Atténuation des surdétections d'un correcteur grammatical de qualité commerciale

Fabrizio Gotti* — **Philippe Langlais*** — **Guy Lapalme***
Simon Charest† — **Éric Brunelle†**

* *RALI/DIRO, Université de Montréal, C.P. 6128, Succ. Centre-Ville, H3C 3J7 Montréal (Québec), Canada[3pt] — {gottif,felipe,lapalme}@iro.umontreal.ca*

† *Druide informatique inc., 1435 rue Saint-Alexandre, bureau 1040, H3A 2G4 Montréal (Québec), Canada[3pt] — developpement@druide.com*

RÉSUMÉ. *Nous décrivons une étude menée à l'initiative de Druide informatique inc. conjointement avec le* RALI *visant à mettre au point un détecteur de surdétections, c'est-à-dire un système capable d'identifier les détections proposées à tort par un correcteur grammatical. Plusieurs classifieurs ont été entraînés de manière supervisée sur quatorze types de fautes détectées par un correcteur grammatical de qualité commerciale. Huit des quatorze classifieurs développés dans ce travail font partie de la plus récente édition du correcteur, intégré dans un assistant de rédaction grand public. Ce travail mené sur une période de six mois est un exemple réussi de déploiement d'une approche d'ingénierie statistique au service d'une application langagière robuste.*

ABSTRACT. *We describe a study conducted on the proposal of Druide informatique inc., in collaboration with* RALI*, aiming at developing a system capable of detecting "overdetections", i.e. designed for filtering detections erroneously flagged by a grammar checker. Various families of classifiers have been trained in a supervised way for 14 types of detections made by a commercial grade French grammar checker. Eight of the 14 classifiers we devised are now part of the latest edition of the grammar checker, embedded in a popular writing assistant. This project was conducted over a six-month period and is an interesting illustration of how a machine learning component can be successfully embedded in a robust, popular natural language application for commercial use.*

MOTS-CLÉS : *correction grammaticale, classification de détections, ingénierie langagière.*

KEYWORDS: *grammatical checking, classification of detections, natural language engineering.*

1. Introduction

Pour bien des utilisateurs de systèmes informatiques actuels, la correction grammaticale par ordinateur s'avère la manifestation tangible de plusieurs découvertes en traitement automatique des langues naturelles. Ces modules de correction grammaticale sont typiquement intégrés à des outils d'aide à la rédaction, et sont de plus en plus sollicités lors de la rédaction de la masse grandissante des documents qui sont rédigés en ligne (Ehsan et Faili, 2012). Ils ont également une vocation pédagogique évidente, notamment dans l'apprentissage des langues secondes (Morin, 1995 ; Knutsson *et al.*, 2003 ; Napolitano et Stent, 2009). Leacock *et al.* (2010) dressent une cartographie complète des travaux menés en détection d'erreurs pour les apprenants d'une langue seconde.

Ces outils incluent des ressources linguistiques aussi riches que disparates, tels des dictionnaires à large couverture (voir notamment (Fontenelle, 2005)), des modèles de langue probabilistes (par exemple (Napolitano et Stent, 2009)), des analyseurs syntaxiques avec relâchement de contraintes (Clément *et al.*, 2009) ou tout simplement des règles symboliques (Sofkova Hashemi, 2001).

On le devine donc, le développement d'un correcteur grammatical est un processus complexe nécessitant de délicats choix d'ordres scientifique et industriel, comme en témoignent par exemple les efforts menés chez Microsoft pour développer un correcteur grammatical multilingue (Helfrich et Music, 2000). L'amélioration d'un produit ayant déjà fait ses preuves peut s'avérer tout aussi difficile.

Le projet *Scorali* que nous présentons ici porte justement sur l'amélioration d'une technologie de correction grammaticale de qualité commerciale appelée *Analytix* et développée par la société Druide.

Ce travail, fruit d'un partenariat entre un laboratoire de recherche universitaire et un industriel renommé, était encadré par un cahier des charges présentant certains défis. Notamment, Druide a puisé dans son expertise du domaine pour fixer des objectifs d'amélioration clairs et chiffrés qui ont su nous guider dans notre travail. Druide a également fixé une contrainte importante : le traitement doit se faire sur la sortie du correcteur, sans accès ni modification à son fonctionnement interne.

On a donc privilégié la conception d'un filtre posté *en aval* des détections produites. Ce filtre doit retenir ou rejeter les détections afin d'améliorer la justesse du correcteur. La stratégie a plusieurs mérites. Il nous est ainsi possible d'éviter une chirurgie au cœur d'une application au code propriétaire, ce qui ne serait pas nécessairement plus simple. Nous nous épargnons ainsi l'examen de plusieurs modules du correcteur grammatical, qui doit, quant à lui, analyser la phrase, localiser les fautes et proposer les corrections.

De plus, ainsi découplée du produit qu'elle cherche à améliorer, notre approche pourrait théoriquement être adaptée à un correcteur autre que celui utilisé dans notre

étude. Il serait ainsi fort intéressant d'étudier notre stratégie sur Cordial[1] (Laurent *et al.*, 2009) réputé pour sa qualité d'analyse. Certes, le travail que nous décrivons requiert l'accès à un nombre important d'informations produites lors de l'analyse, mais il peut néanmoins constituer un premier indice que l'idée d'un filtre en post-traitement est viable pour un analyseur reposant sur des ressources riches et variées. On verra enfin que, indépendamment de cette stratégie de découplage du filtre et du correcteur, nous aurons quand même la possibilité de fournir une rétroaction intéressante sur le fonctionnement de l'analyseur.

Dans un sens, notre travail revient à effectuer une *fouille d'erreurs* sur les sorties d'une chaîne de traitement de TAL, afin d'en guider l'amélioration éventuelle. D'autres nous ont précédés. L'étude de Van Noord (2004) a proposé de vérifier automatiquement les sorties d'un analyseur syntaxique. Son approche consiste à faire l'analyse d'un grand nombre de phrases néerlandaises à l'aide d'Alpino[2], un analyseur de dépendances syntaxiques à large couverture, puis à comparer la fréquence des mots et séquences de mots dans les phrases qui font échouer l'analyseur avec la fréquence des mêmes mots et séquences dans les phrases qui ne posent pas de problèmes. L'isolation des séquences suspectes a permis à Van Noord d'identifier à la fois des erreurs dans les ressources linguistiques utilisées par Alpino et dans les règles de sa grammaire de détection. Ainsi, il a isolé des erreurs intervenant lors de la segmentation en mots, dans le lexique, dans le traitement des expressions idiomatiques ou de certaines entités nommées. Ceci a permis l'amélioration d'Alpino.

Sagot et de la Clergerie (2009) s'inspirent de l'étude de Van Noord et vont plus loin en tentant d'identifier, pour chacune des phrases pour lesquelles l'analyse a échoué, le mot qui est le plus susceptible d'avoir causé cet échec. Ils appellent ce mot *suspect principal*. Les auteurs étendent leur stratégie aux bigrammes de formes et de lemmes suspects. Leur étude utilise non pas un, mais deux analyseurs syntaxiques du français (FRMG et SXLFG-fr) utilisant le même lexique et la même chaîne de prétraitements, sur plusieurs millions de phrases. En utilisant un algorithme de point fixe, ils calculent le taux de suspicion moyen des mots et bigrammes, permettant d'identifier ceux qui sont vraisemblablement source d'erreurs. Ceux-ci agissent comme révélateurs de problèmes de segmentation, de bogues informatiques divers ou encore d'erreurs dans les entrées du Lefff[3]. Le fait d'avoir utilisé deux analyseurs partageant des ressources importantes leur permet également d'attribuer certaines erreurs à la partie commune du pipeline d'analyse lorsque les résultats sont les mêmes, pour un certain suspect, entre les deux chaînes.

Contrairement à ces chercheurs, cependant, nous considérerons le correcteur comme une « boîte noire » et nos efforts d'amélioration ne conduiront pas à des rectifications internes du correcteur, mais plutôt à la création d'un filtre en post-traitement de ses sorties.

1. `www.synapse-fr.com/`
2. `www.let.rug.nl/vannoord/alp/Alpino/`
3. `alpage.inria.fr/~sagot/lefff.html`

Cette approche s'est également avérée féconde en traduction automatique. En 2007, lors de l'atelier de l'ACL sur la traduction automatique statistique, deux expériences ont été menées indépendamment pour rehausser la qualité des traductions machine, en aval du système de référence. Dugast *et al.* (2007) et Simard *et al.* (2007) ont tous deux eu l'idée de placer un module de *postédition* statistique appliqué sur les sorties de SYSTRAN, un système de traduction commercial à base de règles. Les deux groupes de chercheurs ont observé une amélioration très significative de la qualité des traductions en ajoutant la postédition statistique, sans pour autant avoir dû modifier SYSTRAN. Ces deux dernières études sont proches de notre cas de figure, en particulier parce qu'elles ne prétendent pas connaître le fonctionnement de la chaîne de traitement dont les sorties doivent être rectifiées.

La suite de cet article est organisée comme suit. Nous présentons dans ses grandes lignes le projet *Scorali* en section 2. Nous décrivons en section 3 notre approche à la détection de surdétection et présentons nos résultats en section 4. Une discussion du travail réalisé et de ses perspectives est proposée en section 5.

2. Le projet *Scorali*

Le projet *Scorali* cherche à créer un module de post-traitement au moteur d'analyse et de correction grammaticales Analytix, servant à repérer et atténuer les surdétections dans les sorties du correcteur. Nous appelons surdétections les corrections grammaticales proposées à tort par le correcteur, là où le texte est en réalité irréprochable. Le travail présenté est le fruit d'une réflexion menée initialement chez Druide, et a été réalisé en partenariat avec des chercheurs du RALI.

