Reducing Overdetections in a French Symbolic Grammar Checker by Classification

Fabrizio Gotti[1], Philippe Langlais[1], Guy Lapalme[1],
Simon Charest[2], and Éric Brunelle[2]

[1] DIRO/Univ. de Montréal
C.P. 6128, Succ Centre-Ville
H3C 3J7 Montréal (Québec) Canada
http://rali.iro.umontreal.ca
[2] Druide Informatique
1435 rue Saint-Alexandre, bureau 1040
H3A 2G4 Montréal (Québec) Canada
http://www.druide.com

Abstract. We describe the development of an "overdetection" identifier, a system 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 French grammar checker. Eight of these were integrated in the most recent commercial version of the system. This is a striking illustration of how a machine learning component can be successfully embedded in `Antidote`, a robust, commercial, as well as popular natural language application.

1 Introduction

Even though most modern writers use, often unknowingly, the grammar checker embedded in Microsoft Word, few NLP researchers have addressed the problem of improving the quality of the grammatical error detection algorithms [1,2]. Clément et al. [3] suggest that this could be explained by the lack of an annotated error corpus and by the close link that exists between a grammar checker and the proprietary word processor that embeds it.

Bustamante and Léon [4] present a typology of errors often encountered in Spanish and describe how the GramCheck project dealt with them. They distinguish structural errors (e.g. bad prepositional attachments) from non structural ones (e.g. subject verb agreement). The former is dealt with by crafting rules encoding typical errors that are added to the language parsing rules or by using auxiliary grammars on an ad hoc basis. The latter is dealt by loosening the unification process within the parser. These developments are quite complex and require a fine tuning of linguistic heuristics used within the parsing process.

Two main approaches to grammar checking have been taken by researchers. The first approach consists in comparing the sentence to proofread against a model of proper language use (a positive grammar). For instance, [5] propose using n-grams to create a language model of lemmas and part-of-speech tags (POS) occurring in proper English text. The second strategy seeks to create

A. Gelbukh (Ed.): CICLing 2011, Part II, LNCS 6609, pp. 390–401, 2011.

negative grammars in order to represent erroneous language constructs, like in [6] or [7]. Both approaches will often use a corpus of correct and faulty sentences to learn sequences of words that are then compared with the text to check. [8] propose the use of grammar error rules derived from a normal grammar's rule so that the relationship between the correct rule and its derived error rules reflects a possible error as well as the correction to apply. Finding a representative training corpus is a challenge for these approaches, although one could use the one described in [9] or derived from it [10], but more important is the fact that regular expressions cannot reliably detect errors between distant words.

Sofkova Hashemi [11] also uses (positive) regular grammars on POS tags to detect grammatical errors. Using a notion of automata subtraction, she builds on the idea that if a coarse grammar (e.g. not taking into account number and gender agreement) can parse a sentence but not a more precise one, then there is probably an error in this sequence and a detection is made.

It is natural to think that a good grammar checker should strive to reach two conflicting goals : detect all errors present, offering a good *recall*, while avoiding false flags (henceforth *overdetections*), offering a good *precision*. But these goals are not equal in the eyes of the end-user. Indeed, a low precision is a major source of dissatisfaction for them: The overdetections give an (often false) impression that the grammar checker is incorrect in all of its suggestions. Indeed, when Microsoft researched their customers in order to properly design the grammar checker incorporated into their Office suite, they concluded [12] that "increasing precision and decreasing the false flag per page rate have had a higher priority than recall for these grammar checkers." This is corroborated by [13].

In this paper, we focus on a new NLP task: the identification of *overdetections*, i.e. grammatical error detections erroneously made by the system on flawless excerpts of text. We hope to demonstrate that this task presents interesting scientific challenges while offering some feedback to a large community of end-users.

We propose to tackle this task by training classifiers in a supervised way in order to recognize these overdetections. Our work is very different from the ones alluded to above because we are positioned *downstream from* the grammar checker. Resorting to a post-processing strategy has several potential benefits. First, the approach we propose can in principle be adapted to another grammar checker or to other types of errors than those we studied here (see Section 2.3). Second, we already mentioned that modifying an existing parser to account for ill-formed input is a difficult enterprise, one that we avoid here. In fact, the task we address is relatively simpler: we do not locate the errors or suggest a correction because this has already been done by the grammar checker.

