RECITAL 2008, Avignon, 9–13 juin 2008

Vers une nouvelle approche de la correction grammaticale automatique

Agnès Souque
Laboratoire LIDILEM - Université Stendhal - Grenoble 3
1491 rue des Résidences
38040 Grenoble Cedex 09
asouque@gmail.com

Résumé. La correction grammaticale automatique du français est une fonctionnalité qui fait cruellement défaut à la communauté des utilisateurs de logiciels libres. Dans le but de combler cette lacune, nous avons travaillé à l'adaptation au français d'un outil initialement développé pour une langue étrangère. Ce travail nous a permis de montrer que les approches classiques du traitement automatique des langues utilisées dans le domaine ne sont pas appropriées. Pour y remédier, nous proposons de faire évoluer les formalismes des correcteurs en intégrant les principes linguistiques de la segmentation en *chunks* et de l'unification. Bien qu'efficace, cette évolution n'est pas suffisante pour obtenir un bon correcteur grammatical du français. Nous envisageons alors une nouvelle approche de la problématique.

Abstract. Free software users community is sorely lacking French grammar checking. With the aim of filling this gap, we have worked on the adaptation to French of a tool originally developped for a foreign language. Thanks to this work, we could show that classic natural language processing approaches used in grammar checking are not suitable. To remedy it, we suggest an evolution of grammar checkers that includes linguistic principles such as chunking and unification. Despite its efficiency, this evolution is not sufficient to get a good French grammr checker. We are then thinking of a new approach of the problem.

Mots-clés : correction grammaticale, syntagme, unification.

Keywords: grammar checking, chunk, unification.

1 Introduction

Dans le domaine de la correction grammaticale automatique, les outils existants sont généralement des logiciels propriétaires aux coûts d'intégration élevés, qui sont peu ou pas décrits dans la littérature, et fermés à toute amélioration externe. Nos travaux de recherche visent donc à développer un outil de correction grammaticale libre pour le français, dont les ressources langagières seront accessibles et modifiables, et dont le formalisme générique le rendra facilement adaptable à d'autres langues.

Dans (Souque, 2007), nous avons étudié plusieurs correcteurs libres existant dans des langues étrangères. L'adaptation au français de l'un d'entre eux nous a permis de mettre à la disposition de la communauté scientifique, mais aussi de la communauté des utilisateurs de logiciels

libres (OpenOffice.org, WEB), un formalisme, un outil informatique et une base conséquente de règles de correction grammaticale. Cependant, nous avons mis en évidence l'inadéquation des approches classiques du traitement automatique des langues à la problématique.

Dans la section 1 de cet article, nous commençons par expliquer dans quelle mesure le principe de fonctionnement des correcteurs libres que nous avons étudiés nuit à leur efficacité. Nous exposons plus particulièrement le problème de la détection des erreurs qui se fonde sur des règles sans abstraction et sur un système de *pattern-matching* rigide, dont une des conséquences est la nécessité d'énumérer dans des règles toutes les fautes possibles.
Pour simplifier la détection des erreurs, nous proposons dans la section 2 une évolution des formalismes des correcteurs grammaticaux, qui combine deux approches syntaxiques issues d'écoles opposées : la segmentation en *chunks* et l'unification de structures de traits.
La section 3 est consacrée à la validation de l'évolution présentée dans la section précédente et aux bénéfices qu'elle peut apporter. Nous avons pour cela développé un prototype de calculateur d'unification auquel nous avons soumis des phrases contenant des erreurs de grammaire.

2 Une correction grammaticale limitée

Il ne nous a pas été possible d'analyser le fonctionnement des correcteurs grammaticaux propriétaires, ces derniers n'étant pas documentés. Les outils que nous avons étudiés sont donc exclusivement des outils libres, tel que An Gramadóir (AnGramadóir, WEB) de Kevin Scannell, ou (LanguageTool, WEB) développé par Daniel Naber (Naber, 2003).

