HAL open science

Un analyseur de surface non déterministe pour le français

François Trouilleux

To cite this version:

François Trouilleux. Un analyseur de surface non déterministe pour le français. Traitement automatique des langues naturelles (TALN), Jun 2009, Senlis, France. http://www-lipn.univ-paris13.fr/taln09/paper/paper_TALN_6.html. hal-00403924

HAL Id: hal-00403924

https://hal.science/hal-00403924v1

Submitted on 14 Jul 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

TALN 2009, Senlis, 24–26 juin 2009

Un analyseur de surface non déterministe pour le français

François Trouilleux
LRL, Université Blaise-Pascal
http://www.univ-bpclermont.fr/LABOS/lrl/

Cet article a été publié dans les actes de la 16ème Conférence sur le Traitement Automatique des Langues Naturelles 2009. La publication originale est disponible sur le site http://www-lipn.univ-paris13.fr/taln09/

Résumé. Les analyseurs syntaxiques de surface à base de règles se caractérisent par un processus en deux temps : désambiguïsation lexicale, puis reconnaissance de patrons. Considérant que ces deux étapes introduisent une certaine redondance dans la description linguistique et une dilution des heuristiques dans les différents processus, nous proposons de définir un analyseur de surface qui fonctionne sur une entrée non désambiguïsée et produise l'ensemble des analyses possibles en termes de syntagmes noyau (*chunks*). L'analyseur, implanté avec NooJ, repose sur la définition de patrons étendus qui annotent des *séquences* de syntagmes noyau. Les résultats obtenus sur un corpus de développement d'environ 22 500 mots, avec un rappel proche de 100 %, montrent la faisabilité de l'approche et signalent quelques points d'ambiguïté à étudier plus particulièrement pour améliorer la précision.

Abstract. Rule-based chunkers are characterized by a two-tier process : part-of-speech disambiguation, and pattern matching. Considering that these two stages introduce some redundancy in the linguistic description and a dilution of heuristics over the different processes, we propose to define a chunker which parses a non-disambiguated input, and produces all possible analysis in terms of chunks. The parser, implemented with NooJ, relies on the definition of extended patterns, which annotate *sequences* of chunks. The results obtained on an approx. 22500 word corpus, with almost 100 % recall, demonstrate the feasability of the approach, and signal which ambiguities should be further studied in order to improve precision.

Mots-clés : Analyse syntaxique de surface, automates à états finis, déterminisme, désambiguïsation.

Keywords: Chunking, finite-state automata, determinism, disambiguation.

1 Introduction

Dans un récent article, M. Cori (2008) distingue deux grands types de méthodes en TAL. Le « TAL théorique » y est caractérisé par référence aux différents modèles de grammaires syntagmatiques, tandis qu'un logiciel de « TAL robuste » y est caractérisé par une série de critères: il traite du texte « tout venant », il produit toujours une analyse, il sélectionne l'analyse supposée la meilleure, et il se prête à des procédures d'évaluation quantitative de ses performances. L'article oppose par ailleurs TAL théorique et TAL robuste sur leurs objectifs d'analyse, en soulignant que le TAL robuste vise souvent une analyse syntaxique *partielle*, typiquement en syntagmes noyau (*chunks*).

En jouant sur l'un ou l'autre des critères identifiés par Cori pour le TAL robuste, on peut chercher de nouvelles frontières entre les deux pratiques. Le travail présenté dans cet article trouve son origine dans la volonté de faire varier le paramètre qui consiste à ne produire qu'une analyse en sélectionnant la meilleure. On a ainsi tenté de développer un analyseur de surface (*chunker*) pour le français, capable de traiter du texte tout venant, qui produise toujours une analyse, mais qui, pour chaque phrase du texte analysé, produise *un ensemble d'analyses possibles* en termes de syntagmes noyau, étant donné une description de chaque modèle de syntagme noyau.

Notre analyseur s'écarte de deux pratiques omniprésentes dans les analyseurs de surface: le déterminisme et l'incrémentalité. Nous justifierons ce choix en le mettant en perspective avec ces pratiques canoniques (alinéa 2.1), puis nous exposerons les problèmes posés par l'application directe de la reconnaissance de patrons sur une entrée ambigüe (alinéa 2.2). Notre analyseur, implanté avec NooJ, propose une solution à ces problèmes grâce à l'utilisation de patrons étendus annotant des séquences de syntagmes noyau (alinéa 3). Les résultats obtenus sur un corpus de développement de 22 500 mots sont présentés dans l'alinéa 4.