We present our project in section 2. In section 3, we describe our approach to the overdetection problem. Results are presented in section 4. Contributions and new perspectives are presented in section 6.

2 ScoRali

The development of a grammar checker or its improvement is a complex endeavor involving many strategic choices as described by [12]. Here, we describe

ScoRali, resulting from the close collaboration between Druide Informatique (Druide) and researchers from Rali. Druide has been, for many years, actively developing a symbolic French parsing technology called Analytix which is based on rich symbolic description dictionaries and on a dependency parser both built and maintained manually by a team of linguists. The parser can deal with complex syntactic phenomena such as coordination (complete and elliptic), extraposition, correlation, punctuation use and some categorial inference. Particular care has been devoted to the parser robustness in the case of lexical and syntactical errors. A correction module uses the syntactic trees to pinpoint errors and suggest appropriate corrections. This technology is embedded in many commercial products, including Antidote, a writing assistant developed for the French language.

2.1 Requirements

The goal of the project was to train classifiers in a supervised way to detect overdetections for 14 common error types processed by Analytix. The error types have been selected by Druide according to their frequency and their overdetection risk (see section 2.3). Used after Analytix processing pipeline, these classifiers would judge the quality of the detections in order to filter out those that would most probably not be appropriate in this context. To be considered worthwhile, a classifier should identify at least 66% of overdetections and not remove more than 10% of correct detections. The classifiers should fit Analytix's processing pipeline.

2.2 Methodology

Data preparation[1] was crucial in this project. For each type of error, Druide prepared a sample of about 1000 detections, separated on average into an equal number of overdetections and legitimate detections, produced by Analytix on "real" texts, representing different types of use of the application. Each occurrence was then annotated by a linguist as being an overdetection or not and was associated with the syntactical parse produced by Analytix. This parse gives the position of each word, its grammatical category and about fifty morphosyntactic attributes such as gender, number (before and after correction) and, for verbs, their mode, tense and person. Moreover, all syntactic relations between words in the sentence were given, allowing to rebuild the syntactic parse tree of the sentence including, for each node, its grammatical category and a confidence weight. Each word was associated with a number of semantic-syntactic tags chosen from more than a thousand available.

This data was used as a basis for the features that we extracted for training our classifiers (see section 3). To help us determine the best ones, Druide also provided, for each type of detection, a summary characterization of the most frequent overdetection contexts and a linguistically motivated estimate of what they felt were the most suitable identification features.

[1] This data is unfortunately not available to the community.

2.3 Types of Errors

After many annotation and development cycles, 14 detection types were studied. Some of them are quite specific with respect to the linguistic phenomena they detect, e.g. the confusions between two words, like QUE/DONT — confusion between *que* ("that") and *dont* ("of that", used with a verb requiring a preposition) — or OU/OÙ — confusion between the conjunction *ou* ("or") and the adverb and pronoun *où* ("where"), which share the same pronunciation and differ only by the grave accent, a common source of error in French texts. An example of a good detection and a incorrect one is shown in Figure 1 for QUE/DONT. In these examples, the underlined word is the site of the detection and ∗ indicates an overdetection. An English translation of the original text and its correction is also provided.

`Je comprends ce que tu dis mais pas ce que` *`[dont]`* `tu parles.`
I understand what you say but not **what** *[of what]* you speak.

`Mais bon dieu que` *`[*dont]`* `les adultes s'amusent!`
But gosh **what** *[*of what]* fun these adults have!

Fig. 1. An example of a good detection (top) and of an overdetection (bottom) for QUE/DONT. This type of detection is concerned with the confusion between 2 French words, "que" (what) and "dont" (of what).

Other detections are more general, in the sense that a given detection, like PP/VC (confusion between the past participle of a verb and its other conjugations) could encompass numerous different linguistic manifestations, given that it applies to many inflected forms of different verbs, sometimes with intervening words within the ill-formed construct. We give an example of such an overdetection for PP/VC in Figure 2 below.