Ces systèmes de correction grammaticale ont une structure en couche qui effectue le traitement du texte en plusieurs étapes successives. Dans un premier temps, le texte est segmenté en phrases, puis en *tokens* [1]. L'étape suivante consiste à étiqueter morpho-syntaxiquement chacun de ces *tokens*. Lors de cette étape, les mots peuvent recevoir plusieurs étiquettes s'ils sont ambigus. Un traitement supplémentaire est alors nécessaire pour les désambiguïser, c'est-à-dire pour réduire le nombre d'étiquettes qu'ils possèdent, selon une méthode statistique ou fondée sur des règles. La dernière étape consiste enfin à détecter les erreurs de grammaire. Cette détection utilise le principe du *pattern-matching* à partir d'une base de règles de correction. Ce principe rigide consiste à comparer des séquences de *tokens* du texte avec des patrons de séquences de *tokens* décrits dans les règles. Une erreur est détectée lorsqu'il y a correspondance exacte entre les deux. Le correcteur LanguageTool que nous avons adapté au français dans (Souque, 2007), fonctionne selon ce principe.

Dans cette section, nous présentons les principales conséquences de l'utilisation du *pattern-matching* rigide dans les correcteurs grammaticaux que nous avons étudiés. Ces outils nécessitent un très grand nombre de règles de correction, ils se retrouvent pris au piège d'un cercle vicieux et se révèlent incapables de détecter des erreurs, pourtant fréquentes, impliquant des mots distants.

[1] Un *token* est généralement un signe de ponctuation ou un mot, au sens forme graphique. En effet, le mot en linguistique peut désigner une unité sémantique constituée de plusieurs formes graphiques. Par exemple, "pomme de terre" est une unité composée de 3 formes graphiques, qui correspondent à 3 *tokens.*

2.1 L'explosion combinatoire des règles

Pour qu'une erreur grammaticale soit détectée, elle doit donc être décrite dans une règle. Ceci implique que pour avoir une efficacité maximale, le correcteur doit posséder une règle de correction pour chaque erreur possible.

Si nous prenons l'exemple simple des syntagmes nominaux. Ils sont principalement constitués de déterminants, de noms et d'adjectifs, en quantité variable, chacun se déclinant en plusieurs genres et nombres. Les correcteurs An Gramadóir ou LanguageTool utilisent par exemple un lexique dans lequel les déterminants, les noms et les adjectifs peuvent être de genre masculin, féminin ou épicène, et de nombre singulier, pluriel ou invariable.

Si nous voulons écrire une règle pour chaque erreur d'accord possible dans un syntagme nominal, la non abstraction des motifs dans les règles nous oblige d'une part à énumérer toutes les positions que peut prendre chaque mot au sein du syntagme en fonction de sa catégorie, et d'autre part à tenir compte pour chacun de toutes les combinaisons genre-nombre qu'il peut avoir. Nous sommes confrontés ainsi rapidement à une explosion combinatoire du nombre de règles, telle que le montre la figure 1.

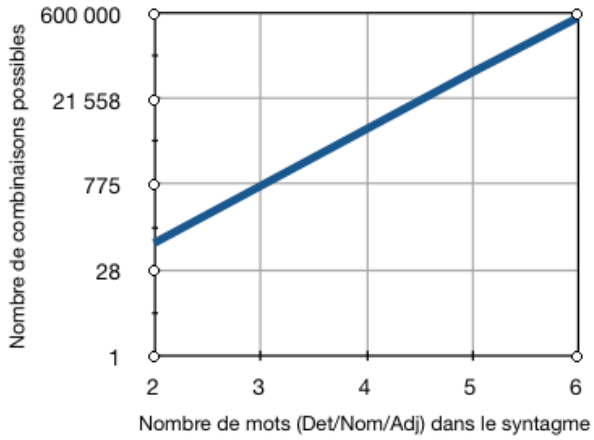

FIG. 1 – Explosion combinatoire du nombre de règles pour un syntagme nominal

Nous retrouvons le même problème pour tous les types d'erreurs. Le fait de devoir décrire tous les contextes de fautes implique qu'il faut imaginer toutes les combinaisons erronées de mots, ce qui est impossible.
Il s'ensuit de nombreux problèmes de silence ou de bruit dans la détection des fautes : du silence lorsqu'une erreur n'a pas été prévue et n'est pas décrite dans une règle ; du bruit lorsque plusieurs règles, parmi les très nombreuses existantes, détectent une même erreur.
Par exemple, dans un énoncé tel que `*"La petit aiguille indique l'heure"` l'erreur de genre de `"petit"` est détectée à deux reprises par le correcteur LanguageTool :
– une première règle s'applique car un déterminant féminin est suivi d'un adjectif masculin ;
– une seconde règle s'applique car un adjectif masculin est suivi d'un nom féminin.

