau scénario* permet de définir des séquences d'activités en fonction du parcours de l'apprenant. Un profil permettra de définir des parcours individualisés.

L'évaluation des réponses devrait comparer la réponse de l'apprenant à la réponse attendue sur trois niveaux :

i. le *niveau orthographique* si le mot ne peut être retrouvé. Dans ce cas, soit on corrige automatiquement, soit on fournit une liste de mots ;
ii. le *niveau morphosyntaxique*, où sont déterminés le lemme et les traits potentiels afin de fournir un diagnostic précis ;

iii. le *niveau sémantique*, où on tente de découvrir s'il existe une relation sémantique proche grâce à *EuroWordNet* (§3.5.3).

Le prototype *ExoGen* (Kraif et Ponton, 2007; Blanchard *et al.*, 2009), en cours d'implémentation, utilise une stratégie de triangulation didactique basée également sur des techniques basiques éprouvées, comme l'analyse morphologique (Blanchard, 2007). La réponse de l'apprenant est annotée avec ces outil, ainsi que la réponse attendue par le système et les deux sorties sont comparées afin de générer un feedback. Le système utilise un corpus de textes étiquetés et lemmatisés pour générer des exercices à l'aide d'exemples authentiques correspondant à un certain schéma. Le système utilise l'analyse d'erreurs du corpus FRIDA (§4.1) pour bâtir des arbres de décision afin d'établir un diagnostic.

B.2.22 Nihongo-CALI, Banzai et Robo-Sensei

Nihongo-CALI (Nagata, 1995; Kang et Maciejewski, 2000) est un logiciel d'apprentissage du japonais. Il est basé sur un analyseur LFG (§3.3.5) qui traite les phrases actives est passive.

Successeur de *Nihongo-CALI*, *Banzai* (Nagata, 1997, 2002) est un logiciel d'enseignement du japonais qui utilise un analyseur basé sur un automate à états finis (§3.3.4.1). Il est écrit en *Java* (§2.7.5) et était disponible sur le *web* [4]. Il contient 24 leçons de japonais de niveau débutant à avancé. Le système utilise un lexique, un générateur morphologique (§3.1.1.4), un segmenteur de mots (§3.1.1.2) et des analyseurs morphologique et syntaxique (§3.3) doté d'un détecteur d'erreurs. *Banzai* analyse la phrase correcte stockée dans le système. Les parties du discours de cette phrase sont étiquetées pour désambiguïser. Ensuite, *Banzai* se sert de l'analyse de la réponse attendue pour effectuer un diagnostic de l'erreur de l'apprenant. Le concepteur d'exercice peut spécifier des variantes lexicales acceptables dans la phrase modèle.

Deux types de *feedback* sont proposés à l'apprenant :

- un *feedback déductif* explique les règles de grammaire ;
- un *feedback inductif* propose des exemples en réponse à l'erreur de l'apprenant.

4. Indisponible actuellement, probablement à cause de l'existence d'une version commerciale.

Enfin, créé par la même auteure dans la continuité des autres projets, le logiciel *Robo-Sensei : Personal Japanese Tutor* (Nagata, 2009) est un logiciel commercialisé qui offre 24 leçons de japonais avec un *feedback* soigné, du niveau débutant à un niveau avancé, en complément d'un manuel écrit. Comme il existe plusieurs graphies du japonais et l'ordre des mots est relativement libre, il existe de nombreuses possibilités de phrases correctes. Le système utilise un schéma de réponse et la phrase de l'apprenant pour générer les réponses correctes ; grâce à un analyseur ascendant basé sur des règles indépendantes du contexte, il analyse ensuite les phrases correctes et la phrase de l'apprenant pour les comparer et déterminer un diagnostic.

B.2.23 NooJ

NooJ est un système de traitement de corpus (§3.1.7) dans le cadre de l'apprentissage des langues et de la linguistique (Silberztein et Tutin, 2004). Il améliore les fonctionnalités du logiciel *INTEX* développé par le même auteur, qui permet de développer des lexiques et des grammaires, notamment pour l'analyse de corpus. La gestion des concordances à retrouver dans le corpus est simple et intuitive. Les grammaires sont développées sous forme de transducteurs (§3.3.4.1). Les corpus sont annotés semi-automatiquement grâce à des grammaires simples basées sur des transducteurs, ce qui permet de générer facilement des exercices.

B.2.24 PARSER

PARSER (Bailin et Thomson, 1988) enseigne la structures syntaxique de l'anglais en identifiant les éléments des phrases qu'il génère automatiquement. Le système tourne sur PC et est écrit en *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Il génère une phrase et demande à l'apprenant d'identifier certains éléments (nom, verbe, syntagme prépositionnel). Si l'apprenant tape une mauvaise réponse, *PARSER* lui indique quel élément il a indiqué par erreur et lui indique une réponse correcte.

L'algorithme de génération (§3.1.5) se base sur les règles de réécriture de la grammaire pour constituer des phrases. Des traits sémantiques empêchent la création de phrases sans signification. Pour éviter que des phrases trop longues et complexes soient engendrées, les règles récursives sont remplacées par des syntagmes intermédiaires ou sous-types de syntagmes, qui permettent d'autoriser certaines combinaisons et d'interdire une trop grande complexité. Des syntagmes nominaux vides permettent aussi de lier des

traces, comme dans la théorie GB (§3.3.5).

B.2.25 Passive Voice Tutor, WebPVT

Passive Voice Tutor (Virvou *et al.*, 2000; Virvou et Tsiriga, 2001) est un système d'ALIAO pour l'apprentissage de la structure du passif en anglais, écrit en *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Outre des questions à choix multiples, l'apprenant doit réécrire des phrases complètes de l'actif au passif ou inversement. Le système expert contient des connaissances sur le processus de passage de la voix active au passif ainsi que sur la conjugaison des verbes. Il est aussi capable de générer de nouveaux exercices, sur la base de connaissances syntaxiques et sémantiques. Le diagnostic d'erreurs est basé sur une typologie des erreurs communément commises par les apprenants et contient des règles d'erreurs. Il génère la phrase correcte avec ses connaissances et la compare avec la phrase de l'apprenant. Il existe une version sur *Internet* nommée *WebPVT*.

B.2.26 Piléface

PILEFACE ou *Piléface* (*Programme Intelligent pour les Langues Etrangères Favorisant l'Approche Communicative de l'Enseignement*, Lelouche, 1991) est un logiciel d'enseignement du français pour la communication. L'apprenant doit apprendre à communiquer dans diverses situations, avec des registres et des conventions de politesse variés. Il comprend un module de génération et d'analyse qui corrige des erreurs syntaxiques, sémantiques et pragmatiques. Aucune précision sur le traitement des erreurs n'est fournie.

B.2.27 PLATO

PLATO (*Programmed Logic for Automatic Teaching Operations*, Clancy, 1973; Hart, 1995) est une projet au long cours de l'Université de l'Illinois qui s'est étendu du début des années 1960 à la fin des années 1970, au cours de quatre phases. Le système, fonctionnant sur un gros ordinateur, a été conçu pour l'apprentissage individualisé et interactif, pour un grand nombre d'apprenants.

Le système n'était pas exclusivement réservé à l'apprentissage des langues. Il comportait un système de communication interne, sous la forme d'une

sorte de courrier électronique et de la transmission de notes de cours. Le système avait aussi un module de gestion des apprenants et des tuteurs. La gestion des modules était très sophistiquée. L'individualisation des parcours était possible.

Faute de financement, le projet *PLATO* est mort avec l'arrivée des micro-ordinateurs. Le manque de puissance des machines, l'absence de connectivité entre elles et l'inadéquation du langage *BASIC* (§2.7.1) pour l'enseignement ne rendaient pas ces machines intéressantes, bien qu'il y ait un certain intérêt pour les capacités graphiques supérieures des micro-ordinateurs ainsi que la possibilité d'exécuter *Lisp* (Gazdar et Mellish, 1989; Loritz, 1987) ou *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2).

Les cours de français couvraient quatre semestres, avec des modules de vocabulaire, grammaire, linguistique, culture et dialogue (Demaizière et Dubuisson, 1989). Les modules entraînaient l'écrit et la lecture (notamment la reconnaissance et production de structures grammaticales). L'apprenant devait jouer un rôle de négociant en bourse ou de propriétaire de restaurant. Il avait le choix entre plusieurs actions comme faire un emprunt à la banque dans le premier rôle ou engager du personnel dans le second. La séquence pédagogique variait alors selon ces choix.

Un outil de synthèse vocale (§3.1.4) a été intégré dans les dernières versions. Un système permettait aussi aux apprenants d'enregistrer leurs productions. Des écrans tactiles et disques vidéo étaient aussi utilisés.

Le traitement des phrases complètes était fait par reconnaissance de formes (*pattern matching*, §3.1.1.1), où le concepteur devait donner la bonne réponse ainsi que les réponses incorrectes anticipées (Chapelle et Jamieson, 1983; Pusack, 1984). Il pouvait aussi indiquer les mots alternatifs ou facultatifs. Les erreurs de typographie (inversion, substitution etc.) pouvaient être anticipées.

A l'origine, *PLATO* I était simplement utilisable par un terminal attaché à l'ordinateur hôte. *PLATO* III fonctionnait dans une petite salle de terminaux. En 1969, le développement de *PLATO* IV, qui prévoyait des changements conséquents, a bénéficié d'un financement substantiel de la *National Science Fundation*, de même que le projet *TICCIT* (§B.2.36). Des prévisions voyaient le projet *PLATO* relier des milliers voire des millions de terminaux. Les possibilités graphiques des terminaux étaient très avancées pour l'époque. Les entrées tapées par les étudiants étaient traitées caractère par caractère, et non par ligne, ce qui permettait un *feedback* immédiat.

Les tutoriels étaient écrits grâce à un langage de programmation propre, *TUTOR*, spécialisé pour l'écriture de tutoriels mais présentant aussi les caractéristiques d'un langage de programmation complet.

B.2.28 Système de Pulman

Dans un petit système de dialogue à domaine restreint destiné aux apprenants de l'anglais, Pulman (1984) utilise une grammaire GPSG (Gazdar *et al.*, 1985) qu'il compile pour obtenir un réseau de transition récursif. Le système tente d'analyser la phrase aussi loin que possible et transmet le résultat à un système de diagnostic qui signale à l'apprenant quel élément il s'attendrait à trouver dans cette position. Le système tourne sous un système Vax et est écrit dans un langage dérivé de *LISP* (Gazdar et Mellish, 1989; Loritz, 1987).

B.2.29 Système de Reuer

Reuer (1999, 2000, 2003) décrit un prototype de tutoriel de l'apprentissage de l'allemand pour débutants. Le système de dialogue utilise la technique des automates (§3.3.4.1). L'analyseur utilise un vérificateur orthographique (§3.2) également basé sur des automates, et une grammaire LFG (§3.3.5) utilisant l'algorithme d'Earley (Earley, 1970) et traitant les erreurs par relâchement de contraintes. Les erreurs couvertes par le système sont les erreurs d'accord, de sous-catégorisation et de cas. Les erreurs d'omission sont traitées par des listes de catégories lexicales qui pourraient remplir la fonction manquante. Les éléments mal placés sont stockés dans le système pour être insérés plus tard à la bonne position. Les éléments superflus ou douteux ne sont pas encore traités par le système. Pour formuler son diagnostic, le système choisit l'erreur qui obtient le score minimal.

B.2.30 SAFRAN

SAFRAN (*Système d'Apprentissage du FRANçais*, Hamel, 1996; Hamel et Vandeventer, 1998; Hamel, 1998; Hamel et Vandeventer, 2000; L'haire, 2000) est un prototype de logiciel multimédia qui vise à tester l'utilisation et l'adaptation d'outils de traitement du langage pour un logiciel d'apprentissage des langues (synthèse vocale, §3.1.4, analyse syntaxique, §3.3, dictionnaire conceptuel et conjugueur automatique, 3.1.1.4). Ce projet a duré de

1995 à 1999. Le public cible est des apprenants universitaires anglophones de niveau intermédiaire à avancé avec connaissances de linguistique.

L'objectif est d'apprendre à l'apprenant de reconnaître et d'utiliser des stratégies propres aux différentes fonctions communicatives de la langue (converser, raconter, faire agir, persuader, informer et s'exprimer) à travers différents types de texte. Les tutoriels contiennent de nombreux exemples grammaticaux ou oraux qui sont liés dynamiquement aux outils d'analyse syntaxique ou à la synthèse vocale.

Tous les outils de traitement du langage sont basés sur l'analyseur syntaxique *Fips* (§5, Wehrli, 1997). L'analyseur n'est capable d'analyser que des phrases grammaticalement correctes. Il a été adapté pour afficher l'analyse de la phrase sous forme d'une analyse parenthétique simplifiée. En outre, l'analyse peut aussi être présentée sous la forme d'une interface graphique en couleur ; les mots sont écrits d'une couleur différente selon leur catégorie lexicale (déterminant, verbe, etc.) ; les différentes fonctions (prédicat, sujet, complément circonstanciel, etc.) sont marquées par des soulignements par des lignes en couleur ; si une phrase contient des subordonnées, certains mots sont soulignés sur plusieurs niveaux. Enfin, il existe une représentation sous forme d'arbre syntaxique qui reprend graphiquement les étiquettes de la structure parenthétique simplifiée.

SAFRAN utilise le synthétiseur vocal *FipsVox* (Gaudinat et Wehrli, 1997; Gaudinat et Goldman, 1998), qui permet d'écouter n'importe quelle phrase. Un outil de phonétisation permet de lire la transcription de la phrase dans l'alphabet phonétique international (API).

Un dictionnaire conceptuel *FRTool* (Hamel *et al.*, 1995; Singleton *et al.*, 1998) permet de favoriser l'apprentissage de vocabulaire par un apprenant. Le dictionnaire fournit une représentation non linéaire des connaissances. Les mots peuvent être écoutés grâce à la synthèse vocale. Enfin, un conjugueur permet de consulter la conjugaison de tous les verbes à tous les modes et à tous les temps. Les formes verbales peuvent également être prononcées par le synthétiseur.

Divers modules d'exercices sont offerts à l'apprenant. Pour entraîner l'expression orale, des exercices proposent à l'apprenant d'enregistrer un mot et de comparer sa production avec celle du synthétiseur vocal. D'autres types d'exercices sur les mots sont décrits chez L'haire (2000).

Des exercices de grammaire demandent à l'apprenant de trouver divers éléments lexicaux ou de repérer des fonctions syntaxiques dans un texte.

Un troisième type d'exercice grammatical demande à l'apprenant de repérer les antécédents des pronoms ou les mots avec lesquels les participes sont accordés. Par manque de robustesse de l'analyseur, les textes utilisés pour les exercices ne sont pas entièrement traités automatiquement : les étiquettes lexicales et les fonctions syntaxiques doivent être révisées par un expert humain ; quant au repérage de l'antécédent des pronoms ou des participes passés, il doit être fait à la main.

La première version du tutoriel de *SAFRAN* a été développée pour le système d'exploitation *Windows* 3.1. Les exemples des tutoriels étaient stockés dans une base de données et liés aux tutoriels par un code propriétaire très simple. Une seconde interface a été développée sous *Windows* 95 pour afficher les tutoriels codés en *HTML*. Les tutoriels peuvent également être consultés sur *Internet* [5]. Un éditeur *HTML* permet à l'enseignant d'écrire relativement facilement des tutoriels et d'insérer des exemples en vérifiant directement le bon fonctionnement de l'analyseur ou de la synthèse vocale.

B.2.31 SigmaStar

SigmaStar (Ott *et al.*, 2005) est un projet de recherche qui vise à développer des applications d'apprentissage de l'allemand niveau débutant à travers des jeux sur téléphones mobiles. Le projet utilise l'analyseur morphologique *MORPH* (Hanrieder, 1994, §3.1.1.4) pour la génération et l'analyse de textes. *SigmaStar* utilise également le dictionnaire sémantique *GermaNet* (Lemnitzer et Kunze, 2002), dérivé de *Wordnet* (§3.5.3) etc. Tout le dispositif est programmé en *Java* (§2.7.5) ; les fichiers sont de taille considérable (40 MB) mais le logiciel peut être ensuite utilisé sans connexion téléphonique.

B.2.32 The Spanish Verb

The Spanish Verb (Soria, 1997) est un logiciel d'apprentissage de la conjugaison de l'espagnol doté d'un analyseur morphologique (§3.1.1.4) et d'un module de diagnostic.

5. `http://www.latl.unige.ch/safran/`, dernier accès le 23 juillet 2006

B.2.33 SPANLAP

SPANLAP (Jehle, 1987) est un logiciel d'apprentissage de l'espagnol qui simule un dialogue dans le but d'apprendre à communiquer. Le logiciel simule une conversation de base en espagnol en amenant les apprenants à se corriger eux/elles-mêmes. Le logiciel procède à une analyse morphologique, syntaxique et sémantique. Il est basé sur un lexique de 3000 mots environ. L'algorithme est écrit en *Lisp*. Il fonctionnait sur un système VAX.

B.2.34 STyLE

STyLE (*Scientific Terminology Learning Environment*, Boytcheva *et al.*, 2000, 2001, 2004; Angelova *et al.*, 2002; Vitanova, 2004) est un environnement sur la toile pour l'acquisition de terminologie scientifique et financière en anglais, qui fait partie du projet *LARFLAST* (*Learning Foreign Language Scientific Terminology*).

Les connaissances de la base sont encodées par des graphes conceptuels construits à la main. Ces graphes sont ensuite automatiquement traduits en représentations *Prolog* (§3.3.4.2) pour les moteurs d'analyse et d'inférence. *STyLE* offre sept types d'exercices : QCM, textes à trous, mots croisés, phrase à remettre en ordre, exercices d'appariement, exercices de mise en ordre et textes libres. Le modèle de l'apprenant est mis à jour grâce à un dialogue entre l'apprenant et le système (Dimitrova *et al.*, 2000).

Le module de diagnostic est basé sur l'analyseur *Parasite* (Ramsay, 2005), qui représente la phrase comme une forme logique. Cette représentation est comparée avec deux phrases type entrées dans le système, l'une avec le contenu minimal pour obtenir un score correct, l'autre avec le contenu maximal, qui regroupe toutes les informations disponibles. Le système utilise aussi la LSA (3.1.8 p. 75) pour mesurer la pertinence des documents collectés sur la toile.

B.2.35 Thai Learning System

Le *Thai Learning System* (TLS, Dansuwan *et al.*, 2001) est un logiciel d'apprentissage du thaï sur *Internet*. Il est destiner à enseigner l'emploi des cas. L'analyseur est écrit en *DCG Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) et vérifie la consistance sémantique de la phrase

à l'aide de traits sémantiques simples. Les sorties sont traitées par *Perl* (§2.7.5) pour afficher un arbre syntaxique si la phrase est correcte, ou un diagnostic d'erreurs. L'article ne détaille pas l'étendue de la couverture ni le type d'erreurs qui peuvent être détectées.

B.2.36 TICCIT et CLIPS

TICCIT (*Time-Shared, Interactive, Computer-Controlled Information Television*, Hendricks *et al.*, 1983; Jones, 1995) est un projet qui a débuté en 1971 à l'Université Brigham Young. Comme son nom l'indique, le projet combinait la télévision, les bandes audio et l'ordinateur. La stratégie d'enseignement incluse dans le système contraignait fortement le classement des leçons dans le système, afin de faciliter l'apprentissage par une présentation uniforme. L'apprenant avait de nombreuses possibilité de contrôler le système grâce à un clavier *ad hoc* pour demander l'affichage de règles, exemples, conseils, etc. et pour contrôler le niveau de difficulté. Dans une version plus récente tournant sur micro-ordinateur, cette rigidité de conception du matériel d'enseignement a quelque peu été assouplie.

Dès le début des années 1990, les modules de TICCIT ont été transférés sur des programmes tournant sur PC avec MS-DOS et portent désormais le nom de *CLIPS* (*Computerized Language Instruction and Practice Software*). Il existe actuellement des versions sous *Windows*, *MacIntosh* et des modules sur *Internet*.

B.2.37 VERBCON

VERBCON est un logiciel sur PC écrit en Pascal (Bailin et Thomson, 1988). Son but est d'enseigner aux apprenants l'utilisation de formes verbales de l'anglais écrit. Les exercices consistent à mettre au temps correct un verbe à l'infinitif. La réponse correcte est donnée au bout de trois essais. Le système analyse la forme verbale de chaque réponse de l'apprenant. De plus, en cas d'erreur, il donne des indices sur la bonne réponse. A la fin de chaque exercice, il dresse la statistique des temps verbaux utilisés. Pour reconnaître les verbes, *VERBCON* dispose d'un analyseur morphologique (§3.1.1.4) qui se sert des formes de base (infinitif, indicatif troisième personne du singulier, prétérit, participes présent et passé) pour déduire les autres formes. L'analyseur est basé sur un algorithme d'analyse situation-action (Winograd, 1983, 401 ss.), qui prévoit d'appliquer des actions lorsqu'il rencontre une certaine situation ; les actions peuvent être de rechercher d'autres propriétés du verbe

ou l'attachement d'étiquettes grammaticales au verbe. L'analyseur utilise une stratégie déterministe.

En outre, le système dispose d'une grammaire des formes verbales. Des explications et des exemples illustrent l'utilisation des temps, modes, aspects et voix.

B.2.38 VINCI

VINCI (Levison et Lessard, 1996; Levison *et al.*, 2000; Levison et Lessard, 2004; Lessard et Levison, 2007) [6] est un système de génération de textes (§3.1.5) capable de créer des phrases selon les spécifications de l'utilisateur, développé pendant plus de 20 ans (Thomas *et al.*, 2004a). Il est écrit en DCG-Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) et détecte les erreurs à l'aide de règles d'erreur. Le système est multilingue et traite l'anglais et le français ; des tests ont également été faits sur l'espagnol, l'italien, le russe et le chinois. Les lexiques peuvent fournir une traduction d'un mot et contiennent des relations de synonymie, antonymie etc.

Levison et Lessard (1996) décrivent les exercices qui peuvent être générés à partir du lexique : exercices de vocabulaire, de morphologie, textes à trous, transformations, compréhension de phrases simples, etc. *VINCI* peut également générer des contes de fées, des calembours, des charades, de courts poèmes et des questions de compréhension de textes (Lessard et Levison, 2007). Les histoires des contes de fée sont basés sur des scénarios typiques et des personnages récurrents. Il peut être couplé à un moteur de synthèse vocale (Thomas *et al.*, 2004a, §3.1.4). Les réponses sont analysées grâce à une comparaison de phrases avec la réponse attendue. Les erreurs sont détectées par une mesure de distance entre la chaîne attendue et la chaîne écrite par l'apprenant. Des règles phonologiques et morphologiques complètent le module de diagnostic. *VINCI* peut également générer des jeux de mots et des poèmes humoristiques dans un but didactique.

B.2.39 Visual Active Syntax Learning (VISL)

Visual Active Syntax Learning (*VISL*, Bick, 2004, 2005) est une interface graphique d'enseignement de l'analyse grammaticale sur *Internet*, disponible pour 7 langues. L'analyseur hybride à niveaux multiples utilise le formalisme de la grammaire de contraintes (*Constraint Grammar*, Karlsson,

6. `http://www.cs.queensu.ca/CompLing/`, dernier accès le 26/9/07.

1990). D'autres descriptions syntaxiques peuvent être utilisées à l'affichage. Des jeux grammaticaux permettent un enseignement ludique de la grammaire. Les applications d'enseignement sont programmées en *Java* et en *Perl* (§2.7.5).

B.2.40 VP^2

Le logiciel VP^2 (Schuster, 1986) est un tuteur pour l'enseignement de l'anglais langue seconde. Il est centré sur l'acquisition des constructions verbes + particule et verbe + syntagme prépositionnel. Le système est implémenté en Prolog en utilisant les DCG (§3.3.4.2, Pereira et Shieber, 1987; Covington, 1994) sur un système VAX. Si l'analyse de la phrase de l'apprenant échoue, le système tente de l'analyser avec les règles de la langue maternelle.

La grande originalité de VP^2 est de se baser sur une grammaire explicite de la langue maternelle de l'apprenant, afin de détecter les interférences avec la langue seconde. Le système dispose de l'arbre d'analyse syntaxique de la phrase à traduire dans les deux langues. Il compare ces arbres avec le résultat de l'analyse de la phrase de l'apprenant, compare les structures argumentales des verbes correspondants dans les deux langues et cherche à déduire la cause de l'erreur de l'apprenant pour adapter son *feedback*. Ainsi, pour la phrase suivante, l'apprenant doit traduire la phrase (76a) :

(76) a. Pensaba en estudiar Español.
b. *I thought in studying Spanish.
c. I thought of/about studying Spanish
d. Je songeais à étudier l'espagnol / je pensais (à) étudier l'espagnol.

Il répond (76b) alors que les formes correctes sont en (76c). Le système signalera que *en* se traduit souvent par *in*, mais que ce n'est pas le cas ici.

Le système traite les particules manquantes, les prépositions incorrectes et les prépositions superflues. Si la traduction est incorrecte et les mots anglais ne correspondent pas, le système demande à l'apprenant de recommencer.

Le système pourrait être étendu pour traiter les erreurs d'accord ou de temps verbaux ainsi que de structures de phrase. Il est toutefois à noter que la comparaison de phrases ne sera plus d'utilité dans le cas de l'accord ; en

revanche, la technique de comparaison des temps est utilisable pour l'utilisation des temps verbaux.

B.2.41 Why2-Atlas et ITSPOKE

Why2-Atlas (VanLehn *et al.*, 2002; Rosé *et al.*, 2002) n'est pas un système d'ALIAO mais un tuteur intelligent de physique. Le but est de dialoguer avec un apprenant pour lui apprendre à raisonner en utilisant des phrases en langue naturelle. Le tutoriel vise à aider l'apprenant à construire un raisonnement qualitatif en éliminant les mauvaises conceptions, à travers un dialogue. L'analyseur construit une représentation logique de la phrase, puis un module du discours bâtit les étapes du raisonnement à l'aide d'un automate implémentant le formalisme *Knowledge Construction Dialogues* (KCD). Un module tuteur gère les actions pédagogiques. Enfin, un module du dialogue gère l'interaction avec l'apprenant. L'analyseur est construit sur une grammaire hors contexte augmentée d'éléments de LFG (§3.3.5) et de Grammaire Fonctionnelle (*Functional Grammar*, Halliday, 1985) et permet de traiter les erreurs, comme les phrases incomplètes et les erreurs d'unification nécessitant un relâchement des contraintes. *ITSPOKE* (*Intelligent Tutoring SPOKEn dialogue system*, Litman et Silliman, 2004; Ward et Litman, 2005; Forbes-Riley et Litman, 2005, 2006; Forbes-Riley *et al.*, 2006; Litman et Forbes-Riley, 2006) est une version orale de *Why2-Atlas*, où les phrases de l'apprenant sont reconnues par un système *Sphinx-2* et les réponses du systèmes sont synthétisées (§3.1.4) par le système commercial *Cepstral*.

B.2.42 XTrA-TE

XTrA-TE (*English Chinese Sentence TRAnslator to Teach English*, Chen et Kurtz, 1989; Danna, 1997) est un tuteur d'apprentissage de l'anglais pour apprenants chinois, qui doivent traduire des phrases d'une langue à l'autre. Le diagnostic est fait par passes successives en relâchant les contraintes syntaxiques et sémantiques, ainsi que l'ordre obligatoire des mots. Si ces tentatives échouent, le logiciel tente encore une traduction mot à mots.

B.3 Micromondes

B.3.1 Projet Athena

Né en 1983 et terminé officiellement en 1994, le projet *Athena Language Learning Project* au Massachusetts Institute of Technology (MIT) vise à appliquer les principes de l'apprentissage communicatif des langues aux nouvelles technologies de TAL et multimédia. Les langues concernées sont le français, l'espagnol, l'allemand et le russe (Kramsch *et al.*, 1985). De nombreux logiciels développées sur *MacIntosh* sont issus de ce projet.

Lingo (*Language Instruction through Graphic Operations*, Murray, 1995) est un micromonde graphique où l'apprenant doit diriger un esprit frappeur pour mettre un appartement sens dessus dessous. *No recuerdo* est un environnement vidéo interactif qui utilise le même outil de TAL pour l'espagnol. L'activité consiste à guider un scientifique amnésique. *Topicks* (Kramsch *et al.*, 1985) est un logiciel d'étude de vocabulaire allemand qui permet aussi de simuler des dialogues avec des phrases complètes. *Reverse Eliza* (Kramsch *et al.*, 1985) est un système de dialogue où l'apprenant doit tirer les vers du nez d'un informateur peu coopératif. Ainsi, des questions directes n'obtiendront pas de réponses, tandis que des réactions empathiques, des questions indirectes, réactions, paraphrases, interprétations, inférences, etc. obtiendront des réponses.

Ces projets fonctionnent grâce à une série complexe d'outils de TAL, au niveau morphologique (§3.1.1.4), syntaxique (§3.3), sémantique et pragmatique, en analyse comme en génération (§3.1.5). Les actions à accomplir sont alors déterminées par une représentation conceptuelle indépendante de la langue. Le système de génération est utilisé pour simuler un dialogue ou pour la correction d'une production erronée de l'apprenant (Felshin, 1995). Certains logiciels issus du projet sont également dotés d'outils de reconnaissance vocale (§3.1.3) destinés à aider à la prononciation à l'aide d'une comparaison de spectrographes.

B.3.2 BELLOC

BELLOC (Chanier *et al.*, 1992) est un système intelligent d'apprentissage du français basé sur une situation de résolution de problèmes. L'apprenant doit découvrir les relations familiales entre plusieurs personnes en posant des questions au système. Le contexte des dialogues est restreint de sorte

à rendre possible une représentation sémantique des connaissances. Le système comprend un modèle de l'apprenant basé sur une représentation des connaissances sous forme de règles applicables (*applicable rules*) qui guident le dialogue entre le système et l'apprenant. Si le système détecte une erreur, il cherche à découvrir quelle règle l'apprenant a mal appliquée en lui proposant des phrases correctes ou incorrectes et en lui demandant de les évaluer. Sur demande de l'apprenant, il peut lui fournir une courte explication grammaticale. Un modèle de l'apprenant (§2.7.4.2) a été prévu mais n'a pas été implémenté.

B.3.3 BRIDGE et MILT

BRIDGE (Criswell *et al.*, 1992; Holland *et al.*, 1993; Holland, 1994; Sams, 1995; Weinberg *et al.*, 1995; Kreyer et Criswell, 1995; Kaplan et Holland, 1995; Dorr *et al.*, 1995; Kaplan *et al.*, 1998), est un logiciel d'enseignement de l'allemand avec une extension pour l'arabe. Son successeur MILT (*Military Intelligent Language Tutor*) enseigne l'arabe. Les deux logiciel sont basés sur des micromondes qui ont pour but d'entraîner les membres du renseignement de l'armée pour l'interrogatoire de prisonniers. Ils sont développé à l'*Army Research Institute* en Virginie. BRIDGE et MILT sont basé sur un analyseur basé sur la théorie GB (§3.3.5), qui accepte les ellipses et permet de corriger les erreurs grammaticales.

Basé sur un environnement graphique simple, BRIDGE vise à entraîner la description des lieux et l'orientation spatiale. Les activités sont relativement simples, notamment des cartes à cliquer. La progressivité des exercices est soit fixe, soit basée sur les performances, au choix de l'enseignant. L'apprenant dispose d'outils graphiques pour évaluer ses performances et sa progression.

MILT est doté d'un module d'analyse sémantique du dialogue parlé et écrit (Holland *et al.*, 1999), en simulant l'interrogatoire d'un prisonnier et le remplissage d'une fiche de renseignements. L'apprenant interagit avec un agent animé. Un micromonde piloté par des phrases demande à l'apprenant de découvrir des documents dans une pièce. Même si MILT est conçu pour des tâches de type militaire, il peut être utilisé également par des apprenants universitaires.

Les exercices écrits sont gérés grâce aux LCS (§3.4.2), qui permettent de comparer la réponse de l'apprenant avec la réponse stockée dans le système. MILT est doté d'un système commercial de reconnaissance vocale en parole discontinue (Kaplan *et al.*, 1998). En outre, 70 phrases sont enregistrées en

parole continue, ce qui contourne le caractère artificiel d'une prononciation mot à mot mais est par contre très sensible aux erreurs et à la variation.

B.3.4 FLUENT

Développé au *Massachusetts Institute of Technology* de Boston sur *MacIntosh*, *FLUENT* (*Foreign Language Understanding Engendered by Naturalistic Techniques*; Hamburger et Hashim, 1992; Hamburger, 1995; Schoelles et Hamburger, 1996; Reeder *et al.*, 1999) est un micromonde graphique dirigé par écrit pour l'apprentissage de l'espagnol. Il est basé sur le même analyseur que le logiciel *Lingo* du projet Athena (§§B.3.1, 3.3.5) bien que n'étant pas un logiciel issu de ce projet. FLUENT est construit sur les bases de l'apprentissage par immersion en représentant des environnement familiers comme un bureau ou une cuisine. Le contrôle de l'environnement est laissé tantôt au tuteur, tantôt à l'apprenant. Le but est d'apprendre à réaliser des actions et à donner des ordres, à poser des questions et à y répondre. Les dialogues gèrent le *feedback* des erreurs en appelant à l'autocorrection. Le fait d'utiliser un domaine restreint permet de limiter les problèmes dus à l'utilisation d'un analyseur. Par exemple, les questions globales appellent une réponse en un mot (*oui/non*) et les questions partielles (ou questions-*qu*) un syntagme nominal. Schoelles et Hamburger (1996) décrivent l'outil de création d'exercices.

B.3.5 Herr Komissar

Développé par *amber productions* sur *MacIntosh*, *Herr Kommissar* (DeSmedt, 1995) est un jeu commercial de type micromonde, où l'apprenant joue le rôle d'un policier et interroge des suspects dans le cadre d'une enquête criminelle. Le système de dialogue est limité à un sens, ce qui fait que le système ne doit pas extraire de nouveaux éléments d'information à partir des phrases de l'apprenant. Le moteur d'analyse est basé sur une technique *predication-driven parsing* qui part du verbe et cherche les arguments qui lui sont attachés. Le lexique couvre 2300 mots. Le module d'interprétation permet de traiter les paraphrases, l'ellipse, l'anaphore et la déixis. Les connaissances du système sont réalisées avec un formalisme sémantique appelé *Knowledge Representation System* (KRS), qui fait intervenir une ontologie hiérarchique de 2500 concepts.