2.1. *Analytix*

Druide est une société montréalaise connue notamment pour ses outils d'aide à la rédaction, et ce, depuis 1996.

Ces outils font appel à une technologie d'analyse symbolique et de détection appelée Analytix, intégrant une grammaire de dépendance syntaxique à large couverture ainsi que plusieurs ressources linguistiques riches, pour la plupart élaborées et maintenues manuellement. Analytix traite ainsi des phénomènes complexes et un soin particulier a été investi pour assurer sa robustesse face aux déviances grammaticales de tout acabit. Ce module logiciel cherche notamment à détecter les fautes d'orthographe, de syntaxe, de sémantique (par exemple confusion d'homophones, impropriété), de typographie et de style (registre de langue inadéquat, par exemple).

Les détections sont effectuées automatiquement à partir de l'inspection des arbres d'analyse produits. Nous avons travaillé dans cette étude avec la version d'Analytix en vigueur en janvier 2009. Les versions subséquentes du moteur intègrent une partie du travail présenté ici.

2.2. *Cahier des charges*

Bien qu'un assistant de rédaction soit bien plus qu'un correcteur grammatical (Genthial et Courtin, 1992), le cahier des charges du projet *Scorali* porte entièrement sur cet aspect de l'application. Plus précisément, l'objectif du projet consistait à entraîner de manière supervisée des classifieurs à reconnaître les surdétections pour plusieurs types de fautes que traite le correcteur grammatical.

Pour être considéré comme intéressant, un classifieur devait repérer au moins 66 % des surdétections et ne pas éliminer plus de 10 % des détections légitimes. Ces seuils ont été fixés par Druide, à partir de leur expertise dans le domaine, et de leur connaissance du comportement de leur logiciel et des rétroactions des utilisateurs. Au final, c'est l'utilisateur du logiciel qui juge de la qualité de celui-ci après modifications, et c'est un jugement subjectif (Helfrich et Music, 2000). Nous considérons que Druide est experte lorsqu'il s'agit d'estimer les seuils qui influenceront ce jugement.

Le cahier des charges faisait également mention que les classifieurs développés devaient être intégrables facilement dans la chaîne de traitement d'Analytix. C'est donc dire qu'à performances égales ou comparables, nous privilégions toujours les classifieurs rapides et simples d'implémentation.

2.3. *Méthodologie*

Pour chaque type de faute, Druide a préparé un ensemble d'environ 1 000 exemples de bonnes et mauvaises détections produites par Analytix (voir le tableau 1 pour des statistiques détaillées). Ces exemples constituent notre corpus d'entraînement. Chaque détection a été annotée par les linguistes de la société à l'aide d'une interface dédiée. Une annotation indique si une détection est correcte ou pas (surdétection). Les exemples ont été choisis de manière à contenir des textes représentatifs de différents utilisateurs de l'application. Là encore, la constitution de ce corpus est soumise à une nécessaire subjectivité, mais les chercheurs de Druide sont les mieux placés pour connaître les textes qui sont typiquement soumis à leurs outils de correction. En plus des annotations associées à chaque faute, un document en format libre présente pour chaque type de détection une analyse sommaire des problèmes rencontrés ainsi qu'une indication de traits pertinents à leur détection. Le travail de préparation des données a été décisif à la bonne réalisation du projet.

2.4. *Fautes étudiées*

Après plusieurs cycles d'annotation et de développement, un total de quatorze types de détections a été étudié. Nous distinguons trois grandes classes de fautes : des fautes d'accord et de conjugaison, des fautes spécifiques, comme la confusion que/dont (Q/DONT) ou la confusion ou/où (OU/OÙ) ainsi que des fautes de nature typographiques, comme l'omission d'une apostrophe (APOS) ou des fautes de casse

(MAJ). Nous présentons dans la suite chaque type de faute à l'aide d'exemples où le site de la détection est indiqué en **gras souligné**, suivi de la correction marquée *[entre crochets]*. Un astérisque distingue les surdétections des bonnes détections.

ACCORD identifie les fautes d'accord, notamment en genre ou en nombre, entre deux entités linguistiques. L'exemple ACCORD$_1$ contient plusieurs fautes et mérite que nous en offrions une version corrigée : « *Ils veulent un libéralisme vrai, acceptent le marché, mais refusent son truquage…* ».

Pour cet exemple, il est intéressant de noter que le correcteur intégré à Microsoft Office semble procéder différemment. En effet, depuis Office 2007, un module de correction contextuelle cherche à détecter les fautes où un mot est confondu avec un autre (par exemple *fosse *sceptique* pour *fosse septique*, ou **pear of shoes* pour *pair of shoes*, en anglais). Dans notre cas, ce module intervient le premier et signale l'accent manquant sur *marché*. Hirst (2008) décrit ce module et démontre notamment qu'il a un faible rappel, mais une haute précision. Le lecteur intéressé par le correcteur contextuel pourra lire le billet que Thierry Fontenelle, un des concepteurs de ce module, a écrit sur le sujet, dans le blogue du *Microsoft Developer Network*[4].

ACCORD$_1$ Ils veulent un libéralisme VRAI, accepte **le** *[*la]* marche, mais refusent son truquage…

ACCORD$_2$ Je lisais un article qui se trouvait à la **une** *[*un]* (première page) d'un journal.

CONJUG signale les fautes de conjugaison des verbes, notamment les erreurs de nombre, de mode ou de temps. Pour CONJUG$_1$, la forme *inclue*, qui est en fait le subjonctif présent du verbe *inclure* à la troisième personne du singulier a manifestement été confondue avec la forme *incluse* qui est suggérée par le correcteur. Nous soulignons que certaines fautes touchant la flexion du verbe sont signalées par les détections ER/EZ, PP/INF et PP/VC décrites ci-après.

CONJUG$_1$ Il est clair que si l'on parle de mariage, la question de l'adoption est ***inclue** *[incluse]* !

CONJUG$_2$ Pour les majors, l'ennemi n'est plus le concurrent (peut-on encore parler de concurrence quand quatre sociétés se **partagent** *[*partageant]* le monde)…

ER/EZ marque la confusion faite fréquemment entre la forme infinitive d'un verbe du premier groupe et sa forme conjuguée à la deuxième personne du pluriel. Les deux exemples de surdétection ci-dessous illustrent une correction erronée d'une terminaison *-ez* vers une terminaison *-er* (ER/EZ$_1$) et *vice versa* (ER/EZ$_2$). Notons que ce type de détection ne s'intéresse pas à la confusion entre des mots comme *cher* et *chez*, qui

4. blogs.msdn.com/b/correcteurorthographiqueoffice/archive/2009/07/16/un-correcteur-contextuel-fran-ais-dans-office-2010.aspx

ne sont pas des verbes. De plus, même si elle constitue une faute de conjugaison, elle a mérité une détection distincte de CONJUG, vue plus haut.

ER/EZ$_1$	**Regardez** *[*Regarder]* les yeux de Faber (interprété par Luchini), une histoire s'y déroule…
ER/EZ$_2$	Pour Fabrice **Ferrer** *[*Ferrez]* 4 octobre 2007 à 20 : 50 (CEST)

MODE distingue parmi les fautes de conjugaison celles qui sont imputables à une confusion sur le mode du verbe. L'exemple MODE$_1$ illustre une surdétection où la construction *que je suis* appelle correctement l'indicatif et non le subjonctif *suive*, comme le suggère le correcteur.

MODE$_1$	En résumé, et pour te dire Koko que je **suis** *[*suive]* parfaitement en accords avec…
MODE$_2$	Et **vive** *[*vivre]* le sport…

PP/INF signale une autre confusion fréquente entre la forme infinitive d'un verbe du premier groupe et son participe passé.

PP/INF$_1$	En ce sens, des engagements ont été pris, tels que **développer** *[*développés]* des technologies plus propres…
PP/INF$_2$	Elle nous apprend à ne plus ***convoitée** *[convoiter]* le bien d'autrui.

PP/VC marque une confusion entre le participe passé d'un verbe et l'une de ses autres formes.

PP/VC$_1$	Une fois installés, **fini** *[*finirent]* les soucis !
PP/VC$_2$	Par ce jeu de reflet, la forme devient évanescente, la matière **confondue** *[*confondit]*.

PP INV souligne la confusion entre la forme fléchie et la forme invariable d'un participe passé.

PP INV$_1$	Par contre j'ai une préférence pour la mozza **rapée** *[*rapé]*.
PP INV$_2$	Démarrée en 1993, cette série a ***apportée** *[apporté]* une certaine renommée à Paul Grist.

INVAR détecte un changement de forme pour un mot qui est invariable. La correction est toujours le mot invariable, dépouillé de la flexion incorrecte. Comme pour ACCORD$_1$, la surdétection de l'exemple INVAR$_2$ est due à une faute à un autre endroit dans la phrase : *plus part* est mal écrit, et perturbe le correcteur lorsqu'il rectifie le genre de l'article.

INVAR$_1$	Trois Casques bleus du Sri-Lanka avaient été blessés par ***balles** *[balle]* mercredi
INVAR$_2$	deje pas beaucoup de marocain construise leur maison ici c vrai parce qu'il savent ce qui se passe et **la** *[*le]* plus part des maisons detruite sont a des francais

MAJ caractérise les problèmes de casse. Cette détection implique donc souvent les entités nommées ou les symboles.