Il est généralement préférable pour l'utilisateur que le silence soit privilégié au bruit. Les fausses détections sont en effet très gênantes, et lorsqu'elles sont trop fréquentes, elles conduisent souvent l'utilisateur à ne plus utiliser le correcteur.

2.2 Le cercle vicieux

Le nombre de règles n'est pas l'unique responsable du bruit et du silence. En effet, pour que les règles de correction puissent s'appliquer et que la détection des erreurs se fasse correctement, selon le principe du *pattern-matching*, il faut que la séquence de *tokens* contenant l'erreur et le patron décrit dans une règle correspondent parfaitement. Cependant, les erreurs de grammaire ou d'orthographe dans le texte peuvent conduire à des erreurs d'étiquetage, qui à leur tour nuisent à l'application des règles, car la correspondance entre le texte et les motifs des règles ne peut plus se faire.

Par exemple, la règle suivante décrit un motif permettant de détecter l'oubli de la particule `"ne"` de la négation : pronom sujet + verbe + `"pas"`
En présence d'un énoncé tel que : `*"Il travail pas assez."` la règle ne peut pas être appliquée. En effet, tel qu'il est orthographié, le mot `"travail"` porte l'étiquette nom. La séquence ne correspond donc pas au motif décrit dans la règle et l'erreur concernant l'oubli de `"ne"` ne peut pas être détectée.

Nous nous heurtons ainsi au problème du cercle vicieux en correction grammaticale : la bonne détection des erreurs dépend d'un bon étiquetage, qui lui-même dépend d'un texte sans erreurs...

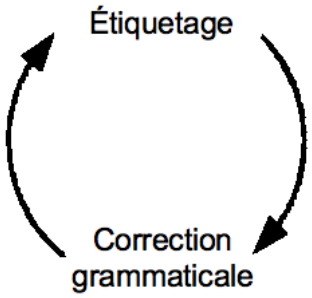

FIG. 2 – Cercle vicieux

2.3 La limitation au contexte immédiat

Le principe du *pattern-matching* a également pour conséquence de rendre la détection des erreurs portant sur des relations distantes très difficile, voire impossible. Prenons par exemple l'énoncé ci-dessous :

`*"Les personnes en situation de test s'autosurveille énormément."`

Le verbe `"autosurveille"` n'est pas accordé correctement avec le syntagme nominal sujet `"Les personnes"`. Pour détecter cette erreur d'accord sur le verbe, il faut une règle décrivant un motif du type : Det fém plur + Nom fém plur + 5 mots + Verbe 3e pers sing
Une telle règle convient dans notre exemple précis, mais ne fonctionne plus si les mots qui doivent s'accorder sont séparés d'un nombre de *tokens* différent de 5. Il faudrait donc créer une règle pour chaque quantité de mots, quantité qu'il est impossible de prévoir.
D'autre part, nous pouvons évoquer à nouveau, avec cet exemple, le problème de l'explosion combinatoire des règles. En dehors du nombre de "mots séparateurs" à prévoir, il faut également créer une règle pour chaque possibilité de sujet du verbe, c'est-à-dire avec chaque combinaison de mots pouvant constituer un syntagme nominal sujet.

2.4 Conclusion

Le principe du *pattern-matching* rigide, avec la non abstraction des motifs décrits dans les règles, oblige à prévoir toutes les erreurs possibles et rend ainsi très coûteux le travail de constitution du nombre démesuré de règles nécessaires, travail qui doit par ailleurs être répété pour chaque langue.
Nous pensons qu'il est possible de simplifier ce travail en utilisant des principes linguistiques mieux adaptés, permettant d'intégrer un niveau d'abstraction dans les règles et de remplacer ainsi les nombreuses règles spécifiques par quelques règles génériques simples.

3 Une amélioration possible des correcteurs

La simplification des règles est indispensable à une correction plus efficace et moins coûteuse. Pour y parvenir, nous avons combiné, de manière atypique, les deux concepts de la segmentation en *chunks* et de l'unification de structures de traits (Miller & Toris, 1990).
Dans cette partie, nous présentons donc ces deux concepts et l'intérêt de les combiner pour simplifier la détection des fautes.