2 Problématique

Les analyseurs syntaxiques de surface robustes à base de règles [1] se caractérisent par le fait qu'ils sont *déterministes*, dans le sens où ils ne produisent qu'une seule analyse, et mettent en œuvre un processus d'analyse *incrémental*. Par ailleurs, dans ces systèmes, les règles sont en général appliquées selon le mode de la *reconnaissance de patrons*.

2.1 Déterminisme et incrémentalité

De manière générale, l'argument en faveur du déterminisme de l'analyseur est utilitaire. Ainsi, Hindle (1994, p. 107) considère qu'« une sortie unique facilite grandement l'utilisation d'un analyseur. Si un analyseur devait produire plusieurs analyses, pour la plupart des tâches d'analyse textuelle il serait nécessaire de choisir parmi les alternatives avant d'utiliser les résultats ». Hindle précise que le déterminisme n'est pas une nécessité ; notre projet est de l'éviter. La conséquence de ce choix est que l'intérêt de notre analyseur ne résidera pas dans ses apports à une chaine de traitements globale, mais dans le fait qu'il fournira une sortie propre à mettre en lumière les difficultés de l'analyse linguistique. Notre analyseur se veut un outil d'étude linguistique, plutôt qu'un composant d'une application.

[1] On laisse de côté dans cet article les approches à base de modèles statistiques.

Les systèmes de TAL robustes sont évalués de façon uniquement quantitative, ce qui conduit les linguistes qui les définissent à faire des choix motivés par des considérations statistiques. Ainsi la plupart de ces systèmes échoueront à analyser correctement la phrase suivante, parce que statistiquement un *des* en début de phrase est le plus souvent un déterminant:

(1) Des pétunias en bordure de l'allée montait par bouffée une odeur de cassis.

Une de nos motivations est de pouvoir repérer dans des corpus des configurations telles que celle qu'on observe dans cet exemple.

Les analyseurs syntaxiques de surface à base de règles se caractérisent par un processus en deux temps: désambiguïsation lexicale, puis applications de règles. On pourrait multiplier les exemples de tels analyseurs: (Hindle, 1994), (Kinyon, 2001), (Aït-Mokhtar *et al.*, 2002), (Bourigault *et al.*, 2005), etc. On notera également que le Natural Language Toolkit, dont on peut penser que, en tant que « boite à outils », il met en œuvre des approches paradigmatiques, définit un analyseur syntaxique de surface en ces termes:

> A chunker finds contiguous, non-overlapping spans of related tokens and groups them together into chunks. Chunkers often operate on tagged texts, and use the tags to make chunking decisions. (Bird *et al.*, 2009, alinéa 7.2)

De fait, le système de segmentation en syntagmes noyau du NLTK fonctionne par application d'expressions régulières sur du texte étiqueté de façon non ambigüe.

L'utilisation d'un tagger pour désambiguïser les unités lexicales est avant tout, comme le déterminisme, utilitaire: les taggers, qu'ils soient à base de règles ou statistiques, sont en général extrêmement rapides et permettent de réduire sensiblement la complexité du processus d'analyse global. Ainsi, (Roussanaly *et al.*, 2005) justifient l'inclusion d'un tagger dans la chaine de traitement mise en œuvre pour leur participation à la campagne EASY par les « temps d'analyse rédhibitoires » provoqués par « des ambiguïtés multiples ».

Une alternative à la désambiguïsation lexicale est proposée par (Vergne, 1999), dont l'analyseur fonctionne sur des unités lexicales ayant explicitement une catégorie *par défaut*. L'objectif est également de réduire les possibilités combinatoires pour la suite de l'analyse.

Le principal inconvénient d'un processus d'analyse en deux temps est bien connu: les erreurs d'analyse au premier niveau vont affecter le second. Or il est généralement admis que l'information utilisée par les taggers est *trop limitée* pour permettre une analyse toujours correcte. On abandonne ainsi le traitement de certaines ambiguïtés, celle du *des* de (1), par exemple. Cela étant, nous voyons dans l'utilisation d'un tagger deux autres inconvénients.