MAJ$_1$	Tout le monde semble prendre le **pb** *[*Pb]* au sérieux.
MAJ$_2$	1574 : Tycho Brahé entreprend dans la petite île de **Ven** *[*VEN]* (ou Hveen) près de Copenhague au Danemark, Uraniborg la construction du château

APOS annote les mauvais emplois de l'apostrophe, qu'il s'agisse d'une omission ou au contraire de sa présence erronée, par exemple lors d'une élision de mauvais aloi. APOS$_1$ illustre une erreur relevant de la correction orthographique, puisque *decu* n'appartient pas au français. Le correcteur est d'ailleurs conçu pour ces détections d'ordre lexical (voir section 2.1), mais emploie ici une règle de correction malheureuse, là où une simple distance d'édition aurait pu faire mouche. Notons enfin que, pour les détections de ces deux exemples, ni le texte original ni le texte corrigé ne sont acceptables. Pour APOS$_2$, il manque une espace entre *-20* et l'unité de température.

APOS$_1$	Le gouvernement canadien est terriblement ***décu** *[*d'écu]* et avec raison ;
APOS$_2$	Qu'à -20***C** *[*ce]*, même sans neige ou glace, vos pneus n'ont plus la même adhérence ! ? !

ÉLISION signale les élisions non nécessaires ou fautives.

ÉLISION$_1$	**M'bassidjé** *[*Me bassidjé]* François fut un Roi Abé.
ÉLISION$_2$	Ça ***s'** *[se]*peut vraiment.

OU/OÙ signale une confusion entre la conjonction de coordination *ou* et l'adverbe et pronom relatif *où*, deux formes homophones.

OU/OÙ$_1$	Là ***ou** *[où]* ça se gâte, c'est lorsqu'il commence à sauter...
OU/OÙ$_2$	**Où** *[*Ou]* il y a le coq, la poule ne chante pas.

LA/LÀ marque une confusion entre l'article ou pronom *la* avec l'adverbe ou interjection *là*. Il s'agit là encore d'une confusion entre deux formes homophones.

LA/LÀ$_1$	L'objectif est de rouvrir **la** *[*là]* ou les coronaires bouchées ou en train de se boucher ...
LA/LÀ$_2$	***La** *[Là]*, la france n'est pas mal lotie.

Q/DONT signale une confusion fréquente dans le langage parlé entre les deux pronoms *que* et *dont*.

Q/DONT$_1$	Je comprends ce que tu dis, mais pas ce ***que** *[dont]* tu parles.
Q/DONT$_2$	Mais bon dieu **que** *[*dont]* les adultes s'amusent !

On observe que les phrases du corpus d'entraînement appartiennent à des registres extrêmement variés. On retrouve ainsi des phrases de Wikipédia dans un excellent français, aux côtés d'extraits aux déviances typiques de la langue des messages SMS (voir les exemples INVAR$_2$ et MAJ$_1$), où les diacritiques manquent cruellement et où l'orthographe devient presque phonétique. D'autres encore sont des phrases tronquées, télégraphiques ou même des titres.

Il est également manifeste que les fautes détectées sont caractéristiques de clientèles bien différentes. En effet, la plupart des francophones sont susceptibles de commettre une faute ACCORD ici et là, alors que la faute Q/DONT (par exemple : *la personne ⋆que je parle*) est peu probable dans le texte d'un scripteur natif et compétent qui rédige une lettre.

Les caractéristiques principales du corpus annoté sont décrites dans le tableau 1. Chaque type de faute comporte en moyenne 1 251 exemples dont le nombre de surdétections (48 %) est globalement équilibré avec le nombre de bonnes détections. La haute proportion de surdétections n'est bien sûr pas représentative du fonctionnement habituel d'Analytix. Il est même difficile d'imaginer pareil scénario : le correcteur

Type de faute	Nb. d'exemples	Nb. moyen de mots par phrase	% surdétections	Nb. moyen de détections par phrase
ACCORD	1 506	35,5	45 %	3,8
CONJUG	1 325	25,6	34 %	2,7
ER/EZ	1 600	22,0	49 %	2,2
MODE	1 517	29,7	62 %	2,6
PP/INF	1 157	29,8	49 %	3,5
PP/VC	1 155	26,3	43 %	2,3
PP INV	1 119	33,9	45 %	4,4
INVAR	1 825	34,8	22 %	3,9
MAJ	1 012	31,0	50 %	3,7
APOS	1 086	35,3	46 %	6,3
ÉLISION	997	27,9	63 %	3,4
OU/OÙ	1 241	25,8	40 %	3,1
LA/LÀ	1 249	19,3	50 %	2,2
Q/DONT	732	26,5	71 %	2,5
Moyenne	1 251	35,0	48 %	3,3

Tableau 1. *Description du corpus d'entraînement utilisé dans cette étude*

en serait lourdement handicapé. Au contraire, les surdétections sont surreprésentées dans le corpus d'entraînement à dessein, ce qui nous permet d'étudier aussi bien les caractéristiques et les contextes d'occurrence des détections légitimes que ceux des surdétections.

On observe qu'un peu plus de trois détections sont identifiées en moyenne par phrase, ce qui semble corroborer nos observations sur la mauvaise qualité de certains des textes dans le corpus d'entraînement. Les occurrences des fautes (et leurs détections) ne sont donc pas toutes indépendantes : la qualité d'une détection à un site donné peut être influencée par une ou plusieurs fautes ailleurs dans la phrase. ACCORD_1 est un exemple d'une telle phrase où la surdétection illustrée est la résultante de l'absence (non identifiée) d'un accent sur le nom *marché*.

3. Approche

Comme nous l'avons mentionné, le projet *Scorali* consistait à entraîner de manière supervisée un classifieur par type de faute ; l'application d'un tel classifieur se faisant en aval de la chaîne de détection d'Analytix. Une partie importante du projet a donc été dédiée à l'ingénierie et à l'extraction de traits (*features*) représentant chaque exemple. Nous la décrivons en section 3.1. La sélection et l'entraînement des classifieurs ont également été le fruit de nombreuses réflexions qui sont résumées en section 3.2.

En pratique, nous avons divisé notre étude en deux parties séparées temporellement. La première, *Scorali–A*, visait à respecter en tout point le cahier des charges fixé par Druide. C'est à l'issue de ce sous-projet qu'ont été livrés à Druide les classifieurs. Dans la seconde partie, nommée *Scorali–B*, nous nous autorisions l'usage de traits plus lourds à calculer ainsi que des classifieurs plus complexes.

3.1. *Ingénierie des traits pour l'encodage des exemples*

Nous extrayons environ 1 200 traits décrivant la détection, le mot au site de la détection et les mots dans son voisinage (voir les sections 3.1.1.1 et 3.1.1.2). À ceux-là, nous ajoutons environ 50 traits issus de la désambiguïsation sémantique (voir la section 3.1.1.3). Pour *Scorali–B*, nous explorons également environ 50 traits dérivés de modèles de langue (section 3.1.2). Dans la mesure où nous ne nous intéressons qu'à un seul site dans la phrase, ces traits sont comparativement nombreux, en grande partie grâce à la richesse des informations produites par Analytix. Il n'en demeure pas moins que le nombre de traits utilisés dans cette étude peut sembler modeste lorsque comparé à ceux dérivés d'une phrase complète. Par exemple, dans le projet Alpino, Van Noord (2007) décrit une approche de reclassement (*reranking*) exploitant 42 000 traits.

3.1.1. *Traits partagés par Scorali–A et Scorali–B*

3.1.1.1. Traits communs à chaque détection

L'examen des annotations de la section précédente, ainsi que celui de centaines d'autres exemples, indique que les mots dans le voisinage des détections permettent souvent de guider la classification des détections comme bonne ou mauvaise. Par exemple, dans la confusion OU/OÙ, il est manifeste que si le mot *là* précède la confusion, c'est presque toujours le mot *où* qui est attendu ensuite. Il existe naturellement des contre-exemples, ainsi *J'irai là ou ailleurs*. De même, pour la confusion Q/DONT, un verbe transitif direct appelle le mot *que*, alors qu'un verbe transitif indirect appelle le mot *dont*. Ce voisinage peut être simplement constitué des mots avant et après la détection ou, puisque le matériel d'entraînement inclut l'arbre d'analyse syntaxique produit par l'analyseur, des mots parents ou enfants dans cet arbre d'analyse.

Certains traits du mot qui fait l'objet de la détection importent également. On observe notamment qu'il est peu indiqué de corriger un mot ayant une capitale lorsqu'il suit un autre mot avec une capitale : on est vraisemblablement en présence d'une entité nommée.

La liste suivante décrit les principaux traits utilisés qui sont communs à l'ensemble des détections.

– Les traits du mot au site de la détection : soit, tout d'abord, sa casse, sa longueur en lettres, sa catégorie morphosyntaxique, sa position dans la phrase, puis son genre et son nombre (lorsque c'est applicable) ou son temps et sa personne (dans le cas d'un verbe), et, enfin, certaines caractéristiques de l'analyseur de Druide, par exemple « verbe se terminant en *-yer* pouvant être confondu avec un nom ». Des traits de préfixe ont été utilisés également.

– Les mêmes traits que ceux du point précédent, mais cette fois pour les mots précédant et suivant le mot détecté, ainsi que pour son mot parent dans l'arbre d'analyse syntaxique.

– Les traits de la détection elle-même : la certitude avec laquelle le correcteur la propose, les traits de la correction proposée pour remplacer le mot fautif.

– Les traits de la phrase dans laquelle on trouve la détection : sa longueur en mots, le nombre de relations détectées, le nombre de mots inconnus[5], le nombre total de détections.

– La nature des relations syntaxiques impliquant le mot détecté ; nommément, les relations qu'il entretient avec son parent et ses enfants éventuels dans l'arbre d'analyse. On retrouve les relations habituelles, telles que « complément du nom » ou d'autres, plus spécifiques au correcteur, comme « tel que », qu'on utiliserait dans la phrase de l'exemple PP/INF_1.