3.1 La segmentation en *chunks*

(Abney, 1991) définit les *chunks* de la manière suivante :

> *« The typical chunk consists of a single content word surrounded by a constellation of function words, matching a fixed template »*

Un *chunk* est donc un groupe de mots contigus, réunis autour d'une tête lexicale dont ils dépendent. Ces relations de dépendance font de la structure interne des *chunks* une structure relativement figée, dans laquelle les contraintes d'accord sont assez fortes, ce qui nous intéresse particulièrement pour la correction grammaticale.

Les *chunks* sont principalement de type nominal ou verbal, et plus rarement adjectival ou adverbial. Ils sont délimités grâce aux mots grammaticaux, à la ponctuation ou aux marques morphologiques, et sont donc relativement faciles à définir. En général, un *chunk* commence par un mot grammatical ou juste après une ponctuation, et se termine juste avant une ponctuation ou le mot grammatical suivant, qui marquent alors le début d'un nouveau *chunk*. Ainsi, la définition d'un *chunk* ne se fait pas en fonction de son contenu, mais en fonction de ses marqueurs de début et de fin.

Les *chunks* sont également appelés syntagmes, mais ils font partie des grammaires robustes du TAL dans lesquelles la notion de syntagme diffère de celle des grammaires chomskyennes, d'où est issu le principe d'unification. Dans (Chomsky, 1979), les syntagmes sont récursifs par exemple, ce qui signifie qu'ils peuvent contenir d'autres syntagmes du même type. Ainsi `"un code élaboré et un code restreint"` est considéré comme un syntagme nominal constitué de deux autres syntagmes nominaux.
Pour les grammaires robustes au contraire, l'exemple ci-dessus consiste en trois syntagmes distincts, ou *chunks* : `"[un code élaboré] [et] [un code restreint]"`. Ces syntagmes ne sont pas récursifs, mais constituent au contraire un niveau hiérarchique entre les mots et la phrase. Ainsi, chaque *chunk* est constitué d'éléments du niveau hiérarchique infé-

rieur (les mots) et constitue à son tour un élément du niveau hiérarchique supérieur (la phrase) (Vergne & Giguet, 1998).
Lorsque nous employons le mot syntagme dans la suite de cet article, nous faisons référence aux syntagmes non récursifs, c'est-à-dire aux *chunks*.

L'exemple que nous donnons en 2.3 est segmenté en *chunks* de la manière suivante :

```
[*Les personnes] [en situation] [de test] [s'autosurveille
énormément].
```

Au sein d'une phrase, les *chunks* entretiennent également des relations de dépendance. Un syntagme dépend généralement de son prédécesseur, sauf dans le cas d'un *chunk* verbal. Ce dernier dépend du *chunk* sujet, qui correspond en principe au premier syntagme nominal à gauche. Ces propriétés de dépendance vont s'avérer très utiles pour vérifier les accords entre différents syntagmes.

3.2 L'unification de structures de traits

Les étiquettes morpho-syntaxiques attribuées à chaque mot lors de l'étiquetage du texte constituent des structures de traits. Elles décrivent chaque élément des phrases en énumérant ses caractéristiques linguistiques, sous la forme de liste de couples attribut-valeur.

L'unification est définie de manière générale dans (Abeillé, 1993) :

> *« L'unification de deux structures de traits A et B (notée A ∪ B) est la structure minimale qui est à la fois une extension de A et de B. Si une telle extension n'existe pas, l'unification «échoue» (ce qui est noté ⊥) ».*

Plus précisément, dans le cadre de notre travail, nous dirons que l'unification consiste à combiner les traits de deux étiquettes et à vérifier leur compatibilité. Deux structures de traits peuvent s'unifier si elles ont les mêmes valeurs pour des attributs identiques.
Par exemple, dans la figure 3, l'unification échoue entre l'étiquette du nom `"genoux"` et celle du déterminant `"le"`, car la valeur de l'attribut nombre n'est pas identique.