`Roman ou récit, la ≪ Collection blanche ≫ de Gallimard éblouit`
`[*ébloui]``.`
Novel or story, the ≪ Collection blanche ≫ from Gallimard **dazzles** *[*dazzled]*.

Fig. 2. An example of an overdetection for PP/VC, the confusion between the past participle of a verb and its other forms

It should be pointed out that there are many different types of texts in the training corpus: Some sentences were extracted from Wikipedia articles (including some headers), others from Internet discussion boards or scientific texts, etc. Some were even text messages without any diacritical marks, sometimes resorting to *phonetic* spelling. The quality therefore greatly varies.

One observable consequence of the poor quality of some of these sentences is that some overdetections result from other problems in the same sentence. In

the example of Figure 3, the correction of ***le***[*the (masc.)*] into ***la***[*the (fem.)*] is proposed because the word ***marché***[*deal (masc.)*] is misspelled ***marche***[*step (fem.)*]. As explained by [12] and [5], the (obviously poor) French of the writer could have been taken into account when making corrections, here.

`Ils veulent un libéralisme VRAI, accepte le [*la] marche mais...`
They want a TRUE liberalism, accepts **the (masc.)** *[*the (fem.)]* step but...

Fig. 3. An example of an overdetection for ACCORD: an agreement error either in number or gender. In the translation, we purposely introduced errors in "TRUE" (capitalization), "accept" and "step" (the writer misspelled "deal") to illustrate the errors in the French original text, for the corresponding words.

3 Classification of Detections

3.1 Feature Extraction

The manual inspection of hundreds of instances of correct and incorrect detections (like those presented in Figures 2 and 3) shows that words in the neighborhood of the detections made by `Analytix` can guide the classification of a given detection. This context can simply be words before and after the detection or, since the training data includes the syntactical parse produced by `Analytix`, head words or dependents. For instance, for the confusion OU/OÙ, it is rather obvious for a French speaker that, whenever the word "là" precedes a potential confusion, a "où" is expected, rather than "ou". Similarly, for the confusion QUE/DONT, if the head word for the site of the confusion is a verb calling for a prepositional object, then "dont" should be used. Other features of the word flagged as a detection are important. For instance, when detecting capitalization errors, it is not advisable to correct a capitalized word when it follows another capitalized word: they probably participate in a named entity.

As a consequence of the previous observations, we selected a set of more than 1200 features per detection, among which:

The features of the word at the site of the detection. They are : its case, its length in letters, its gender, its number (for a noun), its tense, its number (for a verb), its part of speech, its position in the sentence, as well as features specific to the parser used by `Analytix`, for instance "verbs ending in -yer than can be confused with a noun".

The features of the words surrounding the site of the detection. The same features as those of the previous item, but this time for the words preceding and following the detection, as well as for the head word of the detection, in the parse.

The features of the detection itself. Precisely, the certainty with which `Analytix` proposes a correction, and the features of the correction proposed as a replacement for the word detected.

The features of the sentence in which the detection is found. That is, its length in words, the numbers of dependency relations identified, and the number of unknown words[2].

The nature of the dependency links in which the word detected participates. Namely, the links between the detection and its head word or its possible dependents. Here, we identify the usual relations, like "noun adjunct" or others, more specific to the grammar checker, e.g. the French "tel que" ("for instance").

Some ad hoc features, specific to each type of detection. For instance, we attempted to reframe the classification problem at hand as a word sense disambiguation (WSD) task, for the confusion QUE/DONT. Indeed, if we examine the example in Figure 1, we can consider the site of the detection as a placeholder, and "que" and "dont" as potential "semantic" labels. The disambiguation process allows us then to determine which one of those labels to insert at the site of the ambiguity. This is similar in spirit to the strategy adopted in [14] for fixing context-sensitive spelling corrections, that is, spelling mistakes resulting in existing words (e.g. *piece* versus *peace*). Among other strategies, we used an adaptation of the technique described by [15]. Unfortunately, our incursion into the WSD territory meant that we had to build and use external texts for modeling the context of *que* and *dont*, which precluded the integration of this classifier in `Analytix`.