5. Notons que la plupart des traits qui sont des dénombrements sont doubles : il y a un trait qui est le compte et un autre qui est le compte normalisé par la longueur de la phrase.

Notons que nous n'avons pas combiné manuellement les traits dont nous avons fait l'extraction afin d'en créer de nouveaux. Notre objectif d'ingénierie des traits était d'extraire le plus d'information possible des sources relativement riches à notre disposition, et d'offrir la possibilité aux classifieurs utilisés d'y découvrir des règles ou combinaisons, ce qui constitue une de leurs grandes forces (ce n'est cependant pas assuré).

3.1.1.2. Traits spécifiques à certaines détections

À ces traits génériques s'ajoutent quelques-uns qui reflètent nos intuitions quant aux particularités de certaines détections, à la suite de l'examen attentif du corpus d'apprentissage. Par exemple, pour la détection PP/INF, nous tenons compte spécifiquement de la présence des verbes *pouvoir* et *faire* avant le site de la détection. L'étude du corpus d'apprentissage révélait en effet que la présence de ces verbes permettait souvent de confirmer ou d'infirmer la détection faite : ces verbes sont typiquement (mais pas toujours) suivis de l'infinitif. De même que la présence du pronom *vous* avant le site de la détection nous a semblé un indice pertinent. En effet, lorsqu'il existe une hésitation entre le participe passé et l'infinitif d'un verbe à un site précédé du mot *vous*, c'est presque toujours la forme infinitive qui l'emporte, comme dans *Il faut vous réconcilier.* Une exception notable est l'interrogation, qui autorise *Vous êtes-vous réconcilié ?*. Puisque les traits utilisés couvrent déjà l'information d'interrogation, nous n'avons pas combiné manuellement ces traits.

3.1.1.3. Traits de désambiguïsation sémantique

Pour la confusion Q/DONT, il nous a paru intéressant de reformuler une partie de notre problème de classification comme une application de la désambiguïsation sémantique (*word sense disambiguation* ou *WSD*). En effet, on peut voir le site de la détection comme un mot fictif de sens indéterminé et *que* et *dont* comme ses étiquettes sémantiques possibles. La désambiguïsation permet alors de savoir laquelle doit être insérée à l'endroit de l'ambiguïté. Nous tentons donc de désambiguïser en ajoutant des traits qui explorent le contexte de la confusion et en le comparant aux contextes dans lesquels on retrouve habituellement *que* et *dont*, dans des textes (*a priori*) irréprochables. Cette reformulation du problème nous autorise dès lors à emprunter quelques outils à la littérature du domaine de la désambiguïsation sémantique.

Nous utilisons notamment une adaptation de la technique proposée par Yarowsky (1995) et décrite dans (Manning et Schütze, 1999). Ainsi, pour un mot en position i détecté par le correcteur comme une faute de type Q/DONT, nous ajoutons à ceux précédemment décrits les traits suivants où $\mathcal{C}$ dénote le contexte (différents contextes allant d'un mot à la phrase entière privée du mot au site de la détection sont considérés) :

– les probabilités $P(s|\mathcal{C})$ pour $s \in \{$*que*,*dont*$\}$;
– les ratios associés $r_{\mathcal{C}} = P(que|\mathcal{C})/P(dont|\mathcal{C})$;

– des valeurs indicatrices $v_s \equiv r_{\mathcal{C}} \geq s$, pour une vingtaine de seuils entre $1/50$ et 50[6].

Pour une partie des traits, nous définissons $\mathcal{C}$ simplement comme le mot qui précède la détection. Un site de détection en position i concernant le mot w_i dans une phrase de n mots $w_0 \ldots w_i \ldots w_n$ explorera donc les probabilités $P(s|w_{i-1})$ pour les étiquettes s. Pour d'autres variantes de ces traits, nous considérons $\mathcal{C}$ comme les mots de fenêtres qui peuvent être $w_{i-5} \ldots w_{i-1}$ ou encore $w_{i+1} \ldots w_{i+5}$. Nous avons utilisé des fenêtres plus grandes encore : $w_0 \ldots w_{i-1}$ et $w_{i+1} \ldots w_n$. Deux mots spéciaux BOS et EOS ont été ajoutés respectivement au début et à la fin de la phrase pour dénoter ces deux événements linguistiquement importants. Nous utilisons de plus une variante de tous ces contextes à plusieurs mots, en ignorant les mots grammaticaux (*stopwords*) qui pourraient s'y trouver et qui sont des indicateurs pauvres du contexte, car ubiquistes.

Notons que dans le cas où le contexte $\mathcal{C}$ contient plusieurs mots, $P(s|\mathcal{C})$ est estimé par : $argmax_{w \in \mathcal{C}} P(s|w)$, conformément à l'approche de (Yarowsky, 1995) qui se fie à une seule collocation pour trancher entre plusieurs étiquettes sémantiques. Enfin, nous utilisons une variante de ces contextes à plusieurs mots qui ignore les mots grammaticaux qui pourraient s'y trouver.

Nous avons précalculé $p(s|w)$ en nous fondant sur la version française du Hansard canadien[7]. Pour chacune des étiquettes *que* et *dont*, nous avons extrait aléatoirement un corpus de 40 000 phrases les contenant. La probabilité $p(s|w)$ est estimée par fréquence relative. Ces modèles statistiques calculés à l'avance ont le mérite d'être relativement compacts, une qualité essentielle selon le cahier des charges. En effet, après un peu d'ingénierie (élimination de mots peu fréquents, encodage des mots sur deux octets, etc.) un modèle pour une définition de contexte donnée (par exemple un mot avant et après le site de la détection) prend environ 30 ko. Naturellement, si nous avions eu plus d'étiquettes « sémantiques », l'approche aurait été d'autant plus lourde. Cette stratégie n'est donc pas applicable à des confusions impliquant un grand nombre de formes, par exemple toutes les flexions d'un verbe donné.

Nous n'avons pas étendu l'approche aux confusions LA/LÀ et OU/OÙ à cause des résultats décevants (voir section 4.3) de cette approche coûteuse sur Q/DONT.

3.1.2. *Traits spécifiques à Scorali–B*

On l'a vu à la section 1, certains correcteurs statistiques s'appuient exclusivement sur des phénomènes de surface : ils vérifient dans quelle mesure une séquence de mots est plausible dans une langue donnée. Si elle est improbable, alors elle pourrait être le site d'une faute. La séquence *ce que tu parles* est par exemple certainement moins fréquente que *ce dont tu parles* dans un corpus de qualité.

6. Une grande valeur de s indique une grande conviction que *que* devrait être choisi.
7. Ces débats sont disponibles à `www.parl.gc.ca/common/chamber.asp`.

Nous utilisons un modèle de langue statistique pour capturer cette intuition. Dans notre cas, nous avons entraîné à l'aide de la boîte à outils SRILM (Stolcke, 2002) un modèle trigramme (de type Kneser-Ney) sur 3,3 M de phrases françaises d'une tranche du Hansard canadien, qui concerne les débats effectués entre 1986 et 1994.

Soit une détection en position i dans la phrase, w_i le mot s'y trouvant et w'_i la correction proposée par le correcteur. Il nous est possible de créer les traits correspondant aux probabilités $p(w_i|w_{i-2}w_{i-1})$, $p(w'_i|w_{i-2}w_{i-1})$, $p(w_i|w_{i+2}w_{i+1})$ et $p(w'_i|w_{i+2}w_{i+1})$, ce qui revient à calculer les plausibilités respectives de la séquence qui fait l'objet de la détection et de celle qui est proposée par le correcteur.

Nous soulignons que notre modèle de langue requiert 500 Mo sur disque et environ la même quantité en mémoire vive. Son chargement sur un ordinateur cadencé à 3 GHz nécessite de l'ordre de 20 secondes. S'il existe des solutions pour réduire le stockage et le temps d'accès aux probabilités (Pauls et Klein, 2011), l'usage de tels modèles et de leurs traits dérivés n'était toutefois pas envisageable en l'état dans un correcteur destiné au grand public.

3.2. *Classifieurs étudiés*

Afin d'étudier et de créer les classifieurs requis, nous avons fait appel au logiciel libre Weka (Hall *et al.*, 2009)[8]. Il permet d'explorer de nombreuses familles de classifieurs et possède des fonctions intéressantes, tels la visualisation et le prétraitement des données d'entraînement. Il est également possible d'appeler un classifieur à partir de la ligne de commande pour du traitement par lots (*batch processing*), ce qui s'est avéré indispensable dans notre cas.

Weka permet de prototyper une cinquantaine de classifieurs, regroupés en huit familles, dont les classifieurs bayésiens, les arbres de décision, les perceptrons, les séparateurs à vaste marge (SVM) et les métaclassifieurs. Ces derniers combinent d'autres classifieurs, par exemple en les faisant voter. Chacun de ces classifieurs est typiquement contrôlé par une vingtaine d'hyperparamètres discrets ou continus.

Au-delà de la sélection et du paramétrage des classifieurs, Weka permet également diverses stratégies de prétraitement des données. Nous avons d'abord filtré les traits, pour enlever ceux qui ne variaient pas assez, donc qui sont non discriminants. De plus, pour chaque détection, nous avons essayé différentes stratégies de filtrage des traits (incluant une variante les gardant tous), réduisant dans certains cas les 1 200 traits d'origine à une douzaine. Ceci simplifie naturellement l'entraînement et le test des classifieurs, mais aussi leur implémentation subséquente.