Le premier est que dans l'approche à deux étapes, on a une description linguistique *redondante*, dans le sens où on dira souvent deux fois la même chose [2]. Par exemple, au premier niveau on aura une règle ou une statistique qui dira qu'un clitique est plus probable qu'un article ou un nom après la forme *ne*, et au deuxième niveau on aura un automate ou une expression régulière spécifiant des syntagmes NV qui intègrera la même information.

Le second inconvénient est qu'en répartissant le processus d'analyse sur deux niveaux, on aboutit à une *dilution des heuristiques*. Des choix statistiques (plus ou moins explicites) sont faits aux deux niveaux et on ne sait plus forcément dans l'évaluation globale du système d'où viennent

[2] Et non dans le sens où deux informations différentes permettraient d'aboutir à une même conclusion, ce qui serait positif.

les erreurs. Nous pensons qu'un analyseur produisant une segmentation en syntagmes noyau ambigüe permettra de spécifier par la suite et d'évaluer clairement les heuristiques à mettre en œuvre pour désambiguïser cette sortie. L'analyseur que nous proposons désambiguïsera certaines formes, mais seulement sur la base de la structure interne des syntagmes noyau.

2.2 Limites de la reconnaissance de patrons

Bon nombre d'analyseurs robustes tirent leur robustesse du fait que les règles qu'ils implantent sont appliquées selon le mode de la « reconnaissance de patrons ». L'algorithme qui parcourt la chaine d'entrée ne cherche pas à associer une analyse axiomatique à l'ensemble de la chaine, mais se contente de reconnaitre que des fragments de cette chaine satisfont aux règles spécifiées. Cet aspect de l'algorithme est essentiel parce que c'est lui qui permet de passer outre des mots inconnus, toujours présents dans les corpus. NooJ (Silberztein, 2004), logiciel que nous utilisons, permet de déclarer des automates servant à la reconnaissance de patrons. Trois modes de reconnaissance des patrons définis par une grammaire sont disponibles: la plus courte, la plus longue ou toutes les correspondances.

Considérons en premier lieu un modèle simple qu'on notera ainsi:

```
Main = :GN + :GP + :NV + :PV + :GR + :GA ;
```

où les suites `:GN`, `:GP`, etc. identifient chacune un automate à états finis définissant respectivement l'ensemble des GN, GP, etc. possibles [3].

Avec une telle grammaire, les deux premiers modes sont clairement inadaptés à notre projet. Ils sont déterministes, et il est facile de trouver des exemples où ils échoueront. En voici deux, pour la sélection de la plus longue et de la plus courte correspondance, respectivement:

```
(2) a. <GN>La petite ferme</GN> <GN>les yeux</GN>.
    b. <GN>La petite</GN> <GN>fille</GN> le <NV>regarde</NV>.
```

Notons que ces deux exemples démontrent qu'on ne peut appliquer une reconnaissance de patrons simples de façon déterministe sur une entrée ambigüe. Cette observation jette une nouvelle lumière sur le processus mis en œuvre dans les analyseurs robustes à base d'expressions régulières ou d'automates finis: désambiguïser le texte avant d'appliquer la reconnaissance de patrons n'est pas uniquement motivé par une question d'efficacité, c'est une *nécessité*.

Reste le mode « toutes les correspondances ». Il présente l'inconvénient de surgénérer exagérément. Deux cas de figure méritent en particulier l'attention.

Le premier est illustré par l'exemple suivant, où une phrase composée d'un seul syntagme NV, donne lieu à quatre analyses.

```
(3) <NV>Il <NV>le <NV>lui <NV>donne</NV></NV></NV></NV>.
```

Le propre de la reconnaissance de patrons est de pouvoir ignorer des mots ; ici on voudrait pouvoir dire au système de ne pas laisser hors d'un syntagme les mots qu'il peut intégrer.

Le second peut être illustré par la séquence suivante:

```
(4) <GP>Pour cent francs</GP> <GP>par an</GP>
```

Pour les cinq unités lexicales de cette séquence, notre dictionnaire a les catégories suivantes:

```
Pour,PREP|N  cent,NUM  francs,ADJ|N  par,PREP|N  an,N
```

[3] On utilise les catégories de la campagne d'évaluation EASY (cf. (Gendner & Vilnat, 2004)).