3.2 Classifiers Studied

To create and put to the test the required classifiers, we used the free software package Weka [16], written in Java[3]. This package allows the easy experimentation of numerous families of classifiers and possesses valuable features, like the visualization of data and classifiers as well as the preprocessing of training data. Moreover, it is possible to bypass the graphical interface and launch a classifier from the command line, which proved invaluable in our case when batch-processing thousands of classifier commands.

Weka allows the prototyping of roughly 50 classifiers, grouped into 8 families, e.g. Bayesian classifiers, decision trees, perceptrons, SVM and meta-classifiers. The latter combine other classifiers, by making them vote, for instance. Each of these classifiers is typically controlled by 1 to 20 hyperparameters, discrete or continuous. This causes a combinatorial explosion in the number of possible classifiers. Therefore, our first efforts focused on the exploration of classifiers that are rapid during training and classification, partly to satisfy the specifications for the project. Additionally, we preferred classifiers which were conceptually "simple", in order to facilitate their tuning, design, and eventual implementation within `Analytix`. These reasons led us to concentrate our efforts on symbolic classifiers (but see section 4.4), namely `rules.ConjunctiveRule`, `rules.DecisionTable`,

[2] It is noteworthy that most of the features which are numerical counts are doubled: one is the count itself, the other is the count normalized by the length of the sentence.

[3] `www.cs.waikato.ac.nz/ml/weka/`

`rules.JRip` (a propositional rule learner, like RIPPER [17]), `trees.ADTree` (alternating decision trees), `trees.DecisionStump`, `trees.J48` (C4.5 decision trees) and `trees.J48graft`.

Beyond the selection and parametering of the classifiers, Weka allows diverse pre-processing strategies on the training data. We first filtered features, to remove those which did not vary enough or varied too much among the instances of the training set (these features cannot be used to discriminate). Furthermore, for every type of detection, we tried different filtering strategies for their features, reducing in some case the 1200 features to a mere dozen. Naturally, this simplifies the training and test of the classifiers, but also their eventual implementation. We also varied the cost that Weka attributes to false negatives and false positives. Ultimately, for each of the 14 detection types, we tested about 4000 classifier settings in order to find one which would meet the requirements set by `Druide`. This exploration was made on a 16 dual-core computer cluster, with a computing time of 5 days for each detection.

4 Results

It is impossible to fit all the results obtained on all classifiers tested in this article. For this reason, in this section, we will detail the results for the detection PP/VC, for which we provided an example earlier in Figure 2. We chose this example because it lent itself quite successfully to the approach, and because it is representative of the results we obtained on several other detections. Also, it is an illustration of a detection arising in very varying contexts, which proves challenging. We will nonetheless present a global overview of the results for all the detection types later in this article.

4.1 The pp/vc Detection

Figure 4 shows the performance of more than 3000 classifiers for the project SCORALI (it is the scatter plot cluster labeled "System". The x-axis represents the percentage of good corrections erroneously flagged as bad corrections by the classifiers (false positives), while the y-axis represents the percentage of bad corrections correctly identified as such (true positives). For each classifier, this data was obtained through 10-fold cross-validation on the training set.

Remarkably, the scatter plot cluster is relatively compact, forming a band encompassing the possible compromises each classifier offers. The choice of a classifier is made manually, based on this kind of figure, while striving to meet the requirements set by `Druide` (section 2.1). In our case, the classifier recommended for `Analytix` is the one whose data point is circled in the figure. It is a C4.5 decision tree, grafted and pruned, allowing the identification of 77 % of overdetections, at the cost of a loss of 8 % of good corrections. The decision tree classifies 88 % of instances correctly, with a substantial agreement of $\kappa = 0.76$ between all 10 folds of the cross-evaluation. Other classifiers with similar performances were discarded, either because they were too complex to implement or because they used too many features.