Nous avons également fait varier les coûts que Weka attribue aux erreurs de type faux positifs et faux négatifs (voir le tableau 3) lors de l'entraînement. Nous avons expérimenté avec des coûts pour les deux types d'erreurs (I et II) variant de 1 (le coût par

8. `www.cs.waikato.ac.nz/ml/weka/`

défaut) à 30. Les matrices de coût ne peuvent être couplées à tous les classifieurs, aussi avons-nous pris soin de choisir des classifieurs compatibles aussi bien pour *Scorali–A* que *Scorali–B*.

Il est à noter que le choix du classifieur, sa paramétrisation, les stratégies de prétraitement et le type de critère utilisé pour l'entraînement engendrent une explosion combinatoire du nombre de classifieurs possibles. Nous privilégions l'approche proposée par Hoos (2012) qui encourage notamment les chercheurs à entretenir autant d'algorithmes concurrents que possible lorsqu'ils sont confrontés à un problème de notre type. Conformément à cette approche, écarter d'emblée des pistes de solution sur la base de critères trop souvent subjectifs (notamment la taille de l'espace de recherche d'hyperparamètres) pourrait nous coûter une solution utile. Nous avons donc tenté d'explorer des classifieurs de familles aussi diverses que possible, en termes de capacité des modèles produits, de taille de l'espace de configuration, de facilité d'interprétation, tout en tenant compte de leur facilité d'implémentation, de leur temps d'entraînement et de classification.

3.2.1. *Scorali–A*

Nous avons dans un premier temps limité notre exploration aux classifieurs rapides lors de l'entraînement et la classification, notamment pour satisfaire le cahier de charges. De plus, nous avons privilégié les classifieurs conceptuellement « simples », de façon à faciliter leur mise au point puis leur implémentation au cœur du correcteur grammatical. Ceci nous a menés à concentrer nos efforts sur des classifieurs symboliques succinctement décrits au tableau 2[9]. Ces classifieurs s'appuient sur des réponses à une suite de questions fermées et sont bien adaptés à notre tâche. Ils sont en effet extrêmement rapides à tester (et à entraîner) et sont de plus interprétables par un expert (Duda *et al.*, 2001), ce qui rend leur maintenance plus simple.

3.2.2. *Scorali–B*

Dans un deuxième temps, nous avons étudié différentes variantes d'un classifieur SVM. Un SVM est un classifieur évolué qui fonctionne en représentant les données dans un espace vectoriel en haute dimension, puis en tentant de trouver un hyperplan qui sépare le mieux possible les deux classes dans cet espace. Ce classifieur s'illustre dans la littérature du traitement des langues naturelles, notamment par sa stabilité face aux disparités entre corpus d'entraînement et de test. Il est également directement applicable aux problèmes de séparation entre deux classes, ce qui est justement notre cas. Ces avantages se gagnent malheureusement au prix d'une interprétabilité réduite et de difficultés d'exportation accrues des classifieurs vers un code industriel.

Le classifieur SVM utilisé est entraîné en utilisant l'algorithme d'optimisation séquentielle minimale décrite par Platt (1998). L'implémentation de Weka gère les traits manquants et transforme également les attributs discrets en attributs aux valeurs continues. Le SVM de base peut s'avérer limité en cela qu'il cherche un hyperplan sépara-

9. Le lecteur est invité à consulter la documentation de Weka pour plus de détails à leur sujet.

Nom du classifieur Weka	Description
`rules.ConjunctiveRule`	Apprentissage de règles conjonctives (ET logiques)
`rules.DecisionTable`	Table de décisions simples apprises avec l'algorithme IDTM (Kohavi, 1995), parfois surpassant les arbres de décision, même avec traits continus
`rules.JRip`	Apprentissage de règles prédictives de type RIPPER (Cohen, 1995)
`trees.ADTree`	Arbre de décision alternatif, implémenté pour deux classes, soit notre cas
`trees.DecisionStump`	Arbre de décision de profondeur 1, extrêmement simple
`trees.J48`	Algorithme correspondant à l'arbre de décision C4.5, testé avec et sans élagage
`trees.J48graft`	Algorithme J48, avec greffe de nœuds supplémentaires, réduisant l'erreur de classification, le temps de calcul et la complexité de l'arbre produit

Tableau 2. *Description succincte des classifieurs explorés pour le projet Scorali–A*

teur dans l'espace linéaire des traits utilisés. Pour augmenter la capacité de cet algorithme, nous avons eu recours à l'« astuce du noyau », ce qui est la pratique habituelle. Nous avons ainsi testé plusieurs noyaux de classifieurs, dont les noyaux polynomiaux de degrés 1, 2 et 3 et le noyau de fonction à base radiale (RBF).

4. Résultats

Étant donné le nombre restreint d'instances dans le corpus d'entraînement pour une faute donnée, de même que l'absence de corpus de test aisément accessible, nous avons recouru à la validation croisée en dix blocs (avec blocs de tailles égales), avec répartition aléatoire des instances dans chaque bloc.

En tenant compte des prétraitements différents des traits, des classifieurs variés (et de leurs réglages respectifs), ainsi que des matrices de coûts, chaque type de faute a demandé l'entraînement et l'évaluation de 2 300 classifieurs pour *Scorali–A*. Pour *Scorali–B*, nous avons exploré les mêmes 2 300 classifieurs, cette fois-ci avec les traits additionnels issus des modèles de langue décrits à la section 3.1.2. Les classifieurs de type SVM, quant à eux, sont au nombre de 600. Nous les avons testés dans un premier temps sur les mêmes traits que *Scorali–A*, et dans un deuxième temps sur les traits de *Scorali–A* additionnés des traits des modèles de langue.

Les expériences ont été réalisées sur une grappe de seize machines Linux 3 GHz à deux processeurs chacune, en un temps moyen de cing jours par faute. Il n'est pas

		Sortie du classifieur	
		Surdétection	Détection légitime
Condition réelle	Surdétection	Vrai positif	Faux négatif (erreur de type II)
	Détection légitime	Faux positif (erreur de type I)	Vrai négatif

Tableau 3. *Matrice de confusion appliquée à notre étude des surdétections*

possible pour des raisons de place de présenter ici tous les résultats obtenus[10]. Dans cette section, nous détaillons les résultats de deux détections représentatives, l'une (PP/VC) pour laquelle nous avons identifié un classifieur compatible avec le cahier des charges, l'autre (Q/DONT) pour laquelle nous avons échoué.

4.1. *Les sorties du classifieur : terminologie*

Dans ce travail, les classifieurs tranchent entre deux classes : surdétection ou détection légitime (abusivement « mauvaise » et « bonne » détection, respectivement). La classe que nous cherchons à identifier est la surdétection. Le tableau 3 récapitule les sorties possibles d'un classifieur pour le cas qui nous intéresse et clarifie les notions de vrai et faux positifs.

Le taux de faux positifs (FP) est le pourcentage de détections légitimes erronément classées comme surdétections. Un haut taux de FP signifie un moins bon rappel de l'identification des fautes commises par le scripteur. On a vu à la section 2.2 que le cahier des charges bornait supérieurement cette statistique à 10 %.

Le taux de faux négatifs (FN) est le pourcentage de surdétections erronément classées comme détections légitimes. Un haut taux de FN expose l'utilisateur à beaucoup de surdétections. Le taux de vrais positifs (VP) est relié au précédent ; il vaut $\text{VP} = 1 - \text{FN}$. Le cahier des charges de la section 2.2 stipulait une borne minimale $\text{VP} \geq 66\ \%$.

4.2. *Étude de la faute* PP/VC

La figure 1 montre les performances des 2 006 classifieurs entraînés dans la sous-tâche *Scorali–A* pour lesquels $\text{FP} < 20$ (286 classifieurs ne satisfont pas ce filtre). L'axe des abscisses représente le pourcentage de détections légitimes éliminées par erreur par les classifieurs (faux positifs) et l'axe des ordonnées, le pourcentage de surdétections correctement identifiées (vrais positifs).

10. Le lecteur intéressé peut consulter (Gotti, 2011) pour plus d'information.

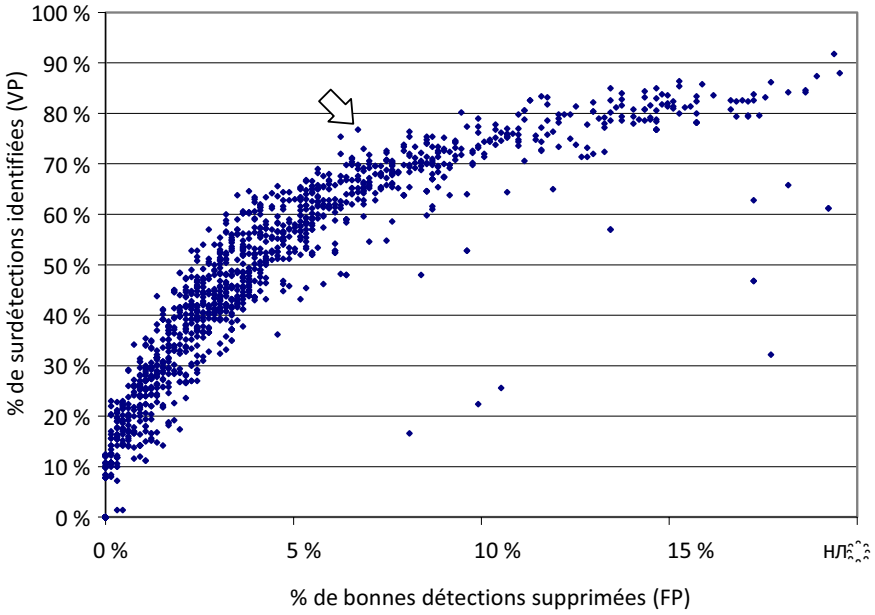

Figure 1. *Sous-ensemble des classifieurs testés pour la détection* PP/VC *dont le taux de faux positifs est inférieur à 20 %. Chaque point représente les performances d'un classifieur. La flèche indique le classifieur recommandé, en conformité avec le cahier des charges présenté à la section 2.2.*

Il est remarquable de constater que le nuage de points est relativement compact, traçant une bande qui marque les compromis possibles avec ces classifieurs entraînés sur les données à notre disposition. Le choix du classifieur est fait manuellement à partir de cette figure, en tenant compte des contraintes énumérées dans le cahier des charges (section 2.2). Dans le cas qui nous préoccupe, le classifieur recommandé à Druide est celui marqué d'une flèche dans la figure 1. Il s'agit d'un arbre de décision C4.5 avec greffes, élagué, permettant l'identification de 77 % des surdétections, au prix de la perte de 8 % des bonnes détections. L'arbre de décision classifie 86 % des instances correctement, avec un accord substantiel $\kappa = 0,76$ entre les résultats des dix blocs de l'évaluation. D'autres classifieurs aux performances similaires ont été écartés, car ils étaient trop complexes à implémenter ou utilisaient un trop grand nombre de traits.