Demander la reconnaissance de toutes les correspondances possibles avec cette entrée ambigüe aboutira à de multiples analyses, parmi lesquelles on trouvera par exemple:

(5) `<GN>Pour</GN> <GN>cent</GN> <GN>francs</GN>`
`<GN>par</GN> <GN>an</GN>`

Ici, il semble raisonnable de privilégier les lectures de *pour* et *par* comme prépositions, et leur association avec le nom qui les suit. L'analyseur que nous présentons par la suite permettra d'exprimer cela.

3 Un analyseur de surface non déterministe avec NooJ

Nous avons montré que la reconnaissance de patrons définissant des syntagmes noyau ne pouvait s'appliquer directement sur une entrée ambigüe. Néanmoins, nous avons pu implanter avec NooJ un analyseur qui satisfasse notre objectif de non-déterminisme et qui fonctionne en mode *reconnaissance de patrons*, donc avec une certaine robustesse. La modélisation s'appuie d'une part sur les caractéristiques fondamentales des syntagmes noyau en français, d'autre part sur la possibilité de définir dans NooJ des patrons qui identifient des *séquences* de syntagmes noyau.

3.1 Caractéristiques des syntagmes noyau

On s'appuie dans ce travail sur la définition des syntagmes noyau de (Abney, 1991). Si l'on prend cette définition [4] au pied de la lettre, il faut reconnaitre deux modèles de syntagmes noyau possibles:

1. un mot lexical seul,
2. et une suite de un ou plusieurs mots lexicaux, précédée et/ou suivie de un ou plusieurs mots fonctionnels.

Dans la pratique, on s'écarte un peu de cette définition en considérant aussi comme un syntagme nominal noyau une suite comme *belle marquise* ou *pauvre petit garçon*, parce qu'on y reconnait des séquences qui intègreraient un même syntagme nominal noyau si elles étaient précédées d'un déterminant.

Muni de cette définition stricte, on peut faire sur le français deux observations intéressantes:

1. un mot fonctionnel à droite du syntagme doit en principe être lié au mot lexical qui le précède par un tiret (*donne-le-moi, ce garçon-là*) [5],
2. il y a seulement deux cas où un syntagme noyau contient plusieurs mots lexicaux:
   (a) un adverbe dans un syntagme verbal à l'infinitif (ex. *pour mieux labourer*),
   (b) un GN ou GP avec adjectif antéposé au noyau (ex. *avec la petite fille*)

Une conséquence qu'on peut tirer de la première observation est que, s'il n'est pas rattaché par un tiret à sa gauche, un mot fonctionnel ouvre un syntagme noyau ou s'intègre dans un syntagme noyau ouvert par un autre mot fonctionnel. Si on veut privilégier la lecture des mots

[4] I define chunks in terms of *major heads*. Major heads are all content words except those that appear between a function word f and the content word that f selects. For example, *proud* is a major head in *a man proud of his son*, but *proud* is not a major head in *the proud man*, because it appears between the function word *the* and the content word *man* selected by *the*.

[5] Cela vaut si on analyse des mots comme *durant* ou *excepté* comme des participes, ce que nous faisons.

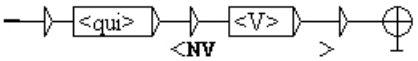
fonctionnels comme tels (cf. l'exemple 5), une stratégie de type « reconnaissance de la plus longue chaine » sera adaptée: par exemple, étant donné *avec la fille*, on évitera de fermer un syntagme après *avec* ou *la*. Notons que cette propriété des mots fonctionnels est à la base de l'algorithme de (Kinyon, 2001).

En ce qui concerne la deuxième observation, nous manquons de données pour commenter le premier cas, et nous nous concentrerons donc par la suite sur le second, beaucoup plus fréquent. Pour l'heure, le point important est que la présence de plusieurs mots lexicaux est susceptible de conduire à une ambiguïté de segmentation au niveau de la limite droite du syntagme, avant ou après un mot lexical, comme *brise* dans l'exemple classique *La petite brise la glace*.

3.2 Des patrons pour des séquences de syntagmes noyau

NooJ permet de déclarer des automates servant à la reconnaissance de patrons. Un apport essentiel de NooJ dans le système d'analyse que nous présentons ici est qu'il distingue les patrons *spécifiés* par un automate et le ou les patrons *annotés* par cet automate. Les patrons spécifiés sont ceux qui étiquettent un chemin *complet* de l'automate (de l'état initial à l'état final), un patron annoté est un sous-chemin commençant par un état étiqueté `<X` et se terminant par un état étiqueté `>`, avec `X` la catégorie qu'on souhaite donner au syntagme [6].