Le système fait intervenir un policier quand l'apprenant commet une erreur et suggère une correction. Il dispose d'un synthétiseur vocal (§3.1.4)

qui peut être activé optionnellement. Les actions de l'apprenant sont enregistrées et un logiciel externe permet aux enseignants d'en tirer profit. Un logiciel auteur permet de réaliser d'autres énigmes.

B.3.6 Système de Menzel & al.

Heinecke *et al.* (1998); Menzel et Schröder (1998a,b, 1999); Foth *et al.* (2000); Menzel (2004); Foth et Menzel (2006) décrivent un prototype de système de diagnostic destiné à tester l'emploi des prépositions allemandes. L'analyseur est basé sur les grammaires de dépendance (*Weighted Constraint Dependency Grammar*, §3.3.5) et a une couverture limitée au traitement du passif et de subordonnées simples. Dans une version ultérieure, la grammaire comprend environ 1000 règles pour l'allemand. Les performances de l'analyseur sont améliorées par un pré-traitement par l'étiqueteur probabiliste *TnT* (Brants, 2000, §3.3.3.3), qui aide à choisir les bonnes alternatives. Les propositions relatives, les verbes modaux, les négations et la coordinations ne sont pas traités. La grammaire contient 220 contraintes, qui ont une pondération entre 0 et 1. Plus le poids est proche de 1, plus la contrainte peut facilement être violée (Schröder *et al.*, 2000). Ces poids sont ensuite multipliés et l'analyse la plus plausible est celle qui obtient le meilleur score.

Le système d'apprentissage est basé sur un micromonde statique. L'apprenant doit décrire un tableau à un vieux peintre devenu aveugle pour qu'il se remémore s'il a peint le tableau ou non. Un système de diagnostic d'erreurs interagit avec la représentation du micromonde. Un perroquet signale les erreurs à l'apprenant.

Un composant sémantique agit dans les deux sens, tant au niveau de la construction de la représentation dans un système de dialogue que pour désambiguïser.

B.3.7 SAMPRAS

SAMPRAS est un micromonde destiné à l'apprentissage du français langue étrangère (Michel et Lehuen, 2002, 2004). Il consiste à réaliser une recette de cuisine avec l'aide d'un partenaire, le chef, qui s'exprime par écrit à travers des bulles de bande dessinée. Pour l'activité de compréhension, l'apprenant doit manipuler les instruments, meubles et ingrédients du micromonde. Pour l'activité de production, l'apprenant doit donner des instructions par écrit en langue naturelle au chef. L'apprenant est amené à reconnaître ses fautes

(*noticing*, Cross, 2002, §2.3) ou à négocier le sens avec la machine. Le système d'inférences (§3.4.4) peut identifier des mots inconnus (*mettre le truc au frigo*) et demander de préciser. Il vérifie également l'état de l'interface et si l'action demandée est possible ; le chef ne pourra pas mettre des ingrédients au frigo si celui-ci n'est pas ouvert.

B.3.8 SHRDLU

SHRDLU (Winograd, 1972) est un micromonde simple où l'utilisateur dialogue avec l'ordinateur pour lui faire déplacer des objets en utilisant des commandes en langage naturel. Il s'agit en fait d'un des premiers programmes d'intelligence artificielle et de micromonde à commande en langue naturelle. Le objets sont dessinés sur un écran graphique. Les commandes sont analysées grâce à des réseaux de transition augmentés (§3.3.4.1) implémentés en *LISP* (Gazdar et Mellish, 1989) avec un lexique de 200 mots. Une interface sémantique vérifie la cohérence des phrases et des ordres par rapport à l'état du système.

B.3.9 Spion et Syncheck

Développé à la Miami University of Ohio, *Spion* (Molla *et al.*, 1989; Sanders et Sanders, 1995a,b) est un micromonde d'apprentissage de l'allemand né en 1981, où l'apprenant doit guider un espion dans Berlin, à l'époque de la guerre froide. La version finale date de 1987. Les phrases reconnues par le système sont limitées à l'impératif. Le système, qui joue le rôle de l'espion Robotky, décrit des objets et des événements dans la ville, uniquement par des phrases en allemand et sans composante graphique. L'apprenant doit demander des informations à l'espion, le faire se déplacer et lui faire faire des actions. Le système informe l'apprenant s'il ne connaît pas des mots ou si des actions qui ne font pas sens ou ne sont pas possibles logiquement sont ordonnées. Si la phrase n'est pas grammaticalement correcte, un message demande de reformuler. L'apprenant peut demander la traduction d'un mot simplement en écrivant celui-ci suivi d'un point d'interrogation. Un composant sémantique simple vérifie la cohérence des phrases.

Syncheck est un analyseur qui est destiné à une couverture large de l'allemand (Sanders et Sanders, 1987; Sanders, 1991), sans couvrir les erreurs de type sémantique. Les erreurs de morphologie et de syntaxe sont traitées par une grammaire d'erreurs tandis que des heuristiques traitent des erreurs fréquentes chez les apprenants. Il n'a que peu de points en commun avec

Spion mais résulte d'une longue recherche entamée avec ce projet.

B.3.10 Die Sprachmaschine

Die Sprachmaschine (Harroff, 1986) est un micromonde d'apprentissage de l'allemand écrit en langage LOGO (Papert, 1970; Harvey, 1986, §2.7.3). Le but est de construire une phrase complète en allemand à partir d'une liste de mots. Le générateur construit ensuite une phrase en expliquant sa démarche. Les apprenants peuvent poser des questions au programme en allemand en utilisant des phrases très restreintes telles que : *quelle est la forme du verbe* spielen *?*

B.3.11 Te Kaitito

Te Kaitito est un logiciel de dialogue pour l'apprentissage du maori (Vlugter *et al.*, 2004). L'apprenant doit dialoguer avec le système selon un scénario pré-établi, mais qui lui laisse une certaine autonomie dans l'initiative du dialogue. Le système est doté d'un diagnostic d'erreurs grammaticales et sémantiques pour vérifier la cohérence du dialogue.

B.3.12 Wo ist Hans?

Wo ist Hans? (*Où est Hans*, Ward *et al.*, 1999) est un petit logiciel pour l'apprentissage de l'allemand. Il est implémenté en C++ et utilise la technique des automates. L'apprenant doit retrouver Hans, son correspondant allemand, dans Berlin. Une artiste est prête à dessiner un portrait de Hans. L'apprenant doit interroger la grand-mère de Hans qui lui décrit son petit-fils quand il était petit. Une diseuse de bonne aventure décrit son habillement actuel.

B.4 Systèmes intelligents

B.4.1 ALICE

ALICE (*A Little Intelligent Conjunction Expert*) est un logiciel expérimental destiné à enseigner la traduction de conjonctions temporelles et causales d'une langue vers une autre (Cerri, 1989; Cerri *et al.*, 1992). Les langues traitées sont l'anglais, l'italien et le français. La connaissance est codée sous formes de concepts discriminants indépendants de la langue et leurs réalisation dans les différents langues. Les éléments, indépendants ou dépendants de la langue, sont connectés entre eux dans des réseaux multilingues codant les différents schémas syntaxiques.

Le logiciel est implémenté en Common LISP (Gazdar et Mellish, 1989; Loritz, 1987) sur des ordinateurs *VAX* et *SUN* puis *Macintosh*. Le cour du logiciel est constitué par le système *Nobile* d'acquisition de connaissances. Celui-ci est basé sur les *Knowledge Representation System* (KRS, §3.4.4), qui représentent les connaissances de manière explicite. Les GTE (Van Marcke, 1990, §2.7.4.3) permettent de décrire les activités en termes de tâches et de méthodes. Le système est capable de confirmer son diagnostic et de le valider par un dialogue avec l'apprenant.

B.4.2 ALICE-chan

Développé à l'Université Carnegie Mellon à Pittsburgh, ALICE-chan (*Automated Language Instruction/Curriculum Environment* ; *chan* est un suffixe diminutif en japonais ; Levin et Evans, 1995) est un logiciel basé sur le formalisme LFG (§3.3.5) avec environ 100 règles de grammaire et des règles d'erreur. Il est développé pour des apprenants de japonais de première ou deuxième année. Il n'a rien à voir avec le projet ALICE (§B.4.1).

Le système permet d'utiliser plusieurs alphabets concurrents pour le japonais, y compris la translittération en caractères latins. Le lexique compte 1500 mots. La segmentation du texte et l'analyse morphologique sont effectuées d'une seule traite à travers un automate. L'analyseur utilise l'algorithme LR de Tomita (1986) en LISP (Gazdar et Mellish, 1989). En cas d'échec, les phrases sont comparée à une phrase correcte stockée dans une base de données sous forme de f-structures. Le *feedback* n'est pas basé sur des termes grammaticaux de sujet et objet mais d'*acteur* et *objet de l'action*. Comme le japonais s'écrit sans espaces, un travail de segmentation et

d'analyse morphologique doit d'abord être opéré par des automates non déterministes (§3.3.4.1). Attendu que l'ordre des mots en japonais est libre, un analyseur doit se baser sur les cas pour déterminer les fonctions. Cependant, les erreurs de cas sont très fréquentes chez les apprenants. *ALICE-chan* active une série d'heuristiques pour déterminer les fonctions et les rôles. Les structures syntaxiques couvertes par l'analyseur sont sommaires. La grammaire compte environ 100 règles ainsi que des règles d'erreurs. L'analyseur peut aussi désambiguïser des phrases avec l'aide de l'apprenant, de manière interactive, en proposant des frontières de syntagmes.

B.4.3 ALLES

ALLES (*Advanced Long-distance Language Education System*, Schmidt *et al.*, 2004) est un logiciel d'apprentissage à distance pour apprenants avancés de l'anglais, espagnol, allemand et catalan. Un module de reconnaissance vocale (§3.1.3) sert uniquement pour la transcription de la parole, pas pour un diagnostic. Des outils de statistique mesurent la richesse des productions des apprenants en utilisant une analyse superficielle (§3.3.4.3) pour extraire des informations linguistiques, syntaxiques et des marques de discours. Des outils d'extraction de l'information utilisent des techniques de moteurs d'indexation pour retrouver des descripteurs déterminés par des humains dans les textes des apprenants.

Enfin, les erreurs grammaticales sont traitées par deux méthodes. L'analyseur *KURD* (*Kill, Unify, Replace, Delete*, §3.3.4.3, Carl *et al.*, 1997) se base sur une analyse morphologique et utilise un système de règles comprenant une description et une action. Des règles d'erreurs spécifiques sont décrites. D'autre part, l'algorithme *K-DIFF* procède à une reconnaissance de patrons (§3.1.1.1) sophistiquée pour comparer la phrase de l'apprenant avec des réponses possibles stockées dans le système.

B.4.4 Arabic ICALL

Arabic ICALL (Shaalan, 2005) est un tutoriel multimédia intelligent pour l'apprentissage de l'arabe qui comporte plusieurs composantes de TAL. Un analyseur morphologique basé sur des automates (§3.3.4.1) peut décomposer un mot en racine et affixes et déterminer les erreurs éventuelles. L'analyse morphologique peut être utilisée dans le cadre d'exercices à trous.

L'analyseur syntaxique est écrit en DCG Prolog (Pereira et Shieber,

1987; Covington, 1994, §3.3.4.2). L'arabe est une langue difficile à analyser : la syntaxe est complexe à cause d'un ordre relativement libre ; l'absence de voyelles à l'écrit pose de nombreux problèmes ; les pronoms personnels sont élidés. *Arabic ICALL* est limité à une langue littéraire relativement contrainte et n'accepte pas les signes diacritiques pour les voyelles, qui ne sont pas systématiquement notés. Les erreurs orthographiques ne sont pas prises en compte. Un module de *feedback* détecte les erreurs en appliquant des règles heuristiques.

B.4.5 ARCTA

Le projet ARCTA (*Aide à la Rédaction et à la Correction de Textes Anglais de Francophones*) a été réalisé au Laboratoire de Traitement du Langage et de la Parole de l'Université de Neuchâtel (Kübler et Cornu, 1992; Tschichold *et al.*, 1994; Kübler, 1995; Cornu *et al.*, 1996; Cornu, 1997). Développé sous *Windows*, il comporte une série d'outils d'aide à la rédaction en anglais, dont un correcteur grammatical, des aides et un identificateur de problèmes potentiels, à l'aide d'automates (§3.3.4.1).

Les aides sont des dictionnaires divers, des fiches grammaticales, une liste de mots difficiles (faux amis, ...), un conjugueur et un traducteur d'expressions figées. L'identificateur de problèmes potentiels met en évidence les mots qui sont une source d'erreurs fréquentes chez les francophones. L'emploi de chaque mot peut être vérifié.

B.4.6 ArtCheck

ArtCheck (Sentance, 1997) est un système d'ALIAO destiné à l'apprentissage de l'usage du déterminant en anglais. Il est doté d'un système de diagnostic d'erreurs. Le système est basé sur un analyseur tabulaire (*chart parser*, Winograd, 1983; Wehrli, 1997) et des règles indépendantes du contexte (§3.3.2.1) ainsi que des règles morphologiques. Les erreurs sur les articles sont détectées à l'aide d'un module spécifique.

B.4.7 Artificial Co-Learner

Artificial Co-Learner (Greene *et al.*, 2004) est un logiciel destiné aux écoles primaires en Irlande. Les apprenants sont accompagnés par un condis-

ciple artificiel, qui commet parfois des erreurs. L'apprenant peut aussi corriger son collègue artificiel, ce qui améliore la base de connaissances du système. Le système enseigne à distinguer les vrais amis des faux amis entre l'anglais et l'allemand. Il se base sur une liste de mots et des dictionnaires bilingues ainsi que sur un outil de traitement de corpus (§3.1.7) et un lemmatiseur (§3.1.1.4).

B.4.8 Automated German Tutor

Automated German Tutor (Weischedel *et al.*, 1978) est considéré comme le premier système d'ALIAO. Il existe peu de description sur le côté pédagogique. L'analyse des erreurs est basée sur des réseaux de transition augmentés (§3.3.4.1) avec des règles spécifiques aux erreurs est décrit chez Weischedel *et al.* (1978) et Weischedel et Black (1980). Certains arcs (*failable arcs*) peuvent être traversé même si une condition n'a pas été remplie grâce au relâchement de contraintes. Le lieu de franchissement de l'arc est alors gardé en mémoire pour générer un *feedback* adéquat.

Le système est doté d'une analyse sémantique qui traite les informations données (présuppositions) et infère les nouvelles informations de la phrase. Même s'il y a des erreurs syntaxiques, le système est capable de construire l'information grâce à un contexte restreint (vocabulaire et syntaxe limités, traitement de textes simples et connus d'avance dans un système non ouvert). Le système peut faire des inférences d'après un procédé relativement simple et vérifier si ce que l'apprenant a compris est cohérent avec la représentation des connaissances tirée à partir de la question.

B.4.9 AZALEA

AZALEA est un système d'ALIAO en ligne [7] développé par la société japonaise *Sunflare* (Tokuda et Chen, 2004). Ils utilisent un système de modèles de phrases basé sur un réseau de transitions chargé d'accepter ou rejeter des phrases (§3.3.4.1). Si les phrases ne sont pas reconnues par le système, celui-ci essaye de déterminer le modèle qui a le plus de similarité avec la phrase de l'apprenant, grâce à la technique du *Heaviest Common Sequence Algorithm*.

Chen *et al.* (2002) et Chen *et al.* (2005) proposent une autre technique appelée POST-Parser (*Part-of-speech tagged parser*). La technique consiste à prédéterminer une série de modèles de réponses correctes et incorrectes

7. `http://azalea.sunflare.co.jp/`, site donné par l'auteur mais indisponible.

pour les exercices. Les mots ou syntagmes peuvent être désambiguïsés. Le système choisit ensuite le modèle de phrase s'approchant le plus de celle de l'apprenant. En outre, l'analyseur dispose d'une banque de d'arbres désambiguïsés pour chaque séquence possible de parties du discours.

B.4.10 Système de Brown

Brown (2002) et Brown *et al.* (2004) décrivent un système intelligent d'apprentissage du français et de l'anglais, écrit en DCG Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) suivant le schéma X-barre (v. p. 120). Le modèle de l'apprenant est basé sur une grammaire incomplète, qui contient les règles de production correctes et incorrectes acquises par l'apprenant. Les règles incorrectes sont automatiquement extraites à partir d'une analyse partielle. Le logiciel d'enseignement de l'anglais porte sur l'utilisation de dix temps verbaux.

B.4.11 CALLE

Développé par *XEROX Special Information Systems* à Palo Alto, Californie, le projet CALLE (Feuerman *et al.*, 1987; Rypa et Feuerman, 1995) vise à entraîner la compréhension de textes en espagnol (articles de journaux) et se base sur un analyseur LFG (§3.3.5). Dans une première version du système, destinée aux apprenants débutants, un système de diagnostic d'erreurs permet d'analyser les erreurs des apprenants. Ceux-ci ont la possibilité de tester leurs phrases avant de les soumettre comme réponse définitive. Le système dirige un dialogue avec l'apprenant grâce à une modélisation des compétences discursives orientée vers la réalisation d'une tâche, qui illustre un point de grammaire particulier (accord sujet-prédicat, emploi du verbe *ser* [8]).

Une version ultérieure, CALLE90, cible les apprenants de niveau intermédiaire. Les exercices et l'analyseur ont donc été construits pour ce niveau plus élevé. L'apprenant a accès soit à l'analyse morphologique d'un mot, soit à l'analyse d'une phrase et à l'explication des structures syntaxiques en jeu, soit à un diagramme syntaxique sous forme d'arbre simplifié. L'apprenant peut avoir accès à des textes variés qui illustrent les mêmes types de phénomènes et les mêmes niveaux de difficulté. Considérons la phrase

8. Une des deux formes de verbe *être*.

suivante :

(77) Se vende el libro.
pron-réfl / pron-1-sg / marqueur-passif | vendre-pres-3-sg | le livre.

La phrase (77) peut être traduite de trois manières différentes :

(78) a. Il se vend le livre (à lui-même). (*se* objet indirect réfléchi).
b. On vend le livre. (*se* sujet impersonnel).
c. Le livre est vendu. (*se* marqueur du passif).

CALLE90 est un système intelligent, basé sur les quatre modules usuels (§2.7.4) qui doivent interagir entre eux pour guider un apprentissage efficace. Le système doit fournir à l'apprenant des aides à l'apprentissage (analyse morphologique d'un mot ou d'une phrase, explication des structures syntaxiques). Des détecteurs syntaxiques basés sur des patrons doivent détecter certains phénomènes dans les f-structures et afficher des messages d'aide pour des apprenants.

B.4.12 CLEF

CLEF (*Computer assisted Learning Exercises for French*, Holmes et Kidd, 1980) [9] est un logiciel d'apprentissage du français, niveaux débutant et intermédiaire, né en 1978. Paramskas (1993) nomme le logiciel *Computer assisted Lessons for French*. Le logiciel est écrit en *BASIC* sur microordinateur *Compucolor*, qui permettait d'utiliser 8 couleurs différentes. Il consiste en une série d'exercices de drill, où les réponses de l'apprenant sont comparées avec les réponses stockées dans le système. Chaque unité traite d'un point de grammaire particulier illustré par un texte. Le diagnostic d'erreurs permet de donner un *feedback* détaillé.

B.4.13 Compounds

Compounds (Boucher *et al.*, 1993; Boucher et Sébillot, 1993; Danna, 1997; Danna et Sébillot, 1997) est un logiciel intelligent (§2.7.4) destiné à

9. Voir la description sous `http://www.camsoftpartners.co.uk/clef.htm`, dernière consultation le 7.6.10.

l'apprentissage des mots composés anglais. Il permet d'aider à déterminer la signification d'un mot composé comme dans *whale boat* qui signifie *baleinier*, un bateau équipé pour la chasse à la baleine. *To hand weave* signifie "tisser à la main", *truck driver* "conducteur de camion", *data processing* "traitement de données", *killer shark* "requin tueur" et *drinking water* "eau potable". Chacun des exemples ci-dessus est caractérisé par l'emploi d'un schéma différent (N-N, N-V, N-V+er, N-V+ing, V+er-N et V+ing-N), mais chacun de ces composés a l'élément de droite pour tête, soit l'élément le plus significatif du composé, ce qui est le cas pour la plupart des composés anglais. COMPOUNDS est capable de déterminer la signification d'un composé en donnant une définition, et, à l'inverse, de construire un composé à partir d'une définition. Danna (1997) décrit le modèle de l'apprenant qui doit être associé avec le logiciel. Une ébauche de module pédagogique est également proposée.

B.4.14 COOL/CALP

COOL/CALP (van Heuven, 1998) est un logiciel d'enseignement de théorie linguistique. Il était capable de détecter les erreurs des apprenants. Aucune indication n'est disponible sur les théories linguistiques et les techniques utilisées.

B.4.15 DiBEx

DiBEx (*Dialogue-based Explanation*) est un prototype d'ALIAO décrit par Klenner (2004). Il est doté d'un système de raisonnement sur des méta-informations sur les erreurs des apprenants et guide l'apprenant par un dialogue progressif. Il commence par signaler le type de l'erreur, puis progressivement, si l'apprenant n'arrive pas à corriger, il le guide sur le lieu de l'erreur et la nature de celle-ci. Il est basé sur un analyseur LFG (§3.3.5) en Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Les phrases sont connues par avance et stockées dans le système et sont comparées aux phrases de l'apprenant. Le système peut ensuite mettre en œuvre un module de raisonnement chargé d'expliciter les règles de grammaire.

B.4.16 Systèmes de Dublin

Greene *et al.* (2004); Keogh *et al.* (2004) présente deux systèmes d'ALIAO destiné aux apprenants du gaélique et de l'allemand au niveau primaire, de

7 à 13 ans. Pour le gaélique, le système utilise des automates (§3.3.4.1) pour construire des animations grammaticales à l'aide de *Perl* (§2.7.5), *XML* (§2.7.5) et *Flash* (§2.7.5). Les productions libres sont traitées à l'aide d'une grammaire DCG incrémentale (§3.3.4.2), dont la couverture est étendue au fur et à mesure de l'avancement dans le logiciel, ce qui évite la surgénération de structures inutiles.

Pour l'allemand, les structures grammaticales couvertes sont traitées par l'étiqueteur TreeTagger de Schmidt (1994, §3.1.2). De plus, des travaux sont prévus pour construire des automates pour synthétiser des nombres en allemand.

B.4.17 ÉLÉONORE

ÉLÉONORE (Renié et Chanier, 1995, 1996) est un environnement d'apprentissage du français basée sur la théorie de la collaboration (§2.3). L'apprenant doit interagir avec un compagnon simulé par l'ordinateur. Le logiciel est axé sur l'acquisition des structures interrogatives. Le français connaît en effet des structures interrogatives plus variées que l'anglais. Ces variantes sont employées plus ou moins fréquemment selon des paramètres pragmatiques : fonction illocutoire (demande d'action, d'information, suggestion), canal de communication (écrit, oral), unité de dialogue et rôle. Les activités d'apprentissages sont centrées tantôt sur la compréhension, tantôt sur la production et tantôt les deux. ÉLÉONORE propose plusieurs modes d'apprentissage : découverte, explication, exercice et mode collaboratif.

Le système est construit selon le modèle des tuteurs intelligents (§2.7.4), avec un modèle expert, module des activités, modèle de l'apprenant, modèle du compagnon et un gestionnaire de dialogues. Le système de diagnostic utilise un automate à états finis (§3.3.4.1).

B.4.18 English Tutor

English Tutor est un projet voué à construire un système d'enseignement de l'utilisation des verbes anglais pour italophones (Fum *et al.*, 1992). Il utilise le formalisme de la grammaire systémique afin de mieux distinguer quel est le sens porté par la phrase. Il est doté d'un modèle de l'apprenant capable de construire de nouvelles règles d'erreurs lorsqu'une erreur inconnue survient.

B.4.19 ESPADA

ESPADA est un système prototype d'ALIAO en syntaxe pour apprenants germanophones de l'espagnol (Koller, 2003, 2004). Il n'y a pas de module de l'apprenant. Le module d'enseignement illustre les règles grammaticales à l'aide d'animations en *Flash* (§2.7.5), qui utilisent du *XML* (§2.7.5). Le module d'analyse reconnaît les phrases qui comprennent des erreurs d'accord ou au niveau syntaxique et sémantique, dans le cas de phrases simples.

B.4.20 ESPRIT

ESPRIT est un système d'ALIAO pour l'enseignement du français, de l'italien et de l'espagnol pour des locuteurs avancés d'au moins une de ces langues (Greene *et al.*, 2004; Koller, 2005). En effet, la plupart des systèmes et des méthodes d'enseignement ne tirent pas parti des connaissances préalables des apprenants d'autres langues de même type. Le système est pourvu d'un analyseur multilingue qui détecte les erreurs. Des animations grammaticales en *Flash* (§2.7.5) sont fournies en rétroaction et montrent les différences de construction entre les trois langues. Une base de corpus spécialisés fournit des matériaux authentiques d'apprentissage dans différents domaines.

B.4.21 EXCALIBUR

EXCALIBUR est un environnement informatique doté d'intelligence artificielle (§2.7.4) qui donne un cadre de développement et d'enseignement (Lian, 1992). Le système d'expert du domaine d'enseignement contient une base de connaissances et un module de raisonnement. Le module de l'apprenant modélise ses connaissance du domaine et ses capacités cognitives. Le module d'enseignement contient les connaissances pédagogiques génériques et spécifiques au domaine. L'interface en langue naturelle est capable d'analyser et comprendre les entrées de l'utilisateur et de générer des sorties du système en langue naturelle à partir des connaissances encodées. Enfin, le système de raisonnement construit les dialogues avec l'apprenant à partir des connaissance du module expert et du module de l'apprenant. Aucun détail n'est donné sur le module de traitement du langage.

B.4.22 Grammar-Debugger

Grammar-Debuger est un analyseur destiné aux apprenants sinophones de l'anglais (Si-Qing et Luomai, 1990). L'analyseur accepte des phrases grammaticales et non grammaticales et produit un arbre syntaxique en sortie qui marque les relations entre syntagmes. Le cas échéant, des messages pointent sur les erreurs contenues dans la phrase.

L'analyseur comprend 350 règles de grammaire et se base sur un dictionnaire de 60 000 entrées. Les règles comprennent des conditions primaires qui sont obligatoires et des règles secondaires qui peuvent être violées. Il ne fait aucune analyse sémantique à cause de l'absence de traits dans le lexique. Il utilise l'algorithme d'analyse déterministe de Marcus (1985). Par la stratégie de regard en avant, cet algorithme permet de choisir une analyse parmi d'autres sans retour en arrière ; dès lors, lorsque l'analyse échoue à un certain point, il s'agit certainement d'une erreur dans la phrase.

B.4.23 Systèmes de Heift

Heift et Nicholson (2001) décrivent un analyseur qui est utilisé par plusieurs applications d'ALIAO que nous décrirons par la suite. Plusieurs modules interviennent dans le traitement des erreurs : ponctuation, orthographe (par le logiciel libre sous *Unix Ispell* [10]), vérification du contenu de la réponse en vérifiant la présence de mots requis et l'absence de mots superflus, vérification de l'ordre des mots, vérification grammaticale et récupération en cas d'échec de la détection d'erreurs. Le pré-traitement vérifie une série d'erreurs courantes, pour éviter que l'analyseur doive anticiper les erreurs d'ordre, les erreurs morphologiques etc. L'analyseur grammatical est construit en *DCG Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) selon le formalisme ALE (Carpenter et Penn, 2001), qui est un dérivé de HPSG (§3.3.5). La vérification d'erreurs est faite par relâchement de contraintes. Au lieu de ne contenir que les traits corrects, les entrées lexicales listent les traits possibles qui sont marqués soit de la mention *correct*, soit d'une mention *erreur*. Ainsi, le processus d'unification est toujours possible, même en cas d'erreur.

Le *German Tutor* est un système intelligent destiné aux apprenants de l'allemand (Heift et McFetridge, 1999; Heift, 2001; Heift et Nicholson, 2001; Heift, 2002, 2003). Il contient des modules de vocabulaire et de grammaire.

10. `http://www.gnu.org/software/ispell/ispell.html`, dernier accès le 8 août 2006.

Depuis 2003, le logiciel est accessible sur *Internet* sous le nom de *E-tutor*. [11] Les exercices sont implémentés en *Java* (§2.7.5) et affichés sur le client de l'apprenant grâce à une *applet*. Il existe six types d'exercices : dictée, construction d'un syntagme, construction d'une phrase, identification d'un mot différent des autres et mise en ordre des mots d'une phrase. Si la phrase de l'apprenant correspond exactement au modèle de réponse contenu dans le système, les modules de TAL ne sont pas activés. Le système permet de contraindre le contexte des réponses et assure une grande précision des résultats, notamment grâce à un important processus de pré-traitement (Heift et Nicholson, 2001).

Le module de l'apprenant sélectionne les analyses les plus probables parmi les analyses produites. Le modèle de l'apprenant contient des statistiques de violation des contraintes permet ensuite de cibler les lacunes de l'apprenant. Trois niveaux de score ont été identifiés. Une seule erreur à la fois est présentée.

En outre, ces données servent au module de filtrage des *feedbacks* pour savoir si un *feedback* doit être affiché. Celui-ci est modulé en fonction de trois niveaux de maîtrise de la notion de grammaire en jeu. Les catégories erreurs sont mise dans une queue des priorités d'erreurs. Celles-ci servent à déterminer quel *feedback* doit être affiché en premier. Cet ordre peut être changé par le concepteur d'exercices, de manière à mieux cibler l'objet de l'exercice. Heift (2002) présente une étude sur les habitudes de consultation des apprenants.

L'apprenant a accès en tout temps au *Report Manager* (Heift, 2004), qui synthétise les informations du modèle de l'apprenant et permet de revoir les réponses aux exercices et les rétroactions du système. Heift (2004) montre que les apprenants consultent ces données et refont les exercices qu'ils/elles n'ont pas bien réussis.

Toole et Heift (2002) décrivent un logiciel interactif d'apprentissage de vocabulaire et grammaire anglaise. Le *feedback* est individualisé grâce au modèle de l'apprenant, qui contient un score pour chaque construction grammaticale. Si un exercice porte sur une construction particulière, le poids du score de celle-ci peut être accentué. Un logiciel-auteur *Tutor Assistant* permet aux enseignants de créer des exercices et du contenu d'enseignement sans connaissance technique préalable. Trois types d'exercices sont disponibles : construction de phrase à partir d'une liste de mots, assembler une phrase par un glisser-déplacer et textes à trous. Le logiciel *Task Generator* permet d'extraire automatiquement des phrases répondant à des critères pédago-

11. `http://www.e-tutor.org/`, dernier accès le 23 juillet 2006.

giques, en se basant sur un étiqueteur probabiliste. Les exercices disponibles sont des textes à trous, des glisser-déplacer et un exercice de création de phrases.

Heift *et al.* (2000) et Turcato *et al.* (2000) présentent *Greek ILTS*, un prototype intelligent d'enseignement multimédia du grec moderne sur le *web* [12]. Celui-ci est basé sur des exercices de vocabulaire et de grammaire (ordre des mots, textes à trou, construction de phrases complètes). L'analyseur est capable de détecter des erreurs par relâchement de contraintes ou par règles d'erreurs.

B.4.24 ICICLE

ICICLE (*Interactive Computer Identification and Correction of Language Errors*) est un logiciel d'ALAO destiné à enseigner l'anglais écrit à des apprenants sourds (Suri, 1994; Michaud et McCoy, 1999; Michaud *et al.*, 2001) qui sont locuteurs natifs en langue des signes étasunienne (*American Sign Language*). Le système analyse des compositions écrites par les apprenants. Il est basé sur une grammaire hors contexte augmentée de règles d'erreurs. Les erreurs sont mises en évidence dans le texte par une couleur différente par type d'erreur. Puis l'apprenant peut obtenir une explication détaillée.

ICICLE est pourvu d'un module de l'apprenant nommé *SLALOM* (*Steps of Language Acquisition in a Layered Organisation Model*) et destiné à avoir une image aussi précise que possible de celui/celle-ci et d'adapter au mieux la rétroaction. Le modèle est conçu comme une série de notions à apprendre organisées de manière séquentielle et hiérarchisée. Ce modèle est constamment mis à jour avec les nouvelles notions acquises ou non acquises.

B.4.25 Intelligent Language Tutor

Intelligent Language Tutor (Schwind, 1986, 1988, 1995) est un tutoriel d'allemand implémenté en DCG Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Il est basé sur une grammaire de traits et vérifie les erreurs syntaxiques, sémantiques et d'accord. Cette grammaire permet également au système de générer des phrases et de répondre aux questions des apprenants. Lorsqu'une unification n'est pas possible, les deux traits incompatibles sont notés dans la structure. Quant aux erreurs syntaxiques (p. ex.

12. Les articles cités ne donnent pas de nom au prototype, mais le logiciel est connu sous ce nom dans les états de l'art de divers articles.

ordre des mots), elles sont traitées par des règles d'erreur. La partie sémantique utilise des règles sémantiques attachées au lexique et à la syntaxe, sous forme de prédicats et de règles. Ces règles peuvent également être violées et un mécanisme similaire aux règles syntaxiques permet de traiter les erreurs sémantiques.

Le système permet de générer des exercices variés : construction de phrases, traduction, pronominalisation, transformation de phrases, exercices sur la causalité, la temporalité, compréhension de textes et conversation. Les exercices sont proposés en français. L'apprenant peut demander en français ou en allemand la traduction d'éléments, les constructions et conjugaisons des verbes.

B.4.26 Intelligent Tutor

Intelligent Tutor est un logiciel intelligent d'apprentissage de l'anglais académique sur *Internet* doté d'un module de détection d'erreurs (Dodigovic, 2005). Les apprenants doivent répondre à des questions à propos d'un texte qu'ils ont lu. L'analyseur descendant est implémenté en *DCG-Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Les erreurs sont traitées par des règles *ad hoc*. Une seule phrase peut être traitée à la fois. La grammaire et le lexique sont restreints. Il n'y a pas de correction orthographique.

B.4.27 M

M (Yang et Akahori, 1997, 1999) est un logiciel d'ALIAO sur *Internet* pour le japonais. Le but est d'enseigner la structure du passif. Le système est doté d'un analyseur morphologique (*JUMAN*) et syntaxique (*KNP*), capable de détection d'erreurs. Le système est basé sur une grammaire de cas, afin de vérifier la cohérence sémantique. Les erreurs de syntaxe sont vérifiées à l'aide de règles d'erreurs avec une grammaire de dépendance (§3.3.5). Puis un générateur de *feedback* se charge de la rétroaction.