La figure 2 montre un extrait des règles induites par le classifieur sélectionné. L'arbre complet comprend vingt-cinq feuilles, qui correspondent chacune à une classe « bonne détection » ou « surdétection ». Un des branchements fait référence au trait noté φ qui s'applique aux verbes dont la forme au passé simple, à la troisième personne du singulier, ne peut être confondue avec le participe passé. Les paires de nombres

si le nombre de détections au site étudié ≤ 1 **alors**
  si trait du mot du site est φ **alors**
    surdétection (55,52/8,93)
  sinon
    si catégorie du mot du site est participe passé **alors**
      bonne détection (472,85/7,02)
    sinon
      ...
sinon
  si catégorie du mot du site est `participe passé` **alors**
    bonne détection (2,55/0)
  sinon
    si catégorie du mot précédent est `pronom personnel sujet` **alors**
      bonne détection (2,55/0)
    sinon
      surdétection (82,96/2,55)

Figure 2. *Extrait de l'arbre de décision produit pour la détection* PP/VC

(x/y) à la suite de chaque étiquette de classe sont des mesures de pureté ; une paire (x/y) signifie que x exemples correspondent à ce cas de figure et sont correctement classés, et que y exemples correspondent à ce cas de figure, mais ne sont pas correctement classés[11].

On lit par exemple dans la figure 2 que, lorsqu'il n'y a qu'une seule détection impliquant le mot au site de la détection, et que ce mot n'a pas le trait φ et qu'il est un participe passé, alors la détection est légitime, c'est-à-dire que le correcteur est presque toujours (472,85/7,02) en droit de signaler à l'utilisateur qu'il aurait dû utiliser un autre temps de verbe.

4.3. *Étude de la faute* Q/DONT

La détection Q/DONT mérite que l'on s'y attarde, puisqu'elle a donné de mauvais résultats malgré les efforts investis pour ramener le problème de classification à un problème de désambiguïsation sémantique. La série de points *Scorali–A* de la figure 3 montre que nous n'avons pu trouver aucun classifieur respectant le cahier des charges, le meilleur compromis étant un classifieur pour lequel FP $= 9\ \%$ et VP $= 57\ \%$. On constate cependant graphiquement que les traits de désambiguïsation sémantique (voir la section 3.1.1.3) améliorent globalement les résultats.

11. Ce sont des nombres réels, car la matrice de coût vient multiplier ces nombres.

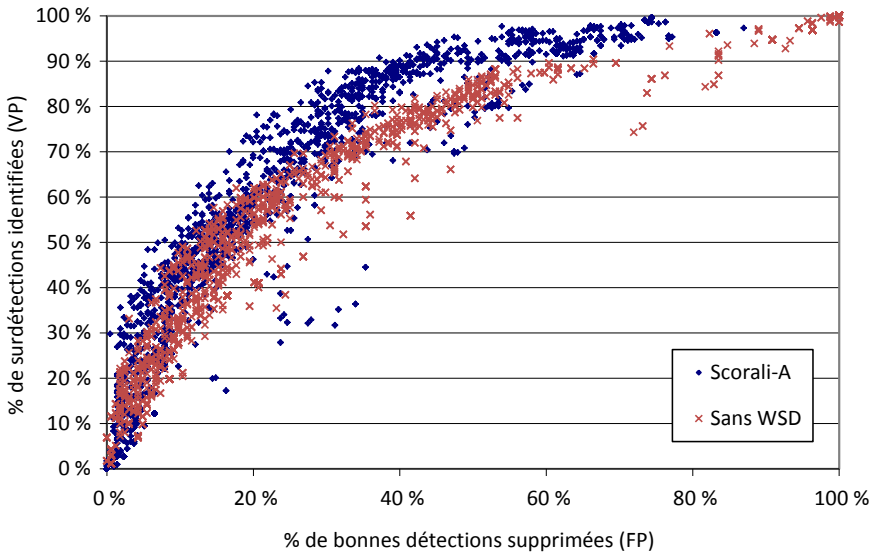

Figure 3. *Classifieurs entraînés pour la faute* Q/DONT. *Aucun d'entre eux ne respecte les contraintes établies dans le cahier des charges fourni (section 2.2), donc aucune recommandation n'est faite pour cette faute.*

4.4. *Vue d'ensemble pour toutes les erreurs*

Nous avons effectué le travail décrit à la section précédente pour les quatorze types de détections fournis par Druide. Les résultats sont présentés au tableau 4. On constate d'abord que nous avons réussi à créer des classifieurs satisfaisants pour neuf des quatorze détections de *Scorali*. Ce sont tous des arbres de décision : à performances égales ou presque, nous sélectionnions un arbre de décision pour uniformiser la tâche d'implémentation. En examinant les arbres, il n'y a pas vraiment de traits ou de famille de traits qui se détachent des autres quant à leur utilité pour un arbre de décision.

Il est difficile d'expliquer pourquoi certaines détections se sont bien prêtées à la classification, et d'autres pas. Malgré nos efforts, Q/DONT n'a pas conduit à un bon classifieur, alors que la confusion ÉLISION, pour laquelle il semblait à première vue bien difficile d'induire des règles (voir les exemples), s'est révélée très facile à traiter.

Naturellement, l'induction de règles de classification est facilitée si Analytix surcorrige une certaine construction de façon systématique. C'est le cas par exemple avec la détection LA/LÀ, où l'examen de l'arbre de décision recommandé montre que 20 % des surdétections surviennent lorsque le mot déterminé par *la* est un complément du nom. Notons d'ailleurs que l'exercice qui consiste à examiner ainsi le classifieur produit a un effet secondaire bien désirable : il permet une analyse de certaines erreurs du moteur d'analyse. En l'occurrence, il serait peut-être souhaitable de vérifier les règles

qui entraînent le rattachement d'un nom à son complément dans le moteur. On n'aurait plus ainsi à corriger ces surdétections en aval, mais à la source même. Toutes les détections ne se prêtent pas à ce genre d'analyse, cependant.

Erreur	% FP	% VP	%acc	Erreur	% FP	% VP	%acc
ÉLISION⋆	7 %	87 %	89 %	PP/INF⋆	7 %	68 %	81 %
LA/LÀ⋆	9 %	84 %	88 %	CONJUG⋆	8 %	66 %	83 %
PP INV⋆	6 %	80 %	91 %	ER/EZ⋆	9 %	65 %	78 %
PP/VC	8 %	77 %	86 %	Q/DONT	9 %	57 %	67 %
APOS	6 %	73 %	84 %	OU/OÙ⋆	9 %	49 %	74 %
INVAR⋆	8 %	71 %	83 %	ACCORD	9 %	40 %	68 %
MAJ	7 %	71 %	82 %	MODE	9 %	27 %	51 %

Tableau 4. *Résultats complets de l'expérience Scorali–A. Un astérisque marque les classifieurs ajoutés aux versions subséquentes du correcteur. %acc désigne le pourcentage de classifications correctes. Les cellules ombrées correspondent aux classifieurs ne vérifiant pas le cahier des charges.*

On peut également avancer que certaines surdétections sont extrêmement difficiles à détecter, même pour un être humain. Les exemples des détections MODE et ACCORD sont éloquents. Pour l'exemple « accepte **le** *[*la]* marche », il faut vraiment comprendre le contexte pour deviner que l'auteur voulait écrire *marché* et non pas *marche*, ce qui explique la surcorrection. Il y a fort à parier que les classifieurs ont de la difficulté avec ce genre de cas, *a fortiori* lorsque ces détections peuvent se faire dans des contextes très différents, et pour des raisons différentes. C'est le cas de ACCORD, qui peut être une faute d'accord en genre ou en nombre. Il faudrait sans nul doute poursuivre les recherches ici en précisant des sous-classes de détections : en genre et en nombre.

4.5. *Validation et intégration*

Druide s'est chargée de valider les classifieurs produits par *Scorali*. Les arbres sérialisés par Weka ont été automatiquement convertis en code C++ par un programme spécialement conçu. Certains arbres de décision trop profonds ont outrepassé les limites du compilateur, et ont dû être remplacés par des candidats élagués aux performances presque identiques.