Le petit automate ci-contre illustre cette possibilité: il spécifie le patron `<qui> <V>`, mais annote seulement le patron `<V>` comme un syntagme de catégorie NV.



Notre grammaire définit ainsi un automate qui spécifie des patrons annotant *plusieurs* syntagmes noyau consécutifs, en intégrant les ambiguïtés de segmentation possibles. La figure 1 illustre le principe de construction de cette grammaire. On y voit l'un des graphes principaux de la grammaire, qui non seulement annote comme PV ou NV des syntagmes, mais annote également d'autres syntagmes à droite de ces syntagmes par les graphes PP, GNb, GR, NVb, et GA-PP. Ces cinq graphes annotent eux-mêmes selon les cas un à trois autres syntagmes:

- PP annote un ou deux syntagmes: un participe passé, optionnellement précédé d'un adverbe (GR), ou d'une forme du pronom *tout* (GN),
- GNb annote un seul syntagme: un nom qui n'est pas ambigu avec un mot fonctionnel, optionnellement précédé d'un adjectif préférentiellement antéposé, lui-même optionnellement précédé d'un adverbe,
- NVb annote un syntagme NV composé d'un verbe, suivi optionnellement de clitiques, plus, optionnellement et si le verbe est un auxiliaire, des syntagmes du graphe PP,
- GR annote un syntagme composé d'un seul adverbe,
- GA-PP annote un syntagme composé d'un seul adjectif ou participe passé.

Le graphe PP a pour fonction de définir la relation spécifique entre auxiliaire et participe passé. Les quatre autres graphes ont pour fonction de définir les syntagmes ne commençant pas par un mot fonctionnel, qu'on appellera « syntagmes B ». C'est avant ou après les mots composants ces syntagmes qu'on peut observer une ambiguïté de segmentation du type évoqué dans notre deuxième observation ci-dessus (fin de l'alinéa 3.1).

[6] En fait, on note `<E>/<X` et `<E>/>` pour indiquer qu'on a d'une part la chaine vide (`<E>`) dans la chaine d'entrée, d'autre part le parenthésage d'un syntagme en sortie.

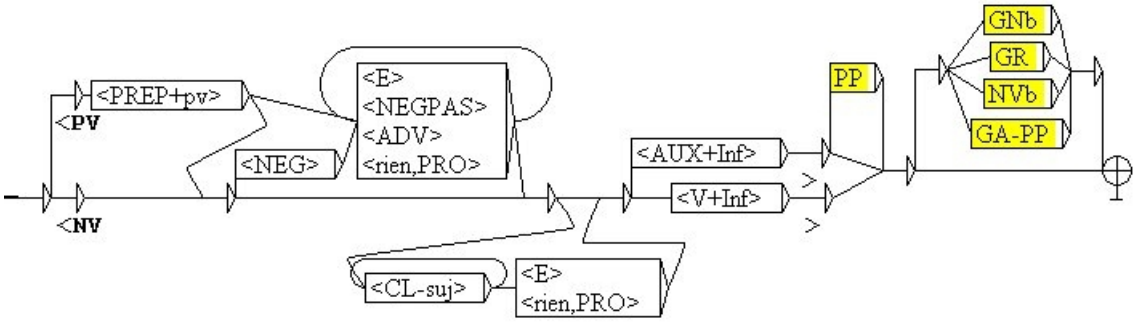


FIG. 1 – Graphe annotant les PV et NV infinitif + des syntagmes unaires optionnels à droite

Si on fait abstraction du graphe PP, la grammaire spécifie des patrons qui annotent un syntagme noyau commençant par un mot fonctionnel ou non (un « syntagme A »), suivi optionnellement d'un syntagme B. Ce modèle est appliqué en mode « reconnaissance de la plus longue chaine » ; la grammaire définit les transitions possibles entre syntagmes A et syntagmes B, en faisant en sorte que les analyses qu'on veut privilégier correspondent à un chemin complet plus long, tandis que celles pour lesquelles on veut des annotations ambigües soient de même longueur.