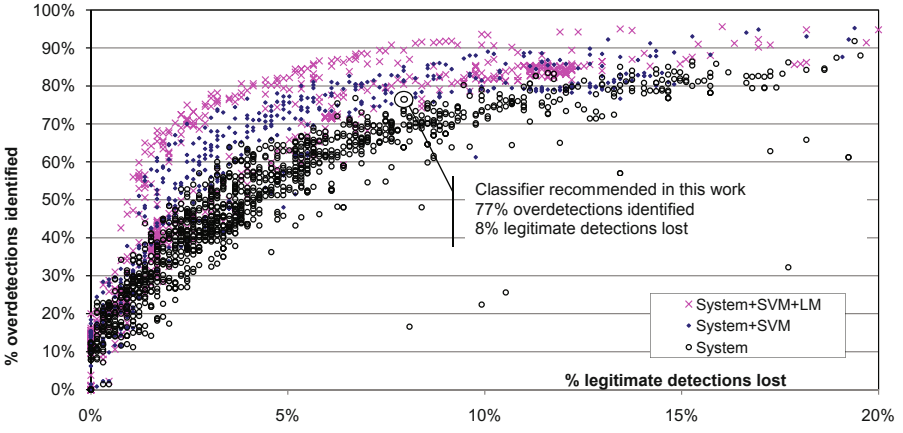

Fig. 4. Classifiers tested for the detection PP/VC. Each point represents the performance of a single classifier, i.e. a compromise between the true and false positives. The graph shows 3 scatter plots, one for SCORALI per se (Section 4, labeled "System") and 2 others, resulting from further research (Section 4.4).

4.2 An Overview of All Detection Types

We applied the strategy described in the previous section for all 14 types of detections provided by `Druide`. Although the lack of place prevents us from describing each error type, the overall results are presented in Table 1. We first observe that we succeeded in creating classifiers meeting the project's requirements for 9 out of the 14 detection types. They all are decision trees: when a group of classifiers proved equally good at classifying detections, we selected a decision tree among them. A closer inspection of the induced rules used in the decision trees did not allow the identification of features consistently present in all or most of the trees.

It is difficult to explain why certain detections lent themselves well to classification, and others not. Despite our best efforts, QUE/DONT could not find a good classifier, whereas some detections proved easy to process, although it seemed at first that rule induction would be difficult because they occurred in extremely different contexts, for different reasons and often in text of very poor quality.

Naturally, classification rules are induced more easily if `Analytix` overdetects within a certain language construct in a systematic way. This is the case for the detection LA/LÀ, where a manual inspection of the decision tree produced shows that 20 % of the overdetections occur when "la" is an article ending an abruptly truncated sentence, like in the instance "recommandations de **la** *[*là]*" (recommendations of **the** *[*there]*). The construct is always the same then: a missing noun adjunct preceded by the article.

It is also obvious that certain overdetections made by Analytix are very difficult to classify, even for a human being. The examples shown in Figure 2 for PP/VC and in Figure 3 are striking. In the latter Figure, the instance "accepte **le** *[*la]* marche", one must really understand the sentence to know that the author meant "accepte le marché" ("accept the deal", where "marché" is masculine) rather than "accepte la marche" ("accept the step", where "marche" is feminine). The overdetection is then due to the missing acute accent on the final letter of "marche". It is highly likely that classifiers have a hard time with these cases, especially when these detections are flagged in varying contexts, for different reasons. It is the case for ACCORD, which can be an agreement error, either in gender or number. It could be interesting to further our research by creating two different detection types: ACCORD for gender and ACCORD in number.

Table 1. Complete results of ScoRali, as delivered to Druide

Detection	% fp	% tp	Detection	% fp	% tp	Detection	% fp	% tp
ÉLISION	7%	87%	INV	8%	71%	QUE/DONT	9%	57%
LA/LÀ	9%	84%	MAJ	7%	71%	OU/OÙ	9%	49%
PP INV	6%	80%	PP/INF	7%	68%	ACCORD	9%	40%
PP/VC	8%	77%	CONJUG	8%	66%	MODE	9%	27%
APOS	6%	73%	ER/EZ	9%	65%			

4.3 Tests at Druide and Implementation

The eleven best classifiers were therefore converted and integrated to Analytix, then tested on a corpus distinct from the one we used for their training, a recommended practice in the industry (see for instance [12]). These tests revealed that 3 classifiers degraded the performance of the grammar checker to a point where they had to be rejected. These classifiers were not necessarily those with the worst performances during training, but had to be removed nonetheless. The 8 remaining classifiers are part of the latest commercial version of Analytix. Among these, 3 are used as they are, and five are subject to an ad hoc test determining whether the engine will use them, based on the context of the detection. Finally, the user interface includes an on/off switch for these classifiers: The users are presented with a checkbox labeled "statistical filtering of detections". The user wishing not to miss any good detection can deactivate this setting, but will be presented with more false detections. The classifiers are on by default.