Les onze meilleurs classifieurs du projet *Scorali–A* ont été validés sur un corpus distinct du corpus d'entraînement fourni, une pratique recommandée dans l'industrie (voir par exemple (Helfrich et Music, 2000)). Ces tests ont révélé que trois classifieurs[12] dégradaient les performances du correcteur au point où il a fallu les exclure. Ces classifieurs n'étaient pas nécessairement les moins bons durant l'entraînement en

12. Ce sont les classifieurs APOS, MAJ et PP/VC.

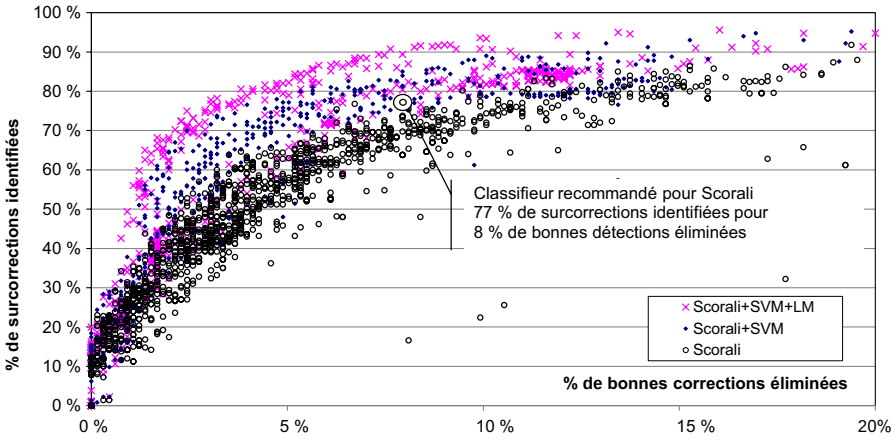

Figure 4. *Sous-ensemble des classifieurs testés pour la détection* PP/VC *dont le taux de faux positifs est inférieur à 20 %. Voir la section 4.6.*

laboratoire. Les huit classifieurs survivants ont été intégrés aux versions subséquentes d'Analytix, et sont désormais utilisés quotidiennement par des milliers de gens.

4.6. *Expériences étendues*

Le projet *Scorali–B* prolonge *Scorali–A*, que l'on considère ici comme ligne de base (*baseline*). Il offre trois nouveaux ensembles de classifieurs : +LM correspond à *Scorali–A* enrichi de traits issus des modèles de langue, +SVM utilise les mêmes traits que *Scorali–A*, mais en utilisant des classifieurs SVM et +LM+SVM indique des SVM entraînés sur les traits de +LM.

La figure 4 montre les résultats obtenus par la ligne de base (*Scorali–A*) ainsi que ceux de +SVM et de +LM+SVM. On tait les résultats de +LM pour alléger la figure. À nouveau, on concentre notre attention sur la région FP $<$ 20 %. On y constate clairement que les tendances observées pour *Scorali–A* ne se démentent pas pour cette détection : les classifieurs sont groupés en trois nuages relativement distincts, en forme de courbe ROC, à quelques points singuliers près.

Une observation centrale est que les classifieurs SVM sont en mesure d'améliorer les résultats de *Scorali–A*. Les classifieurs du nuage +SVM flottent à environ 5 % au-dessus du nuage de *Scorali–A*. L'ajout de traits dérivés des modèles de langue (+LM+SVM) augmente encore le nombre de vrais positifs, pour un gain additionnel de quelque 5 %. Il est bon de noter que ces améliorations sont observées pour toutes les détections, peu importe les résultats qu'elles ont donnés dans *Scorali–A*.

Dans le cas de la faute PP/VC, le choix du meilleur classifieur au sein de chaque population amène aux performances du tableau 5. Même si le processus manuel de sélection est assez subjectif, il apparaît que le passage de la configuration de la ligne de base vers un classifieur +LM+SVM pourrait améliorer considérablement le meilleur classifieur de *Scorali–A*, au prix de complexités et lourdeurs additionnelles dans le transfert du code et de sa nouvelle implémentation dans le produit fini.

Configuration	FP	VP
base	8 %	77 %
+LM	7 %	77 %
+SVM	6 %	81 %
+LM+SVM	6 %	87 %

Tableau 5. *Description des meilleurs classifieurs de Scorali–A et Scorali–B pour la détection* PP/VC

5. Discussion et perspectives

Le projet *Scorali* a permis d'élaborer et d'intégrer à Analytix huit des quatorze classifieurs permettant la détection de surdétections, en aval du moteur de correction. Selon le tableau 4, ces classifieurs épargnent aux utilisateurs 71 % des surdétections au prix de la perte d'environ 8 % des bonnes détections. Cet exercice fructueux de portage de technologies du laboratoire vers un produit commercial a nécessité l'exploration de milliers de stratégies de détection différentes, ainsi qu'un délicat équilibre entre performances et contraintes techniques. Dans un second volet nous avons analysé les performances de classifieurs de type SVM et nous avons ajouté des traits calculés à partir de modèles de langue. Les gains obtenus sont significatifs pour les quatorze types de détections.

Pour le moment, certaines détections semblent ne pas se prêter à l'approche proposée, peut-être parce que la détection se fait dans des contextes trop variés, ce qui défie l'induction de règles. Il se peut également que nous soyons en butte à l'importante variabilité des niveaux de langue des exemples d'entraînement, mais, après tout, c'est là une contrainte nécessaire qui reflète la diversité des textes réels. Le fait que nous avons traité chaque faute de manière indépendante est aussi un point perfectible de notre étude : on observe effectivement un phénomène de cascade, où, par exemple, un diacritique manquant sur un mot confond le correcteur suffisamment pour le faire commettre une surdétection.

Les pistes de recherche que ce travail suggère sont nombreuses ; nous en présentons plusieurs dans la suite. Les trois premières sont des propositions d'amélioration de notre approche, les trois suivantes offrent des perspectives plus ambitieuses et dépassent le projet décrit dans cet article.

Premièrement, les traits que nous avons extraits afin d'alimenter nos classifieurs sont perfectibles. La lexicalisation des contextes (et, partant, leur explosion combinatoire) est une limite bien connue du domaine du traitement des langues naturelles. Comme le soulignent Golding et Roth (1999), deux propriétés caractéristiques de ce genre de problèmes sont leur très grande dimensionnalité et le fait que les classes cibles se réfèrent à un sous-ensemble restreint des traits dans cet espace. Les auteurs proposent de recourir aux classifieurs de type *winnow*, qui sont des perceptrons rapides à entraîner et qui savent manipuler un grand nombre de traits inutiles. C'est une approche que nous avons tentée au tout début du projet *Scorali*, au moment de notre exploration de Weka, sans obtenir de résultats encourageants. Elle mériterait d'être reconsidérée.

Il est également certain qu'il est toujours possible d'ajouter des traits : là où l'on explorait le mot qui précède et qui suit, on peut examiner les mots dans une fenêtre de deux mots avant et après le site de la détection, par exemple. On peut tenter de réduire la dimensionnalité des modèles de langue en construisant des modèles non pas entraînés sur les mots, mais sur les catégories morphosyntaxiques. Nous avons jugé que cet effort d'ingénierie des traits était acceptable pour ce travail lorsque nous en sommes arrivés aux quelque 1 300 traits que nous proposons ici, qui, à notre avis, rendaient accessibles aux classifieurs l'essentiel des informations fournies par le correcteur ainsi que des statistiques riches. Puisque les possibilités de traits sont quasi innombrables, notre choix demeure améliorable.

Deuxièmement, nous avons vu que les styles de rédaction de nos exemples d'entraînement sont extrêmement hétérogènes et peuvent causer des surdétections. Il serait donc sage de reconnaître les styles susceptibles d'engendrer des surdétections (comme les SMS) afin soit de taire les détections, soit d'adapter les classifieurs en conséquence. Les modèles de langue se prêtent à cette identification, mais on pourrait également, pour reconnaître un texte où la détection est périlleuse, s'appuyer sur diverses caractéristiques que nous jugeons très discriminantes, par exemple l'absence systématique de majuscules, de diacritiques (accents, cédilles) ou encore la présence de certaines abréviations (par exemple, *tjs* pour *toujours*) ou mots typiques du style de rédaction de certains internautes[13].

Troisièmement, comme l'ont montré les résultats de *Scorali–B*, les traits dérivés des modèles de langue et les SVM se sont révélés deux stratégies fructueuses pour améliorer encore les résultats. Lorsque combinées, ces dernières permettent une augmentation du taux de vrais positifs qui peut atteindre 15 % sans augmenter le taux de faux positifs. L'intérêt évident de ces stratégies plus sophistiquées pose la question de leur portage efficace vers l'industrie. On pourrait simplifier le vocabulaire sur lequel travaillent les modèles de langue ou encore discrétiser les probabilités qu'ils consignent. Par exemple, Whittaker et Raj (2001) montrent que la quantification (*quantization*) et l'élagage de modèles statistiques de langue pour l'anglais autorisent une diminution de 60 % de la taille en mémoire du modèle, sans aucune perte notable

13. Lire à ce sujet (Langlais *et al.*, 2012) pour des exemples franco-québécois.

dans la distribution. Cette approche pourrait être jumelée à la stratégie d'encodage sans perte proposée dans (Pauls et Klein, 2011) qui rapporte une réduction de 75 % de la taille mémoire nécessaire à l'encodage d'un très grand modèle de langue. En ce qui concerne les SVM, l'étude d'autres boîtes à outils que Weka serait également pertinente[14].