Après un groupe verbal (NV ou PV) ou adverbial (GR), la grammaire autorise une transition vers tout type de syntagme [7]. Pour les autres types de syntagmes (GA, GN et GP), les conditions qui régissent les transitions sont les suivantes:

1. Un syntagme A ayant pour noyau un mot ambigu avec un mot fonctionnel [8] n'admet pas de transition vers un syntagme B ; on privilégie ainsi l'interprétation de ces mots comme fonctionnels si on peut construire un syntagme A avec eux. Par exemple, *la*, nom potentiel (la note de musique), ne peut être noyau d'un GP dans *de la porte*.
2. Pour les syntagmes à noyau nominal ou adjectival, qui sont ceux où l'ambiguïté de segmentation à droite est possible, les transitions sont résumées dans le tableau 1:
   (a) pas de transition vers un GNb après un GA ou après un GN ou GP ayant pour noyau un adjectif (`<ADJ>`) ou une forme du pronom *tout*, (p.ex. on exclut `<GN>la petite</GN> <GN>brise</GN>` ou `<GN>tout</GN> <GN>homme</GN>`)
   (b) si un GN ou GP n'a pas d'épithète antéposée [9] et a pour noyau un nom non ambigu avec un mot fonctionnel, toutes les transitions sont permises,
   (c) si un GN ou GP contient une épithète antéposée à un noyau nominal, l'automate n'enregistre pas de transition vers un GNb ; on fait ainsi en sorte qu'un tel syntagme ait la même longueur qu'une séquence syntagme A + syntagme B. Par exemple,

[7] Les catégories utilisées ici sont celles de (Gendner & Vilnat, 2004), modulo les adaptations suivantes:
– on traite les NV avec participe passé (PP) comme les adjectifs ; les deux catégories ont des distributions proches et les NV avec participe passé sont toujours unaires ;
– NVb = groupe verbal sans mot fonctionnel à gauche ;
– GNb = groupe nominal sans mot fonctionnel ni numéral ;
– GN = groupe nominal avec au moins un mot fonctionnel ou numéral.
– GP0 et GPA: voir la suite du texte.

[8] Information codée dans le lexique par l'ajout d'un trait `+fble` pour ces mots. La contrainte n'a pas été implantée pour les verbes mais s'applique aux adverbes (GR).

[9] Par *épithète antéposée*, on veut dire tout type de groupe adjectival antéposé au noyau du syntagme nominal, par exemple *très bonne* dans *une très bonne idée*.

		NVb	GNb	GA-PP	GR
GA-PP		+	–	+	+
GN – GP	– <tout,PRO>	+	–	+	+
	– <ADJ>	+	–	+	+
	– <N-fble>	+	+	+	+
	– *épithète* <N>	–	–	–	–
GP0		–	–	–	–
GPA		–	–	–	–

TAB. 1 – Transitions entre syntagmes A à noyau nominal ou adjectival (lignes) et syntagmes B (colonnes).

<GN>la petite brise</GN> (sans transition vers un syntagme B) a la même longueur que <GN>la petite</GN> <NV>brise</NV> (synt. A + synt. B).

3. On distingue par ailleurs deux catégories de syntagmes spécifiques:
   (a) GP0 = groupes prépositionnels composés d'une seule préposition, éventuellement précédée de *de*, par exemple il était <GP0>derrière</GP0> ;
   ceux-ci seraient analysés comme GR dans EASY ;
   (b) GPA = construction du type <GN>un fils</GN> <GPA>de malade</GPA>, analysée dans EASY comme <GN>un fils</GN> de <GA>malade</GA>.

   On n'enregistre aucune transition après ces syntagmes, c'est-à-dire qu'on ne les admet que si on ne peut pas interpréter la séquence comme le début d'un GP plus long.

A cela s'ajoute un cas particulier: après une forme du pronom *tout*, on admet une transition supplémentaire vers un syntagme NV composé d'un clitique *le* et d'un verbe. Par ailleurs, pour tout GN ou GP ayant un noyau autre que adjectif ou nom commun (adverbe, nom propre, pronom, numéral ou participe passé), l'automate n'enregistre pas de transition vers un syntagme B.

Les transitions ont été établies expérimentalement sur un corpus décrit ci-après. Pour résumer, on peut dire qu'elles ont essentiellement deux fonctions: privilégier la construction des syntagmes associant les mots fonctionnels à un mot lexical (cf. les exemples 3, 4 et 5 ci-dessus) et gérer les ambiguïtés de segmentation à droite (cf. l'exemple 2).