B.4.28 Mocha

Mocha (Schulze, 2008; Schulze et Penner, 2008) est un système intelligent d'enseignement de l'allemand. Il utilise un modèle de l'apprenant basé

sur la Théorie des Systèmes Dynamiques (DST, §2.3) et les grammaires de construction. Les structures analysées, qu'elles contiennent des erreurs ou non, sont comparées avec les autres résultats déjà connus par le système. Le système tente de bâtir un modèle de l'apprenant en mesurant par diverses formules le degré de complexité des textes des apprenants.

B.4.29 RAFALES

RAFALES (*Recueil Automatique Favorisant l'Acquisition d'une Langue Etrangère de Spécialité*; Zampa, 2004, 2005) est un prototype de logiciel intelligent d'aide à la lecture et à l'acquisition de vocabulaire spécialisé. Le système indexe des documents et évalue les connaissances de l'apprenant en se basant sur l'Analyse sémantique latente (LSA, §3.1.8). Le module pédagogique calcule le modèle de l'apprenant en fonction de ses connaissances lexicales et sélectionne un document qui n'est ni trop proche, ni trop éloigné, afin de favoriser l'acquisition progressive de nouveaux mots par la lecture.

B.4.30 RECALL, CASTLE

RECALL (*Repairing Errors in Computer Aided Language Learning*; Krüger et Hamilton, 1997; Murphy *et al.*, 1998) est un logiciel intelligent multimédia (§2.7.4) basé sur des jeux de rôle où l'apprenant doit répondre à des questions posées par le système pour des tâches communicatives. Le système comporte également des exercices grammaticaux. Le module de diagnostic détecte des erreurs à l'aide d'une grammaire *ad hoc* et tient compte du profil de l'apprenant pour déterminer la cause de l'erreur, puis le module tuteur fournit le bon diagnostic et, le cas échéant, présente les exercices adéquats.

L'analyseur est basée sur une grammaire d'erreurs qui n'est pas décrite en détail. Le système de diagnostic offre une rétroaction graduelle en fonction du nombre d'erreurs du même type dans un exercice. Certaines erreurs ne sont pas signalées en fonction du type d'exercice. Les erreurs sont classées d'après le niveau linguistique (phonétique, orthographe, morphologie etc.). Si une erreur arrive une seule fois, elle est localisée par une surbrillance et le système fournit un indice. Avec deux erreurs du même type, l'erreur est signalée une deuxième fois et une explication du point de grammaire concerné est fourni. Pour trois erreurs, l'erreur est signalée et l'apprenant peut obtenir la solution de l'exercice.

CASTLE (Murphy et McTear, 1997) est une composante du projet *RE-*

CALL qui présente des tâches de communication accompagnées d'aides à la remédiation. Les apprenants peuvent interagir relativement librement avec le système. Le logiciel est doté d'un module de l'apprenant qui est mis à jour en fonction de l'acquisition de nouvelles capacités par l'apprenant, par recoupements avec un modèle du domaine de connaissances, organisé d'après un modèle de dépendances hiérarchiques. A l'aide d'un prétest, les apprenants sont assignés à un modèle stéréotype qui correspond à leur niveau de connaissances.

B.4.31 SCRIPSI

Scripsi (Catt, 1988; Catt et Hirst, 1990) est un logiciel intelligent (§2.7.4) d'apprentissage de l'anglais capable de reconnaître une grande série d'erreurs d'apprenants. Il vise à identifier les causes d'erreurs et à donner une rétroaction corrective. Bien qu'il soit orienté vers les erreurs d'apprenants de langue maternelle française et chinoise, il est utilisable par des apprenants d'autres origines car il reconnaît un grand nombre d'erreurs de surgénéralisation qui ne dépendent pas de la langue maternelle.

Scripsi est basé sur la théorie chomskyenne des grammaires transformationnelles (*théorie standard*, Chomsky, 1965) , avec des emprunts à la théorie GB (§3.3.5) et à la LFG (§3.3.5). Il utilise Prolog (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) et analyse sur la base d'une version étendue de l'algorithme ascendant *shift-reduce* (Aho et Johnson, 1974; Shieber, 1983). Catt (1988) ajoute des traits et des variables aux règles indépendantes du contexte, afin de pouvoir traiter les phénomènes de transformation, tels que l'inversion de l'auxiliaire et du syntagme nominal pour les structures interrogatives. Des règles récursives permettent de traiter les phénomènes d'enchâssement. L'analyseur n'est pas déterministe et permet le retour en arrière en cas d'échec. Il n'y a pas de traitement sémantique.

Le système détecte des erreurs morphologiques, syntaxiques (accord, complémentation, sous-catégorisation, placement de l'adverbe, construction) et lexicales.

Scripsi n'utilise pas de règle pour les syntagmes verbaux mais projette l'information de sous-catégorisation contenue dans le lexique. Au lieu de réduire une règle de réécriture du syntagme verbal, l'algorithme introduit une nouvelle action *project* qui réduit la pile d'élément selon le cadre de sous-catégorisation le plus adéquat. Une opération *transform* est introduite pour permettre d'utiliser les règles de transformation, telles que l'inversion auxiliaire-syntagme nominal pour les interrogatives. Ces deux opérations

ont lieu lorsque tous les symboles de la phrase ont été lus. Les pronoms interrogatifs sont gérés à l'aide d'une pile spéciale. Les structures enchâssées telles que les complétives ne sont pas traitées car le processus de récursion n'a pas été implémenté. Lorsque plusieurs analyses sont possibles, *Scripsi* choisit arbitrairement la première analyse.

Pour traiter des erreurs de transfert, Catt (1988) propose d'appliquer les règles de réécriture et de transformation de la L1 en marquant l'utilisation de règles erronées. Il est néanmoins difficile de traiter les faux amis comme *sa forme actuelle* $\leftrightarrow$ *its present form* $\neq$ **its actual form*.

Pour les erreurs de surgénéralisation morphologiques, des règles d'erreur sont appliquées. Quant aux autres erreurs de surgénéralisation (accord [13], sous-catégorisation, transformation), elles sont traitées par relâchement de contraintes (§3.3.2.2) en permettant l'unification de deux termes non compatibles. Un module de diagnostic analyse ensuite les erreurs détectées pour les reporter à l'apprenant

B.4.32 SPELLER

SPELLER (de Haan et Oppenhuizen, 1994) est un tuteur intelligent (§2.7.4) d'orthographe pour apprenants néerlandophones de l'anglais. Le système identifie les erreurs des apprenants par des règles de conversion graphème-phonème qui peuvent aussi résoudre les erreurs classiques d'inversion, substitution et omission. Il aide l'apprenant à résoudre le problème par un dialogue interactif. *SPELLER* peut montrer à l'apprenant les erreurs du même type qu'il a commises par le passé afin de l'aider à assimiler les règles.

B.4.33 SWIM

SWIM (*See What I Mean*; Zock, 1992) est un logiciel sous plate-forme *MacIntosh* qui a pour but d'aider à exprimer des idées à l'aide d'un générateur (§3.1.5). Il existe plusieurs modes d'utilisation. Dans un premier cas, l'apprenant construit une phrase en répondant à des questions posées par le système. Dans un autre mode, l'apprenant peut bâtir un graphe conceptuel qui représente ce qu'il veut dire. Le système peut demander à l'apprenant d'écrire la phrase correspondant à l'idée qu'il a voulu représenter. Le système

13. Étonnamment, ce type d'erreur est considéré par Catt (1988) comme une erreur de surgénéralisation.

génère la phrase correcte, qui peut alors être comparée à celle de l'apprenant. L'apprenant peut aussi modifier certains paramètres dans la phrase (changement de temps, de voix, pronominalisation etc.) et observer l'effet de ces changements sur la phrase.

B.4.34 TAGARELA

TAGARELA (*Teaching Aid for Grammatcical Awareness, Recognition and Enhancement of Linguistic Abilities*, Amaral et Meurers, 2006, 2008) est un système intelligent d'apprentissage du portugais sur *Internet* en cours de développement à la Ohio State University. Il vise à développer les capacités des apprenants au niveau écrit, lu et compréhension orale. Le système est capable de détecter les erreurs orthographiques, morphologiques, syntaxiques et sémantiques sur des mots, des syntagmes et des phrases.

Une première version de *TAGARELA* utilisait l'analyseur lexical LT TTT (Grover *et al.*, 2000, §3.1.1.4) et l'étiqueteur TnT (Brants, 2000, §3.1.2), entraîné sur un corpus étiqueté d'espagnol (Hana *et al.*, 2006) et basé sur des modèles de Markov cachés (HMM, v. 3.1.2 p. 49). Ensuite, un analyseur de surface basé sur le formalisme HPSG (§3.3.5) sert au système expert d'élaborer une stratégie de rétroaction d'après le module pédagogique et le module de l'apprenant. Une seconde version (Amaral et Meurers, 2008) utilise une première désambiguïsation faite par des automates puis un analyseur ascendant affine le résultat. Une analyse sémantique sommaire compare la réponse de l'apprenant à la réponse attendue par le système.

Les exercices proposés traitent la compréhension de l'écrit et de l'oral, des descriptions d'images, de la pratique de vocabulaire et de reformulation et réécriture. Pour vérifier la cohérence sémantique des réponses, celles-ci sont comparées à une réponse modèle (Bailey et Meurers, 2006) à l'aide d'un algorithme similaire à celui du système d'évaluation de traducteurs automatique *METEOR* (*Metric for Evaluation of Translation with Explicit ORdering*, Banerjee et Lavie, 2005) : les mots qui correspondent exactement sont alignés ; les autres mots sont soumis à un lemmatiseur (§3.1.1.4) et comparés les uns aux autres ; en outre, *WordNet* (§3.5.3) est utilisé pour découvrir les synonymes. Une métrique de proximité est ensuite calculée d'après ces facteurs. Enfin, *TAGARELA* est doté d'un modèle de l'étudiant qui permet de choisir la meilleure stratégie de rétroaction (Amaral et Meurers, 2008). Le modèle de l'apprenant prend en compte les compétences linguistiques et également les compétences stratégiques : plus une tâche est complexe, plus elle est difficile à réaliser. Les erreurs sont analysées en fonction de la tâche à réaliser. Signalons encore que Dickinson et Herring (2008) présente un

système intelligent d'apprentissage du russe basé sur *TAGARELA*.

B.4.35 Textana

Le système *Textana* (Schulze, 1995, 1997, 1999; Schulze et Hamel, 2000) vise à développer les compétences en lecture et écriture des apprenants anglophones de l'allemand d'affaires, d'apprendre du vocabulaire et d'acquérir des structures syntaxiques. Il est axé sur l'écriture de commentaires de textes. *Textana* fournit une rétroaction sur des erreurs typiques d'apprenants à l'aide d'un analyseur basé sur HPSG (§3.3.5). Le type de *feedback* est différencié suivant le degré de connaissance supposé de l'apprenant (Schulze, 1999). Le traitement des erreurs proprement dites est assuré par un module séparé nommé *Paidagogos* (Schulze, 1998) qui rassemble des informations sur l'apprenant et les utilise pour créer le *feedback* approprié.

B.4.36 WIZDOM

WIZDOM (Handke, 1992) est un logiciel intelligent (§2.7.4) tournant sur MS-Dos. Il permet de traiter des textes et d'en tirer des exercices divers, grâce au générateur d'exercices *FLINGER*. Les exercices possible sont la correction orthographique, les exercices de vocabulaire, de traduction, de choix de synonymes, les exercices à trou, les questionnaires à choix multiples, les exercices de syntaxe (repérer des syntagmes), de morphologie, les questions ouvertes restreintes et les exercices de reconstruction de textes. Un module de tutorat évalue et guide l'apprenant.

B.5 Aides à la rédaction

B.5.1 ALCOGRAM

ALCOGRAM (Adriaens et Schreurs, 1992) est un analyseur de l'anglais contrôlé (ou à couverture restreinte) destiné à un logiciel d'ALIAO. Il est basé sur un lexique de 6000 mots. Cent cinquante règles de grammaire permettent de spécifier des règles d'interdiction (*n'utilisez pas X*), de restriction (*utilisez seulement X*) ou de recommandation (*évitez X*). Un composant lexical vérifie l'emploi et la construction des noms, adjectifs, verbes et adverbes. Un composant syntaxique traite de la coordination, la subordination, le

temps et l'aspect. Enfin, un composant stylistique contrôle la ponctuation et les éléments typographiques. Le formalisme et l'algorithme utilisés ne sont pas précisés.

B.5.2 Exploratexte

V. *Correcteur 101* (§C.9 p. 425).

B.5.3 Francophone Stylistic Grammar Checking

Francophone Stylistic Grammar Checking (Brehony et Ryan, 1994) est un logiciel d'apprentissage de l'anglais destiné aux apprenants francophones, fonctionnant sous *Windows* 3.1 en conjonction avec *Microsoft Word* 2.0. Il est basé sur une grammaire de liens (*link grammar*, Sleator et Temperley, 1991).

B.5.4 FrenchWriter

FrenchWriter est un système intelligent (§2.7.4) destiné à l'enseignement du français (Hagen, 1995). Le système comporte une grammaire de référence hypertextuelle, un dictionnaire bilingue doté d'un conjugueur, des exercices de drill de vocabulaire et un vérificateur grammatical *GrammarMaster*, basé sur une grammaire d'unification orientée-objet. L'analyseur, appelé *HANOI*, est inspiré du formalisme HPSG (§3.3.5). Le vérificateur est capable de traiter les phénomènes d'incise, de dislocation, d'élision et de redondance. Il suggère à l'apprenant de vérifier auprès d'un enseignant si la phrase est correcte, même si aucune erreur n'a été détectée.

B.5.5 FROG et French Grammar Analyser

FROG (*French RObust Grammar checker*, Imlah et du Boulay, 1985) est un outil destiné aux apprenants anglais du français. Il ne traite que les phrases déclaratives et ne dispose pas de composante sémantique. Il effectue une analyse morphologique (§3.1.1.4) des mots inconnus pour tenter de proposer un diagnostic d'erreurs. Les erreurs fréquentes d'orthographe sont dans le lexique. Le diagnostic affiché est très détaillé.

French Grammar Analyser (FGA, Barchan *et al.*, 1986) reprend *FROG* pour améliorer ses performances. Il est réécrit de manière modulaire afin de favoriser les développements futurs. La grammaire, les techniques d'analyse et le module de signalisation des erreurs sont clairement séparés. Les règles sont écrites pour une analyse descendante, qui permet d'anticiper des structure incorrectes.

B.5.6 HARRY

HARRY (Holdich *et al.*, 2004) est un système d'aide à l'écriture et la révision de textes pour enfants anglophones, qui aide à améliorer le style des textes. Le système *Check Text* procède à des vérifications de lisibilité au moyen d'outils statistiques. Enfin, les erreurs orthographiques et autres erreurs techniques sont corrigées au moyen du traitement de texte *Microsoft Word* (§C.24).

B.5.7 Hidden Forms / Formes Cachées

Formes Cachées ou *Hidden Forms* (Ward, 1996; Ward *et al.*, 1999) est un logiciel d'aide à l'écriture pour malentendants. Le logiciel simule une conversation avec l'apprenant dans le but de résoudre un jeu de type *Memory* en utilisant la langue écrite. Le système fonctionne sur le *web* avec un programme en *Javascript* (§2.7.5) et la version française *Formes Cachées* est écrite en C++. Les phrases sont construites en appuyant sur des boutons dans le bon ordre. Le logiciel fonctionne grâce à des automates (§3.3.4.1).

B.5.8 ILLICO

ILLICO (Pasero et Sabatier, 1998) est une boîte à outils contenant un analyseur, un générateur et une aide à la composition, qui peuvent être utilisés notamment pour un logiciel d'ALAO. L'analyseur fonctionne avec des règles DCG Prolog (§3.3.4.2). Puis une représentation sémantique est dérivée grâce à des règles associées aux règles syntaxiques. La consistance conceptuelle de la représentation sémantique est vérifiée au moyen du modèle conceptuel. Enfin, le modèle contextuel s'occupe des présuppositions du discours, des implications et des anaphores, d'après un formalisme proche de la DRT (Kamp, 1981, §3.4.1). *ILLICO* est aussi doté d'un générateur qui sélectionne dans le lexique les mots valides pour compléter la phrase en

cours.

Dans le cadre de l'ALAO, *ILLICO* peut servir à construire des jeux sur les mots en les adaptant au niveau de connaissance d'un apprenant. Si un jeu consiste à construire le plus possible de phrases valides à partir d'une liste de mots, le système est capable de générer toutes les réponses possibles. Des contraintes peuvent être ajoutées sur la longueur de la phrase, le type (déclaratif, interrogatif...). *ILLICO* peut aussi servir de base à un système de dialogue pour piloter un micromonde (§2.7.3). Enfin, ce système peut guider un apprenant dans la résolution d'un problème, par exemple dans une tâche de pronominalisation.

B.5.9 Irakazi / Lentillak

Irakazi (Aldabe *et al.*, 2006) est un outil d'analyse d'erreurs d'apprenants pour le basque sur *Internet*. Cet outil est associé à l'application *Erreus* qui stocke les informations sur les erreurs et les déviations [14] des apprenants ; *Irakazi* utilise aussi la plate-formes d'outils TAL et de ressources d'apprentissage *Lentillak*, qui sont à disposition des apprenants. Les productions des apprenants sont stockées dans un corpus et étiquetés par des linguistes informaticiens.

La plate-forme *Lentillak* utilise des outils de TAL développés pour des éléments grammaticaux et adaptés pour la détection d'erreurs. Elle intègre un analyseur morphologique pour lemmatiser les formes, un conjugueur, un déclineur (§3.1.1.4), un concordancier (§3.1.7) [15] pour donner des exemples d'utilisation de mots en contexte, et des dictionnaires. Enfin, la plate-forme donne accès à une aide à la construction de structure de phrase, qui se base sur les informations de sous-catégorisation contenues dans le lexique. De plus, le système fournit une aide pour la construction de structures, comme les comparatives. Pour le traitement des erreurs, le système dispose d'un correcteur orthographique, d'un correcteur morphologique et de vérificateurs grammaticaux. Un des vérificateurs est basé sur une grammaire de contraintes (Karlsson, 1990). Le second marque les erreurs d'accord et utilise une grammaire de dépendance (§3.3.5, Díaz de Ilarraza *et al.*, 2005). Le troisième vérificateur (Alegria *et al.*, 2006) traite la ponctuation à l'aide d'heuristiques statistiques basée sur l'analyse d'un corpus d'apprentissage.

14. Les auteurs appellent erreur toute production agrammaticale. Les déviations sont des cas agrammaticaux ou inappropriés, comme le fait d'éviter des structures ou des répétitions.
15. *Key Word In Context*

B.5.10 Intelligent Writing Tutor System

Intelligent Writing Tutor System (Liou, 1996) est un logiciel d'apprentissage de révision de textes anglais destiné à des apprenants sinophones. Une base de connaissances contient les erreurs fréquentes du public-cible. L'analyseur est basé sur le formalisme GPSG (Gazdar *et al.*, 1985). Les règles comprennent aussi des règles d'erreurs.

B.5.11 LICE

LICE (*Language Independent Composition Environment*, Bowerman, 1992) est un prototype intelligent d'aide à la rédaction en allemand destiné aux apprenants de la langue. Il est basé sur le formalisme FPSG (*Functional Phrase-Structure Grammar*) qui emprunte les f-structures du formalisme LFG (§3.3.5), les structures de constituant du formalisme GPSG (Gazdar *et al.*, 1985) et une sémantique logique attachée aux règles (*s-structures*). Les *c-structures* consistent en des règles basées sur des traits, qui sont, selon l'auteur, plus adéquates pour traiter l'ordre des constituants qui est relativement libre en allemand.

Pour s'attacher à l'organisation du texte, Bowerman (1992) utilise la RST (*Rhetorical Structure Theory*, §3.4.4, Mann et Thompson, 1987; Hovy, 1988) qui relie les phrases d'un texte grâce à un ensemble de 20 relations ; ces relations peuvent à leur tour être reliées entre elles pour décrire l'organisation du texte.

LICE construit les structures FPSG et RST grâce au même analyseur ascendant en largeur. Lorsqu'une erreur est rencontrée, un analyseur d'erreurs examine la structure incomplète construite par l'analyseur ascendant et procède à une analyse descendante en relâchant les contraintes pour permettre une combinaison et obtenir une analyse complète (Mellish, 1989). L'analyseur d'erreurs tente de combiner les constituants complets, d'ajouter des constituants manquants et de relâcher les contraintes pour permettre d'obtenir une analyse complète. Les erreurs sont notées dans le module de l'apprenant.

Le modèle de l'apprenant est basé sur des catégories d'erreurs, organisées hiérarchiquement. Le système calcule une probabilité qu'un type d'erreur est problématique pour l'apprenant. Le module tuteur intervient lorsque la probabilité atteint un certain niveau. Les probabilités d'une catégorie d'erreur d'un niveau inférieur sont propagées et pondérées aux catégories

supérieures dont elle dépend.

Le module tuteur aide l'apprenant à construire un plan pour son texte. Il peut aussi intervenir lorsque des points de grammaire nécessitent une explication.

B.5.12 LINGER, eL et ISCA

LINGER (*Language-INdependent Grammatical Error Reporter*, Yazdani et Uren, 1988; Yazdani, 1991; Bolt et Yazdani, 1998) est un système d'ALIAO écrit en DCG Prolog (Pereira et Shieber, 1987; Covington, 1994) pour le français, l'allemand, l'espagnol, l'italien et l'anglais. Il a pris la succession de *FGA* (§B.5.5). Le projet a été développé de 1986 à 1992. Si une phrase est agrammaticale, le système procède à une correction par des insertions, effacements et transformations pour obtenir une structure correcte et affiche un *feedback* détaillé expliquant ses corrections.

enhanced LINGER (eL, Bolt, 1991) est une extension de *LINGER* qui utilise un analyseur tabulaire (*chart parser*, Winograd, 1983; Wehrli, 1997) pour pallier les difficultés causées par des analyses incomplètes. Les erreurs sont détectées grâce à des conditions et actions associées aux règles.

Le successeur de *LINGER*, *ISCA* (*Interactive Sentence Constructor and Analyser*, Bolt et Yazdani, 1998), avait pour but de pallier les défauts de *LINGER*, jugé trop prescriptif et pas assez fiable, en procédant à une analyse interactive.

B.5.13 LIPSTIC

LIPSTIC (*Limited Intelligence Parser Seeking Typical Interference Constructions*, Pankhurst, 2005) est un prototype de correcteur grammatical destiné aux apprenants de l'anglais. Comme son nom peut l'indiquer, *LIPSTIC* se veut une aide à l'écriture qui ne vise pas à une couverture totale de la langue et son analyseur ne cherche pas à construire une structure complète pour chaque phrase (§3.3.4.3). Ainsi, une structure partielle est bâtie pour retrouver le sujet et les compléments des verbes et des règles d'interférence dépendantes de la langue première de l'apprenant détectent les erreurs. Le logiciel est basé sur un lexique de 5000 mots, qui s'avèrent être suffisants pour des phrases d'apprenants. Aucun détail n'est donné sur le formalisme grammatical.

B.5.14 Literacy Tutor

Literacy Tutor (Horton *et al.*, 1990) est un logiciel d'apprentissage de l'écriture pour adultes. Il a été développé sur un microordinateur *Archimedes*. Il comprend un traitement de texte, un correcteur orthographique et grammatical, un lexique de 3800 mots et un dictionnaire de synonymes. L'analyseur utilisé pour la correction grammaticale utilise le formalisme GPSG (Gazdar *et al.*, 1985). La grammaire contient 1088 règles mais n'a qu'une couverture faible. Les erreurs sont traitées grâce à des règles *ad hoc*.

B.5.15 Systèmes de Menzel

Menzel (1988) décrit un système d'analyse de l'allemand fonctionnant sur un micro-ordinateur. Le système fonctionne selon un réseau de contraintes qui traite les erreurs de cas, d'accord et de syntaxe. Les structures sont déterminées par des procédure de reconnaissance de patron (§3.1.1.1).

Menzel (1990) décrit un système d'apprentissage de l'allemand basé sur le formalisme GPSG Gazdar *et al.* (1985).

B.5.16 Native English Writing Assistant

Native English Writing Assistant est une application commerciale d'aide à la rédaction et à la lecture destinée aux locuteurs germanophones, francophones et hispanophones de l'anglais, développée par *Inso Corporation* (Hu *et al.*, 1998). Il existe plusieurs version du logiciel paramétrées en fonction de la langue maternelle de l'utilisateur. Des corpus d'erreurs ont été récoltés et analysés pour découvrir les erreurs typiques des locuteurs non natifs et les interférences typiques de leur langue maternelle. Un correcteur de style permet de choisir entre quatre degrés de formalisme. L'utilisateur peut sélectionner quelles erreurs il veut que le logiciel corrige. La rétroaction peut être faite en anglais ou dans la langue de l'utilisateur. Aucune description des techniques utilisées n'est disponible.

B.5.17 L'Orthophile

L'Orthophile [16] est un logiciel pédagogique de vérification orthographique programmé en *MSWLogo*, un environnement *Windows* du langage *LOGO* (§2.7.3) destiné à des locuteurs natifs au niveau de la fin de l'école primaire et début du secondaire. Le lexique contient peu de mots. Le programme commence par la vérification orthographique, qui fonctionne avec des règles d'erreurs du type *allors** se réécrit *alors*. Puis il poursuit par une phase de désambiguïsation interactive : par exemple, lorsqu'il rencontre le mot *sont*, il demande: "que peux-tu dire du mot *sont*? 1) je peux le remplacer par *étaient*. 2) je peux le remplacer par *mon*. 3) c'est un nom $\rightarrow$ *le son*." Ensuite, par phases successives, il corrige les erreurs des apprenants en leur demandant de valider ses choix. Tout le processus est cyclique. De nombreuses questions sont posées et la correction peut être longue. Le système couvre les erreurs d'accord, d'auxiliaire, de négation et d'euphonie. Le lexique est accessible par les apprenants, qui peuvent obtenir de courtes explications et ajouter des mots. Le programme dispose également d'un conjugueur. En mode expert, l'apprenant reçoit moins d'aide du programme et doit par exemple identifier et corriger les erreurs d'orthographe. Pour sa part, l'enseignant peut accéder aux phrases de l'apprenant et au bilan dressé par le vérificateur à l'issue de la tâche.

B.5.18 SANTY

SANTY est un correcteur orthographique (§3.2) destiné aux apprenants de l'allemand langue étrangère (Rimrott, 2003). Pour pallier la déficience des correcteurs orthographiques traditionnels, elle propose d'incorporer un analyseur morphologique qui cherche à détecter l'application inadéquate de règles morphologiques à des mots.

B.5.19 STASEL

STASEL (*Stylistic Treatment at SEntence Level*, Payette, 1990) est un logiciel destiné à corriger le style d'apprenants. Deux modules guident le processus d'évaluation stylistique : un analyseur syntaxique et stylistique s'attache aux problèmes d'usage et de structure de phrase, et un analyseur d'objectifs stylistiques qui analyse la clarté structurelle.

16. `http://jeannoel.saillet.free.fr/Orthophile/Orthophile.htm`, dernier accès le 10 mars 2007.

Le logiciel est écrit en *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) et contient plus de 500 règles de grammaire et de style, pour un lexique de 800 mots. Le correcteur procède phrase après phrase avec des entrées sans erreurs grammaticales ni orthographiques. Le lexique contient des informations sur l'impact stylistique des mots. L'analyseur produit un arbre syntaxique et un arbre stylistique.

L'arbre stylistique utilise un formalisme sémantique appelé *frame statement*, dérivé de l'interpréteur sémantique de Hirst. Les informations contenues dans l'arbre stylistique sont la complexité de la phrase, la voix, les caractéristiques stylistiques des mots etc.

L'analyseur d'objectifs stylistiques analyse la clarté structurelle de la phrase. Les analyses sont basées sur des heuristiques comme le nombre d'enchâssements de groupes prépositionnels ou de propositions subordonnées, le nombre de modifications d'un nom, l'équilibre des éléments dans une coordination etc. D'autres objectifs stylistiques comme l'emphase, le caractère concret et le caractère formel sont prévus mais n'ont pas été implémentés.

B.5.20 Système-D

Système-D: Writing Assistant for French (Burston, 1991; New, 1999) est une aide à l'écriture du français pour apprenants anglophones aux niveau débutant et moyen, disponible sous *Windows* et *MacIntosh*. Il comprend un petit dictionnaire bilingue français-anglais, un lexique de terme reliés sémantiquement, un conjugueur et des notes de grammaire. Les actions de l'apprenant sont enregistrées. Le logiciel a été passé en revue par Ledgerwood (2000).

B.5.21 TechWriter

TechWriter (Napolitano et Stent, 2009) est un prototype d'assistant à la rédaction, qui offre des suggestions de correction et propose des aides de remédiation. Un processus d'apprentissage permet au logiciel de corriger automatiquement les erreurs fréquentes des apprenants. Il utilise les grammaires de dépendance (§3.3.5) avec le *Stanford Dependency Parser* (de Marneffe *et al.*, 2006). Les erreurs d'orthographe sont corrigées grâce au correcteur orthographique Jazzy [17], disponible en *open source* et développé en *Java* (§2.7.5).

17. `http://jazzy.sourceforge.net/`, dernier accès le 5.4.10

B.5.22 ESL-WEPS

ESL-WEPS (*Web-based English Proofing System for ESL users* Yi *et al.*, 2008) analyse les phrases d'apprenants asiatiques de l'anglais grâce à un analyseur de surface (§3.3.4.3) basé sur les modèles de Markov cachés (HMM, 3.1.2 p. 49). Puis un système génère des requêtes pour un moteur de recherche sur le *web*, afin de trouver des exemples d'utilisation afin d'aider l'apprenant à reformuler sa phrase.

Annexe C

Description de correcteurs orthographiques et grammaticaux

C.1 Système d'Agirre & al.

Un correcteur orthographique (§3.2) sophistiqué pour le basque est décrit par Agirre *et al.* (1998). Il a pour but de présenter une seule correction à l'utilisateur. Le système est basé sur un analyseur morphologique (§3.1.1.4) et un analyseur syntaxique (§3.3). Pour faire le tri entre plusieurs solutions, il calcule une proximité de sens à l'aide de *WordNet* (§3.5.3). Le correcteur se base également sur des fréquences de mots. Enfin, une série d'heuristiques permettent d'éliminer certains choix pour des erreurs courantes.

C.2 An Gramadóir

An Gramadóir [1] est un correcteur grammatical *open source* développé pour le gaélique mais étendu à d'autres langues, dont le français (Lechelt, 2005). Il est développé en *Perl* (§2.7.5) et dispose d'un moteur d'analyse indépendant de la langue. Il est centré sur un étiqueteur qui utilise des expressions régulières (§3.1.1.1) pour désambiguïser puis pour corriger les erreurs. *An Gramadóir* utilise aussi l'algorithme de Brill (1995) pour établir

1. `http://borel.slu.edu/gramadoir/`, dernier accès le 10 mars 2007.

automatiquement des règles de désambiguïsation.

C.3 Antidote

Antidote (Brunelle, 2004; Brunelle et Charest, 2007) est un correcteur développé depuis 1996 par la société *Druide*. La cinquième version du logiciel est sortie fin 2003, fruit de onze années de développement ou 80 années-personnes. *Antidote* est disponible sous *Windows* et sous *MacIntosh*. Le moteur d'analyse et les règles sont séparés. Ces dernières sont codées dans un métalangage orienté objet basé sur une grammaire et des arbres de dépendance (§3.3.5). Le dictionnaire compte 113 000 mots, chacun faisant l'objet d'une description formelle poussée. Le système gère aussi les expressions figées en encodant environ 50 000 collocations (v. p. 66). La grammaire compte environ 2000 règles syntaxiques. Plus de 1200 types d'erreurs sont corrigées.

L'analyseur peut gérer plusieurs analyses de phrase. Lorsque l'utilisateur sélectionne une variante à l'aide d'infobulles, les autres analyses sont ajustées en conséquence. *Antidote* dispose aussi d'une grammaire de référence en 375 articles qui expliquent les règles appliquées. Un dictionnaire gère les synonymes, antonymes, paronymes et homophones ainsi que les difficultés associées au mot répertoriées par le correcteur. Les anglicismes sont repérés. Un conjugueur permet d'afficher les tableaux des verbes. Ces outils sont bien intégrés entre eux. Une revue de la version 98 du logiciel est disponible chez Burston (1998). Durel (2006) décrit l'utilisation d'*Antidote* pour des activités de révision, où les apprenants reçoivent leurs productions annotées par un tuteur et doivent ensuite réviser leur texte à l'aide de ce correcteur. La version 6 est dotée d'un dictionnaire de cooccurrences (Charest *et al.*, 2007), [2] extrait automatiquement à partir d'un gros corpus de divers genres et diverses variétés du français, qui compte 800 000 expressions illustrées par des exemples. Le dictionnaire sert également à l'analyseur pour sélectionner des analyses préférentielles et au correcteur pour détecter des fautes stylistiques ou les erreurs sémantiques (paronymie).

C.4 Arboretum

Arboretum (Bender *et al.*, 2004) est un prototype de correcteur grammatical de l'anglais. Il est capable d'analyser des phrases contenant des erreurs,

2. Terme qui inclut les collocations (p. 66) et des formes moins figées mais néanmoins significatives.

d'en tirer une représentation sémantique et d'en tirer une correction. Le formalisme utilisé est la grammaire *ERG*, dérivée d'HPSG (§3.3.5), additionnée de règles d'erreurs. La correction est fournie grâce au générateur (§3.1.5).

C.5 CELINE

CELINE est un système automatique ou interactif de détection et correction d'erreurs lexicales et syntaxiques (Courtin *et al.*, 1991; Menézo *et al.*, 1996, 1998). Le système est organisé sous la forme d'une architecture hiérarchique multi-agents. La correction orthographique (§3.2) est assurée par trois méthodes : (*i*) recherche de mots proches par clé-squelette (Pollock et Zamora, 1984, §3.2.3.2) ; (*ii*) recherche phonologique à l'aide d'un transducteur (§3.3.4.1) ; (*iii*) analyseur morphologique, également à l'aide d'un transducteur. L'analyse syntaxique (Courtin *et al.*, 1991; Genthial *et al.*, 1994) est basée sur une grammaire de dépendance (§3.3.5) utilisant un analyseur *Prolog* (Pereira et Shieber, 1987, §3.3.4.2). La collaboration entre divers agents aux différents niveaux permet de confirmer ou infirmer certaines corrections. Les erreurs d'accord sont également vérifiées par un analyseur particulier. Un analyseur stochastique basé sur des HMM (§3.1.2, p. 49) permet d'affiner la désambiguïsation. De nombreux outils collaborent au processus de correction malgré l'hétérogénéité des formalismes et des méthodes en jeu.