4.4 Better Classifiers

As a requirement (see Section 2.1), the classifiers delivered to Druide had to be simple to interpret and to embed within Analytix. In order to measure the improvements that could be made to ScoRali, we studied the performances offered by SVM classifiers and we added features derived from language models

trained on the Canadian Hansard. The gains obtained are shown in Figure 4 by their respective scatter plot cluster, for detection PP/VC. SVMs (labeled "System+SVM") alone allow a gain of about 5% in identification of overdetections, compared to the classifiers delivered to `Druide`, at the cost of an increase in computational needs and a decrease in expressivity of the model created. The addition of language model features (labeled "System+SVM+LM") increases the number of true positives, for a further 10% gain.

5 Related Work

A fair number of studies have been dedicated to spelling correction (e.g. [18,14,19]). Grammar checking, which we believe is a useful component of a writing assistant tool, has received — somehow paradoxically — much less attention. However, we see many advantages to studying grammar checkers on their own. Indeed, we feel this naturally belongs to the field of *grammar engineering*. Behind this expression, we group activities as diverse as making parsing faster and more robust (e.g. [20]), adapting parsers to new domains (e.g. [21]), or simply improving existing parsers (e.g. [22]).

Actually, for certain types of detections, the classifiers we trained proved very useful as error mining tools within `Analytix`'s parsing grammar.[4] For instance, we noticed that `Analytix` has difficulty recognizing the expression "faire partie" (take part) and makes the overdetection "faire **partie** *[*partit]*" (take parted), for detection PP/VC. Although the particular classifier for PP/VC did not include such a feature, it would probably be interesting to detect the presence of the verb "faire" in the context leftward of the detection.

Thus, the work we conducted could prove, as a side effect, to be complementary to the studies made on error identification in wide-coverage grammars [22,23,24], but has the advantage of not requiring the modification of the grammar studied, a delicate task which is not always possible in a complex grammar maintained manually, especially in a commercial context.

6 Conclusion and Future Work

ScoRali allowed the creation and implementation into `Analytix` of 8 out of 14 classifiers identifying overdetections, downstream of the grammar checking engine. This successful transfer of technologies from the laboratory to a commercial product entailed the exploration of thousands of different classifiers, as well as a delicate balance between performance and technical constraints.

We think that this paper clearly shows that statistical and symbolic approaches can go hand in hand, and is indeed a very clear illustration of the kind of balancing act that such a combination requires [25].

[4] This is one argument in favor of classifiers such as decision trees that can be easily interpreted against other ones such as SVMs.

Despite the fact that the corpus we used in this study can not be released to the scientific community, we hope to have shown that overdetection identification constitutes a "real" task in NLP, one that presents interesting scientific challenges while offering some feedback to a large community of end-users.

This work shows some avenues that we think are worth investigating. For the time being, certain detections seem not to lend themselves to the proposed approach, maybe because they occur in contexts which are too varied, thus defying rule induction. The work we conducted on using more features and more robust classifiers (see Section 4.4) for the PP/VC detection shows that there is room for improvement. This suggests further experiments to see if such gains carry over other detections.

One limitation of our work lies in the simplifying assumption that each detection within a sentence is independent of the other possible detections within the same sentence, although evidence shows that one actual error can trigger an overdetection.

Also, we feel the evolution of ScoRali poses a number of exciting questions. The data used to train the classifiers is not frozen in time: it was generated by `Analytix` at a given moment in its life cycle. Although our classifiers passed a number of regression tests at `Druide`, it remains to be seen whether they will withstand the likely changes that will happen over time (within `Analytix`, in detection statistics, etc.) or whether new training (or adaptation) will be required.