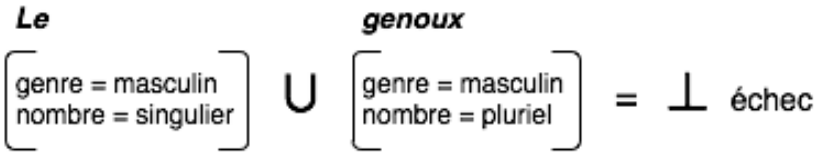

FIG. 3 – Échec de l'unification

3.3 La combinaison des *chunks* et de l'unification

Le fait de combiner le découpage en *chunks* et l'unification de structures de traits va permettre de faciliter la vérification grammaticale, en particulier pour les phénomènes d'accord.
Les *chunks* délimitent des zones de calcul en fonction de leurs frontières, et non plus en fonction de leur contenu, comme c'est le cas avec les patrons de séquences de *tokens* dans les règles. Il n'est alors plus nécessaire de lister exhaustivement toutes les combinaisons de mots.
À l'intérieur de ces zones de calcul, tous les éléments doivent s'accorder, ce qui signifie que tous leurs traits doivent s'unifier, indépendamment de leur catégorie grammaticale. Il est ainsi

possible de détecter les erreurs d'accord, même si un mot est ambigu ou mal étiqueté.
De plus, en attribuant aux *chunks* les traits de leur tête lexicale, ils peuvent alors s'unifier entre eux, et des erreurs entre groupes de mots peuvent ainsi être détectées, comme par exemple l'accord entre le syntagme nominal sujet et le syntagme verbal.
L'utilisation combinée de ces deux principes linguistiques permet donc d'introduire une abstraction dans les motifs des règles, dont le nombre peut être ainsi considérablement réduit.

4 Un prototype de calculateur d'unification

Pour valider notre hypothèse selon laquelle la combinaison des *chunks* et de l'unification peut permettre de détecter des fautes à l'aide de quelques règles simplifiées, nous avons conçu un calculateur d'unification. L'outil prend en entrée des phrases préalablement étiquetées et segmentées en *chunks*, il recherche des erreurs d'accord en calculant l'unification à l'aide d'expressions régulières, puis crée un fichier de sortie contenant des informations sur les erreurs détectées.

4.1 L'étiquetage de phrases test

Pour pouvoir tester notre outil, nous avions besoin de phrases étiquetées et segmentées. Nous avons choisi le formalisme XML pour représenter ces phrases. Bien que plus lourd à traiter par le programme, ce formalisme a plusieurs avantages. Il est facilement lisible et modifiable par des linguistes, qui ne sont pas nécessairement informaticiens, c'est un formalisme ouvert qui permet l'implantation de nombreuses fonctionnalités, et les balises qui le constituent sont faciles à identifier par un outil tel que le nôtre qui utilise des expressions régulières. Il s'agit d'autre part du formalisme d'étiquetage utilisé dans le correcteur LanguageTool sur lequel nous avons travaillé.

Ne disposant pas d'étiqueteur en XML ni de segmenteur en *chunks* compatibles avec ce que nous voulions obtenir, nous avons réalisé le travail manuellement sur quelques phrases, extraites d'un corpus de variations orthographiques (COVAREC, 1994) (Lucci & Millet, 1994). Nous avons obtenu des phrases étiquetées de la manière suivante :

```
*"Les personnes en situation de test s'autosurveille énormément"
   <SN genre="f" nombre="p">
       <D nombre="p">Les</D>
       <N genre="f" nombre="p">personnes</N>
   </SN>
   <SP>
       <P>en</P>
       <N genre="f" nombre="s">situation</N>
   </SP>
   <SP>
       <P>de</P>
       <N genre="m" nombre="s">test</N>
   </SP>
   <SV type="ppal">
       <R type="refl" pers="3" >s'</R>
```

```
        <V mode="ind" temps="pres" pers="3" nombre="s">
        autosurveille</V>
        <A>énormément</A>
    </SV>
```

Les balises SN, SV et SP délimitent respectivement les *chunks* nominaux, verbaux et prépositionnels. Chacune contient à son tour un ou plusieurs élements : dans notre exemple, nous avons les balises D (déterminant), N (nom), P (préposition), V (verbe), R (pronom) et A (adverbe), À l'intérieur des balises, divers attributs définissent les traits morpho-syntaxiques (genre, nombre, temps, etc.) de chaque mot.

4.2 Le fonctionnement du prototype

Notre prototype fonctionne à l'aide d'expressions régulières, qui permettent de trouver facilement des motifs dans les chaînes de caractères. Comme tout l'étiquetage se fonde sur des balises, il est facile de définir des motifs pour repérer les différents niveaux : phrases, *chunks*, éléments et attributs.

L'outil commence par parcourir le fichier XML afin d'isoler et de stocker dans un tableau chaque niveau dans une phrase. Ainsi, pour l'exemple en 4.1, l'outil va commencer par isoler et stocker le *chunk* SN, ses attributs et les 2 éléments D et N avec leurs attributs, puis il va continuer avec les *chunks* suivants jusqu'à la fin de la phrase.