C.6 CheckPoint

CheckPoint (Crysmann *et al.*, 2008) est un correcteur grammatical et stylistique pour l'anglais et l'allemand qui n'est pas tourné vers l'apprentissage des langues. Il utilise une approche hybride, où les erreurs sont d'abord prétraitées par un traitement de surface par l'analyseur *FLAG* puis traitées en profondeurs par un analyseur HPSG (§3.3.5). La correction stylistique est effectuée en reformulant certaines phrases utilisant des constructions à éviter (passif, etc.) à l'aide d'un générateur (§3.1.5).

C.7 CoGrOO

CoGrOO [3] est un correcteur grammatical du portugais brésilien basé sur un étiqueteur (§3.1.2) et des règles d'erreurs. Il est développé en *open source* pour la suite bureautique *OpenOffice*.

C.8 Cordial

Cordial (*CORrecteur D'Imprécisions et Analyseur Lexico-sémantique*) est un analyseur et correcteur du français conçu depuis 1991 par la société *Synapse Développement* (Laurent, 1999; Laurent *et al.*, 2007, 2009a,b). Il est issu du logiciel *Le Rédacteur*, né en 1987, un traitement de textes doté d'outils de correction sur micro-ordinateur *Atari* destinés aux journalistes du quotidien *Libération* [4]. Il tourne actuellement sous *Windows* et *Linux*. Cordial est basé sur un dictionnaire de 900 000 formes de 142 000 lemmes. Le lexique est enrichi de nombreux traits sémantiques, 9 pour les adverbes ou 147 pour les verbes. Constamment amélioré au cours des versions successives, l'analyseur fonctionne à la fois sur des informations statistiques (bi- et trigrammes, fréquences de lectures de mots et des cooccurrences des mots) et sur des règles explicites. L'algorithme d'analyse fonctionne par morceaux (§3.3.4.3 Campione *et al.*, 2005) mais permet de retrouver des relations à longue distance. Le correcteur est basé sur l'analyse d'un corpus de 40 000 erreurs. Une version anglaise est également en développement.

Le logiciel offre de nombreuses aides comme la visualisation d'analyse syntaxique, un conjugueur, un dictionnaire sémantique, une analyse sémantique, une analyse stylistique etc. La version professionnelle donne accès aux dictionnaires *Littré* et *Trésor de la Langue Française* (§3.5.4) ainsi qu'à des dictionnaires bilingues de plusieurs langues vers le français, un dictionnaire des synonymes, un dictionnaire de noms propres et toponymes ainsi qu'une grammaire de référence pour apprenants du français langue étrangère. En outre, *Cordial* permet de comparer le texte soumis avec 2000 textes de référence, de différents domaines. Un outil permet d'obtenir une liste d'occurrences de formes et de lemmes ainsi que de mots classés par parties du discours. Une autre fonctionnalité permet d'extraire les mots les plus fréquents du texte ainsi que des patrons adjectif-nom, nom-préposition-nom etc. L'utilisateur dispose aussi d'un thésaurus classant 58 000 lemmes selon

3. `http://cogroo.incubadora.fapesp.br/`, dernier accès le 10 mars 2007.
4. Voir le site de la société Synapse, `http://www.synapse-fr.com/`, dernier accès le 25 mai 2006.

une ontologie de concepts-clé.

Cordial Universités, maintenant appelé *Cordial Analyseur*, est un outil à disposition des chercheurs. Il ne permet pas de sauvegarder les textes corrigés mais intègre un étiqueteur morphosyntaxique et un concordancier. L'étiqueteur permet non seulement de trouver l'analyse lexicale du mot (130 étiquettes différentes), mais aussi de nombreuses informations sémantiques. Toutes ces fonctionnalités sont utilisables pour la recherche dans un corpus, comme le décrivent Valli et Véronis (1999) et Campione *et al.* (2005).

C.9 Correcteur 101, Exploratexte, Bilingual Corrector, El Corrector et CorText

Correcteur 101 est un correcteur du français (Joscelyne, 1994; Burston, 1995/6; Mogilevski, 1998; Doll et Coulombe, 2004; Gagnon et Da Sylva, 2005) qui est le premier à avoir proposé une analyse syntaxique complète de la phrase. Il est basé sur la théorie des lexiques-grammaires de Gross (1975) et sur les grammaires de dépendance (§3.3.5 et Tesnières, 1959; Kahane, 2001) et la théorie sens-texte (§3.5.2) pour les règles de grammaire. Il contient entre 2500 et 3300 règles de grammaires (selon les sources), dont 300 pour les règles d'accord. Il dispose d'un lexique de plus de 88 000 entrées. *Correcteur 101* revendique un taux de correction de 80%. Il permet de corriger plus de 1500 difficultés de la langue. En cas d'analyse partielle, le système indique aux utilisateurs comment modifier certains segments. Les erreurs peuvent être corrigées manuellement ou automatiquement.

Le prédécesseur de *Correcteur 101*, *Exploratexte* est un logiciel commercial d'apprentissage de la grammaire et l'orthographe du français sous *Windows* 3.1, conçu comme une série de programmes d'exercices et un correcteur grammatical. Les exercices sont corrigés à travers le correcteur grammatical. L'apprenant doit corriger lui même sa faute après un rappel de la règle transgressée. Le logiciel est ouvert et permet aux enseignants de créer de nouveaux exercices.

The Bilingual Corrector est un produit basé sur la technique du *Correcteur 101* pour la partie concernant le français et produit par la même société (v. la revue par Carpenter Binkley, 2002, 2004). Il est destiné aux apprenants anglophones du français et aux apprenants francophones et hispanophones de l'anglais. Il contient des aides à l'apprentissage et à la rédaction telles qu'un dictionnaire de synonymes, un conjugueur etc. Il est également doté d'une mémoire de corrections qui facilite l'édition de grands textes. *El Cor-*

rector Klein (1998) est un correcteur pour l'espagnol basé sur les mêmes techniques.

CorText (Mydlarski, 1999) est un correcteur d'anglais basé sur *Correcteur101*. Il dispose d'un mode de correction globale et d'un mode intermédiaire où les erreurs sont signalées. Les performances du logiciel sont jugées mitigées.

C.10 CorrecText ou Correct Grammar

CorrecText (Dobrin, 1990) est un correcteur grammatical commercial parfois connu sous le nom de *Correct Grammar* dans certaines versions. Il est basé sur une analyse complète de la phrase. Les techniques d'analyses sont peu décrites pour des raisons commerciales. Le correcteur est destiné avant tout à des personnes connaissant la grammaire anglaise et faisant des fautes d'inadvertance. Il est donc peu utile pour des apprenants. La couverture des erreurs corrigées semble assez large, d'après les indications de la firme conceptrice. De nombreuses erreurs semblent corrigées par des règles d'erreurs *ad hoc* (§3.3.3) et d'autres par relâchement de contraintes (§3.3.2.2).

C.11 EasyEnglish

EasyEnglish est un produit d'IBM qui aide à écrire un anglais plus simple, moins complexe et moins ambigu (Bernth, 1997). Il s'agit d'un correcteur pour locuteurs natifs destiné à contrôler la lisibilité des textes. Il contrôle l'ambiguïté structurelle (syntaxique), la complexité des phrases et la violation de contraintes sur le vocabulaire. Le système signale les ambiguïtés et donne une paraphrase possible pour chacun des attachements possibles. Outre le correcteur stylistique, *EasyEnglish* procède aussi à une vérification grammaticale.

C.12 Système d'Emirkanian et Bouchard

Emirkanian et Bouchard (1988, 1989) ont développé un système expert pour l'orthographe, la morphologie et la syntaxe du français tournant sur

MacIntosh. Les connaissances sont représentées de manière explicite dans le système et le chemin du raisonnement est conservé. Les erreurs les plus fréquentes sont également représentées.

Le correcteur orthographique recherche la racine d'un mot dans un dictionnaire et recherche ensuite les suffixes dans un second dictionnaire. En outre, les erreurs de suffixation les plus fréquentes sont associées à la racine du mot.

L'analyseur reconnaît les erreurs de sous-catégorisation et tente de remplacer un mot incorrect grâce à une réinterprétation phonétique et au schéma de sous-catégorisation.

C.13 Epistle et Critique

Epistle (*Evaluation, Preparation, and Interpretation System for Text and Language Entities*, Miller *et al.*, 1981; Heidorn *et al.*, 1982; Jensen *et al.*, 1983; Ravin, 1993; Richardson et Braden-Harder, 1993) est un système de correction de grammaire et de style pour l'anglais courant, écrit dans le formalisme PNLP (Heidorn, 1972) et APSG (Heidorn, 1975, *Augmented Phrase Structure Grammar, APSG*,), en utilisant le relâchement de contraintes, ainsi que des règles d'erreurs pour placer un élément qui pose problème dans une position plausible (*parse fitting*). Les phrases trop longues ou trop complexes sont déconseillées. Le système détecte les erreurs grammaticales puis procède à une évaluation stylistique en détectant le mauvais usage des mots, les redondances, les constructions inadéquates etc. Des statistiques sont générées pour chaque document.

Le système a été rebaptisé *Critique* par la suite (Richardson et Braden-Harder, 1988) et a été intégré dans les gros systèmes IBM, à destination des applications de bureautique, de l'édition de publication et des institutions du domaine éducatif, pour traiter la langue aussi bien de locuteurs natifs que d'apprenants de langue seconde. L'application a aussi été testée sur des PC. Le système a une large couverture de la langue et traite plus de cent erreurs de grammaire et de style. Il peut traiter aussi bien de la correspondance commerciale, des compositions d'étudiants ou de la documentation technique. *Critique* fonctionne de manière interactive et possède trois niveaux de diagnostic : pointage de l'erreur, brève explication de l'erreur et explication détaillée. L'utilisateur peut paramétrer le niveau de correction souhaité.

C.14 GRAC GRAmmar Checker

GRAC GRAmmar Checker (Biais, 2005) [5] est un correcteur *open source* indépendant de la langue. Il est basé sur un étiqueteur probabiliste (3.3.3.3) qui utilise aussi quelques règles pour traiter les mots inconnus. L'étiqueteur utilisé n'est pas mentionné.

C.15 Grammatik

Grammatik (Sanz, 1992) est un logiciel de correction qui s'adapte à la plupart des logiciels de traitement de texte. Il corrige les erreurs de concordance des temps, les confusion fréquentes comme *ce/ se, du/ dû, leur /leurs* etc., les faux amis, les erreurs lexicales etc.

Brock (1990) décrit sommairement les fonctionnalités de *Grammatik IV*, un correcteur commercial destiné aux locuteurs natifs de l'anglais. En plus des statistiques de base et des scores de lisibilité, le logiciel repère les fautes courantes à l'aide de reconnaissance de patrons syntaxiques (§3.1.1.1) et de règles d'analyse. Les règles sont groupées en classes et classées dans un dictionnaire. Un message doit être associé à chaque règle pour signaler l'erreur à l'utilisateur. En outre, *Grammatik IV* est un système ouvert, qui peut être paramétré par les utilisateurs. De nouvelles règles peuvent être introduites.

Wei et Davies (1996) présentent un test de *Grammatik V* pour les apprenants d'anglais. Le logiciel fonctionne selon différents modes plus ou moins interactifs. Les auteurs soulignent que les accords entre verbe et sujet ne fonctionnent pas correctement avec des phrases complexes.

Laenzlinger (1992) présente un test comparatif de correcteurs orthographiques, où *Grammatik 1.0* obtient le meilleur résultat, soit 57,4% de bonnes prédictions. Il est particulièrement efficace pour l'accord du participe passé. Les lexiques sont particulièrement développés, ce qui améliore la qualité de correction. Toutefois, l'ordre des mots est peu traité, ce qui trahit un traitement des erreurs local qui ne repose pas sur une analyse complète de la phrase. Le correcteur peut être utilisé avec la plupart des logiciels de traitement de textes. Sanz (1992) procède à un autre comparatif de logiciels, où *Grammatik 1.0* corrige 60 fautes sur 121. Glencross (1993) procède lui aussi à une étude d'une version non spécifiée du logiciel.

5. `http://sourceforge.net/projects/grac/`, dernier accès le 10 mars 2007.

C.16 GramCheck

GramCheck (Ramírez Bustamante et Sánchez León, 1996; Ramírez Bustamante *et al.*, 2000) est un vérificateur grammatical et stylistique pour l'espagnol et le grec. L'analyseur est basé sur HPSG (§3.3.5) avec le formalisme *ALEP* (Schütz, 1996) et développé en *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2). Pour relâcher les contraintes, le système complète le formalisme original avec des extensions nommées *Constraint Solvers* (CS). Les erreurs structurelles sont détectées par reconnaissance de patrons (§3.1.1.1). Le formalisme est complété par des règles d'erreurs chargées d'anticiper les erreurs. Les corrections sont proposées par la transduction de structures linguistiques, au moyen d'une technique analogue au transfert en traduction automatique (§3.1.6).

C.17 GrammR

GrammR (Sanz, 1992; Burston, 1995/6) est un correcteur grammatical sous *Windows*. Il contient un conjugueur. Il permet de corriger les erreurs d'accord, d'élision, de ponctuation et certaines confusions fréquentes comme *a/à*.

Laenzlinger (1992) soumet *GrammR* à un test comparatif où il ne donne que 32,6% de bonnes prédictions. L'accord sujet-verbe est particulièrement mal traité. Le traitement des erreurs semble très local. Les erreurs doivent être corrigées manuellement par l'utilisateur. Les commentaires des erreurs détectées sont très peu informatifs. Sanz (1992) procède à un autre comparatif de logiciels, où *GrammR* corrige 32 fautes sur 121.

C.18 Système de Holan et al.

Holan *et al.* (1997) présentent un prototype de correcteur grammatical pour le tchèque intégré à *Microsoft Word*. Le correcteur souligne les erreurs orthographiques et syntaxiques ; il peut aussi fournir la liste complète des erreurs et afficher l'analyse d'une phrase sous forme d'arbre. Un analyseur morphologique (§3.1.1.4) traite la formation des mots. Ensuite, l'analyseur syntaxique essaye de créer une analyse complète de la phrase avec une grammaire de dépendance robuste à ordre libre (*Robust Free Order Dependency Grammar, RFODG*, Holan *et al.*, 1997, §3.3.5). Si un problème est détecté,

l'analyse est passée à un processus d'évaluation. Enfin, si aucune analyse ne peut être obtenue, le système ne tente aucune opération car les résultats d'une combinaison d'analyses partielles ne donne pas, selon les auteurs, de résultat probant.

C.19 Hugo Plus

Hugo Plus est un correcteur pour *Windows* (Sanz, 1992; Burston, 1995/6, pour la version 6.0). L'interface est bien construite. Le nombre de fausses détection est plutôt élevé. La détection des fins de phrases est problématique.

Dans un test comparatif, Laenzlinger (1992) teste la version 5.1 du correcteur, qui ne donne que 31,6% de bonnes prédiction. Les erreurs d'accord sont détectées de manière insuffisante, spécialement pour les accords à longue distance comme dans les propositions relatives. Il conclut que l'analyse de la phrase est superficielle et ne traite que les erreurs adjacentes. Sanz (1992) procède à un autre comparatif de logiciels, où *Hugo Plus* version 6.0 détecte 28 erreurs sur 121.

C.20 Système de Jeker

Jeker (1992) présente un correcteur grammatical du français basé sur le formalisme GPSG (Gazdar *et al.*, 1985), qui traite en particulier des structures clivées et coordonnées. Le système est développé pour *MacIntosh*.

C.21 Kant

KANT (*Knowledge-based Accurate Natural language Translation*, Mitamura *et al.*, 2003) est un système de traduction de textes techniques qui utilise une langue contrôlée. Pour aider les auteurs à se conformer à cette langue, le logiciel est pourvu d'un vérificateur qui est capable de détecter les structures incorrectes et parfois de formuler des alternatives correctes. L'analyseur *KANTOO* est écrit dans le formalisme LFG (§3.3.5) et utilise un algorithme inspiré de celui de Tomita (1986).

C.22 Le Patron, BonPatron

Le Patron (`http://www.lepatron.ca/`, dernier accès le 4.7.06) est un outil de correction de textes d'apprenants du français, développé au Canada. Il identifie les erreurs des apprenants par reconnaissance de patrons (§3.1.1.1) et fournit un *feedback* détaillé sans corriger la phrase. L'outil est basé sur une grande base de données d'expressions régulières qui est constamment mise à jour (Stéfan Sinclair, communication personnelle, 7.12.2005). En outre, la correction orthographique est effectuée grâce au logiciel libre ASPELL (`http://aspell.net/`, dernier accès le 4.7.06).

Son successeur est *BonPatron* (Nadasdi et Sinclair, 2008, 2009, `http://bonpatron.com`, dernier accès le 4.4.10). L'évaluation de Burston (2008) indique que le correcteur est capable d'analyser jusqu'à 1500 mots en 60 secondes. Il déplore le passage sous silence de structures entières qui sont ignorées car elles vont au delà des capacités du système. Cependant un test sur un texte d'apprenant affiche un résultat de 88% des erreurs corrigées. Ses performances tiennent la comparaison avec *Antidote* (§C.3) et peuvent être utilisées pour les erreurs de surface.

C.23 Language Tool

Naber (2003) décrit un correcteur grammatical et stylistique de l'anglais destiné à la suite de logiciels libres de bureautique *OpenOffice* [6]. La correction grammaticale (§3.3.3.3) se base sur l'étiqueteur probabiliste *QTag* (Tufis et Mason, 1998) basé sur un corpus d'entraînement d'un million de mots pour l'anglais et sur un petit corpus de 25 000 mots pour l'allemand. Il fonctionne sur une base de trigrammes. Le système traite les mots inconnus à l'aide de règles heuristiques simples. Puis la sortie de l'étiqueteur sert d'entrée pour une analyse par morceaux (§3.3.4.3) basée sur une reconnaissance de patron (§3.1.1.1). Ensuite, la vérification grammaticale utilise des règles décrites dans un formalisme *XML* (§2.7.5) et appliquées également par reconnaissance de patron sur des étiquettes des éléments lexicaux ou des morceaux d'analyse identifiés à l'étape précédente. Un correcteur de style détecte la lisibilité du texte et l'utilisation de certains mots inadéquats. Le projet est toujours en développement dans la communauté des logiciels libres sous le nom de *Language Tool* [7].

6. `http://fr.openoffice.org/`, dernier accès le 10 août 2006.
7. `http://www.danielnaber.de/languagetool/`, dernier accès le 10 mars 2007.

C.24 Microsoft Word

Microsoft Word 97 est un correcteur grammatical commercial intégré au traitement de textes du même nom (Heidorn, 2000). Le système est l'aboutissement de cinq ans de recherche et développement par une cinquantaine de personnes. Il est basé sur un langage spécifique G qui permet de définir des règles et de spécifier des structures de données attributs-valeurs. Les règles sont ensuite compilées en langage C. Le vérificateur traite des niveaux lexical, syntaxique, sémantique et discursif. Les langues traitées sont le chinois, l'anglais, le français, l'allemand, le japonais, le coréen et l'espagnol. Un module de génération n'existe que pour l'anglais. Lors de la segmentation des phrases, un processus spécial traite des entrées à plusieurs mots, des dates, des lieux et des mots en majuscules. L'analyse est basée sur le formalisme de la grammaire de structure syntaxique augmentée (*Augmented Phrase Structure Grammar, APSG*, Heidorn, 1975) avec un analyseur tabulaire (*chart parser*, Winograd, 1983; Wehrli, 1997). Un processus vérifie l'attachement des modifieurs, principalement les groupes prépositionnels et les propositions relatives, qui sont attachés au syntagme le plus proche, et les déplace ensuite vers un meilleur site d'attachement, comme le niveau phrase pour les compléments de temps. En cas d'échec de l'analyse complète, les meilleurs morceaux d'analyse sont combinés. Puis l'analyseur détermine la forme logique de la phrase. Ensuite, un processus de désambiguïsation sémantique détermine le sens le plus adéquat des mots. En dernier lieu, il est prévu de vérifier les structures discursives à l'aide de la *Rhetorical Structure Theory* (*RST*, Mann et Thompson, 1987; Hovy, 1988, §3.4.4), mais cette composante n'est pas incluse dans le produit commercial, tout comme l'attachement des modifieurs à l'aide de critères sémantiques et l'extraction de formes logique pour la plupart des constructions syntaxiques. Le traitement des erreurs se fait à l'aide de deux cents règles d'erreurs. En outre, une liste de cent paires permet de traiter les mots homophones fréquemment confondus.

Le correcteur d'anglais est proposé pour toutes les langues d'interface du traitement de textes. La correction de certaines erreurs peut être désactivée. Plusieurs niveaux de langue peuvent être spécifiés. Un test de lisibilité donne une indication sur la fluidité du texte.

Fontenelle (2006) présente la version française la plus récente du correcteur de Microsoft disponible pour la suite Office 2003. Le lexique des noms propres a été considérablement étendu, avec notamment des noms de personnages publics actuels. L'orthographe moderne est acceptée à côté de l'orthographe traditionnelle. La féminisation des noms de métier, de grade et de fonction est acceptée. Le traitement des séparateurs de mots comme l'apostrophe ou le trait d'union ont été revus.

Le correcteur grammatical a été réécrit pour améliorer la précision, ce qui fait que les fausses alertes ont considérablement diminué. Les erreurs d'accord du participe des verbes pronominaux utilisés transitivement avec des parties du corps ont été améliorées.

L'extraction d'entité nommées (lieux, montants monétaires, dates) bénéficient aussi de synergies avec d'autres groupes de recherche de *Microsoft*, notamment dans le domaine de la recherche textuelle.

On trouvera aussi une description de la version 6 chez Burston (1995/6) sous le nom de *French Proofing Tools*.

C.25 MSLFG

MSLFG (Cornu, 1997) est un prototype de correction grammaticale basé sur un analyseur LFG (§3.3.5) destiné aux apprenants anglophones du français. La couverture est limitée et exclut les structures interrogatives et impératives. Il ne dispose pas d'un correcteur orthographique et ne peut traiter les mots inconnus. Les règles et le lexique sont déclaratifs et leur modification ne requiert pas une recompilation du programme. L'analyseur procède en trois étapes successives, (i) analyse syntaxique par stratégie ascendante, (ii) élimination des structures inadéquates grâce à une analyse fonctionnelle et (iii) analyse des erreurs et proposition de correction.

C.26 Nat

Nat est un outil commercial de traitement du langage, destiné entre autres à une interface de base de données (Coch et Morize, 1990). L'outil comporte un analyseur et un composant sémantique. Il comprend un module de traitement des erreurs et peut lever certaines ambiguïtés à l'aide d'un dialogue avec l'utilisateur.

C.27 Système d'Oliva

Oliva (1997) présente un correcteur du bulgare et du tchèque, deux langues à ordre des mots libres. Le système est basé sur le formalisme HPSG

(§3.3.5) et utilise la technique du relâchement de contraintes (§3.3.2.2). Afin d'améliorer les performances du système, une pré-analyse est effectuée à l'aide d'automates (§3.3.4.1), afin de trouver des éléments lexicaux clés. Ce processus détermine quelles contraintes peuvent être relâchées dans le contexte de la phrase. Les automates permettent aussi de diviser la phrase en clauses, afin de faciliter l'analyse.

C.28 Système de Park & al.

Park *et al.* (1997) décrivent un prototype de correcteur grammatical pour apprenants de l'anglais. Il est écrit en *Prolog* (Pereira et Shieber, 1987; Covington, 1994, §3.3.4.2) et utilise une grammaire catégorielle combinatoire (Bar-Hillel, 1970; Moortgat, 1990). Le système utilise une grammaire d'erreurs encodée dans les entrées lexicales.

C.29 Prolexis

Produit par les éditions Diagonale, *Prolexis* [8] est un outil de correction orthographique et grammaticale du français qui s'intègre aux traitements de texte existants. Le correcteur est très utilisé dans le monde de l'édition et de la presse. Il dispose d'un lexique de 550 000 mots pour le français. Pour l'anglais, un dictionnaire de 150 000 mots permet de vérifier la cohérence d'emploi de l'anglais britannique ou américain. Le vérificateur grammatical n'est disponible que pour le français. Aucun détail n'est disponible sur les techniques utilisées. Il est très rapide et met l'accent sur les erreurs d'accord, mais aussi sur les homonymes grammaticaux, sur la casse et la ponctuation etc. Une aide contextuelle est disponible pour 350 points de grammaire.

Rainero (2004) utilise *Prolexis* pour la transcription écrite de débats sténotypés qui doivent être corrigés en temps réel.

C.30 ReGra

Teixeira Martins *et al.* (1998, 2002) présentent *ReGra*, un correcteur grammatical du portugais brésilien pour locuteurs natifs. Il utilise une gram-

8. Voir `http://www.prolexis.com/`, dernier accès le 26 mai 2006.

maire indépendante du contexte (§3.3.2.1) et procède par relâchement de contraintes (§3.3.2.2), ainsi que par des règles d'erreurs qui s'appliquent en conjonction avec les règles ordinaires. Un prétraitement est réalisé par des automates (§3.3.4.1) qui détectent les erreurs locales.

C.31 Sans Faute / Grammaire

Sans Faute / Gramaire (Dinnematin *et al.*, 1990; Sanz, 1992) est un correcteur disponible sur *MacIntosh* et intégré au traitement de textes *Write Now*, qui est le seul moyen d'accéder à ce correcteur. Le dictionnaire compte 350 000 formes. Il est capable de suggérer des mots phonétiquement voisins. Le logiciel détecte des erreurs grammaticales mais ne propose pas de correction. Il couvre les erreurs d'accord du participe passé mais ne traite pas des erreurs d'élision. Les versions plus récentes sont aussi disponibles sous *Windows*. *Sans-Faute* est destiné à des apprenants du français, ce qui implique une couverture grammaticale plus faible que certains autres correcteurs commerciaux (Murphy-Judy, 2002).

Laenzlinger (1992) présente une étude comparative de correcteurs grammaticaux où *Sans Faute / Grammaire 1.0* donne 48,9% de bonnes prédictions. Le score sur les erreurs d'accord est plutôt bon. Les erreurs d'ordre des mots ne sont pas traitées, ce qui laisse supposer que le traitement des erreurs est superficiel. Les erreurs doivent être corrigées manuellement. Les suggestions de corrections pour les erreurs de syntaxe sont quasi-inexistantes. Une autre étude du logiciel est présentée par Murphy-Judy (2002). Sanz (1992) procède à un autre comparatif de logiciels, où il détecte 85 erreurs sur 121.

C.32 Skryba

Skryba (Nicholas *et al.*, 2004) est un correcteur orthographique du russe destiné aux apprenants bilingues anglais-russe. En effet, les immigrants de seconde génération maîtrisent correctement la forme orale de la langue en la parlant avec leurs parents mais écrivent leur langue de manière phonétique. Le correcteur utilise des règles morphologiques et phonétiques pour corriger les erreurs dues à des phénomènes de transformation et d'assimilation.

C.33 Style Writer

Style Writer [9] est un correcteur grammatical et stylistique commercial pour l'anglais. Il est capable de détecter des milliers d'erreurs de style, dont les mots complexes, l'utilisation de jargon et de mots abstraits, les structures verbeuses, l'utilisation de la voix passive, les clichés et les phrases trop longues. L'éditeur du logiciel prétend pouvoir réduire les phrases de 25%. Le logiciel traite plus de 35 000 règles. Le système est finement paramétrable.

C.34 Unix Writer's Workbench

Unix Writer's Workbench (Cherry et Macdonald, 1983; Reid *et al.*, 1983) est un logiciel d'édition de textes développé par *Bell Labs*. Comme son nom l'indique, il fonctionne sous le système d'exploitation *Unix*. Les différentes composantes du programme doivent être invoquées en ligne de commande. Le logiciel n'utilise que des reconnaissances de patrons (§3.1.1.1) et n'a pas de composante linguistique à proprement parler.

Un dictionnaire permet de vérifier l'orthographe des mots. Un dictionnaire d'usage de 700 mots contient des mots fréquemment confondus ou utilisés à mauvais escient. Un autre dictionnaire complémentaire contient des termes sexistes. Le vérificateur de texte est capable de détecter les erreurs orthographiques et les erreurs de ponctuation, les mots répétés, les éléments mal utilisés et les infinitifs qui ne sont pas collés à la particule *to*. Un analyseur stylistique vérifie la lisibilité du document, selon des standards statistiques : il compte le nombre de syllabes et la longueur moyenne des phrases ; il vérifie en outre l'utilisation appropriée de différents types de phrases (explétives, nominalisations) et de la voix passive. Le logiciel peut également fournir des explications de règles de ponctuation. Il donne la possibilité à l'utilisateur de vérifier la cohérence d'un texte en extrayant la première et la dernière phrase de chaque paragraphe, ce qui devrait servir de base à un résumé du texte. Un autre script extrait les vingt noms et paires nom-adjectif les plus utilisés dans le texte.

Unix Writer's Workbench est destiné à des personnes de langue maternelle anglaise, mais des expériences ont testé l'utilisation du logiciel par des apprenants de l'anglais. La langue écrite n'est pas testée lors du test TOEFL qui est exigé pour les étudiants non anglophones à l'entrée des universités ; le but est donc d'acquérir les capacités à rédiger dans un anglais académique

9. `http://www.stylewriter-usa.com/`, consulté le 10 mars 2007.

en utilisant les structures rhétoriques adéquates. Les auteurs ont élaboré une liste d'erreurs typiques d'apprenants à partir de l'étude d'un corpus et ont ajouté des fonctionnalités au programme existant.

Il n'existe que peu de détails techniques. Dans sa description, Payette (1990) parle de l'utilisation d'un étiqueteur pour déterminer les parties du discours, puis de techniques de reconnaissance de formes pour repérer les constituants et les éventuelles erreurs.

C.35 Système de Véronis

Véronis (1988a,b) présente un système de correction pour un logiciel d'apprentissage de géométrie. Les erreurs phonétiques sont corrigées grâce à un algorithme de substitution de sous-chaînes par d'autres sous-chaînes à la prononciation similaire. Les erreurs d'accords sont traitées par un analyseur simple qui utilise des vecteurs de coûts pour calculer la meilleure correction.

C.36 Système de Vosse

Vosse (1992) présente un correcteur grammatical du néerlandais, doté d'un correcteur orthographique et d'un analyseur morpho-syntaxique. Le correcteur orthographique doit procéder à une analyse morphologique du mot car le néerlandais, à l'instar de l'allemand, connaît de nombreuses possibilités de composer des mots. L'analyseur procède de manière déterministe et choisit la variante combinatoire avec la partie gauche la plus longue possible. Le correcteur combine une analyse de trigrammes et de triphones, associée à un système de score.

Le néerlandais contient de nombreux cas d'homophonie, qui ne peuvent être détectés que si le mot inapproprié n'appartient pas à la même catégorie lexicale. Pour d'autres cas, le changement de flexion verbale n'entraîne pas de changement de prononciation.

Le système utilise le formalisme APSG (Heidorn, 1975). Les règles d'erreurs reçoivent un poids différencié selon leur criticité.

C.37 XUXEN

XUXEN est un vérificateur et correcteur orthographique commercial du Basque (Agirre *et al.*, 1992; Aduriz *et al.*, 1991). Comme le Basque est une langue à morphologie très riche, il est indispensable d'utiliser une analyse morphologique pour tenter de retrouver la forme correcte. *XUXEN* utilise un analyseur morphologique qui traite des transformations entre niveau lexical et niveau de surface.

XuxenII (Aldezabal *et al.*, 1999) est une nouvelle version du correcteur, qui utilise des transducteurs lexicaux. L'utilisateur peut désormais ajouter de nouveaux mots, tant dans la forme de base que dans les formes déclinées.

Annexe D

Fips: informations lexicales

D.1 Informations lexicales pour *héritage*

héritage		
CATÉGORIE	nom	
VALEURS D'ACCORD	NOMBRE	sing
	GENRE	masculin
	PERSONNE	3
CARACTÉRISTIQUES	nom commun, comptable, déverbal	

TAB. D.1 – *Fips – informations lexicales pour* héritage

D.2 Informations lexicales pour *forte*

(1)

forte	
CATÉGORIE	adjectif
VALEURS D'ACCORD	NOMBRE sing GENRE féminin PERSONNE 1 2 3
DEGRÉ	positif

(2)

forte	
CATÉGORIE	adjectif
...	
POSITION	prénominal
DEGRÉ	positif

(3)

forte	
CATÉGORIE	adjectif
...	
DEGRÉ	positif
SOUS-CAT	_ PP [VALEUR PREP en, RÔLE thème, TYPE PREP cause]

(4)

forte	
CATÉGORIE	adverbe

TAB. D.2 – *Fips – informations lexicales pour* forte

D.3 Informations lexicales pour *est*

TAB. D.3 – Fips – informations lexicales pour *est*

(1)

est	
CATÉGORIE	verbe
TYPE	auxiliaire
FORME DE BASE	*être*
VALEURS VERBALES	TEMPS présent MODE indicatif
VALEURS D'ACCORD	NOMBRE sing GENRE masculin, féminin PERSONNE 3
TRAITS	verbe être

(2)

est	
CATÉGORIE	verbe
TYPE	ordinaire
...	
TRAITS	contrôle sujet, sans passif, verbe être, verbe d'état
SOUS-CAT	NP [FONCTION sujet; RÔLE agent] – FP [FONCTION obj. pred; RÔLE prédication; TRAITS SÉLECT. Pred AP, AdvP, PP, NP, DP]

suite page suivante...

TAB. D.3 – Fips – informations lexicales pour *est* (suite)

(3)

est	
CATÉGORIE	verbe
TYPE	ordinaire
...	
TRAITS	contrôle arbitraire, sans passif, verbe être
SOUS-CAT	NP [FNC sujet; RÔLE agent; TRAITS impersonnel] _ FP [FNC obj. pred; RÔLE thème; TR. SEL. Pred AP, AdvP, PP, NP, DP] S [FNC SObj; RÔLE préd.; TRAITS tensée]

(4)

est	
CATÉGORIE	verbe
TYPE	ordinaire
...	
TRAITS	verbe être
SOUS-CAT	NP [FNC sujet; RÔLE thème; TRAITS inanimé] _ S [FNC SObj; RÔLE prédication; TRAITS tensée]

suite page suivante...