Quatrièmement, nous avons remarqué que certaines règles induites par les classifieurs durant l'entraînement pourraient s'ajouter à celles qui ont été entrées manuellement dans le correcteur ou encore contribuer à rectifier certaines règles existantes. Notre travail pourrait ainsi constituer un outil de fouille d'erreurs automatisée dans les sorties d'un correcteur, dans la veine d'autres travaux menés sur la détection de problèmes dans les grammaires à large couverture (Van Noord, 2004 ; Van Noord, 2007 ; Sagot et de la Clergerie, 2009). Afin de parvenir à fouiller automatiquement les sorties qui nous intéressent ici, il est nécessaire d'établir des corpus d'entraînement annotés de manière automatique. Plusieurs approches sont envisageables. À l'instar de Sagot et de la Clergerie (2009) nous pourrions tenter de soumettre des textes à plusieurs correcteurs et établir une procédure de vote afin de repérer automatiquement les fausses alertes. Une autre piste intéressante consiste à transformer les phrases d'un corpus sans fautes en phrases imitant le niveau de langue des utilisateurs ciblés du correcteur. Bigert *et al.* (2003) décrivent par exemple Missplel, un logiciel en mesure de générer des fautes artificielles de syntaxe, d'homophonie, d'orthographe et même de frappe. Il reste cependant un fossé entre les fautes générées par un tel système (par exemple, *je peu*) et celles que nous avons traitées dans cette étude. Les travaux récents de Max et Wisniewski (2010) offrent une solution de rechange potentielle à la création manuelle d'un corpus fautif. Les auteurs ont en effet créé un corpus contenant notamment 74 100 corrections grammaticales à partir de l'historique des révisions des articles français de Wikipédia. Il serait alors réalisable de renverser ces modifications colligées *in vivo* afin d'insérer des fautes là où le texte original est autrement irréprochable.

Cinquièmement, nous pensons que notre approche pourrait s'avérer utile pour l'apprentissage d'une langue seconde. Comme le soulignent Leacock *et al.* (2010), de nombreux travaux s'intéressent à l'apprenant de l'anglais, en partie à cause de l'omniprésence de cette langue dans les communications internationales, avec plusieurs centaines de millions de personnes la pratiquant comme langue seconde. Notre approche pourrait être appliquée aux corpus disponibles d'erreurs commises par les apprenants d'une langue seconde[15] et comparée aux approches existantes. Nous aimerions également pousser les travaux menés, plus particulièrement sur le français langue seconde.

Sixièmement, le milieu réel dans lequel évoluent maintenant les classifieurs intégrés à Analytix change avec le temps. Les données d'entraînement utilisées dans notre étude ont été réunies à un moment particulier de l'histoire du moteur de correction. Son évolution au cours des versions ultérieures modifiera son fonctionnement et

14. SVMlight et libsvm sont deux librairies C++ populaires.

15. Voir (Leacock *et al.*, 2010) [chap. 4] pour une longue liste de corpus disponibles.

son comportement de détection (et donc de surdétection). Il faudrait déterminer si un simple réentraînement périodique serait suffisant à rétablir l'efficacité des filtres, ou s'il faut constituer un nouveau corpus d'entraînement, ou bien encore s'il faut apporter des changements plus importants. Par ailleurs, puisque la langue française évolue également, il est bon de s'interroger sur la robustesse des classifieurs aux prises avec ces changements graduels. Avec l'avènement de formes neuves de communication électronique, comme le microblogage (*microblogging*) et l'étendue grandissante de celles comme les messages SMS, il est fort probable que de nouvelles fautes verront le jour. Ces changements pourront désarmer certains des classifieurs livrés dans ce projet. À nouveau, la question de la procédure la plus efficace pour répondre à ces changements reste ouverte.

En dernière analyse, nous pensons avoir présenté de manière claire que les approches statistiques et symboliques peuvent faire bon ménage, dans l'esprit qui sous-tend (Klavans et Resnik, 1996). Nous espérons de plus avoir montré que l'atténuation de surdétections dans un correcteur grammatical constitue une tâche réelle du traitement des langues qui présente des défis scientifiques intéressants tout en offrant un retour bénéfique à une large communauté d'utilisateurs.

Remerciements

Nous sommes particulièrement reconnaissants à Mala Bergevin d'avoir annoté le corpus de fautes sans lequel notre travail n'aurait pas abouti. Nous remercions également les relecteurs de la revue TAL.

6. Bibliographie

Bigert J., Ericson L., Solis A., « AutoEval and Missplel : Two Generic Tools for Automatic Evaluation », *Proceedings of Nodalida, Reykjavik, Iceland*, 2003.

Clément L., Gerdes K., Marlet R., « Grammaires d'erreur – correction grammaticale avec analyse profonde et proposition de corrections minimales », *Seizième TALN*, Senlis, France, 2009.

Cohen W. W., « Fast Effective Rule Induction », *In Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, p. 115-123, 1995.

Duda R. O., Stork D. G., Hart P. E., *Pattern classification*, John Wiley and Sons, 2001.

Dugast L., Senellart J., Koehn P., « Statistical post-editing on SYSTRAN's rule-based translation system », *Proceedings of the Second Workshop on Statistical Machine Translation*, p. 220-223, 2007.

Ehsan N., Faili H., « Grammatical and context-sensitive error correction using a statistical machine translation framework », *Software : Practice and Experience*, 2012. Online publication.

Fontenelle T., « Dictionnaires et outils de correction linguistiques », *Rev. franç. de linguistique appliquée*, vol. X-2, p. 119-128, 2005.

Genthial D., Courtin J., « From Detection/Correction to Computer Aided Writing », *14th COLING*, Nantes, France, p. 1013-1018, 1992.

Golding A. R., Roth D., « A Winnow-Based Approach to Context-Sensitive Spelling Correction », *Mach. Learn.*, vol. 34, n° 1-3, p. 107-130, February, 1999.

Gotti F., *« L'atténuation statistique des surdétections d'un correcteur grammatical symbolique »*, Master's thesis, Université de Montréal, Sept., 2011.

Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I. H., « The WEKA Data Mining Software : An Update », *SIGKDD Explorations*, 2009.

Helfrich A., Music B., « Design and evaluation of grammar checkers in multiple languages », *Project notes and demonstration at the 18th COLING*, Saarbrücken, Germany, p. 1036-1040, 2000.

Hirst G., « An Evaluation of the Contextual Spelling Checker of Microsoft Office Word 2007 », , "ftp.cs.toronto.edu/pub/gh/Hirst-2008-Word.pdf", January, 2008.

Hoos H., « Programming by Optimisation – Towards a new Paradigm for Developing High-Performance Software », , ls11-www.cs.tu-dortmund.de/_media/staff/preuss/asta/ppsn2012/hoos-ppsn-12-tutorial-slides.pdf, 2012.

Klavans J. L., Resnik P. (eds), *The Balancing Act : Combining Symbolic and Statistical Approaches to Language*, MIT Press, Dec., 1996.

Knutsson O., Pargman T. C., Eklundh K. S., « Transforming grammar checking technology into a learning environment for second language writing », *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*, p. 38-45, 2003.

Kohavi R., « The Power of Decision Tables », *Proceedings of the European Conference on Machine Learning*, Springer Verlag, p. 174-189, 1995.

Langlais P., Drouin P., Paulus A., Brodeur E. R., Cottin F., « Texto4Science : a Quebec French Database of Annotated Short Text Messages », *8th LREC*, Istanbul, Turkey, may, 2012.

Laurent D., Nègre S., Ségéla P., « L'analyseur Cordial dans Passage », *Seizième TALN*, 2009.

Leacock C., Chodorow M., Gamon M., Tetreault J., *Automated Grammatical Error Detection for Language Learners*, vol. 3 of *Synthesis Lectures on Human Language Technologies*, Morgan Claypool, 2010.

Manning C. D., Schütze H., *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts, 1999.

Max A., Wisniewski G., « Mining Naturally-occurring Corrections and Paraphrases from Wikipedia's Revision History », *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC)*, Valletta, Malta, may, 2010.

Morin R., *« Sur l'intégration du correcticiel à la didactique du thème français »*, Master's thesis, Université d'Ottawa, 1995.

Napolitano D., Stent A., « TechWriter : An Evolving System for Writing Assistance for Advanced Learners of English », *CALICO*, vol. 26(3), p. 611-625, 2009.

Pauls A., Klein D., « Faster and smaller N-gram language models », *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies - Volume 1*, p. 258-267, 2011.

Platt J., « Machines using Sequential Minimal Optimization », *in* B. Schoelkopf, C. Burges, A. Smola (eds), *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1998.

Sagot B., de la Clergerie E., « Fouille d'erreurs sur des sorties d'analyseurs syntaxiques », *Traitement Automatique des Langues*, vol. 49, n° 1, p. 41-60, 2009.

Simard M., Ueffing N., Isabelle P., Kuhn R., « Rule-based translation with statistical phrase-based post-editing », *StatMT '07 : Proceedings of the Second Workshop on Statistical Machine Translation*, Association for Computational Linguistics, Morristown, NJ, USA, p. 203-206, 2007.

Sofkova Hashemi S., « Detecting Grammar Errors in Children's Writing : A Finite State Approach », *13th Nordic Conference on Computational Linguistics*, Uppsala, Sweden, May, 2001.

Stolcke A., « SRILM-an extensible language modeling toolkit », *Proceedings International Conference on Spoken Language Processing*, p. 257-286, November, 2002.

Van Noord G., « Error mining for wide-coverage grammar engineering », *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, 2004.

Van Noord G., « Using self-trained bilexical preferences to improve disambiguation accuracy », *IWPT '07 : Proceedings of the 10th International Conference on Parsing Technologies*, Association for Computational Linguistics, Morristown, NJ, USA, p. 1-10, 2007.

Whittaker E. W. D., Raj B., « Quantization-based language model compression », *7th European Conference on Speech Communication and Technology (EuroSpeech)*, Aalborg, Denmark, p. 33-36, 2001.

Yarowsky D., « Unsupervised word sense disambiguation rivaling supervised methods », *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, p. 189-196, 1995.