Viennent ensuite les calculs d'unification. Ils sont d'abord effectués au sein des *chunks*, à partir des données stockées précédemment. Le programme compare la valeur de chaque attribut d'un élément avec la valeur du même attribut dans tous les autres éléments du *chunk* traité. Si deux mêmes attributs comparés n'ont pas une valeur identique, l'unification échoue et renvoie "faux", ce qui signifie qu'il y a une erreur d'accord. Dans ce cas, le programme indique dans le fichier de sortie (en HTML) le *chunk* concerné, la phrase dans laquelle il se situe, un commentaire sur le type d'erreur détectée et une suggestion de correction.

Une fois toutes les vérifications *intra-chunks* effectuées, le programme procède alors aux calculs d'unification entre les *chunks*. Il s'agit dans ce cas généralement de détection d'erreurs d'accord entre le sujet et le verbe. Ainsi, le verbe du *chunk* verbal principal doit avoir un attribut nombre de même valeur que l'attribut nombre du premier *chunk* nominal à gauche, celui-ci étant généralement sujet. De même, il doit y avoir unification entre les attributs genre et nombre du *chunk* nominal sujet et ceux d'un éventuel participe passé dans le *chunk* verbal. Lorsque le sujet est un pronom personnel, celui-ci étant inclus au *chunk* verbal, une erreur d'accord sera détectée lors des calculs d'unification *intra-chunks*.
En cas d'échec de l'unification, le même type d'indications que lors de la vérifications *intra-chunk* est retourné dans le fichier de sortie.

4.3 Les résultats des tests

Nous avons testé notre prototype sur les phrases que nous avons préalablement étiquetées manuellement et nous avons obtenus des résultats plutôt encourageants.
Les erreurs d'accord dans les syntagmes nominaux ont toutes été détectées, sans qu'il soit fait usage d'une base démesurée de règles de correction. Le correcteur libre An Gramadóir (AnGramadóir, WEB) par exemple possède une base d'environ 450 règles, utilisant des expressions

régulières et des macros complexes, pour la détection des erreurs d'accord dans les syntagmes nominaux. En utilisant les propriétés des *chunks* et l'unification, ces règles ne sont plus nécessaires. Pour détecter ce type d'erreurs, il suffit de calculer l'unification entre tous les constituants d'un *chunk*.
Les fautes d'accord entre les syntagmes nominaux sujet et leur verbe ou participe passé ont également été détectées correctement. Par exemple, si nous reprenons notre énoncé :
`"Les personnes en situation de test s'autosurveille énormément"`
l'erreur de personne sur le verbe a été signalée. De telles erreurs sont généralement difficilement détectables par la plupart des correcteurs, car elles sont souvent associées à des relations de dépendance distantes. Avec les propriétés de dépendance que les *chunks* entretiennent entre eux, ce type d'erreur peut non seulement être détecté, mais en plus détecté avec un nombre très réduit de règles simples.

Notre prototype ne détecte pour le moment que quelques types d'erreurs que nous venons de mentionner (accords dans les SN, accords sujet-verbe, accords sujet-participe passé). Il permet néanmoins de montrer que la combinaison *chunks*-unification est efficace pour détecter les erreurs d'accord. La détection se fait correctement et à moindre coût car elle ne nécessite plus la rédaction de très nombreuses règles.

5 Conclusion et perspectives

Les correcteurs grammaticaux libres que nous connaissons ont tous à peu près la même structure et se fondent sur un système de *pattern-matching*, que l'absence d'abstraction des patrons décrits dans les règles rend très rigide. Comme nous l'avons vu dans cet article, ceci limite de manière importante les performances des correcteurs. Ces derniers doivent disposer d'un nombre démesuré de règles (notamment pour les fautes d'accord), qui ne permettent cependant pas de détecter toutes les erreurs et sont également souvent source de fausses détections.
Les approches TAL mises en oeuvres n'étant pas adaptées à la problématique, nous nous sommes donc tournée vers d'autres approches : la segmentation en *chunks* et l'unification. Le fait de les combiner permet d'intégrer une abstraction et ainsi de simplifier les règles et d'en réduire considérablement le nombre. Les *chunks* constituent en effet des zones de calcul à l'intérieur desquelles les traits de tous les éléments doivent s'unifier.