TAB. D.3 – Fips – informations lexicales pour *est* (suite)

(5)

est	
CATÉGORIE	verbe
TYPE	ordinaire
...	
TRAITS	contrôle arbitraire, verbe être
SOUS-CAT	NP [FNC sujet; RÔLE thème; TRAITS inanimé] _ S [FNC SObj; RÔLE prédication; TRAITS infinitive, complément de]

(6)

est	
CATÉGORIE	verbe
TYPE	ordinaire
...	
TRAITS	verbe être
SOUS-CAT	NP [FNC sujet; RÔLE thème; TRAITS inanimé] _

suite page suivante...

TAB. D.3 – Fips – informations lexicales pour *est* (suite)

(7)	*est* CATÉGORIE: nom TYPE: commun VALEURS D'ACCORD: [NOMBRE: sing; GENRE: masculin; PERSONNE: 3] CARACTÉRISTIQUES: lieu, point cardinal	
(8)	*est* CATÉGORIE: adjectif VALEURS D'ACCORD: [NOMBRE: sing, plur; GENRE: masculin, féminin; PERSONNE: 3, 6]	

Annexe E

Distance lexicographique

E.1 Algorithmes

```
Fonction EstCarProche( a, b : char) : booléen
    c, d : char
    EnleverAccent(a, c)
    EnleverAccent(b, d)
    Retourner c=d
Fin
```

Algorithme 1: Fonction EstCarProche

E.2 Matrices des distances

Voici quelques exemples de matrices au tableau (E.1). La distance entre *manger* et *changer* est de $\frac{2}{6+7} = 0{,}1\overline{153846}$. Pour **tres* et *très*, elle est de $\frac{0{,}1}{4+4} = 0.0125$, ou de 0,125 si on ne traite pas les caractère accentués de manière particulière. Entre **coutumace* et *contumace*, elle est de $\frac{1}{9+9} = 0.0\overline{5}$. Enfin, entre **huereux* et *heureux*, la distance est de $\frac{1}{7+7} = 0.0\overline{714285}$.

Dans le tableau (E.2) p. 448, entre *proffesionel* et *professionnel*, la distance est de $\frac{0.3}{12+13} = 0.012$, au lieu de $\frac{3}{12+13} = 0{,}12$.

	src	0	1	2	3	4	5	6
cib		#	*m*	*a*	*n*	*g*	*e*	*r*
0	#	0,0	1,0	2,0	3,0	4,0	5,0	6,0
1	*c*	1,0	1,0	2,0	3,0	4,0	5,0	6,0
2	*h*	2,0	2,0	2,0	3,0	4,0	5,0	6,0
3	*a*	3,0	3,0	2,0	3,0	4,0	5,0	6,0
4	*n*	4,0	4,0	3,0	2,0	3,0	4,0	5,0
5	*g*	5,0	5,0	4,0	3,0	2,0	3,0	4,0
6	*e*	6,0	6,0	5,0	4,0	3,0	2,0	3,0
7	*r*	7,0	7,0	6,0	5,0	4,0	3,0	2,0

	src	0	1	2	3	4	0	1	2	3	4
cib		#	*t*	*r*	*e*	*s*	#	*t*	*r*	*e*	*s*
0	#	0,0	1,0	2,0	3,0	4,0	0	1	2	3	4
1	*t*	1,0	0,0	1,0	2,0	3,0	1	0	1	2	3
2	*r*	2,0	1,0	0,0	1,0	2,0	2	1	0	1	2
3	*è*	3,0	2,0	1,0	0,1	1,1	3	2	1	1	2
4	*s*	4,0	3,0	2,0	1,1	0,1	4	3	2	2	1

	src	0	1	2	3	4	5	6	7	8	9
cib		#	*c*	*o*	*u*	*t*	*u*	*m*	*a*	*c*	*e*
0	#	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
1	*c*	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0
2	*o*	2,0	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0
3	*n*	3,0	2,0	1,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0
4	*t*	4,0	3,0	2,0	2,0	1,0	2,0	3,0	4,0	5,0	6,0
5	*u*	5,0	4,0	3,0	2,0	2,0	1,0	2,0	3,0	4,0	5,0
6	*m*	6,0	5,0	4,0	3,0	3,0	2,0	1,0	2,0	3,0	4,0
7	*a*	7,0	6,0	5,0	4,0	4,0	3,0	2,0	1,0	2,0	3,0
8	*c*	8,0	7,0	6,0	5,0	5,0	4,0	3,0	2,0	1,0	2,0
9	*e*	9,0	8,0	7,0	6,0	6,0	5,0	4,0	3,0	2,0	1,0

	src	0	1	2	3	4	5	6	7
cib		#	*h*	*u*	*e*	*r*	*e*	*u*	*x*
0	#	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0
1	*h*	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0
2	*e*	2,0	1,0	1,0	1,0	2,0	3,0	4,0	5,0
3	*u*	3,0	2,0	1,0	1,0	2,0	3,0	3,0	4,0
4	*r*	4,0	3,0	2,0	2,0	1,0	2,0	3,0	4,0
5	*e*	5,0	4,0	3,0	2,0	2,0	1,0	2,0	3,0
6	*u*	6,0	5,0	4,0	3,0	3,0	2,0	1,0	2,0
7	*x*	7,0	6,0	5,0	4,0	4,0	3,0	2,0	1,0

TAB. E.1 – *Exemple de matrices des distances*

```
Fonction Distance-lexicographique( source : car[] cible : car[]) : entier
    cib, src, longCib, longSrc : entier
    distance : réel[]
    longCib ← Longueur(cible); longSrc ← Longueur(source);
    Minuscules(cible); Minuscules(source);
    [Affecte dynamiquement la longueur de la matrice]
    AffecterLongueur(distance, longSrc+1, longCib+1)
    Pour cib de 0e à longCib faire
        distance[0,cib] ← cib; [Initialisation de la première colonne]
    Fin Pour
    Pour src de 0 à longSrc faire
        distance[src,0] ← src; [Initialisation de la première ligne]
    Fin Pour
    Pour src de 1 à longSrc faire
        Pour cib de 1 à longCib faire
            Si (source[src] =cible[cib]) Alors
                coût ← 0;
            Sinon
                Si (EstCarProche(source[src],cible[cib])) Alors
                    coût ← 0,1;
                Sinon
                    coût ← 1;
                Fin Si
            Fin Si
            distance[src,cib] ←
                min( distance[src-1, cib]   +  1,  [insertion]
                     distance[src-1, cib-1] +  coût,
                     distance[src, cib-1]   +  1)  [effacement]
            Si (cib > 1 ET src > 1) Alors

                Si ( source[src] = cible[cib-1] ET source[src -1] = cible[cib]) Alors
                    distance[src ,cib] ← [permutation]
                        min( distance[src, cib]     ,
                             distance[src-2, cib-2] +  coût)
                Fin Si
                Si (EstConsonne(cible[cib])) ET (cible[cib] = source[src])) Alors
                    Si ((cible[cib] = cible[cib-1]) ET (cible[cib-1] ≠ source[src-1]))
                    Alors
                        distance[src][cib] ← [insertion 2e consonne]
                            min( distance[src, cib]     ,
                                 distance[src-1][cib-2] +  0,1 )
                    Fin Si
                    Si ((source[src] = source[src-1]) ET (cible[cib-1] ≠ source[src-
                    1])) Alors
                        distance[src][cib] ← [suppression 2e cons.]
                            min( distance[src, cib]     ,
                                 distance[src-2][cib-1] +  0,1)
                    Fin Si

                Fin Si
            Fin Si

        Fin Pour
    Fin Pour
    Retourner distance[longSrc, longCib] / longSource + longCible

Fin
```

Algorithme 2: Fonction Distance-lexicographique

Matrice avec traitement spécial des doubles consonnes

	src	0	1	2	3	4	5	6	7	8	9	10	11	12
cib		#	*p*	*r*	*o*	*f*	*f*	*e*	*s*	*i*	*o*	*n*	*e*	*l*
0	#	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0
1	*p*	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0
2	*r*	2,0	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0
3	*o*	3,0	2,0	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
4	*f*	4,0	3,0	2,0	1,0	0,0	0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1
5	*e*	5,0	4,0	3,0	2,0	1,0	1,0	0,1	1,1	2,1	3,1	4,1	5,1	6,1
6	*s*	6,0	5,0	4,0	3,0	2,0	2,0	1,1	0,1	1,1	2,1	3,1	4,1	5,1
7	*s*	7,0	6,0	5,0	4,0	3,0	3,0	2,1	0,2	1,1	2,1	3,1	4,1	5,1
8	*i*	8,0	7,0	6,0	5,0	4,0	4,0	3,1	1,2	0,2	1,2	2,2	3,2	4,2
9	*o*	9,0	8,0	7,0	6,0	5,0	5,0	4,1	2,2	1,2	0,2	1,2	2,2	3,2
10	*n*	10,0	9,0	8,0	7,0	6,0	6,0	5,1	3,2	2,2	1,2	0,2	1,2	2,2
11	*n*	11,0	10,0	9,0	8,0	7,0	7,0	6,1	4,2	3,2	2,2	0,3	1,2	2,2
12	*e*	12,0	11,0	10,0	9,0	8,0	8,0	7,0	5,2	4,2	3,2	1,3	0,3	1,3
13	*l*	13,0	12,0	11,0	10,0	9,0	9,0	8,0	6,2	5,2	4,2	2,3	1,3	0,3

Matrice sans traitement spécial des doubles consonnes

	src	0	1	2	3	4	5	6	7	8	9	10	11	12
cib		#	*p*	*r*	*o*	*f*	*f*	*e*	*s*	*i*	*o*	*n*	*e*	*l*
0	#	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0
1	*p*	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0
2	*r*	2,0	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0
3	*o*	3,0	2,0	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0
4	*f*	4,0	3,0	2,0	1,0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0
5	*e*	5,0	4,0	3,0	2,0	1,0	1,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0
6	*s*	6,0	5,0	4,0	3,0	2,0	2,0	2,0	1,0	2,0	3,0	4,0	5,0	6,0
7	*s*	7,0	6,0	5,0	4,0	3,0	3,0	3,0	2,0	2,0	3,0	4,0	5,0	6,0
8	*i*	8,0	7,0	6,0	5,0	4,0	4,0	4,0	3,0	2,0	3,0	4,0	5,0	6,0
9	*o*	9,0	8,0	7,0	6,0	5,0	5,0	5,0	4,0	3,0	2,0	3,0	4,0	5,0
10	*n*	10,0	9,0	8,0	7,0	6,0	6,0	6,0	5,0	4,0	3,0	2,0	3,0	4,0
11	*n*	11,0	10,0	9,0	8,0	7,0	7,0	7,0	6,0	5,0	4,0	3,0	3,0	4,0
12	*e*	12,0	11,0	10,0	9,0	8,0	8,0	7,0	7,0	6,0	5,0	4,0	3,0	4,0
13	*l*	13,0	12,0	11,0	10,0	9,0	9,0	8,0	8,0	7,0	6,0	5,0	4,0	3,0

TAB. E.2 – *Autres exemples de matrices des distances*

Annexe F

FipsOrtho: interfaces

FipsOrtho peut être librement testé sur *Internet* [1]. Afin d'affiner les statistiques de notre corpus, nous invitons les utilisateurs à se connecter à l'aide d'un nom d'utilisateur et d'un mot de passe. Nous leur demandons également leur sexe, leur âge et leur langue première. Les enseignants peuvent également inscrire leurs classes et consulter librement les entrées de leurs apprenants. Si les utilisateurs sont inscrits dans une classe, nous disposons également de leur niveau en français [2] et de leur pays de provenance.

L'apprenant visualise ses erreurs avec l'interface présentée à la figure (F.1).

Dans la phrase, les mots comportant des erreurs sont marqués en rouge. Ensuite, ces mots sont listés aux côtés des propositions éventuelles retrouvées par le système. Pour chaque erreur détectée, l'apprenant peut soit sélectionner une proposition, soit proposer une correction manuelle, soit laisser le mot tel quel. En passant la souris sur les mots de la liste, l'utilisateur peut mettre en surbrillance le mot dans la phrase. Enfin, l'apprenant peut utiliser un conjugueur (§3.1.1.4) qui affiche toutes les formes verbales, et un dictionnaire bilingue anglais-français doté d'une sortie vocale.

Par ailleurs, les apprenants et leurs enseignants peuvent accéder au détail des phrases qu'ils ont soumises au correcteur, comme l'illustrent les figures

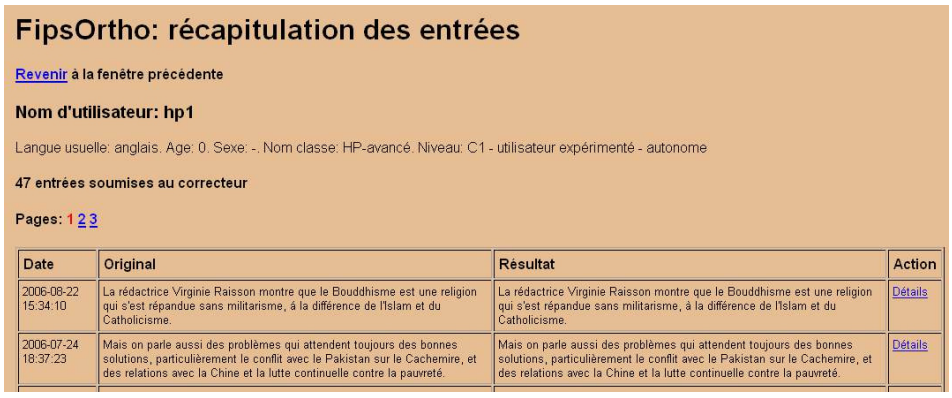
1. `http://latlcui.unige.ch/spellchecker`, dernier accès le 24.3.2009.
2. Nous utilisons la classification selon le cadre européen commun de référence pour l'apprentissage / enseignement des langues, défini par la Direction pour l'Education du Conseil de l'Europe. V. `http://www.coe.int/t/dg4/linguistic/Illustrations_FR.asp`, dernière consultation le 24.1.2008.

FIG. F.1 – FipsOrtho *: exemple de l'interface utilisateur*

(F.2) et (F.3). L'apprenant peut ainsi visualiser côte à côte sa phrase et le résultat final corrigé. Il peut aussi voir chaque phrase en détail, avec toutes les propositions de correction.

Poursuivons avec l'interface de l'expert. A leur connexion au système, les experts voient une liste de phrases qui nécessitent une validation. Ceux-ci peuvent rejeter les phrases, par exemple en cas de doublon ou de doute sur l'authenticité des erreurs. La figure (F.4) montre une partie de l'interface de balisage par l'expert.

Pour chaque entrée, le système calcule le nombre de mots, le nombre de



FipsOrtho: récapitulation des entrées

Revenir à la fenêtre précédente

Nom d'utilisateur: hp1

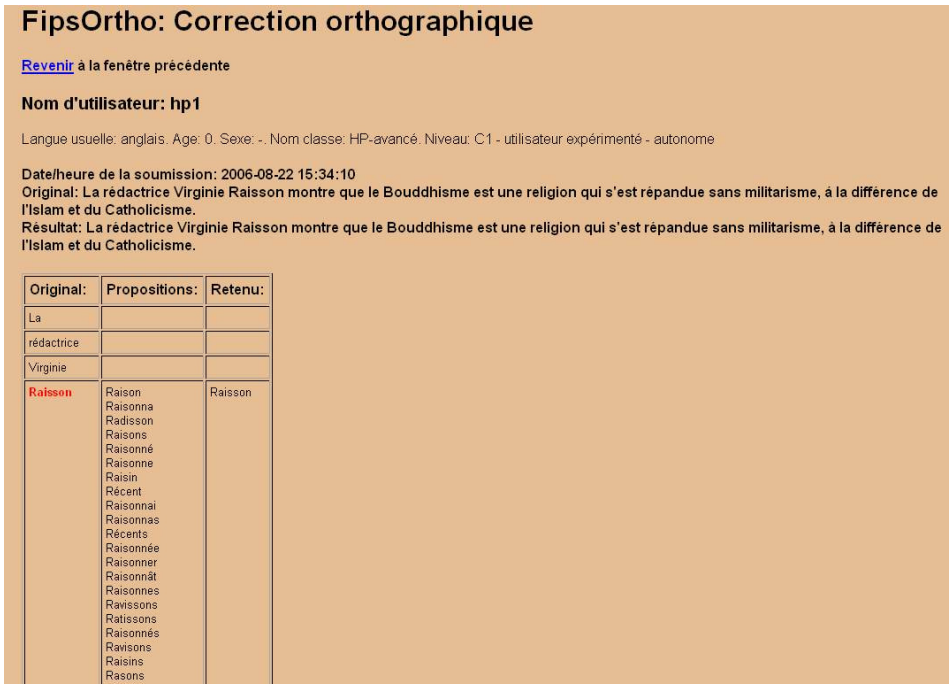
Langue usuelle: anglais. Age: 0. Sexe: -. Nom classe: HP-avancé. Niveau: C1 - utilisateur expérimenté - autonome

47 entrées soumises au correcteur

Pages: 1 2 3

Date	Original	Résultat	Action
2006-08-22 15:34:10	La rédactrice Virginie Raisson montre que le Bouddhisme est une religion qui s'est répandue sans militarisme, à la différence de l'Islam et du Catholicisme.	La rédactrice Virginie Raisson montre que le Bouddhisme est une religion qui s'est répandue sans militarisme, à la différence de l'Islam et du Catholicisme.	Détails
2006-07-24 18:37:23	Mais on parle aussi des problèmes qui attendent toujours des bonnes solutions, particulièrement le conflit avec le Pakistan sur le Cachemire, et des relations avec la Chine et la lutte continuelle contre la pauvreté.	Mais on parle aussi des problèmes qui attendent toujours des bonnes solutions, particulièrement le conflit avec le Pakistan sur le Cachemire, et des relations avec la Chine et la lutte continuelle contre la pauvreté.	Détails

FIG. F.2 – FipsOrtho *: liste des phrases de l'apprenant*



FipsOrtho: Correction orthographique

Revenir à la fenêtre précédente

Nom d'utilisateur: hp1

Langue usuelle: anglais. Age: 0. Sexe: -. Nom classe: HP-avancé. Niveau: C1 - utilisateur expérimenté - autonome

Date/heure de la soumission: 2006-08-22 15:34:10
Original: La rédactrice Virginie Raisson montre que le Bouddhisme est une religion qui s'est répandue sans militarisme, à la différence de l'Islam et du Catholicisme.
Résultat: La rédactrice Virginie Raisson montre que le Bouddhisme est une religion qui s'est répandue sans militarisme, à la différence de l'Islam et du Catholicisme.

Original:	Propositions:	Retenu:
La		
rédactrice		
Virginie		
Raisson	Raison Raisonna Radisson Raisons Raisonné Raisonne Raisin Récent Raisonnai Raisonnas Récents Raisonnée Raisonner Raisonnât Raisonnes Ravissons Ratissons Raisonnés Ravisons Raisins Rasons	Raisson

FIG. F.3 – FipsOrtho *: détail d'une phrase*

phrases, le nombre de mots inconnus et le nombre de non-erreurs [3]. L'expert doit valider le nombre de mots et de phrases, car le découpage de mots et de phrases par *Fips* peut être faussé par les erreurs contenues dans la phrase. Lorsque l'expert valide ses choix, le système met à jour le nombre de non-erreurs et d'erreurs. En outre, le commentaire éventuel de l'expert est enregistré.

3. Erreurs faussement détectées par le système.

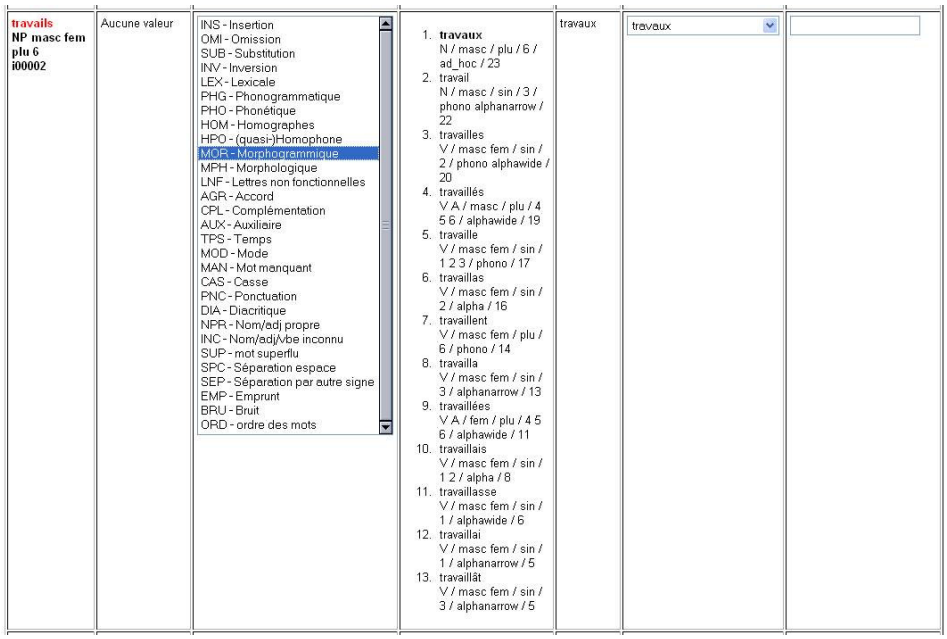


FIG. F.4 – FipsOrtho *: interface de balisage par l'expert*

Examinons maintenant les informations disponibles pour l'expert que nous voyons à la figure (F.4). Dans la première colonne, le système affiche la catégorie et l'analyse lexicales attribuée au mot par le système. Dans la seconde colonne figurent la ou les catégories d'erreurs actuellement attribuées au mot. La troisième colonne permet de sélectionner des catégories d'erreurs. La colonne suivante liste les propositions, avec leur catégorie lexicale, le genre, le nombre, la personne, la ou les méthodes utilisées et le score attribué à la proposition. La proposition sélectionnée par l'apprenant est mise en gras, celle validée par l'expert est en italiques. Dans la cinquième colonne, la proposition finalement retenue est affichée. Enfin, la dernière colonne permet à l'expert de proposer sa solution. Ainsi, chaque entrée est examinée lors d'un assez long processus.

Passons au corpus proprement dit. Les utilisateurs accèdent d'abord à une page contenant les statistiques générales. Ils peuvent consulter toutes les entrées du corpus ou des listes d'erreurs, classées par méthodes de recouvrement des propositions retenues et par catégories d'erreurs ; par ailleurs, toutes les propositions du correcteur sont indexées et sont accessibles par phrase ou par méthode de recouvrement. La figure (F.5) illustre la manière d'afficher les phrases du corpus. La phrase originale est affichée, ainsi que celle corrigée par l'apprenant et celle validée par l'expert. Les statistiques générales de l'entrée sont également affichées.

Date/heure de la soumission: 2006-07-03 16:49:57
Original: Samuel de Champlain joué le rôle du fondateur de la ville de Québec.
Résultat: Samuel de Champlain joué le rôle du fondateur de la ville de Québec.
Expert: Samuel de Champlain a joué le rôle du fondateur de la ville de Québec.

Nb phrases: 1 (auto: 1). Nb mots: 13 (auto: 13)
Nb mots inconnus: 2. Nb. non-erreurs: 2. Nb non-détectés: 1

Commentaire: joué -> jouait, -> erreur morpho

[EDITER] [Affichage XML] [CORPUS] [SUIVANTE]

Original:	Tag(s) erreur actuel(s)	Propositions:	Retenu:
Samuel NP i00001	NPR - Nom/adj propre	1. Saule (10) N / masc / sin / 3 / alphanarrow 0.1818181818181818 (0.2090909090909091) 2. Salue (7) V / masc fem / sin / 1 2 3 / alphanarrow 0.1818181818181818 (0.2090909090909091) 3. Salué (7) V / masc / sin / 1 2 3 / alphanarrow 0.1909090909090909 (0.2090909090909091) 4. Salues (6) V / masc fem / sin / 2 / alphanarrow 0.1666666666666667 (0.1916666666666667) 5. Saluée (5) V / fem / sin / 1 2 3 / alphanarrow 0.175 (0.1916666666666667) 6. Salués (4) V / masc / plu / 4 5 6 / alphanarrow 0.175 (0.1916666666666667)	Samuel
de PP i00002	Aucune valeur		
Champlain DP i00003	NPR - Nom/adj propre	pas de prop	Champlain
joué VP i00004	PHG - Phonogrammatique TPS - Temps MAN - Mot manquant	corr manuelle *a joué*	a joué
le DP i00005	Aucune valeur		

FIG. F.5 – FipsOrtho *: affichage compact d'une phrase du corpus*

FipsOrtho: validation du corpus. Liste d'erreurs

Revenir à la fenêtre des statistiques

Résultats: code: OMI (54 résultats retournés)

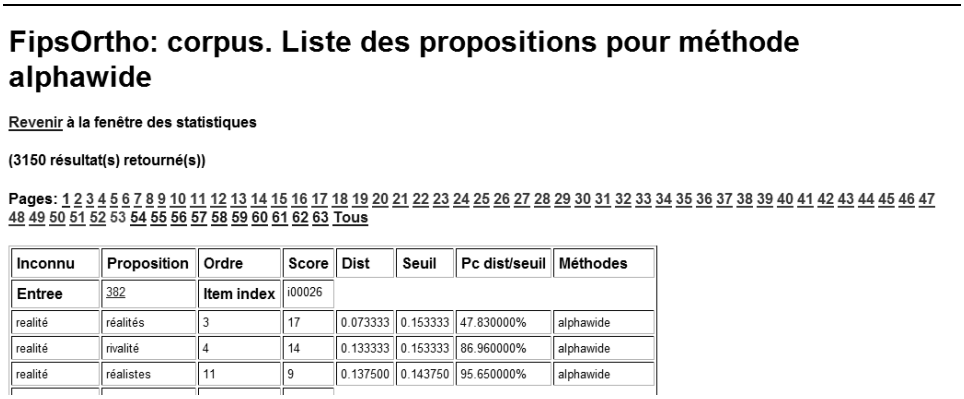
Pages: 1 2 Tous

N°	Ent	It. Idx	Err. idx	Man	Inconnu	Sel	Cor	Nb p.	Ord sel.	Methodes	Catégories
18	7	i00002	p00001	n	origin	origine	origine	2	1	alphawide	LNF PHO OMI
19	7	i00009	p00003	n	fourures	fourrures	fourrures	23	1	alpha phono	OMI
32	14	i00007	h1	n	poque		époque	3	0		OMI

FIG. F.6 – FipsOrtho *: liste d'erreurs*

Par ailleurs, la figure (F.6) illustre comment sont affichées les différentes listes d'erreurs. L'utilisateur peut accéder au contexte de l'erreur, en affichant la phrase comme dans la figure (F.5) avec l'erreur mise en surbrillance. Il peut également savoir si l'erreur a été détectée manuellement ou automa-

tiquement, quel mot a été sélectionné par l'apprenant, quelle correction a été validée par l'expert, le nombre de propositions du système, l'ordre de la proposition correcte le cas échéant, les méthodes utilisées pour recouvrer la proposition et les catégories d'erreurs.



FipsOrtho: corpus. Liste des propositions pour méthode alphawide

Revenir à la fenêtre des statistiques

(3150 résultat(s) retourné(s))

Pages: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 Tous

Inconnu	Proposition	Ordre	Score	Dist	Seuil	Pc dist/seuil	Méthodes
Entree	382	Item index	i00026				
realité	réalités	3	17	0.073333	0.153333	47.830000%	alphawide
realité	rivalité	4	14	0.133333	0.153333	86.960000%	alphawide
realité	réalistes	11	9	0.137500	0.143750	95.650000%	alphawide

FIG. F.7 – FipsOrtho *: liste de propositions par méthode*

Enfin, la figure (F.7) montre l'affichage global de toutes les propositions retrouvées par une méthode. L'utilisateur voit l'ordre et le score des propositions, ainsi que le ratio entre la distance et le seuil. Ici également, il est possible d'accéder au contexte de l'erreur en affichant la phrase.

Annexe G

Résultats de la liste de mots

La liste de mots inspirée de Dinnematin *et al.* (1990), Burston (1998) et Softissimo (2002) contient aussi des variations orthographiques de notre main.

G.1 Alphacode normal

La colonne *oui* donnent le nombre de mots retrouvés et sélectionnés par la méthode. La colonne *non* donne le nombre de mots retrouvés par la méthode qui ont été éliminés parce qu'ils dépassaient le seuil de distance lexicographique (§6.1.2.1). La colonne *élim* donne le nombre d'éléments dont la distance est égale ou inférieure au seuil mais qui sont éliminés (première lettre différente). Enfin, *lex* donne le nombre total de mots, y compris les doublons (*manger* sera retrouvé deux fois, comme nom et comme verbe, mais listé comme une seule proposition).

Tab. G.1 – Liste de résultats pour la méthode alpha-code

Mot	**Correction**	**A oui**	**A non**	**A élim**	**A lex**
absorpsion	absorption	0	1	0	1
acessit	accessit	0	6	1	6
acceuil	accueil	3	2	0	7
aceuil	accueil	3	2	0	7
acolite	acolyte	0	3	0	3
addresse	adresse	11	35	0	49
address	adresse	9	37	3	49

TAB. G.1 – Liste de résultats pour la méthode alpha-code (suite)

Mot	Correction	A oui	A non	A élim	A lex
aigüe	aiguë	1	0	0	1
aigue	aiguë	1	0	0	1
algorythme	algorithme	0	0	0	0
appas	appât	1	9	4	11
appogiature	appog(g)iature	0	5	0	5
aéropage	aréopage	1	8	2	10
arome	arôme	1	2	0	3
asujettir	assujettir	8	11	0	19
attrapper	attraper	5	50	5	62
azalé	azalée	1	1	0	2
azalee	azalée	1	1	0	2
barette	barrette	1	6	0	7
barete	barrette	1	6	0	7
béquée	becquée	0	0	0	0
beckee	becquée	0	0	0	0
bifteek	bifteck	1	0	0	1
biftèque	bifteck	0	0	0	0
boîter	boiter	3	4	0	9
boursouffler	boursoufler	8	1	0	14
braîment	braiment	1	15	0	16
celà	cela	3	9	1	19
charriot	chariot	1	7	1	8
chariau	chariot	0	1	0	1
charette	charrette	4	37	0	44
chrysalyde	chrysalide	0	0	0	0
chrysantème	chrysanthème	2	0	0	2
comparition	comparution	0	4	0	4
comparison	comparaison	4	6	0	13
comcombre	concombre	0	0	0	0
concurent	concurrent	9	58	0	71
concuran	concurrent	1	1	0	2
congruement	congrûment	0	0	0	0
connection	connexion	1	17	0	22
consonnant	consonant	0	28	0	28
contigüe	contiguë	2	0	0	2
contigue	contiguë	2	0	0	2
controle	contrôle	5	27	0	36
control	contrôle	0	1	0	1
convaint	convainc	3	14	0	17

Tab. G.1 – Liste de résultats pour la méthode alpha-code (suite)

Mot	**Correction**	**A oui**	**A non**	**A élim**	**A lex**
convin	convainc	1	0	0	1
coordonateur	coordinateur	1	3	0	4
courier	courrier	2	1	0	3
coutumace	contumace	0	3	0	6
cyprés	cyprès	1	0	0	1
cipre	cyprès	0	9	0	10
débarasser	débarrasser	12	11	0	28
déclancher	déclencher	1	2	0	3
déguingandé	dégingandé	0	3	0	4
dérilection	déréliction	0	4	0	4
dévôt	dévot	2	0	0	3
dilemne	dilemme	0	0	0	0
disgrâcier	disgracier	12	3	0	16
disparâte	disparate	5	49	0	60
drôlatique	drolatique	0	0	0	0
dislexie	dyslexie	0	0	0	0
échaufourée	échauffourée	1	8	0	10
anthropie	entropie	0	9	0	9
erronné	erroné	2	9	1	15
éthymologique	étymologique	0	1	0	1
filigramme	filigrane	0	0	0	0
gaité	gaieté	2	7	3	12
gheto	ghetto	1	0	0	1
guéto	ghetto	0	8	0	10
gueto	ghetto	0	8	0	10
halucination	hallucination	1	1	0	2
hypothénuse	hypoténuse	2	0	0	2
hipotenus	hypoténuse	0	0	0	0
imbécilité	imbécillité	1	0	0	1
infractus	infarctus	1	0	0	1
infarctusse	infarctus	0	9	0	9
inommé	innommé	0	5	0	6
insassiable	insatiable	1	37	0	43
intenssemment	intensément	2	38	0	43
intensement	intensément	3	37	0	43
macchiavélique	machiavélique	1	0	0	1
malaîse	malaise	8	37	0	50
malapris	malappris	0	0	0	0
malapri	malappris	0	0	0	0

TAB. G.1 – Liste de résultats pour la méthode alpha-code (suite)

Mot	Correction	A oui	A non	A élim	A lex
malgrès	malgré	0	8	0	8
malgres	malgré	0	8	0	8
malgre	malgré	1	3	0	4
mapemonde	mappemonde	1	0	0	1
marâsme	marasme	5	64	1	84
marasm	marasme	3	5	1	8
négligeamment	négligemment	0	15	0	15
negligement	négligemment	1	5	0	6
aurenge	orange	0	9	0	10
occurence	occurrence	1	16	2	26
ocurance	occurrence	0	6	0	6
pannacée	panacée	1	2	0	3
panassée	panacée	10	10	0	25
pantomine	pantomime	1	4	0	5
pécunière	pécuniaire	0	2	0	3
pélerine	pèlerine	2	2	0	4
piqure	piqûre	2	12	0	16
picur	piqûre	0	0	0	0
picure	piqûre	0	1	0	1
pickure	piqûre	0	0	0	0
précéde	précède	6	3	0	11
profesionel	professionnel	4	10	0	18
proffessionel	professionnel	4	10	0	18
proffesionel	professionnel	3	11	0	18
profesionnel	professionnel	4	10	0	18
protége	protège	4	0	0	9
psychadélique	psychédélique	0	0	0	0
psiquédélique	psychédélique	0	3	0	3
râtisser	ratisser	10	261	2	292
recoit	reçoit	1	13	0	16
ressoi	reçoit	0	13	0	15
reswa	reçoit	0	0	0	0
réddhibitoire	rédhibitoire	1	0	0	1
redibitoir	rédhibitoire	0	1	0	1
remerciment	remerciement	3	9	0	12
renumération	rémunération	1	21	2	22
shéma	schéma	0	0	0	0
schema	schéma	2	11	0	13
séborhée	séborrhée	0	0	0	0