References

1. Fontenelle, T.: Dictionnaires et outils de correction linguistiques. Rev. franç. de linguistique appliquée X-2, 119–128 (2005)
2. Véronis, J.: Texte: Correcteurs orthographiques en panne? Blog du (July 6, 2005), `http://aixtal.blogspot.com/2005/07/texte-correcteurs-orthographiques-en.html`
3. Clément, L., Gerdes, K., Marlet, R.: Grammaires d'erreur – correction grammaticale avec analyse profonde et proposition de corrections minimales. In: 16è TALN, Senlis, France (2009)
4. Bustamante, F.R., León, F.S.: Gramcheck: A grammar and style checker. In: 16th COLING, Denmark, pp. 175–181 (1996)
5. Napolitano, D., Stent, A.: TechWriter: An Evolving System for Writing TechWriter: An Evolving System for Writing Assistance for Advanced Learners of English. CALICO 26(3), 611–625 (2009)
6. Rider, Z.: Grammar checking using pos tagging and rules matching. In: Proceedings of the Class of 2005, Senior Conference, Computer Science Department, Swarthmore College, pp. 14–19 (2005)
7. Souque, A.: Vers une nouvelle approche de la correction grammaticale automatique. In: Récital, Avignon, France (2008)
8. Foster, J., Vogel, C.: Parsing ill-formed text using an error grammar. Artif. Intell. Rev. 21(3-4), 269–291 (2004)
9. Foster, J.: Good Reasons for Noting Bad Grammar: Empirical Investigations into the Parsing of Ungrammatical Written English. PhD thesis, Department of Computer Science - University of Dublin (May 2005)

10. Foster, J.: Treebanks gone bad: Parser evaluation and retraining using a treebank of ungrammatical sentences. Int. J. Doc. Anal. Recognit. 10(3), 129–145 (2007)
11. Sofkova Hashemi, S.: Detecting grammar errors in children's writing: A finite state approach. In: 13th Nordic Conference on Computational Linguistics, Uppsala, Sweden (May 2001)
12. Helfrich, A., Music, B.: Design and evaluation of grammar checkers in multiple languages. In: Project notes and demonstration at the 18th COLING, Saarbrücken, Germany, pp. 1036–1040 (2000)
13. Bernth, A.: Easyenglish: a tool for improving document quality. In: Proceedings of the Fifth Conference on Applied Natural Language Processing, Morristown, NJ, USA, pp. 159–165. Association for Computational Linguistics (1997)
14. Golding, A.R., Roth, D.: A winnow-based approach to context-sensitive spelling correction. CoRR cs.LG/9811003 (1998)
15. Yarowsky, D.: Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In: 33rd Meeting of the ACL, Cambridge, MA, pp. 189–196 (1995)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1), 10–18 (2009)
17. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann, San Francisco (1995)
18. Damerau, F.: A technique for computer detection and correction of spelling errors. Commun. ACM 7(3), 171–176 (1964)
19. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: ACL 2000: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA, pp. 286–293. Association for Computational Linguistics (2000)
20. Kiefer, B., Krieger, H.U., Carroll, J., Malouf, R.: A bag of useful techniques for efficient and robust parsing (1999)
21. Rimell, L., Clark, S.: Adapting a lexicalized-grammar parser to contrasting domains. In: EMNLP 2008: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Morristown, NJ, USA, pp. 475–484. Association for Computational Linguistics (2008)
22. van Noord, G.: Using self-trained bilexical preferences to improve disambiguation accuracy. In: IWPT 2007: Proceedings of the 10th International Conference on Parsing Technologies, Morristown, NJ, USA, pp. 1–10. Association for Computational Linguistics (2007)
23. Sagot, B., de la Clergerie, E.: Fouille d'erreurs sur des sorties d'analyseurs syntaxiques. Traitement Automatique des Langues 49(1), 41–60 (2009)
24. de Kok, D., Ma, J., van Noord, G.: A generalized method for iterative error mining in parsing results. In: Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF 2009), Suntec, Singapore, pp. 71–79. Association for Computational Linguistics (August 2009)
25. Klavans, J.L., Resnik, P. (eds.): The balancing act: combining symbolic and statistical approaches to language. MIT Press, Cambridge (1996)