Pour vérifier la faisabilité d'une telle combinaison *chunk*-unification, nous avons développé un prototype de calcul d'unification. En présence de phrases préalablement étiquetées en XML et segmentées en *chunks*, l'outil s'est révélé parfaitement capable de détecter les erreurs d'accord, aussi bien dans qu'entre les *chunks*. Cet outil nous a montré qu'il est donc possible, avec un nombre très réduit de règles simples, de détecter les erreurs d'accord quelle que soit la distance qui sépare les mots concernés, alors que ces détections ne sont actuellement pas toujours possibles et nécessitent énormément de règles, parfois très complexes.

Les bénéfices que nous avons mentionnés sont importants mais selon nous pas suffisants pour obtenir un outil capable de corriger efficacement le français. Il nous semble nécessaire de sortir d'une part de l'approche énumérative des règles, d'autre part de l'approche en couches traditionnelle, où le traitement du texte est décomposé en plusieurs étapes successives et où la détection des fautes est prisonnière d'un cercle vicieux (figure 2).

Nous envisageons donc une approche innovante de la correction grammaticale, que nous pourrions nommer "gauche-droite". Le principe est de construire simultanément l'analyse morpho-

syntaxique et la correction au fur et à mesure de la rédaction / lecture de la phrase, et de rechercher des incohérences grammaticales plutôt que d'énumérer exhaustivement toutes les erreurs possibles. Plus précisément, en se fondant sur le principe des latences (Tesnière, 1959) ou d'attentes réciproques (Lebarbé, 2002), construire la structure syntaxique dans l'ordre de lecture des *tokens*, valider les attentes réciproques (un déterminant attend un nom), construire les *chunks* et tester l'unification : l'échec de l'un des trois constitue une détection d'erreur de grammaire et donne son explication.

L'analyse de `*"Les premiers linguistes on donc d'abord écouté"` générerait une erreur sur le mot `"on"`, le syntagme nominal qui le précède ayant une latence droite pour un verbe ou une préposition.

```
"Les premiers linguistes on donc d'abord écouté"
[SN--------------------] erreur
```

Un message du type "un verbe est attendu" serait alors généré pour indiquer l'erreur à l'utilisateur.
Cette approche, sur laquelle nous travaillons dans le cadre de notre thèse, implique une reconsidération complète des formalismes existants (déclaration non plus des fautes, mais de ce qui est attendu) et du traitement.

Références

ABEILLÉ A. (1993). *Les nouvelles syntaxes. Grammaires d'unification et analyse du français*. Armand Colin.

ABNEY S. (1991). Parsing by chunks. In *Principle-based parsing : Computation and Psycholinguistics*, p. 257–278. Kluwer Academic Publishers.

ANGRAMADÓIR (WEB). http ://borel.slu.edu/gramadoir.

CHOMSKY N. (1979). *Structures syntaxiques*. Éditions Seuil.

COVAREC (1994). Corpus de variations orthographiques. Laboratoire LIDILEM, Université Stendhal-Grenoble 3.

LANGUAGETOOL (WEB). http ://www.languagetool.org/.

LEBARBÉ T. (2002). *Hiérarchie inclusive des unités linguistiques en analyse syntaxique coopérative ; le segment, unité intermédiaire entre chunk et phrase dans le traitement linguistique par système multi-agents.* PhD thesis, Université de Caen.

LUCCI V. & MILLET A. (1994). *L'orthographe de tous les jours, enquête sur les pratiques orthographiques des français*. Éditions Champion.

MILLER P. & TORIS T. (1990). *Formalisme pour le TALN*. Hermès.

NABER D. (2003). *A rule-based and grammar checker*. PhD thesis, Technische Fakultät, Universität Beilefeld.

OPENOFFICE.ORG (WEB). http ://fr.openoffice.org/.

SOUQUE A. (2007). *Conception et développement d'un formalisme de correction grammaticale automatique - Application au français*. Mémoire de master 2 recherche sciences de langage, Université Stendhal - Grenoble 3.

TESNIÈRE L. (1959). *Éléments de syntaxe structurale*. Klincksieck.

VERGNE J. & GIGUET E. (1998). Regards théoriques sur le "tagging". *Actes de la cinquième conférence Le Traitement Automatique des Langues Naturelles*.