Tab. G.1 – Liste de résultats pour la méthode alpha-code (suite)

Mot	**Correction**	**A oui**	**A non**	**A élim**	**A lex**
soufle	souffle	6	16	2	39
soufl	souffle	0	1	0	2
subbit	subit	3	2	0	7
subcidiaire	subsidiaire	0	0	0	0
subsidière	subsidiaire	0	0	0	0
substanciel	substantiel	0	3	0	3
succint	succinct	2	0	0	2
suxin	succinct	0	0	0	0
superfètatoire	superfétatoire	2	0	0	2
simptomatique	symptomatique	0	0	0	0
sinptomatik	symptomatique	0	0	0	0
sindrome	syndrome	0	77	0	95
syndrôme	syndrome	2	0	0	2
synthése	synthèse	2	0	0	2
sintèz	synthèse	1	4	0	7
sinthèse	synthèse	0	5	0	6
sizygie	syzygie	0	0	0	0
traditionnaliste	traditionaliste	2	49	0	55
troglodite	troglodyte	0	0	0	0
esplication	explication	0	10	0	10
jud'aurenge	jus d'orange	0	0	0	0
quesque	qu'est-ce que	0	1	0	1
impère	impair	0	27	3	43
san	sans	1	4	1	6
adissione	additionne	0	16	0	20
adicione	additionne	0	5	0	5
chifre	chiffre	4	3	0	9
contien	contient	3	15	0	22
angletterre	Angleterre	1	23	0	25
presentimen	pressentiment	3	17	0	21
so	saut	1	3	1	4
premiere	première	2	25	0	43
videttes	vedettes	0	5	0	10
emploiés	employés	2	10	3	13
ecrire	écrire	5	16	12	29
raissonable	raisonnable	2	36	0	41
particuliaire	particulière	1	24	0	27
fenaitre	fenêtre	0	105	0	106
puit	puits	0	0	0	0

TAB. G.1 – Liste de résultats pour la méthode alpha-code (suite)

Mot	Correction	A oui	A non	A élim	A lex
nonpossible	impossible	0	0	0	0
chauvinistiques	chauvins	0	0	0	0
votages	votes	0	1	0	1
political	politique	0	4	0	4
promoter	promouvoir	3	18	0	28
abandonnement	abandon	0	1	0	1
rivauté	rivalité	0	8	0	8
campaganer	faire campagne	0	0	0	0
sociéte	société	4	7	0	12
éffondrement	effondrement	1	4	0	5
extrème	extrême	1	0	0	2
gôut	goût	1	0	0	1
puls	plus	2	0	0	6
qunad	quand	1	0	0	2
obtinément	obstinément	0	8	0	9
environment	environnement	1	4	0	5
deveint	devient	3	10	0	15
ily a	il y a	1	0	0	1
y-a-t-il	y a-t-il	0	0	0	0
bi-lingue	bilingue	1	0	0	2
co-opération	coopération	1	48	0	49
extra-terrestre	extraterrestre	2	2	0	6
c'est-à dire	c'est-à-dire	1	49	0	55
autravers	au travers	1	16	0	19
autrechose	autre chose	1	10	0	11
déducation	déduction	0	2	1	2
parceque	parce que	2	0	0	2
pourcent	pour-cent	1	32	0	36
weekend	week-end	1	0	0	1
touts	tout	2	0	0	3
notres	nôtres	1	77	0	79
cettes	ces	0	2	1	2
journeaux	journaux	0	0	0	0
journals	journaux	0	0	0	0
acroiront	accroîtront	1	31	0	35
apprende	apprend	3	13	0	16
apprendent	apprennent	0	27	0	31
applaudent	applaudissent	0	1	0	0
augment	augmente	5	2	0	8

TAB. G.1 – Liste de résultats pour la méthode alpha-code (suite)

Mot	**Correction**	**A oui**	**A non**	**A élim**	**A lex**
avont	ont	0	1	0	2
connaise	connaisse	2	36	0	41
constitute	constitue	6	17	0	30
corromprent	corrompent	1	13	0	14
croirent	croient	1	43	0	45
croyent	croient	0	2	0	2
croi	croit	0	1	0	1
devenira	deviendra	2	14	0	16
devienderé	deviendrai	0	7	0	8
deviendrent	deviennent	3	16	0	20
developpenet	développent	1	0	0	1
devont	doivent	0	0	0	0
seriont	serions	3	226	1	267
seron	serons	2	33	2	38
soyent	soient	0	3	0	3
etaients	étaient	1	144	0	168
experiment	expérience	5	5	0	13
faisent	font	2	71	1	77
faitent	font	2	32	1	38
faient	font	4	30	0	38
faissent	font	2	71	1	77
insist	insiste	0	2	0	2
limit	limite	0	0	0	0
not	note	0	3	2	5
pens	pense	7	1	0	11
peus	peux	4	1	0	5
pousent	poussent	1	8	2	9
préfér	préfère	4	0	0	7
prévoyent	prévoient	0	0	0	0
rest	reste	6	31	4	50
romp	rompt	0	0	0	0
voyera	verra	0	0	0	0
voirai	verrai	0	2	0	3
donze	d'onze	1	0	0	1
acquerie	acquise	3	1	0	4
connerais	connais	2	122	1	136

G.2 Alphacode élargi

Le tableau suivant donne les résultats de la liste de mots pour l'alphacode élargi. La colonne *oui* donnent le nombre de mots retrouvés et sélectionnés par la méthode. La colonne *non* donne le nombre de mots retrouvés par la méthode qui ont été éliminés parce qu'ils dépassaient le seuil de distance lexicographique (§6.1.2.1). La colonne *elim* donne le nombre d'éléments dont la distance est égale ou inférieure au seuil mais qui sont éliminés (première lettre différente). La colonne *lex* donne le nombre total de mots, y compris les doublons. Enfin, les dernières colonnes donnent respectivement le nombre d'alphacodes donnant un résultat sur le nombre total d'alphacode, le pourcentage des alphacodes donnant un résultat et la moyenne des résultats par alphacode.

Tab. G.2 – Liste de résultats pour la méthode alpha-code élargie

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
absorpsion	absorption	2	5	0	8	3 / 18	16,67	0,44
acessit	accessit	0	306	0	326	15 / 20	75	16,3
acceuil	accueil	1	53	0	60	12 / 20	60	3
aceuil	accueil	1	53	0	60	12 / 20	60	3
acolite	acolyte	0	72	0	77	9 / 19	47,37	4,05
addresse	adresse	8	438	3	496	15 / 21	71,43	23,6
address	adresse	10	436	7	496	15 / 21	71,43	23,6
aigüe	aiguë	3	43	0	57	8 / 21	38,1	2,71
aigue	aiguë	3	43	0	57	8 / 21	38,1	2,71
algorythme	algorithme	0	0	0	0	0 / 16	0	0
appas	appât	5	65	25	87	15 / 23	65,22	3,78
appogiature	appog(g)iature	0	17	0	17	4 / 17	23,53	1

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
aéropage	aréopage	1	53	0	58	9 / 20	45	2,9
arome	arôme	1	69	0	78	13 / 21	61,9	3,71
asujettir	assujettir	2	31	0	35	4 / 18	22,22	1,94
attrapper	attraper	8	434	0	484	14 / 21	66,67	23
azalé	azalée	2	24	1	29	11 / 22	50	1,32
azalee	azalée	2	24	0	29	11 / 22	50	1,32
barette	barrette	5	92	2	104	13 / 21	61,9	4,95
barete	barrette	5	92	2	104	13 / 21	61,9	4,95
béquée	becquée	0	3	0	5	2 / 22	9,09	0,23
beckee	becquée	0	0	0	0	0 / 22	0	0
bifteek	bifteck	2	0	0	2	2 / 20	10	0,1
biftèque	bifteck	0	0	0	0	0 / 19	0	0
boîter	boiter	4	60	2	67	11 / 20	55	3,55
boursouffler	boursoufler	4	4	0	8	3 / 18	16,67	0,44
braîment	braiment	1	72	0	73	11 / 18	61,11	4,06
celà	cela	1	180	5	255	17 / 22	77,27	11,6
charriot	chariot	1	31	0	34	4 / 19	21,05	1,79
chariau	chariot	0	8	0	8	5 / 20	25	0,4
charette	charrette	5	186	0	198	12 / 20	60	9.9
chrysalyde	chrysalide	2	0	0	2	1 / 17	5,88	0,12
chrysantème	chrysanthème	0	0	0	0	0 / 16	0	0
comparition	comparution	1	40	0	42	5 / 17	29,41	2,47

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
comparison	comparaison	0	34	0	36	5 / 17	29,41	2,12
comcombre	concombre	1	6	0	11	3 / 20	15	0,55
concurent	concurrent	7	218	0	235	16 / 19	84,21	12,4
concuran	concurrent	5	20	0	27	6 / 20	30	1,35
congruement	congrûment	0	3	0	3	2 / 17	11,76	0,18
connection	connexion	9	234	0	287	16 / 20	80	14,4
consonnant	consonant	3	227	0	247	11 / 20	55	12,4
contigüe	contiguë	1	2	0	3	2 / 18	11,11	0,17
contigue	contiguë	1	2	0	3	2 / 18	11,11	0,17
controle	contrôle	6	135	1	149	14 / 19	73,68	7,84
control	contrôle	6	35	0	46	4 / 20	20	2,3
convaint	convainc	1	31	0	33	5 / 19	26,32	1,74
convin	convainc	14	9	0	28	4 / 21	19,05	1,33
coordonateur	coordinateur	2	35	0	39	9 / 17	52,94	2,29
courier	courrier	12	46	2	64	10 / 20	50	3,2
coutumace	contumace	1	24	0	28	6 / 19	31,58	1,47
cyprés	cyprès	1	0	0	1	1 / 20	5	0,05
cipre	cyprès	1	110	0	146	10 / 21	47,62	6,95
débarasser	débarrasser	5	98	0	112	12 / 20	60	5,6
déclancher	déclencher	0	8	0	8	5 / 18	27,78	0,44
déguingandé	dégingandé	0	20	0	21	7 / 19	36,84	1,11
dérilection	déréliction	0	32	0	32	6 / 17	35,29	1,88

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
dévôt	dévot	2	0	0	3	1 / 21	4,76	0,14
dilemne	dilemme	0	9	0	9	4 / 20	20	0,45
disgrâcier	disgracier	3	10	0	14	6 / 18	33,33	0,78
disparâte	disparate	1	133	0	136	11 / 18	61,11	7,56
drôlatique	drolatique	0	4	0	4	4 / 16	25	0,25
dislexie	dyslexie	2	0	0	2	1 / 20	5	0,1
échaufourée	échauffourée	1	19	0	20	5 / 18	27,78	1,11
anthropie	entropie	1	60	0	63	7 / 17	41,18	3,71
erronné	erroné	5	213	22	265	17 / 22	77,27	12
éthymologique	étymologique	0	2	0	2	2 / 15	13,33	0,13
filigramme	filigrane	0	2	0	2	1 / 18	5,56	0,11
gaité	gaieté	3	131	6	140	14 / 21	66,67	6,67
gheto	ghetto	1	0	0	1	1 / 21	4,76	0,05
guéto	ghetto	0	45	0	53	10 / 21	47,62	2,52
gueto	ghetto	0	45	0	53	10 / 21	47,62	2,52
halucination	hallucination	1	6	0	8	2 / 17	11,76	0,47
hypothénuse	hypoténuse	0	3	0	3	2 / 17	11,76	0,18
hipotenus	hypoténuse	0	4	0	5	3 / 17	17,65	0,29
imbécilité	imbécillité	1	0	0	1	1 / 19	5,26	0,05
infractus	infarctus	0	20	0	23	2 / 17	11,76	1,35
infarctusse	infarctus	0	23	0	23	5 / 16	31,25	1,44
inommé	innommé	0	55	1	74	10 / 21	47,62	3,52

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
insassiable	insatiable	2	147	0	162	12 / 19	63,16	8,53
intenssemment	intensément	2	307	0	354	15 / 20	75	17,7
intensement	intensément	2	307	0	354	15 / 20	75	17,7
macchiavélique	machiavélique	1	0	0	1	1 / 16	6,25	0,06
malaîse	malaise	8	255	0	273	16 / 20	80	13,7
malapris	malappris	0	33	0	34	3 / 19	15,79	1,79
malapri	malappris	0	25	0	26	3 / 20	15	1,3
malgrès	malgré	0	32	0	35	6 19	31,58	1,84
malgres	malgré	0	32	0	35	6 / 19	31,58	1,84
malgre	malgré	0	25	0	29	8 / 20	40	1,45
mapemonde	mappemonde	1	2	0	3	3 / 19	15,79	0,16
marâsme	marasme	3	511	1	551	17 / 21	80,95	26,2
marasm	marasme	15	87	1	117	10 / 22	45,45	5,32
négligeamment	négligemment	0	39	0	39	9 / 18	50	2,17
negligement	négligemment	0	26	0	27	7 / 19	36,84	1,42
aurenge	orange	0	82	0	93	14 / 20	70	4,65
occurence	occurrence	1	155	0	174	13 / 20	65	8,7
ocurance	occurrence	1	58	1	61	8 / 19	42,11	3,21
pannacée	panacée	4	14	0	19	5 / 21	23,81	0,9
panassée	panacée	6	111	0	127	13 / 21	61,9	6,05
pantomine	pantomime	1	91	0	96	8 / 18	44,44	5,33
pécunière	pécuniaire	1	13	0	15	5 / 19	26,32	0,79

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
pélerine	pèlerine	2	34	0	38	7 / 20	35	1,9
piqure	piqûre	3	43	0	53	8 / 20	40	2,65
picur	piqûre	0	3	0	3	2 / 21	9,52	0,14
picure	piqûre	1	17	0	20	8 / 20	40	1
pickure	piqûre	0	0	0	0	0 / 19	0	0
précéde	précède	10	20	0	35	6 / 21	28,57	1,67
profesionel	professionnel	0	36	0	38	4 / 17	23,53	2,24
proffessionel	professionnel	0	36	0	35	4 / 17	23,53	2,06
proffesionel	professionnel	0	36	0	38	4 / 17	23,53	2,24
profesionnel	professionnel	0	36	0	38	4 / 17	23,53	2,24
protége	protège	9	14	0	28	5 / 20	25	1,4
psychadélique	psychédélique	0	0	0	0	0 / 14	0	0
psiquédélique	psychédélique	0	2	0	2	1 / 18	5,56	0,11
râtisser	ratisser	5	1547	4	1684	17 / 20	85	84,2
recoit	reçoit	0	126	2	139	12 / 20	60	6,95
ressoi	reçoit	1	297	2	362	15 / 21	71,43	17,2
reswa	reçoit	0	1	0	0	1 / 21	4,76	0
réddhibitoire	rédhibitoire	1	0	0	1	1 /18	5,56	0,06
redibitoir	rédhibitoire	1	26	0	28	7 / 19	36,84	1,47
remerciment	remerciement	3	66	0	69	9 / 19	47,37	3,63
renumération	rémunération	0	115	1	117	10 / 17	58,82	6,88
shéma	schéma	2	29	0	31	8 / 21	38,1	1,48

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
schema	schéma	0	72	0	78	13 / 20	65	3,9
séborhée	séborrhée	0	18	0	19	5 / 20	25	0,95
soufle	souffle	5	33	1	43	12 / 20	60	2,15
soufl	souffle	7	26	0	50	6 / 21	28,57	2,38
subbit	subit	5	40	0	47	8 / 21	38,1	2,24
subcidiaire	subsidiaire	0	2	0	2	2 / 17	11,76	0,12
subsidière	subsidiaire	1	21	0	24	4 / 19	21,05	1,26
substanciel	substantiel	0	13	0	13	5 / 16	31,25	0,81
succint	succinct	1	32	0	38	4 / 20	20	1,9
suxin	succinct	0	3	0	5	1 / 21	4,76	0,24
superfètatoire	superfétatoire	0	8	0	8	2 / 16	12,5	0,5
simptomatique	symptomatique	2	5	0	0	4 / 16	25	0
sinptomatik	symptomatique	0	0	0	0	0 / 17	0	0
sindrome	syndrome	0	188	0	213	9 / 18	50	11,8
syndrôme	syndrome	0	0	0	0	0 / 18	0	0
synthése	synthèse	2	5	0	8	1 / 20	5	0,4
sintèz	synthèse	2	71	0	100	12 / 20	60	5
sinthèse	synthèse	0	77	0	86	8 / 20	40	4,3
sizygie	syzygie	0	1	0	1	1 / 20	5	0,05
traditionnaliste	traditionaliste	2	128	0	137	9 / 16	56,25	8,56
troglodite	troglodyte	0	7	0	7	2 / 18	11,11	0,39
esplication	explication	1	94	0	98	8 / 16	50	6,13

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
jud'aurenge	jus d'orange	0	1	0	1	1 / 18	5,56	0,06
quesque	qu'est-ce que	1	41	3	55	11 / 22	50	2,5
impère	impair	2	141	0	186	13 / 21	61,9	8,86
san	sans	8	75	4	96	17 / 23	73,91	4,17
adissione	additionne	0	279	0	311	14 / 19	73,68	16,4
adicione	additionne	0	81	0	84	7 / 19	36,84	4,42
chifre	chiffre	6	32	2	44	6 / 20	30	2,2
contien	contient	8	235	0	288	16 / 20	80	14,4
angletterre	Angleterre	0	107	0	109	9 / 19	47,37	5,74
presentimen	pressentiment	0	158	0	177	9 / 18	50	9,83
so	saut	14	24	8	52	16 / 24	66,67	2,17
premiere	première	4	139	0	186	13 / 21	61,9	8,86
videttes	vedettes	1	50	3	60	7 / 20	35	3
emploiés	employés	0	41	3	49	11 / 19	57,89	2,58
ecrire	écrire	20	149	13	206	16 / 22	72,73	9,36
raissonable	raisonnable	0	131	1	141	11 / 17	64,71	8,29
particuliaire	particulière	0	63	0	65	11 / 17	64,71	3,82
fenaitre	fenêtre	0	407	0	411	14 / 19	73,68	21,6
puit	puits	4	3	0	9	4 / 22	18,18	0,41
nonpossible	impossible	0	6	0	6	2 / 18	11,11	0,33
chauvinistiques	chauvins	0	0	0	0	0 / 15	0	0
votages	votes	1	3	0	4	4 / 19	21,05	0,21

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
political	politique	0	23	0	24	7 / 19	36,84	1,26
promoter	promouvoir	6	107	0	126	12 / 20	60	6,3
abandonnement	abandon	0	10	0	10	3 / 18	16,67	0,56
rivauté	rivalité	0	58	0	61	10 / 19	52,63	3,21
campaganer	faire campagne	0	8	0	10	2 / 18	11,11	0,56
sociéte	société	0	126	0	146	14 / 20	70	7,3
éffondrement	effondrement	1	7	0	9	4 / 18	22,22	0,5
extrème	extrême	2	2	0	5	4 / 21	19,05	0,24
gôut	goût	8	4	3	14	4 / 22	18,18	0,64
puls	plus	4	24	0	32	7 / 22	31,82	1,45
qunad	quand	0	3	0	3	2 / 21	9,52	0,14
obtinément	obstinément	1	51	0	57	8 / 19	42,11	3
environment	environnement	1	13	0	14	5 / 18	27,78	0,78
deveint	devient	4	58	0	66	7 / 20	35	3,3
ily a	il y a	0	1	0	1	1 / 22	4,55	0,05
y-a-t-il	y a-t-il	0	7	0	7	2 / 21	9,52	0,33
bi-lingue	bilingue	1	0	0	2	1 / 19	5,26	0,11
co-opération	coopération	1	191	0	198	10 / 17	58,82	11,6
extra-terrestre	extraterrestre	0	41	0	43	9 / 20	45	2,15
c'est-à dire	c'est-à-dire	0	161	0	164	12 / 18	66,67	9,11
autravers	au travers	0	62	0	63	8 / 19	42,11	3,32
autrechose	autre chose	0	26	0	27	8 / 17	47,06	1,59

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
déducation	déduction	0	50	2	53	8 / 17	47,06	3,12
parceque	parce que	0	0	0	0	0 / 19	0	0
pourcent	pour-cent	0	55	0	58	8 / 18	44,44	3,22
weekend	week-end	1	0	0	1	1 / 21	4,76	0,05
touts	tout	8	30	10	44	12 / 22	54,55	2
notres	nôtres	2	633	4	703	15 / 20	75	35,2
cettes	ces	17	48	5	74	10 / 22	45,45	3,36
journeaux	journaux	0	0	0	0	0 / 18	0	0
journals	journaux	0	1	0	1	1 / 18	5,56	0,06
acroiront	accroîtront	1	328	0	356	13 / 19	68,42	18,7
apprende	apprend	4	79	0	93	9 / 20	45	4,65
apprendent	apprennent	2	131	0	138	10 / 19	52,63	7,26
applaudent	applaudissent	0	7	0	7	5 / 18	27,78	0,39
augment	augmente	8	28	0	38	7 / 19	36,84	2
avont	ont	1	27	1	33	7 / 21	33,33	1,57
connaise	connaisse	5	400	2	446	13 / 19	68,42	23,5
constitute	constitue	6	212	0	246	13 / 18	72,22	13,7
corrompent	corrompent	1	55	0	57	7 / 18	38,89	3,17
croirent	croient	5	544	0	597	14 / 19	73,68	31,4
croyent	croient	1	8	0	9	5 / 19	26,32	0,47
croi	croit	9	15	1	27	11 / 22	50	1,23
devenira	deviendra	0	77	0	79	7 / 19	36,84	4,16

TAB. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
devienderé	deviendrai	2	59	0	65	6 / 20	30	3,25
deviendrent	deviennent	3	76	1	84	7 / 19	36,84	4,42
developpenet	développent	3	1	0	4	3 / 18	16,67	0,22
devont	doivent	4	7	0	12	4 / 20	20	0,6
seriont	serions	2	1143	0	1277	16 / 19	84,21	67,2
seron	serons	11	482	10	540	18 / 21	85,71	25,7
soyent	soient	0	16	0	17	5 / 20	25	0,85
etaients	étaient	0	1285	1	1396	19 / 20	95	69,8
experiment	expérience	7 /	18	0	28	4 / 18	22,22	1,56
faisent	font	1	299	0	318	13 / 19	68,42	16,7
faitent	font	3	277	0	291	14 / 20	70	14,6
faient	font	9	271	0	291	14 / 20	70	14,6
faissent	font	1	299	0	318	13 / 19	68,42	16,7
insist	insiste	11	124	0	183	6 / 22	27,27	8,32
limit	limite	8	15	0	30	3 / 22	13,64	1,36
not	note	5	50	6	67	12 / 23	52,17	2,91
pens	pense	22	45	2	84	8 / 22	36,36	3,82
peus	peux	9	129	1	177	13 / 22	59,09	8,05
pousent	poussent	2	69	2	73	8 / 19	42,11	3,84
préfér	préfère	7	18	0	36	6 / 22	27,27	1,64
prévoyent	prévoient	2	0	0	3	1 / 18	5,56	0,17
rest	reste	8	329	7	412	17 / 22	77,27	18,7

Tab. G.2 – Liste de résultats pour la méthode alpha-code élargie (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
romp	rompt	6	6	2	14	8 / 22	36,36	0,64
voyera	verra	1	2	0	3	2 / 20	10	0,15
voirai	verrai	2	19	1	23	9 / 21	42,86	1,1
donze	d'onze	1	22	1	26	11 / 21	52,38	1,24
acquerie	acquise	6	53	0	65	8 / 19	42,11	3,42
connerais	connais	14	801	4	873	15 / 18	83,33	48,5

G.3 Alphacode restreint

Le tableau suivant donne les résultats de la liste de mots pour l'alphacode restreint. La colonne *oui* donnent le nombre de mots retrouvés et sélectionnés par la méthode. La colonne *non* donne le nombre de mots retrouvés par la méthode qui ont été éliminés parce qu'ils dépassaient le seuil de distance lexicographique (§6.1.2.1). La colonne *elim* donne le nombre d'éléments dont la distance est inférieure au seuil mais qui sont éliminés (première lettre différente) La colonne *lex* donne le nombre total de mots, y compris les doublons. Enfin, les dernières colonnes donnent respectivement le nombre d'alphacodes donnant un résultat sur le nombre total d'alphacode, le pourcentage des alphacodes donnant un résultat et la moyenne des résultats par alphacode.

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. N**	**% props**	**Moy. / prop**
absorpsion	absorption	0	67	0	87	3 / 8	37,5	10,88
acessit	accessit	0	91	1	107	6 / 6	100	17,83
acceuil	accueil	0	30	1	44	5 / 6	83,33	7,33
aceuil	accueil	1	29	1	44	5 / 6	83,33	7,33
acolite	acolyte	1	28	0	29	4 / 7	57,14	4,14
addresse	adresse	0	96	4	120	4 / 5	80	24
address	adresse	3	93	0	120	4 / 5	80	24
aigüe	aiguë	3	5	1	9	3 / 5	60	1,8
aigue	aiguë	3	5	1	9	3 / 5	60	1,8
algorythme	algorithme	0	0	0	0	0 / 10	0	0
appas	appât	0	5	1	6	3 / 3	100	2
appogiature	appog(g)iature	0	40	0	40	7 / 9	77,78	4,44
aéropage	aréopage	1	26	0	30	5 /6	83,33	5
arome	arôme	4	33	4	49	2 / 5	40	9,8
asujettir	assujettir	12	115	0	146	8 / 8	100	18,25
attrapper	attraper	2	119	0	160	5 / 5	100	32
azalé	azalée	2	5	2	8	2 / 4	50	2
azalee	azalée	1	6	1	8	2 / 4	50	2
barette	barrette	4	66	6	95	5 / 5	100	19
barete	barrette	11	59	9	95	5 / 5	100	19
béquée	becquée	1	3	0	6	2 / 4	50	1,5
beckee	becquée	0	1	0	1	1 / 4	25	0,25

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
bifteek	bifteck	0	0	0	0	0 / 6	0	0
biftèque	bifteck	0	1	0	1	1 / 7	14,29	0,14
boîter	boiter	6	13	0	24	5 / 6	83,33	4
boursouffler	boursoufler	0	21	0	23	4 / 8	50	2,88
braîment	braiment	6	229	2	238	8 / 8	100	29,75
celà	cela	3	7	1	12	4 / 4	100	3
charriot	chariot	2	37	0	39	5 / 7	71,43	5,57
chariau	chariot	2	9	0	11	4 / 6	66,67	1,83
charette	charrette	3	101	0	133	6 / 6	100	22,17
chrysalyde	chrysalide	0	0	0	0	0 / 9	0	0
chrysantème	chrysanthème	0	14	0	14	3 / 10	30	1,4
comparition	comparution	0	38	0	40	6 / 9	66,67	4,44
comparison	comparaison	1	59	0	73	8 / 9	88,89	8,11
comcombre	concombre	0	16	0	20	3 / 6	50	3,33
concurent	concurrent	8	178	0	221	7 / 7	100	31,57
concuran	concurrent	0	11	0	11	4 / 6	66,67	1,83
congruement	congrûment	1	10	0	11	2 / 9	22,22	1,22
connection	connexion	0	49	1	66	6 / 6	100	11
consonnant	consonant	1	74	0	79	6 / 6	100	13,17
contigüe	contiguë	4	8	0	15	3 / 8	37,5	1,88
contigue	contiguë	4	8	0	15	3 / 8	37,5	1,88
controle	contrôle	3	107	0	126	7 / 7	100	18

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
control	contrôle	0	2	0	2	1 / 6	16,67	0,33
convaint	convainc	5	56	0	62	5 / 7	71,43	8,86
convin	convainc	1	4	1	5	3 / 5	60	1
coordonateur	coordinateur	0	94	0	102	6 / 9	66,67	11,33
courier	courrier	4	24	0	34	4 / 6	66,67	5,66
coutumace	contumace	1	13	0	14	5 / 7	71,43	2
cyprés	cyprès	0	6	0	10	1 / 6	16,67	1,66
cipre	cyprès	4	44	3	71	4 / 5	80	14,2
débarasser	débarrasser	0	100	0	118	4 / 6	66,67	19,67
déclancher	déclencher	1	14	0	18	5 / 8	62,5	2,25
déguingandé	dégingandé	2	25	0	32	6 / 7	85,71	4,57
dérilection	déréliction	0	86	0	93	8 / 9	88,89	10,33
dévôt	dévot	0	8	0	11	3 / 5	60	2,2
dilemne	dilemme	1	12	0	16	3 / 6	50	2,67
disgrâcier	disgracier	3	112	0	122	6 / 8	75	15,25
disparâte	disparate	1	457	0	482	8 / 8	100	60,25
drôlatique	drolatique	0	12	0	13	3 / 10	30	1,3
dislexie	dyslexie	0	16	0	24	2 / 6	33,33	4
échaufourée	échauffourée	0	27	0	30	5 / 8	62,5	3,75
anthropie	entropie	0	113	0	115	8 / 9	88,89	12,78
erronné	erroné	0	6	0	6	2 / 4	50	1,5
éthymologique	étymologique	1	0	0	1	1 / 11	9,09	0,09

Tab. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
filigramme	filigrane	0	18	1	18	3 / 8	37,5	2,25
gaité	gaieté	8	31	7	51	5 / 5	100	10,2
gheto	ghetto	0	3	0	3	2 / 5	40	0,6
guéto	ghetto	4	4	0	10	5 / 5	100	2
gueto	ghetto	4	4	0	10	5 / 5	100	2
halucination	hallucination	0	15	0	15	4 / 9	44,44	1,67
hypothénuse	hypoténuse	0	3	0	3	1 / 9	11,11	0,33
hipotenus	hypoténuse	0	12	0	14	3 / 9	33,33	1,56
imbécilité	imbécillité	1	4	0	8	2 / 7	28,57	1,14
infractus	infarctus	0	32	0	32	8 / 9	88,89	3,56
infarctusse	infarctus	1	165	0	168	8 / 10	80	16,8
inommé	innommé	0	17	4	26	4 / 5	80	5,2
insassiable	insatiable	0	120	0	139	7 / 7	100	19,86
intenssemment	intensément	0	150	0	200	5 / 6	83,33	33,33
intensement	intensément	0	150	0	200	5 / 6	83,33	33,33
macchiavélique	machiavélique	0	0	0	0	0 / 10	0	0
malaîse	malaise	3	149	2	184	6 / 6	100	30,67
malapris	malappris	1	17	0	19	4 / 7	57,14	2,71
malapri	malappris	1	17	0	19	4 / 6	66,67	31,17
malgrès	malgré	2	106	3	114	5 / 7	71,43	16,29
malgres	malgré	2	106	3	114	5 / 7	71,43	16,29
malgre	malgré	1	55	4	67	4 / 6	66,67	11,17

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
mapemonde	mappemonde	0	8	0	13	2 / 7	28,57	1,86
marâsme	marasme	1	130	3	161	5 / 5	100	32,2
marasm	marasme	1	15	1	18	4 / 4	100	4,5
négligeamment	négligemment	2	102	0	106	7 / 8	87,5	13,25
negligement	négligemment	1	28	0	32	6 / 7	85,71	4,57
aurenge	orange	2	80	0	100	5 / 6	83,33	16,67
occurence	occurrence	0	61	1	79	6 / 6	100	13,17
ocurance	occurrence	1	51	0	63	6 / 7	85,71	9
pannacée	panacée	1	20	0	26	4 / 5	80	5,2
panassée	panacée	3	65	0	91	5 /5	100	18,2
pantomine	pantomime	0	84	0	88	6 / 8	75	11
pécunière	pécuniaire	0	11	0	11	4 / 7	57,14	1,57
pélerine	pèlerine	1	19	0	24	3 / 6	50	4
piqure	piqûre	5	9	1	25	4 / 6	66,67	4,17
picur	piqûre	0	1	0	1	1 / 5	20	0,2
picure	piqûre	2	22	0	26	4 / 6	66,67	4,33
pickure	piqûre	0	1	0	1	1 / 7	14,29	0,14
précéde	précède	0	17	0	26	4 / 5	80	5,2
profesionel	professionnel	2	78	0	94	5 / 9	55,56	10,44
proffessionel	professionnel	2	78	0	94	5 / 9	55,56	10,44
proffesionel	professionnel	2	78	0	94	5 / 9	55,56	10,44
profesionnel	professionnel	2	78	0	94	5 / 9	55,56	10,44

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
protége	protège	3	21	0	31	3 / 6	50	5,17
psychadélique	psychédélique	2	0	0	2	1 / 12	8,33	0,17
psiquédélique	psychédélique	0	13	0	19	4 / 8	50	2,38
râtisser	ratisser	7	454	1	550	6 / 6	100	91,67
recoit	reçoit	3	38	1	52	6 / 6	100	8,67
ressoi	reçoit	2	52	1	77	4 / 5	80	15,4
reswa	reçoit	0	63	0	78	1 / 5	20	15,6
réddhibitoire	rédhibitoire	0	1	0	1	1 / 8	12,5	0,13
redibitoir	rédhibitoire	0	23	0	31	6 / 7	85,71	4,43
remerciment	remerciement	0	84	0	95	6 / 7	85,71	13,57
renumération	rémunération	0	196	0	196	9 / 9	100	21,78
shéma	schéma	2	24	0	30	1 / 5	20	6
schema	schéma	0	52	0	64	5 / 6	83,33	10,67
séborhée	séborrhée	0	23	0	27	3 / 6	50	4,5
soufle	souffle	5	29	2	45	6 / 6	100	7,5
soufl	souffle	2	5	0	10	3 / 5	60	2
subbit	subit	3	5	0	11	4 / 5	80	2,2
subcidiaire	subsidiaire	3	16	0	20	4 / 9	44,44	2,22
subsidière	subsidiaire	0	33	0	38	4 / 7	57,14	5,43
substanciel	substantiel	3	28	0	31	7 / 10	70	3,1
succint	succinct	0	15	0	16	3 / 6	50	2,67
suxin	succinct	0	4	0	10	1 / 5	20	2

Tab. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
superfètatoire	superfétatoire	0	77	0	77	7 / 10	70	7,7
simptomatique	symptomatique	0	13	0	13	3 / 10	30	1,3
sinptomatik	symptomatique	0	9	0	11	1 / 9	11,11	1,22
sindrome	syndrome	0	195	0	235	8 / 8	100	29,38
syndrôme	syndrome	0	13	0	13	1 / 8	12,5	1,63
synthése	synthèse	0	1	0	1	1 / 6	16,67	0,17
sintèz	synthèse	1	67	5	101	4 / 6	66,67	16,83
sinthèse	synthèse	0	66	0	97	4 / 6	66,67	16,17
sizygie	syzygie	0	4	0	5	1 / 6	16,67	0,83
traditionnaliste	traditionaliste	0	467	0	500	10 / 10	100	50
troglodite	troglodyte	0	5	0	6	3 / 8	37,5	0,75
esplication	explication	0	199	0	219	10 / 10	100	21,9
jud'aurenge	jus d'orange	0	6	0	6	3 / 8	37,50	0,75
quesque	qu'est-ce que	1	19	0	31	2 / 4	50	7,75
impère	impair	1	25	0	35	3 / 5	60	7
san	sans	2	4	4	7	2 / 3	66,67	2,33
adissione	additionne	1	107	0	127	7 / 7	100	18,14
adicione	additionne	0	26	0	28	7 / 7	100	4
chifre	chiffre	1	9	0	15	3 / 6	50	2,5
contien	contient	5	44	1	49	6 / 6	100	8,17
angletterre	Angleterre	0	149	0	163	6 / 7	85,71	23,28
presentimen	pressentiment	5	162	2	187	6 / 8	75	23,38

Tab. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
so	saut	1	0	0	2	1 / 2	50	1
premiere	première	1	25	0	35	3 / 5	60	7
videttes	vedettes	5	74	2	105	5 / 6	83,33	17,5
emploiés	employés	5	42	1	65	7 / 7	100	9,29
ecrire	écrire	0	13	1	17	4 / 4	100	4,25
raissonable	raisonnable	0	177	0	188	9 / 9	100	8,56
particuliaire	particulière	0	72	1	77	9 / 9	100	8,56
fenaitre	fenêtre	0	445	3	474	7 / 7	100	67,71
puit	puits	3	1	0	4	2 / 4	50	1
nonpossible	impossible	0	38	0	58	4 / 8	50	7,25
chauvinistiques	chauvins	0	1	0	1	1 / 11	9,09	0,09
votages	votes	3	7	1	10	4 / 7	57,14	1,43
political	politique	0	29	0	30	6 / 7	85,71	4,29
promoter	promouvoir	0	41	0	55	6 / 6	100	9,17
abandonnement	abandon	1	15	0	16	4 / 8	50	2
rivauté	rivalité	0	58	0	64	4 / 7	57,14	9,14
campaganer	faire campagne	1	2	0	3	2 / 8	25	0,38
sociéte	société	0	23	0	31	6 / 6	100	5,17
éffondrement	effondrement	3	67	0	71	6 / 8	75	8,88
extrème	extrême	0	7	0	7	1 / 5	20	1,4
gôut	goût	0	3	1	6	2 / 4	50	1,5
puls	plus	3	1	1	7	3 / 4	75	1,75

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	Correction	oui	non	elim	lex	props / Nb. W	% props	Moy. / prop
qunad	quand	0	1	0	1	1 / 5	20	0,2
obtinément	obstinément	0	40	0	46	6 / 7	85,71	6,57
environment	environnement	2	81	0	87	7 / 8	87,5	10,88
deveint	devient	13	42	0	65	5 / 6	83,33	10,83
ily a	il y a	0	6	1	6	1 / 4	25	1,5
y-a-t-il	y a-t-il	0	20	0	20	3 / 5	60	4
bi-lingue	bilingue	0	6	2	6	4 / 7	57,14	0,86
co-opération	coopération	1	409	1	421	3 / 21	14,29	20,05
extra-terrestre	extraterrestre	0	121	0	138	4 / 6	66,67	23
c'est-à dire	c'est-à-dire	0	347	0	375	7 / 8	87,5	46,88
autravers	au travers	1	160	1	183	7 / 7	100	26,14
autrechose	autre chose	0	101	0	106	7 / 9	77,78	11,78
déducation	déduction	1	46	0	52	6 / 9	66,67	5,78
parceque	parce que	2	15	0	20	3 / 7	42,86	2,88
pourcent	pour-cent	0	110	1	116	7 / 8	87,5	14,5
weekend	week-end	0	0	0	0	0 / 5	0	0
touts	tout	4	6	2	17	4 / 4	100	4,25
notres	nôtres	6	180	7	208	6 / 6	100	34,67
cettes	ces	4	32	11	47	3 / 4	75	11,75
journeaux	journaux	1	6	0	8	3 / 8	37,5	1
journals	journaux	1	3	1	4	3 / 8	37,5	0,5
acroiront	accroîtront	0	121	2	127	7 / 7	100	18,14

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
apprende	apprend	1	45	2	52	5 / 6	83,33	8,67
apprendent	apprennent	4	170	0	196	7 / 7	100	28
applaudent	applaudissent	0	12	0	12	6 / 8	75	1,5
augment	augmente	0	28	0	30	5 / 7	71,43	4,29
avont	ont	2	26	1	33	5 / 5	100	6,6
connaise	connaisse	7	147	0	205	7 / 7	100	29,29
constitute	constitue	0	142	0	176	8 / 8	100	22
corromprent	corrompent	1	91	0	94	8 / 8	100	11,75
croirent	croient	5	197	1	232	7 / 7	100	33,14
croyent	croient	0	90	2	104	3 / 7	42,86	14,86
croi	croit	5	3	2	9	4 / 4	100	2,25
devenira	deviendra	1	75	1	80	6 / 7	85,71	11,43
devienderé	deviendrai	0	51	0	64	5 / 6	83,33	10,67
deviendrent	deviennent	1	117	0	134	6 / 7	85,71	19,14
developpenet	développent	0	1	0	1	1 / 8	12,5	0,13
devont	doivent	6	15	0	25	4 / 6	66,67	4,17
seriont	serions	11	448	3	568	7 / 7	100	81,14
seron	serons	0	54	1	80	5 / 5	100	16
soyent	soient	2	33	1	41	3 / 6	50	6,83
etaients	étaient	3	318	1	394	6 / 6	100	65,67
experiment	expérience	1	44	1	47	6 / 8	75	5,88
faisent	font	8	260	3	317	7 / 7	100	45,29

TAB. G.3 – Liste de résultats pour la méthode alpha-code restreint (suite)

Mot	**Correction**	**oui**	**non**	**elim**	**lex**	**props / Nb. W**	**% props**	**Moy. / prop**
faitent	font	5	75	1	104	6 / 6	100	17,33
faient	font	5	75	3	104	6 / 6	100	17,33
faissent	font	6	262	3	317	7 / 7	100	45,29
insist	insiste	0	2	0	2	1 / 4	25	0,5
limit	limite	1	4	3	6	2 / 4	50	1,5
not	note	2	2	2	6	2 / 3	66,67	2
pens	pense	6	10	5	24	3 / 4	75	6
peus	peux	8	22	5	47	4 / 4	100	11,75
pousent	poussent	4	26	1	36	6 / 7	85,71	5,14
préfér	préfère	0	16	2	22	2 / 4	50	5,5
prévoyent	prévoient	0	9	0	10	2 / 8	25	1,25
rest	reste	0	51	11	74	3 / 4	75	18,5
romp	rompt	0	0	0	0	0 / 4	0	0
voyera	verra	0	0	0	0	0 / 6	0	0
voirai	verrai	6	12	0	20	2 / 5	40	4
donze	d'onze	3	5	3	13	2 / 5	40	2,6
acquerie	acquise	1	20	0	22	6 / 7	85,71	3,14
connerais	connais	4	497	5	565	8 / 8	100	70,63

G.4 Méthode phonétique

Voici les résultats de la méthode phonétique. La première colonne donne le nombre de chaînes phonétiques donnant un résultat sur le nombre total de chaînes phonétiques. *Prop* donne le nombre de propositions retrouvées et *Nouv* le nombre de propositions nouvelles (qui n'ont pas également été retrouvées par alpha-code). Les dernières colonnes donnent le pourcentage des chaînes donnant une proposition et la moyenne des propositions par chaîne.

Tab. G.4 – Liste de résultats pour la méthode phonétique

Mot	**Correction**	**Props / nb. Pho**	**Prop**	**Nouv**	**% props**	**Moy. / prop**
absorpsion	absorption	1 / 8	2	0	12,5	0,25
acessit	accessit	0 / 4	0	0	0	0
acceuil	accueil	0 / 6	0	0	0	0
aceuil	accueil	0 / 6	0	0	0	0
acolite	acolyte	1 / 4	2	2	25	0,5
addresse	adresse	1 / 4	3	1	25	0,75
address	adresse	1 / 4	3	1	25	0,75
aigüe	aiguë	1 / 2	4	1	50	2
aigue	aiguë	0 / 2	0	0	0	0
algorythme	algorithme	1 / 4	2	2	25	0,5
appas	appât	2 / 4	3	1	50	0,75
appogiature	appog(g)iature	0 / 8	0	0	0	0
aéropage	aréopage	0 / 16	0	0	0	0
arome	arôme	1 / 4	2	0	25	0,5
asujettir	assujettir	2 / 4	4	0	50	1
attrapper	attraper	0 / 12	0	0	0	0

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
azalé	azalée	1 / 8	2	0	12,5	0,25
azalee	azalée	0 / 4	0	0	0	0
barette	barrette	1 / 4	2	0	25	0,5
barete	barrette	0 / 6	0	0	0	0
béquée	becquée	0 / 4	0	0	0	0
beckee	becquée	0 / 2	0	0	0	0
bifteek	bifteck	0 / 1	0	0	0	0
biftèque	bifteck	1 / 2	4	4	50	2
boîter	boiter	0 / 6	0	0	0	0
boursouffler	boursoufler	0 / 3	0	0	0	0
braîment	braiment	0 / 4	0	0	0	0
celà	cela	2 / 6	2	1	33,33	0,33
charriot	chariot	1 / 4	2	0	25	0,5
chariau	chariot	1 / 4	2	2	25	0,5
charette	charrette	1 / 4	2	0	25	0,5
chrysalyde	chrysalide	1 / 2	2	0	50	1
chrysantème	chrysanthème	1 / 4	2	0	25	0,5
comparition	comparution	0 / 8	0	0	0	0
comparison	comparaison	0 / 8	0	0	0	0
comcombre	concombre	0 / 4	0	0	0	0
concurent	concurrent	1 / 4	2	0	25	0,5
concuran	concurrent	1 / 4	2	2	25	0,5

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
congruement	congrûment	0 / 4	0	0	0	0
connection	connexion	1 / 8	4	2	12,5	0,5
consonnant	consonant	0 / 8	0	0	0	0
contigüe	contiguë	1 / 2	2	1	50	1
contigue	contiguë	1 / 2	2	0	50	1
controle	contrôle	1 / 4	3	0	25	0,75
control	contrôle	1 / 4	3	1	25	0,75
convaint	convainc	1 / 4	5	2	25	1,25
convin	convainc	1 / 4	5	1	25	1,25
coordonateur	coordinateur	0 / 48	0	0	0	0
courier	courrier	1 / 2	4	0	50	2
coutumace	contumace	0 / 2	0	0	0	0
cyprés	cyprès	1 / 2	1	0	50	0,5
cipre	cyprès	0 / 1	0	0	0	0
débarasser	débarrasser	0 / 24	0	0	0	0
déclancher	déclencher	1 / 12	2	2	8,33	0,17
déguingandé	dégingandé	0 / 16	0	0	0	0
dérilection	déréliction	0 / 8	0	0	0	0
dévôt	dévot	1 / 4	2	0	25	0,5
dilemne	dilemme	0 / 2	0	0	0	0
disgrâcier	disgracier	2 / 4	10	2	50	2,5
disparâte	disparate	1 / 4	2	0	25	0,5

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
drôlatique	drolatique	0 / 4	0	0	0	0
dislexie	dyslexie	1 / 2	2	0	50	1
échaufourée	échauffourée	1 / 8	2	0	12,5	0,25
anthropie	entropie	0 / 4	0	0	0	0
erronné	erroné	1 / 8	4	0	12,5	0,5
éthymologique	étymologique	1 / 8	2	1	12,5	0,25
filigramme	filigrane	0 / 2	0	0	0	0
gaité	gaieté	2 / 4	14	8	50	3,5
gheto	ghetto	0 / 6	0	0	0	0
guéto	ghetto	1 / 4	2	2	25	0,5
gueto	ghetto	0 / 6	0	0	0	0
halucination	hallucination	1 / 8	2	0	12,5	0,25
hypothénuse	hypoténuse	0 / 4	0	0	0	0
hipotenus	hypoténuse	0 / 6	0	0	0	0
imbécilité	imbécillité	1 / 8	2	0	12,5	0,25
infractus	infarctus	0 / 4	0	0	0	0
infarctusse	infarctus	1 / 4	1	0	25	0,25
inommé	innommé	0 / 4	0	0	0	0
insassiable	insatiable	1 / 8	2	0	12,5	0,25
intenssemment	intensément	0 / 16	0	0	0	0
intensement	intensément	0 / 24	0	0	0	0
macchiavélique	machiavélique	1 / 8	2	0	12,5	0,25

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
malaîse	malaise	1 / 4	2	0	25	0,5
malapris	malappris	0 / 4	0	0	0	0
malapri	malappris	0 / 4	0	0	0	0
malgrès	malgré	1 / 4	1	0	25	0,25
malgres	malgré	0 / 2	0	0	0	0
malgre	malgré	0 / 2	0	0	0	0
mapemonde	mappemonde	1 / 12	2	0	8,33	0,17
marâsme	marasme	1 / 4	2	0	25	0,5
marasm	marasme	1 / 4	2	0	25	0,5
négligeamment	négligemment	1 / 8	1	0	12,5	0,13
negligement	négligemment	0 / 18	0	0	0	0
aurenge	orange	1 / 4	3	3	25	0,75
occurence	occurrence	1 / 4	2	0	25	0,5
ocurance	occurrence	1 / 4	2	1	25	0,5
pannacée	panacée	1 / 8	2	0	12,5	0,25
panassée	panacée	1 / 8	2	1	12,5	0,25
pantomine	pantomime	0 / 4	0	0	0	0
pécunière	pécuniaire	1 / 4	2	1	25	0,5
pélerine	pèlerine	0 / 6	0	0	0	0
piqure	piqûre	0 / 1	0	0	0	0
picur	piqûre	1 / 1	2	2	100	2
picure	piqûre	1 / 1	2	2	100	2

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
pickure	piqûre	1 / 1	2	2	100	2
précéde	précède	1 / 4	3	1	25	0,75
profesionel	professionnel	0 / 24	0	0	0	0
proffessionel	professionnel	1 / 16	4	0	6,25	0,25
proffesionel	professionnel	0 / 24	0	0	0	0
profesionnel	professionnel	0 / 24	0	0	0	0
protége	protège	1 / 4	3	0	25	0,75
psychadélique	psychédélique	0 / 4	0	0	0	0
psiquédélique	psychédélique	0 / 4	0	0	0	0
râtisser	ratisser	0 / 6	0	0	0	0
recoit	reçoit	0 / 6	0	0	0	0
ressoi	reçoit	1 / 6	2	1	16,67	0,33
reswa	reçoit	0 / 4	0	0	0	0
réddhibitoire	rédhibitoire	1 / 4	2	0	25	0,5
redibitoir	rédhibitoire	0 / 6	0	0	0	0
remerciment	remerciement	1 / 12	2	0	8,33	0,17
renumération	rémunération	0 / 24	0	0	0	0
shéma	schéma	1 / 4	2	0	25	0,5
schema	schéma	0 / 6	0	0	0	0
séborhée	séborrhée	0 / 8	0	0	0	0
soufle	souffle	1 / 1	3	1	100	3
soufl	souffle	1 / 1	3	1	100	3

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
subbit	subit	1 / 1	7	2	100	7
subcidiaire	subsidiaire	0 / 2	0	0	0	0
subsidière	subsidiaire	1 / 2	2	0	50	1
substanciel	substantiel	1 / 4	4	0	25	1
succint	succinct	1 / 2	2	0	50	1
suxin	succinct	1 / 2	2	2	50	1
superfètatoire	superfétatoire	1 / 16	2	0	6,25	0,13
simptomatique	symptomatique	1 / 8	2	0	12,5	0,25
sinptomatik	symptomatique	1 / 8	2	2	12,5	0,25
sindrome	syndrome	1 / 4	2	2	25	0,5
syndrôme	syndrome	1 / 4	2	0	25	0,5
synthése	synthèse	1 / 4	2	0	25	0,5
sintèz	synthèse	1 / 4	2	2	25	0,5
sinthèse	synthèse	1 / 4	2	2	25	0,5
sizygie	syzygie	0 / 1	0	0	0	0
traditionnaliste	traditionaliste	1 / 8	2	0	12,5	0,25
troglodite	troglodyte	1 / 4	2	2	25	0,5
esplication	explication	0 / 8	0	0	0	0
jud'aurenge	jus d'orange	0 / 4	0	0	0	0
quesque	qu'est-ce que	0 / 2	0	0	0	0
impère	impair	1 / 4	4	3	25	1
san	sans	2 / 2	10	7	100	5

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
adissione	additionne	1 / 4	3	2	25	0,75
adicione	additionne	1 / 4	3	3	25	0,75
chifre	chiffre	1 / 1	3	1	100	3
contien	contient	0 / 4	0	0	0	0
angletterre	Angleterre	0 / 8	0	0	0	0
presentimen	pressentiment	0 / 12	0	0	0	0
so	saut	1 / 2	8	6	50	4
premiere	première	0 / 3	0	0	0	0
videttes	vedettes	0 / 2	0	0	0	0
emploiés	employés	0 / 8	0	0	0	0
ecrire	écrire	0 / 3	0	0	0	0
raissonable	raisonnable	0 / 8	0	0	0	0
particuliaire	particulière	1 / 4	2	0	25	0,5
fenaitre	fenêtre	1 / 6	2	2	16,67	0,33
puit	puits	1 / 1	2	1	100	2
nonpossible	impossible	0 / 4	0	0	0	0
chauvinistiques	chauvins	0 / 2	0	0	0	0
votages	votes	0 / 4	0	0	0	0
political	politique	0 / 4	0	0	0	0
promoter	promouvoir	1 / 12	2	1	8,33	0,17
abandonnement	abandon	0 / 48	0	0	0	0
rivauté	rivalité	0 / 4	0	0	0	0

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	**Correction**	**Props / nb. Pho**	**Prop**	**Nouv**	**% props**	**Moy. / prop**
campaganer	faire campagne	0 / 24	0	0	0	0
sociéte	société	0 / 4	0	0	0	0
éffondrement	effondrement	1 / 24	2	0	4,17	0,08
extrème	extrême	1 / 4	2	0	25	0,5
gôut	goût	0 / 2	0	0	0	0
puls	plus	0 / 1	0	0	0	0
qunad	quand	0 / 2	0	0	0	0
obtinément	obstinément	0 / 8	0	0	0	0
environment	environnement	0 / 8	0	0	0	0
deveint	devient	1 / 6	3	1	16,67	0,5
ily a	il y a	1 / 2	1	0	50	0,5
y-a-t-il	y a-t-il	0 / 2	0	0	0	0
bi-lingue	bilingue	1 / 2	2	0	50	1
co-opération	coopération	1 / 32	3	1	3,13	0,09
extra-terrestre	extraterrestre	1 / 16	2	0	6,25	0,13
c'est-à dire	c'est-à-dire	0 / 4	0	0	0	0
autravers	au travers	1 / 8	1	0	12,5	0,13
autrechose	autre chose	1 / 12	1	0	8,33	0,08
déducation	déduction	0 / 8	0	0	0	0
parceque	parce que	0 / 6	0	0	0	0
pourcent	pour-cent	1 / 2	1	0	50	0,5
weekend	week-end	0 / 2	0	0	0	0

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
touts	tout	1 / 1	3	1	100	3
notres	nôtres	2 / 2	3	0	100	1,5
cettes	ces	1 / 2	5	1	50	2,5
journeaux	journaux	1 / 2	1	0	50	0,5
journals	journaux	1 / 2	1	0	50	0,5
acroiront	accroîtront	0 / 8	0	0	0	0
apprende	apprend	0 / 4	0	0	0	0
apprendent	apprennent	0 / 8	0	0	0	0
applaudent	applaudissent	0 / 8	0	0	0	0
augment	augmente	0 / 4	0	0	0	0
avont	ont	2 / 4	5	4	50	1,25
connaise	connaisse	0 / 4	0	0	0	0
constitute	constitue	0 / 2	0	0	0	0
corromprent	corrompent	1 / 8	2	1	12,5	0,25
croirent	croient	1 / 4	2	1	25	0,5
croyent	croient	2 / 4	3	3	50	0,75
croi	croit	1 / 2	8	2	50	4
devenira	deviendra	0 / 18	0	0	0	0
devienderé	deviendrai	0 / 36	0	0	0	0
deviendrent	deviennent	1 / 12	2	1	8,33	0,17
developpenet	développent	0 / 108	0	0	0	0
devont	doivent	2 / 6	3	3	33,33	0,5

TAB. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
seriont	serions	1 / 6	1	0	16,67	0,17
seron	serons	1 / 6	2	0	16,67	0,33
soyent	soient	1 / 4	1	1	25	0,25
etaients	étaient	0 / 4	0	0	0	0
experiment	expérience	0 / 12	0	0	0	0
faisent	font	2 / 6	3	4	33,33	0,67
faitent	font	2 / 4	2	1	50	0,5
faient	font	2 / 2	5	5	100	2,5
faissent	font	0 / 4	0	0	0	0
insist	insiste	0 / 2	0	0	0	0
limit	limite	0 / 1	0	0	0	0
not	note	1 / 2	1	1	50	0,5
pens	pense	2 / 2	10	9	100	5
peus	peux	1 / 3	4	3	33,33	1,33
pousent	poussent	0 / 2	0	0	0	0
préfér	préfère	1 / 4	3	1	25	0,75
prévoyent	prévoient	2 / 8	3	2	25	0,38
rest	reste	0 / 2	0	0	0	0
romp	rompt	2 / 2	6	5	100	3
voyera	verra	0 / 12	0	0	0	0
voirai	verrai	0 / 4	0	0	0	0
donze	d'onze	0 / 2	0	0	0	0

Tab. G.4 – Liste de résultats pour la méthode phonétique (suite)

Mot	Correction	Props / nb. Pho	Prop	Nouv	% props	Moy. / prop
acquerie	acquise	0 / 6	0	0	0	0
connerais	connais	0 / 12	0	0	0	0

G.5 Récapitulatif des résultats

Le tableau suivant récapitule les résultats. La colonne *Met* donne la ou les méthodes qui ont trouvé le résultat correct. La colonne *Sel* donne le nombre total de propositions. Les colonnes *A*, *W*, *N* et *P* récapitulent le nombre de mots retrouvés par chaque méthode. Enfin, *Auto* donne le nombre de mots qui passent le filtre de la distance lexicographique ; la différence avec *Sel* s'explique par le fait que les mots retrouvés par la méthode phonologique sont systématiquement sélectionnés.

Tab. G.5 – Récapitulatif des résultats

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
absorpsion	absorption	WP	2	1	7	67	2	75	2
acessit	accessit	-	0	6	306	91	0	403	0
acceuil	accueil	A	4	5	54	30	0	89	4
aceuil	accueil	A	5	5	54	30	0	89	5
acolite	acolyte	P	3	3	72	29	2	106	3
addresse	adresse	AP	20	46	446	96	3	589	20
address	adresse	AP	23	46	446	96	3	589	23

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
aigüe	aiguë	AP	8	1	46	8	4	56	8
aigue	aiguë	A	7	1	46	8	0	55	7
algorythme	algorithme	P	2	0	0	0	2	2	2
appas	appât	P	7	10	70	5	3	86	7
appogiature	appog(g)iature	-	0	5	17	40	0	62	0
aéropage	aréopage	A	3	9	54	27	0	90	3
arome	arôme	AP	6	3	70	37	2	110	6
asujettir	assujettir	AP	24	19	33	127	4	179	22
attrapper	attraper	A	15	55	442	121	0	618	15
azalé	azalée	AP	5	2	26	7	2	35	5
azalee	azalée	A	4	2	26	7	0	35	4
barette	barrette	AP	10	7	97	70	2	174	10
barete	barrette	A	17	7	97	70	0	174	17
béquée	becquée	-	1	0	3	4	0	7	1
beckee	becquée	-	0	0	0	1	0	1	0
bifteek	bifteck	W	3	1	2	0	0	3	3
biftèque	bifteck	P	4	0	0	1	4	5	4
boîter	boiter	A	13	7	64	19	0	90	13
boursouffler	boursoufler	A	12	9	8	21	0	38	12
braîment	braiment	A	8	16	73	235	0	324	8
celà	cela	AP	8	12	181	10	2	204	8
charriot	chariot	AP	4	8	32	39	2	79	4
chariau	chariot	P	4	1	8	11	2	22	4

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
charette	charrette	AP	12	41	191	104	2	336	12
chrysalyde	chrysalide	WP	2	0	2	0	2	2	2
chrysantème	chrysanthème	AP	2	2	0	14	2	16	2
comparition	comparution	W	1	4	41	38	0	83	1
comparison	comparaison	A	5	0	34	60	0	104	5
comcombre	concombre	W	1	0	7	16	0	23	1
concurent	concurrent	AP	24	67	225	186	2	478	24
concuran	concurrent	P	8	2	25	11	2	40	8
congruement	congrûment	-	1	0	3	11	0	14	1
connection	connexion	AP	12	18	243	49	4	312	12
consonnant	consonant	-	4	28	230	75	0	333	4
contigüe	contiguë	A	8	2	3	12	2	18	8
controle	contrôle	AP	14	32	141	110	3	283	14
control	contrôle	WP	8	1	41	2	3	45	7
convaint	convainc	NP	11	17	32	61	5	112	11
convin	convainc	WP	17	1	23	5	5	30	17
coordonateur	coordinateur	W	3	4	37	94	2	135	3
courier	courrier	AP	18	3	58	28	4	89	18
coutumace	contumace	W	2	3	25	14	0	42	2
cyprés	cyprès	AP	2	1	1	6	1	8	2
cipre	cyprès	-	5	9	111	48	0	168	5
débarasser	débarrasser	A	17	23	103	100	0	226	17
déclancher	déclencher	N	4	3	8	15	2	28	4

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
déguingandé	dégingandé	N	2	3	20	27	0	50	2
dérilection	déréliction	-	0	4	32	86	0	122	0
dévôt	dévot	AP	4	2	2	8	2	12	4
dilemne	dilemme	N	1	0	9	13	0	22	1
disgrâcier	disgracier	AP	21	15	13	115	10	145	20
disparâte	disparate	AP	7	54	134	458	2	646	7
drôlatique	drolatique	-	0	0	4	12	0	16	0
dislexie	dyslexie	WP	2	0	2	16	2	18	2
échaufourée	échauffourée	AP	2	9	20	27	2	56	2
anthropie	entropie	-	1	9	61	113	0	183	1
erronné	erroné	AP	7	11	218	6	4	235	7
éthymologique	étymologique	NP	2	1	2	1	2	5	2
filigramme	filigrane	-	0	0	2	18	0	20	0
gaité	gaieté	AP	23	9	134	39	14	190	21
gheto	ghetto	A	2	1	1	3	0	5	2
guéto	ghetto	P	6	8	45	8	2	63	6
gueto	ghetto	-	4	8	45	8	0	61	4
halucination	hallucination	AP	2	2	7	15	2	24	2
hypothénuse	hypoténuse	A	2	2	3	3	0	8	2
hipotenus	hypoténuse	-	0	0	4	12	0	16	0
imbécilité	imbécillité	AP	3	1	1	5	2	7	3
infractus	infarctus	A	1	1	20	32	0	53	1
infarctusse	infarctus	NP	1	9	23	166	1	198	1

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
inommé	innommé	-	0	5	55	17	0	77	0
insassiable	insatiable	WP	4	38	149	120	2	307	3
intenssemment	intensément	A	4	40	309	150	0	499	4
intensement	intensément	A	5	40	309	150	0	499	5
macchiavélique	machiavélique	AP	2	1	1	0	2	2	2
malaîse	malaise	AP	19	45	263	152	2	460	19
malapris	malappris	-	1	0	33	18	0	51	1
malapri	malappris	-	1	0	25	18	0	43	1
malgrès	malgré	NP	2	8	32	108	1	148	2
malgres	malgré	N	2	8	32	108	0	148	2
malgre	malgré	A	2	4	25	56	0	85	2
mapemonde	mappemonde	AP	2	1	3	8	2	12	2
marâsme	marasme	AP	9	69	514	131	2	714	9
marasm	marasme	WP	19	8	102	16	2	126	19
négligeamment	négligemment	NP	2	15	39	104	1	158	2
negligement	négligemment	A	2	6	26	29	0	61	2
aurenge	orange	P	5	9	82	82	3	176	5
occurence	occurrence	AP	2	17	156	61	2	234	2
ocurance	occurrence	NP	3	6	59	52	2	118	3
pannacée	panacée	AP	6	3	18	21	2	42	6
panassée	panacée	P	21	20	117	68	2	206	20
pantomine	pantomime	A	2	5	92	84	0	181	2
pécunière	pécuniaire	WP	2	2	14	11	2	28	2

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
pélerine	pèlerine	A	5	4	36	20	0	60	5
piqure	piqûre	A	10	14	46	14	0	74	10
picur	piqûre	P	2	0	3	1	2	6	2
picure	piqûre	P	5	1	18	24	2	45	5
pickure	piqûre	P	2	0	0	1	2	3	2
précéde	précède	AP	17	9	30	17	3	57	17
profesionel	professionnel	A	6	14	36	80	0	130	6
proffessionel	professionnel	AP	6	14	36	80	4	130	6
proffesionel	professionnel	A	5	14	36	80	0	130	5
profesionnel	professionnel	A	6	14	36	80	0	130	6
protége	protège	AP	16	4	23	24	3	51	16
psychadélique	psychédélique	N	2	0	0	2	0	2	2
psiquédélique	psychédélique	-	0	3	2	13	0	18	0
râtisser	ratisser	A	22	271	1552	461	0	2284	22
recoit	reçoit	A	4	14	126	41	0	181	4
ressoi	reçoit	P	5	13	298	54	2	366	4
reswa	reçoit	-	0	0	1	63	0	64	0
réddhibitoire	rédhibitoire	AP	2	1	1	1	2	3	2
redibitoir	rédhibitoire	W	1	1	27	23	0	51	1
remerciment	remerciement	AP	6	12	69	84	2	165	6
renumération	rémunération	A	1	22	115	196	0	333	1
shéma	schéma	WP	4	0	31	26	2	57	4
schema	schéma	A	2	13	72	52	0	137	2

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
séborhée	séborrhée	-	0	0	18	23	0	41	0
soufle	souffle	AP	17	22	38	34	3	95	17
soufl	souffle	WP	10	1	33	7	3	42	10
subbit	subit	AP	13	5	45	8	7	60	13
subcidiaire	subsidiaire	N	3	0	2	19	0	21	3
subsidière	subsidiaire	WP	2	0	22	33	2	55	1
substanciel	substantiel	NP	4	3	13	31	4	47	3
succint	succinct	AP	3	2	33	15	2	50	3
suxin	succinct	P	2	0	3	4	2	9	2
superfètatoire	superfétatoire	AP	2	2	8	77	2	87	2
simptomatique	symptomatique	WP	2	0	7	13	2	20	2
sinptomatik	symptomatique	P	2	0	0	9	2	11	2
sindrome	syndrome	P	2	77	188	195	2	462	2
syndrôme	syndrome	AP	2	2	0	13	2	15	2
synthése	synthèse	AP	4	2	7	1	2	10	4
sintèz	synthèse	P	6	5	73	68	2	148	6
sinthèse	synthèse	P	2	5	77	66	2	150	2
sizygie	syzygie	-	0	0	1	4	0	5	0
traditionnaliste	traditionaliste	AP	4	51	130	467	2	648	4
troglodite	troglodyte	P	2	0	7	5	2	14	2
esplication	explication	-	1	10	95	199	0	304	1
jud'aurenge	jus d'orange	-	0	0	1	6	0	7	0
quesque	qu'est-ce que	-	2	1	42	20	0	63	2

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
impère	impair	P	6	27	143	26	4	199	6
san	sans	AP	18	5	83	6	10	101	18
adissione	additionne	P	4	16	279	108	3	405	3
adicione	additionne	P	3	5	81	26	3	115	3
chifre	chiffre	AP	12	7	38	10	3	56	12
contien	contient	A	16	18	243	49	0	310	16
angletterre	Angleterre	A	1	24	107	149	0	280	1
presentimen	pressentiment	A	8	20	158	167	0	345	8
so	saut	P	22	4	38	1	8	49	22
premiere	première	A	7	27	143	26	0	196	7
videttes	vedettes	N	6	5	51	79	0	135	6
emploiés	employés	-	7	12	41	47	0	100	7
ecrire	écrire	A	25	21	169	13	0	203	25
raissonable	raisonnable	A	2	38	131	177	0	346	2
particuliaire	particulière	AP	2	25	63	72	2	160	1
fenaitre	fenêtre	P	2	105	407	445	2	959	2
puit	puits / puis	WP	8	0	7	4	2	12	8
nonpossible	impossible	-	0	0	6	38	0	44	0
chauvinistiques	chauvins	-	0	0	0	1	0	1	0
votages	votes	-	4	1	4	10	0	15	4
political	politique	-	0	4	23	29	0	56	0
promoter	promouvoir	-	10	21	113	41	2	176	10
abandonnement	abandon	-	1	1	10	16	0	27	1

TAB. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
rivauté	rivalité	-	0	8	58	58	0	124	0
campaganer	faire campagne	-	1	0	8	3	0	11	1
sociéte	société	A	4	11	126	23	0	160	4
éffondrement	effondrement	AP	5	5	8	70	2	83	5
extrème	extrême	AP	3	1	4	7	2	12	3
gôut	goût	A	9	1	12	3	0	16	9
puls	plus	A	9	2	28	4	0	34	9
qunad	quand	A	1	1	3	1	0	5	1
obtinément	obstinément	W	1	8	52	40	0	100	1
environment	environnement	A	4	5	14	83	0	102	4
deveint	devient	A	21	13	62	55	3	131	21
ily a	il y a	AP	1	1	1	6	1	8	1
y-a-t-il	y a-t-il	-	0	0	7	20	0	27	0
bi-lingue	bilingue	AP	2	1	1	6	2	8	2
co-opération	coopération	AP	4	49	192	410	3	652	4
extra-terrestre	extraterrestre	AP	2	4	41	121	2	166	2
c'est-à dire	c'est-à-dire	A	1	50	161	347	0	558	1
autravers	au travers	AP	2	17	62	161	1	240	2
autrechose	autre chose	AP	1	11	26	101	1	138	1
déducation	déduction	N	1	2	50	47	0	99	1
parceque	parce que	A	4	2	0	17	0	19	4
pourcent	pour-cent	AP	1	33	55	110	1	198	1
weekend	week-end	A	2	1	1	0	0	2	2

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
touts	tout	NP	15	2	38	10	3	51	15
notres	nôtres	AP	9	78	635	186	3	899	9
cettes	ces	-	24	2	65	36	5	104	22
journeaux	journaux	NP	1	0	0	7	1	7	1
journals	journaux	-	3	0	1	4	1	5	1
acroiront	accroîtront	W	2	32	329	121	0	482	2
apprende	apprend	A	8	16	83	46	0	145	8
apprendent	apprennent	N	6	27	133	174	0	334	6
applaudent	applaudissent	-	0	1	7	12	0	20	0
augment	augmente	A	13	7	36	28	0	71	13
avont	ont	-	7	1	28	28	5	61	7
connaise	connaisse	A	14	38	405	154	0	597	14
constitute	constitue	A	12	23	218	142	0	383	12
corromprent	corrompent	A	4	14	56	92	2	163	4
croirent	croient	A	12	44	549	202	2	796	12
croyent	croient	-	4	2	9	90	3	104	4
croi	croit	WP	16	1	24	8	8	35	16
devenira	deviendra	A	3	16	77	76	0	169	3
devienderé	deviendrai	W	2	7	61	51	0	119	2
deviendrent	deviennent	N	8	19	79	118	2	217	8
developpenet	développent	A	4	1	4	1	0	6	4
devont	doivent	-	13	0	11	21	3	35	13
seriont	serions	NP	16	229	1145	459	1	1833	16

Tab. G.5 – Récapitulatif des résultats (suite)

Mot	Correction	Met	Sel	A	W	N	P	Total	Auto
seron	serons	AP	13	35	493	54	2	582	13
soyent	soient	-	3	3	16	35	1	55	3
etaients	étaient	N	4	145	1285	321	0	1752	4
experiment	expérience	-	13	10	25	45	0	80	13
faisent	font	-	14	73	300	268	4	644	14
faitent	font	-	12	35	280	80	2	396	12
faient	font	-	23	34	280	80	5	399	23
faissent	font	-	9	73	300	268	0	641	9
insist	insiste	W	11	2	135	2	0	139	11
limit	limite	W	9	0	23	5	0	28	9
not	note	W	8	3	55	4	1	63	8
pens	pense	A	44	8	67	16	10	100	44
peus	peux	P	24	5	138	30	4	176	24
pousent	poussent	A	7	9	71	30	0	110	7
préfér	préfère	AP	12	4	25	16	3	46	12
prévoyent	prévoient	-	4	0	2	9	3	13	4
rest	reste	A	14	37	337	51	0	425	14
romp	rompt	W	11	0	12	0	6	17	11
voyera	verra	-	1	0	3	0	0	3	1
voirai	verrai	-	8	2	21	18	0	41	8
donze	d'onze	-	5	1	23	8	0	32	5
acquerie	acquise	-	10	4	59	21	0	84	10
connerais	connais	-	20	124	815	501	0	1440	20

Annexe H

Exemple de liste de propositions retrouvées par alpha-code

Exemple : coutumace

Alpha-code : cmtaeou

Le commentaire détaillé de ces résultats se trouve en p. 199.

Méthode	oui	limite [a]	non	Total propositions [b]	Total différents [c]
Alphacode simple	0	0	3	6	3
Alphacode élargi	1	0	24	28	25
Alphacode restreint	0	1	13	14	14
Totaux	1	1	40	48	41

TAB. H.1 – *Récapitulatif des résultats des alphacodes*

[a] Les résultats limite sont les proposition dont la distance lexicographique est égale au seuil de $\frac{2}{longueursource+longueurcible}$.

[b] Nombre de mots retournés par le lexique. Certaines chaînes de caractères peuvent se retrouver plusieurs fois, comme *marche* qui est un substantif et un verbe conjugué.

[c] Nombre de chaînes différentes, somme de "oui", "limite" et "non".

bcmtaeou	1	cdmtaeou	0	cfmtaeou	0	cgmtaeou	0
chmtaeou	0	cjmtaeou	0	ckmtaeou	0	clmtaeou	0
cmntaeou	5	cmptaeou	0	cmqtaeou	0	cmrtaeou	6
cmstaeou	14	cmtvaeou	0	cmtwaeou	0	cmtxaeou	1
cmtzaeou	1	cmtaeiou	0	cmtaeouy	0		

TAB. H.2 – *Récapitulatif des chaînes retrouvées par alphacode élargi*

mtaeou	1	ctaeou	7	cmaeou	0	cmteou	2
cmtaou	3	cmtaeu	1	cmtaeo	0		

TAB. H.3 – *Récapitulatif des chaînes retrouvées par alphacode restreint*

TAB. H.4 – Liste des 41 propositions retrouvées pour *coutumace*

Proposition	**Méth.**	**Alphacode**	**Distance**	**Seuil**
à court terme	W	cmrtaeou	$0.\overline{36}$	$0.\overline{09}$
accoutuma	N	cmtaou	$0.\overline{2}$	$0.\overline{1}$
accoutumâmes	W	cmstaeou	$0.1\overline{952380}$	$0.0\overline{952380}$
accoutumance	W	cmntaeou	$0.\overline{142857}$	$0.\overline{095238}$
accoutumasse	W	cmstaeou	$0.\overline{190476}$	$0.\overline{095238}$
accoutumasses	W	cmstaeou	$0.2\overline{27}$	$0.\overline{09}$
accoutumât	N	cmtaou	0.2157894736842105	0.1052631578947368
accoutumâtes	W	cmstaeou	$0.1\overline{952380}$	$0.\overline{095238}$
accoutume	A	cmtaeou	$0.\overline{2}$	$0.\overline{1}$
accoutumé	A	cmtaeou	$0.2\overline{27}$	$0.\overline{1}$
accoutumée	A	cmtaeou	0.2105263157894737	0.1052631578947368
accoutumées	W	cmstaeou	0.25	0.1
accoutument	W	cmntaeou	0.25	0.1
accoutumer	W	cmrtaeou	0.2631578947368421	0.1052631578947368
accoutumera	W	cmrtaeou	0.25	0.1
accoutumes	W	cmstaeou	0.263157894736842	0.1052631578947368
accoutumés	W	cmstaeou	0.2631578947368421	0.1052631578947368
accoutumez	W	cmtzaeou	0.2631578947368421	0.1052631578947368
au côté	N	ctaeou	0.44375	0.125
automate	N	mtaeou	0.2352941176470588	0.1176470588235294
Cocteau	N	ctaeou	0.3125	0.125
comateuse	W	cmstaeou	$0.\overline{3}$	$0.\overline{1}$
comateuses	W	cmstaeou	0.3684210526315789	0.1052631578947368
comateux	W	cmtxaeou	0.3529411764705883	0.1176470588235294
combattue	W	bcmtaeou	$0.\overline{3}$	$0.\overline{1}$
commuât	N	cmtaou	0.31875	0.125
commuâtes	W	cmstaeou	$0.28\overline{3}$	$0.\overline{1}$
Communauté	W	cmntaeou	0.2684210526315789	0.1052631578947368
communauté	W	cmntaeou	0.2684210526315789	0.1052631578947368
commutateur	W	cmrtaeou	0.3	0.1
contumace	W	cmntaeou	$0.0\overline{5}$	$0.\overline{1}$
coteau	N	ctaeou	$0.\overline{3}$	$0.1\overline{3}$
côteau	N	ctaeou	0.34	$0.1\overline{3}$

TAB. H.4 – Liste des 41 propositions retrouvées pour *coutumace* (suite)

Proposition	**Méth.**	**Alphacode**	**Distance**	**Seuil**
coûtâmes	W	cmstaeou	0.2411764705882353	0.1176470588235294
couteau	N	ctaeou	0.25	0.125
***coutume**	N	cmteou	0.125	0.125
écouta	N	ctaeou	$0.\overline{3}$	$0.1\overline{3}$
écoutâmes	W	cmstaeou	$0.2\overline{7}$	$0.\overline{1}$
écoutât	N	ctaeou	0.31875	0.125
écumât	N	cmtaeu	$0.40\overline{6}$	$0.1\overline{3}$
tout comme	N	cmteou	0.3157894736842105	0.1052631578947368

A = méthode alphacode, W = alphacode élargi, N = alphacode restreint

Annexe I

Exemples de comparaison "sémantique" de phrases

I.1 Phrase (61): qui mange la souris grise?

Question : qui mange la souris grise?
Réponse attendue : le chat noir mange la souris grise
Réponse de l'apprenant : le chat noir

```
************ Q U E S T I O N    I N P U T ************

qui mange la souris grise?


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : partial question
        Predicate      : manger
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : 'WH'
```

```
                Animate          : animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : souris
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : j
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : grise
                    ]CHS
                ]PSS
            }
        ]PSS
    }
]PSS

************ S Y S T E M   I N P U T *************

le chat noir mange la souris grise


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : manger
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : chat
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       : i
            ]DPS
            Satellites  {
```

```
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : noir
                    ]CHS
                ]PSS
            }
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : souris
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : j
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : grise
                    ]CHS
                ]PSS
            }
        ]PSS
    }
]PSS

************* L E A R N E R ' S   I N P U T *************

le chat noir


PSS[{ }
    DPS[
        Property         : chat
        Operator         : the
        Animate          : not animate
        Active          : active
        Gender           : masculine
        Number           : singular
        Ref. index       : i
    ]DPS
    Satellites  {
        PSS[{ }
            Value            : setRestriction
            CHS[
                Characteristic   : noir
            ]CHS
        ]PSS
    }
]PSS

***************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************
```

```
model: chat | learner: chat
type : model: DPS (agent) | learner: DPS
theta : model: agent |  learner:undefined



model: noir | learner: noir
type : model: CHS | learner: CHS


***************** M I S S I N G ****************************

model: manger
type :  CLS

model: souris (theme)  attached to: manger
type :  DPS

model: grise [setRestriction]  attached to: souris
type :  CHS



***************** A D D I T I O N A L ****************************

*** none

***************** M A N D A T O R Y ****************************

mandatory: chat found!
```

I.2 Phrase (62): qu'est-ce que le chat mange?

Question : qu'est-ce que le chat mange?
Réponse attendue : le chat mange une souris grise.
Réponse de l'apprenant : une souris blanche.

```
************* Q U E S T I O N    I N P U T *************

qu'est-ce que le chat mange?


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : partial question
        Predicate      : manger
    ]CLS
```

```
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : chat
                Operator         : the
                Animate          : not animate
                Active           : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
                Operator         : 'WH'
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
    }
]PSS

************* S Y S T E M   I N P U T *************

le chat mange une souris grise.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : manger
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : chat
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
```

```
            DPS[
                Property         : souris
                Operator         : some individual
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : i
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : grise
                    ]CHS
                ]PSS
            }
        ]PSS
    }
]PSS

************ L E A R N E R ' S   I N P U T ************

une souris blanche.


PSS[{ }
    DPS[
        Property         : souris
        Operator         : some individual
        Animate          : not animate
        Active          : active
        Gender           : feminine
        Number           : singular
        Ref. index       : i
    ]DPS
    Satellites  {
        PSS[{ }
            Value            : setRestriction
            CHS[
                Characteristic   : blanche
            ]CHS
        ]PSS
    }
]PSS

***************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************



model: souris | learner: souris
type : model: DPS (theme) | learner: DPS
theta : model: theme |  learner:undefined



model: grise | learner: blanche
type : model: CHS | learner: CHS
different lexical heads
```

```
***************** M I S S I N G ****************************

model: manger
type :  CLS

model: chat (agent)  attached to: manger
type :  DPS


***************** A D D I T I O N A L ****************************

*** none

***************** M A N D A T O R Y ****************************

mandatory: souris found!
```

I.3 Phrase (58a): as-tu vu la voiture rouge?

Question : as-tu vu la voiture rouge?
Réponse attendue : je l'ai vue.
Réponse de l'apprenant : je l'ai vu.

```
************* Q U E S T I O N    I N P U T *************

as-tu vu la voiture rouge?


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : global question
        Predicate      : voir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : second person
                Number           : singular
```

```
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : voiture
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : j
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : rouge
                    ]CHS
                ]PSS
            }
        ]PSS
    }
]PSS

************ S Y S T E M   I N P U T ************

je l'ai vue


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : voir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : first person
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
```

```
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
    }
]PSS

************ L E A R N E R ' S   I N P U T ************

je l'ai vu


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : voir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : first person
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
    }
]PSS
```

```
****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: voir | learner: voir
type : model: CLS | learner: CLS


model: delta | learner: delta
type : model: DPS (agent) | learner: DPS (agent)


model: delta | learner: delta
type : model: DPS (theme) | learner: DPS (theme)
differences in gender



****************** M I S S I N G ****************************

*** none

****************** A D D I T I O N A L ****************************

*** none
```

I.4 Phrase (63): C'est Jean qui m'a donné cette belle pomme rouge

Réponse attendue : C'est Jean qui m'a donné cette belle pomme rouge.
Réponse de l'apprenant : La pomme m'a été donnée par Jean.

```
************* S Y S T E M   I N P U T *************

C'est Jean qui m'a donné cette belle pomme rouge.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : donner
    ]CLS
    Satellites  {
        PSS[{ focus }
```

```
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       : i
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : patient
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : first person
                Number           : singular
                Ref. index       : j
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : pomme
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : k
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : belle
                    ]CHS
                ]PSS
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : rouge
                    ]CHS
                ]PSS
            }
        ]PSS
    }
]PSS

************* L E A R N E R ' S   I N P U T *************

La pomme m'a été donnée par Jean.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
```

```
        Aspect         : (non progressive, perfective)
        Voice          : passive
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : donner
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : pomme
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : patient
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Person           : first person
                Number           : singular
                Ref. index       : j
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS



****************** D I A G N O S I S ******************************

****************** M A T C H E S ******************************



model: donner | learner: donner
type : model: CLS | learner: CLS
voice : active  | learner: passive



model: pomme | learner: pomme
type : model: DPS (theme) | learner: DPS (theme)
```

```
operator : model : demonstrative | learner : the


model: delta | learner: delta
type : model: DPS (patient) | learner: DPS (patient)
differences in gender


model: Jean | learner: Jean
type : model: DPS (agent) | learner: DPS (agent)
features : model: { focus } | learner:{ }


***************** M I S S I N G ****************************

model: belle [setRestriction]  attached to: pomme
type :  CHS

model: rouge [setRestriction]  attached to: pomme
type :  CHS


***************** A D D I T I O N A L ****************************

*** none
```

I.5 Phrase (58c): Qu'est-ce que Jean a acheté?

Question : Qu'est-ce que Jean a acheté?
Réponse attendue : Jean a acheté un perroquet bleu.
Réponse de l'apprenant : un oiseau bleu.

```
************* Q U E S T I O N    I N P U T *************

Qu'est-ce que Jean a acheté?


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : partial question
        Predicate      : acheter
    ]CLS
    Satellites  {
```

```
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
                Operator         : 'WH'
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
    }
]PSS

************ S Y S T E M   I N P U T ************

Jean a acheté un perroquet bleu.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : acheter
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : perroquet
                Operator         : some individual
                Animate          : not animate
                Active          : active
```

```
                Gender           : masculine
                Number           : singular
                Ref. index       : i
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CHS[
                        Characteristic   : bleu
                    ]CHS
                ]PSS
            }
        ]PSS
    }
]PSS

************ L E A R N E R ' S   I N P U T ************

un oiseau bleu.


PSS[{ }
    DPS[
        Property         : oiseau
        Operator         : some individual
        Animate          : not animate
        Active          : active
        Gender           : masculine
        Number           : singular
        Ref. index       : i
    ]DPS
    Satellites  {
        PSS[{ }
            Value            : setRestriction
            CHS[
                Characteristic   : bleu
            ]CHS
        ]PSS
    }
]PSS



***************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************



model: bleu | learner: bleu
type : model: CHS | learner: CHS


model: perroquet | learner: oiseau
type : model: DPS (theme) | learner: DPS
theta : model: theme |  learner:undefined
different lexical heads



***************** M I S S I N G ****************************
```

```
model: acheter
type :  CLS

model: Jean (agent)  attached to: acheter
type :  DPS


***************** A D D I T I O N A L ****************************

*** none

***************** M A N D A T O R Y ****************************

mandatory: perroquet found!
```

I.6 Phrase (58d): Il a lu ces livres.

Réponse attendue : Il a lu ces livres.
Réponse de l'apprenant : Il a lu ses livres.

```
************* S Y S T E M   I N P U T *************

Il a lu ces livres.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : lire
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
```

```
            DPS[
                Property         : livre
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : plural
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS

************* L E A R N E R ' S   I N P U T *************

Il a lu ses livres.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : lire
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : livre
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : plural
                Ref. index       :
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : possessive
                    DPS[
                        Property         : delta
                        Operator         : delta
```

```
                        Animate          : not animate
                        Active          : active
                        Gender           : masculine
                        Person           : third person
                        Number           : singular
                        Ref. index       :
                    ]DPS
                ]PSS
            }
        ]PSS
    }
]PSS


****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: lire | learner: lire
type : model: CLS | learner: CLS


model: delta | learner: delta
type : model: DPS (agent) | learner: DPS (agent)


model: livre | learner: livre
type : model: DPS (theme) | learner: DPS (theme)
operator : model : demonstrative | learner : the



****************** M I S S I N G ****************************

*** none

****************** A D D I T I O N A L ****************************

learner: delta
 [possessive]  attached to: livre type :  DPS
```

I.7 Phrase (58e): quelle langue est parlée au Québec?

Question : quelle langue est parlée au Québec?
Réponse de l'apprenant : le français est parlé au Québec.

```
************* Q U E S T I O N    I N P U T *************

quelle langue est parlée au Québec?
```

```
PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : passive
        Causative      : not causative
        Negation       : not negated
        Utterance type : partial question
        Predicate      : parler
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : langue
                Operator         : 'WH'
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
        PSS[{ }
            Value            : directionIn
            DPS[
                Property         : Québec
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS

************ S Y S T E M   I N P U T ************

Le français est parlé au Québec.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : passive
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : parler
    ]CLS
    Satellites  {
        PSS[{ }
```

```
            Theta role       : theme
            DPS[
                Property         : français
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
        PSS[{ }
            Value            : directionIn
            DPS[
                Property         : Québec
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS
```

I.7.1 Réponse 1: On parle français au Québec.

```
************ L E A R N E R ' S   I N P U T ************

On parle français au Québec.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : parler
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       :
```

```
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : français
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Value            : directionIn
            DPS[
                Property         : Québec
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS


****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: parler | learner: parler
type : model: CLS | learner: CLS
voice : passive  | learner: active



model: français | learner: français
type : model: DPS (theme) | learner: DPS (theme)
operator : model : the | learner : delta



model: Québec | learner: Québec
type : model: DPS | learner: DPS


****************** M I S S I N G ****************************

*** none

****************** A D D I T I O N A L ****************************

learner: delta
 (theme)  attached to: parler type :  DPS


****************** M A N D A T O R Y ****************************
```

```
mandatory: français found!
```

I.7.2 Réponse 2: On parle le français au Québec.

```
************ L E A R N E R ' S   I N P U T ************

On parle le français au Québec.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : parler
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : français
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Value            : directionIn
            DPS[
                Property         : Québec
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
```

```
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS


****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: parler | learner: parler
type : model: CLS | learner: CLS
voice : passive  | learner: active


model: français | learner: français
type : model: DPS (theme) | learner: DPS (theme)


model: Québec | learner: Québec
type : model: DPS | learner: DPS


****************** M I S S I N G ****************************

*** none

****************** A D D I T I O N A L ****************************

learner: delta
 (theme)  attached to: parler type :  DPS


****************** M A N D A T O R Y ****************************

mandatory: français found!
```

I.7.3 Réponse 3: Le français.

```
************* L E A R N E R ' S   I N P U T *************

Le français.


PSS[{ }
    DPS[
        Property         : français
        Operator         : the
        Animate          : not animate
        Active          : active
        Gender           : masculine
        Number           : singular
        Ref. index       :
```

```
    ]DPS
]PSS


****************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************


model: français | learner: français
type : model: DPS (theme) | learner: DPS
theta : model: theme |  learner:undefined


***************** M I S S I N G ****************************

model: parler
type :  CLS

model: Québec [directionIn]  attached to: parler
type :  DPS


***************** A D D I T I O N A L ****************************

*** none

***************** M A N D A T O R Y ****************************

mandatory: français found!
```

I.7.4 Réponse 4: Le français se parle au Québec.

```
************ L E A R N E R ' S   I N P U T ************

Le français se parle au Québec


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : parler
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : theme
```

```
            DPS[
                Property         : français
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
        PSS[{ }
            Value            : directionIn
            DPS[
                Property         : Québec
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS


****************** D I A G N O S I S ******************************

****************** M A T C H E S ******************************


model: parler | learner: parler
type : model: CLS | learner: CLS
voice : passive  | learner: active


model: français | learner: français
type : model: DPS (theme) | learner: DPS (theme)


model: Québec | learner: Québec
type : model: DPS | learner: DPS


****************** M I S S I N G ******************************
```

```
*** none

****************** A D D I T I O N A L ****************************

learner: delta
 (theme)  attached to: parler type :  DPS


****************** M A N D A T O R Y ****************************

mandatory: français found!
```

I.8 Phrase (64): est-ce que Jean a lu ce livre?

Réponse attendue : est-ce que Jean a lu ce livre?
Réponse de l'apprenant : Jean a-t-il lu ce livre?

```
************* S Y S T E M   I N P U T *************

est-ce que Jean a lu ce livre?


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : global question
        Predicate      : lire
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : livre
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
```

```
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS

************* L E A R N E R ' S   I N P U T *************

Jean a-t-il lu ce livre?


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : global question
        Predicate      : lire
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       : j
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : livre
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS



***************** D I A G N O S I S ****************************

****************** M A T C H E S ******************************


model: lire | learner: lire
type : model: CLS | learner: CLS
```

```
model: Jean | learner: Jean
type : model: DPS (agent) | learner: DPS (agent)


model: livre | learner: livre
type : model: DPS (theme) | learner: DPS (theme)


***************** M I S S I N G ****************************

*** none

***************** A D D I T I O N A L ****************************

*** none
```

I.9 Phrase (65): la sœur du capitaine qui a été tuée

Réponse attendue : la sœur du capitaine qui a été tuée
Réponse de l'apprenant : la sœur du capitaine qui a été tué

```
************* S Y S T E M   I N P U T *************

la soeur du capitaine qui a été tuée


PSS[{ }
    DPS[
        Property         : soeur
        Operator         : the
        Animate          : not animate
        Active          : active
        Gender           : feminine
        Number           : singular
        Ref. index       : i
    ]DPS
    Satellites  {
        PSS[{ }
            Value            : possessive
            DPS[
                Property         : capitaine
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Value            : setRestriction
            CLS[
                Mood           : real
                Tense          : E = S
                InfoFunction   : categorical
```

```
                Modality       : undefined
                Aspect         : (non progressive, perfective)
                Voice          : passive
                Causative      : not causative
                Negation       : not negated
                Utterance type : declaration
                Predicate      : tuer
            ]CLS
            Satellites  {
                PSS[{ }
                    Theta role       : theme
                    DPS[
                        Property         : soeur
                        Operator         : the
                        Animate          : not animate
                        Active          : active
                        Gender           : feminine
                        Number           : singular
                        Ref. index       : i
                    ]DPS
                ]PSS
            }
        ]PSS
    }
]PSS

************ L E A R N E R ' S   I N P U T ************

la soeur du capitaine qui a été tué


PSS[{ }
    DPS[
        Property         : soeur
        Operator         : the
        Animate          : not animate
        Active          : active
        Gender           : feminine
        Number           : singular
        Ref. index       :
    ]DPS
    Satellites  {
        PSS[{ }
            Value            : possessive
            DPS[
                Property         : capitaine
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       : i
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CLS[
                        Mood           : real
                        Tense          : E = S
                        InfoFunction   : categorical
                        Modality       : undefined
                        Aspect         : (non progressive, perfective)
```

```
                    Voice          : passive
                    Causative      : not causative
                    Negation       : not negated
                    Utterance type : declaration
                    Predicate      : tuer
                ]CLS
                Satellites  {
                    PSS[{ }
                        Theta role       : theme
                        DPS[
                            Property         : capitaine
                            Operator         : the
                            Animate          : not animate
                            Active          : active
                            Gender           : masculine
                            Number           : singular
                            Ref. index       : i
                        ]DPS
                    ]PSS
                }
            ]PSS
        }
    ]PSS
}
]PSS


****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: soeur | learner: soeur
type : model: DPS (theme) | learner: DPS
theta : model: theme |  learner:undefined


model: capitaine | learner: capitaine
type : model: DPS | learner: DPS


model: tuer | learner: tuer
type : model: CLS | learner: CLS


model: soeur | learner: capitaine
type : model: DPS (theme) | learner: DPS (theme)
different lexical heads
differences in gender


****************** M I S S I N G ****************************

*** none

****************** A D D I T I O N A L ****************************

*** none
```

I.10 Phrase (66): Avec ses jumelles, Jean a regardé cet homme.

Réponse attendue : Avec ses jumelles, Jean a regardé cet homme.
Réponse de l'apprenant : Jean a regardé cet homme avec ses jumelles.

```
************ S Y S T E M   I N P U T ************

Avec ses jumelles, Jean a regardé cet homme.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : regarder
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : homme
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ topic }
            Value            : withManner
            DPS[
                Property         : jumelle
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : plural
                Ref. index       :
            ]DPS
```

```
            Satellites  {
                PSS[{ }
                    Value            : possessive
                    DPS[
                        Property         : delta
                        Operator         : delta
                        Animate          : not animate
                        Active          : active
                        Gender           : feminine
                        Person           : third person
                        Number           : singular
                        Ref. index       :
                    ]DPS
                ]PSS
            }
        ]PSS
    }
]PSS

************ L E A R N E R ' S   I N P U T ************

Jean a regardé cet homme avec ses jumelles.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : regarder
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : homme
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Value            : withManner
```

```
            DPS[
                Property         : jumelle
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : plural
                Ref. index       :
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : possessive
                    DPS[
                        Property         : delta
                        Operator         : delta
                        Animate          : not animate
                        Active          : active
                        Gender           : feminine
                        Person           : third person
                        Number           : singular
                        Ref. index       :
                    ]DPS
                ]PSS
            }
        ]PSS
    }
]PSS


***************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************


model: regarder | learner: regarder
type : model: CLS | learner: CLS


model: Jean | learner: Jean
type : model: DPS (agent) | learner: DPS (agent)


model: homme | learner: homme
type : model: DPS (theme) | learner: DPS (theme)


model: jumelle | learner: jumelle
type : model: DPS | learner: DPS
features : model: { topic } | learner:{ }



model: delta | learner: delta
type : model: DPS | learner: DPS


***************** M I S S I N G ****************************

*** none
```

```
***************** A D D I T I O N A L ****************************

*** none
```

I.11 Phrase (67): Jean a observé cet homme avec ses jumelles.

Réponse attendue : Jean a observé cet homme avec ses jumelles.
Réponse de l'apprenant : Jean a observé cet homme.

```
************* S Y S T E M   I N P U T *************

Jean a observé cet homme avec ses jumelles.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : observer
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : homme
                Operator         : demonstrative
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Value            : withManner
            DPS[
```

```
                Property         : jumelle
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : plural
                Ref. index       :
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : possessive
                    DPS[
                        Property         : delta
                        Operator         : delta
                        Animate          : not animate
                        Active          : active
                        Gender           : feminine
                        Person           : third person
                        Number           : singular
                        Ref. index       :
                    ]DPS
                ]PSS
            }
        ]PSS
    }
]PSS

************ L E A R N E R ' S   I N P U T ************

Jean a observé cet homme.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : observer
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : Jean
                Operator         : delta
                Animate          : not animate
                Active          : active
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : homme
                Operator         : demonstrative
                Animate          : not animate
```

```
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS


****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: observer | learner: observer
type : model: CLS | learner: CLS


model: Jean | learner: Jean
type : model: DPS (agent) | learner: DPS (agent)


model: homme | learner: homme
type : model: DPS (theme) | learner: DPS (theme)


****************** M I S S I N G ****************************

model: jumelle [withManner]  attached to: observer
type :  DPS

model: delta [possessive]  attached to: jumelle
type :  DPS



****************** A D D I T I O N A L ****************************

*** none
```

I.12 Phrase (68): le chien est noir.

```
************ S Y S T E M   I N P U T ************

le chien est noir.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
```

```
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : noir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : chien
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
    }
]PSS
```

I.12.1 Réponse 1: il est noir

```
************* L E A R N E R ' S   I N P U T *************

Il est noir.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : noir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : delta
                Operator         : delta
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Person           : third person
                Number           : singular
                Ref. index       : i
            ]DPS
        ]PSS
    }
]PSS
```

```
****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: noir | learner: noir
type : model: CLS | learner: CLS


model: chien | learner: delta
type : model: DPS (agent) | learner: DPS (agent)
different lexical heads
operator : model : the | learner : delta


****************** M I S S I N G ****************************

*** none

****************** A D D I T I O N A L ****************************

*** none
```

I.12.2 Réponse 2: noir

```
************ L E A R N E R ' S   I N P U T ************

noir


PSS[{ }
    Value            : setRestriction
    CHS[
        Characteristic   : noir
    ]CHS
]PSS


****************** D I A G N O S I S ****************************

****************** M A T C H E S ****************************


model: noir | learner: noir
type : model: CLS | learner: CHS


****************** M I S S I N G ****************************

model: chien (agent)  attached to: noir
```

```
type :  DPS


****************** A D D I T I O N A L ****************************
```

I.13 (69): les enfants que tu as vus hier dorment.

```
************* S Y S T E M   I N P U T *************

les enfants que tu as vus hier dorment.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : dormir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : patient
            DPS[
                Property         : enfant
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : plural
                Ref. index       : i
            ]DPS
            Satellites  {
                PSS[{ }
                    Value            : setRestriction
                    CLS[
                        Mood           : real
                        Tense          : E = S
                        InfoFunction   : categorical
                        Modality       : undefined
                        Aspect         : (non progressive, perfective)
                        Voice          : active
                        Causative      : not causative
                        Negation       : not negated
                        Utterance type : declaration
                        Predicate      : voir
                    ]CLS
                    Satellites  {
                        PSS[{ }
                            Theta role       : agent
                            DPS[
```

```
                                Property         : delta
                                Operator         : delta
                                Animate          : not animate
                                Active          : active
                                Gender           : masculine
                                Person           : second person
                                Number           : singular
                                Ref. index       :
                            ]DPS
                        ]PSS
                        PSS[{ }
                            Theta role       : theme
                            DPS[
                                Property         : enfant
                                Operator         : the
                                Animate          : not animate
                                Active          : active
                                Gender           : masculine
                                Number           : plural
                                Ref. index       : i
                            ]DPS
                        ]PSS
                        PSS[{ }
                            Theta role       : predication
                            Value            : when
                            DPS[
                                Property         : hier
                                Operator         : generic
                                Animate          : not animate
                                Active          : active
                                Gender           : masculine
                                Number           : singular
                                Ref. index       :
                            ]DPS
                        ]PSS
                    }
                ]PSS
            }
        ]PSS
    }
]PSS
```

I.13.1 Reponse 1: les gamins mangent une pomme

```
************* L E A R N E R ' S   I N P U T *************

les gamins mangent une pomme.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
```

```
        Negation       : not negated
        Utterance type : declaration
        Predicate      : manger
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : agent
            DPS[
                Property         : gamin
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : plural
                Ref. index       :
            ]DPS
        ]PSS
        PSS[{ }
            Theta role       : theme
            DPS[
                Property         : pomme
                Operator         : some individual
                Animate          : not animate
                Active          : active
                Gender           : feminine
                Number           : singular
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS



****************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************

*** no matches

***************** M I S S I N G ****************************

model: dormir
type :  CLS

model: enfant (patient)  attached to: dormir
type :  DPS

model: voir [setRestriction]  attached to: enfant
type :  CLS

model: delta (agent)  attached to: voir
type :  DPS

model: enfant (theme)  attached to: voir
type :  DPS

model: hier (predication)  [when]  attached to: voir
type :  DPS
```

```
****************** A D D I T I O N A L ****************************

learner: manger
 type :  CLS
learner: gamin
 (agent)  attached to: manger type :  DPS
learner: pomme
 (theme)  attached to: manger type :  DPS
```

I.13.2 Reponse 2: les gamins dorment.

```
************ L E A R N E R ' S   I N P U T ************

les gamins dorment.


PSS[{ }
    CLS[
        Mood           : real
        Tense          : E = S
        InfoFunction   : categorical
        Modality       : undefined
        Aspect         : (non progressive, non perfective)
        Voice          : active
        Causative      : not causative
        Negation       : not negated
        Utterance type : declaration
        Predicate      : dormir
    ]CLS
    Satellites  {
        PSS[{ }
            Theta role       : patient
            DPS[
                Property         : gamin
                Operator         : the
                Animate          : not animate
                Active          : active
                Gender           : masculine
                Number           : plural
                Ref. index       :
            ]DPS
        ]PSS
    }
]PSS


***************** D I A G N O S I S ****************************

***************** M A T C H E S ****************************


model: dormir | learner: dormir
type : model: CLS | learner: CLS


model: enfant | learner: gamin
```

```
type : model: DPS (patient) | learner: DPS (patient)
different lexical heads



****************** M I S S I N G ******************************

model: voir [setRestriction]  attached to: enfant
type :  CLS

model: delta (agent)  attached to: voir
type :  DPS

model: enfant (theme)  attached to: voir
type :  DPS

model: hier (predication)  [when]  attached to: voir
type :  DPS



****************** A D D I T I O N A L ******************************

*** none
```