4 Analyse de corpus

Pour définir cet analyseur, nous nous sommes appuyés sur un corpus de 22 557 mots (*word forms* de NooJ), en quatre parties: *Un cœur simple* de Flaubert (11 581 mots, 51 %), 157 exemples extraits de l'entrée *de* du TLF (5 309 mots, 23,5 %), la transcription de la cassette de Jean-Claude Mery (3 449 mots, 15,5 %), et neuf articles du journal *La Tribune* (2 218 mots, 10 %). Le tableau 2 donne les mesures de rappel et précision obtenues sur ce corpus. Il ne s'agit en aucun cas d'une évaluation globale du système sur texte inconnu telle qu'on la pratique habituellement, mais simplement d'observations faites dans une situation contrôlée sur le corpus de développement. En particulier, on s'est donné un dictionnaire idéal en ajoutant les noms propres et mots inconnus du corpus, et, comme on le voit sur le tableau, on s'est attaché à obtenir un rappel le plus proche possible de 100 %. L'idée, on le rappelle, est développer un outil qui permettra d'observer les ambiguïtés au niveau des syntagmes noyau.

	total	GA-PP	GN	GNb	GP	GP0	GPA	GR	NV	PV
possible	11144	1205	2716	175	3064	16	9	877	2754	328
effectif	14744	1334	4053	1196	3568	17	66	1140	3037	333
correct	11140	1205	2713	174	3064	16	9	877	2754	328
rappel	99,96	100	99,89	99,43	100	100	100	100	100	100
précision	75,56	90,33	66,94	14,55	85,87	94,12	13,64	76,93	90,68	98,50

TAB. 2 – Mesures observées sur le corpus de développement.

On relève quatre erreurs de rappel, toutes au niveau des GN-GNb. Une est anecdotique (un titre en anglais), les trois autres se manifestent dans les deux exemples suivants:

(6) En voilà `<GN>`une Mme Lehoussais`</GN>`, qui au lieu de prendre un jeune homme...

(7) `<GP>`De tels arguments`</GP>` paraissent étonnants

En (6) le système identifie un GN là où il en faudrait deux. Nous n'avons pas travaillé sur cette erreur parce qu'un syntagme composé d'un indéfini suivi d'un nom propre est possible (p.ex. *un Picasso*), et surtout, on a là une phrase qui à l'oral serait dite avec une intonation propre à séparer *une* et *Mme Lehoussais*, si bien qu'on serait en droit d'attendre une virgule à cet endroit. On atteint là, nous semble-t-il, le point de conflit entre exemple attesté et jugement de bonne formation. Cela étant ce cas est le seul où l'association d'un mot fonctionnel (*une*) à un mot lexicale est une erreur. La stratégie consistant à privilégier la construction des syntagmes associant les mots fonctionnels est donc globalement bonne [10].

En (7), le système identifie un GP alors qu'on aurait voulu une analyse ambigüe GN-GP. La source de cette erreur est l'ambiguïté de *tels*, DET ou ADJ. L'analyse de la séquence comme GN est bien prévue, mais comme il y a un adjectif antéposé, il n'y a pas à sa droite de transition vers un syntagme B, alors que l'analyse GP peut se faire avec une séquence PREP + DET + N, qui admet une transition vers un syntagme B (ici le NV *paraissent*), et est donc privilégiée. La construction *de* `<ADJ+pl> <N+pl>`, en ce qu'elle exige la combinaison de *de* et l'épithète pour former un GN, pourrait peut-être donner lieu à une exception.

Le chiffre de la précision donne une idée du résultat qu'on peut obtenir avec une information limitée à la seule constitution interne des syntagmes, à savoir un bruit assez important. La place manque ici pour analyser les erreurs en détail, mais on peut déjà noter deux causes principales:

- l'ambiguïté des syntagmes commençant par *des, de* ou *du* (GN, ou GP, ou GPA), responsable d'environ 38,5% du bruit,
- l'ambiguïté adjectif/nom, assez systématique dans le dictionnaire utilisé: on trouve 547 GA analysé comme GNb, et 238 analyses scindant un syntagme nominal avec épithète en deux, de type `<GN>un jeune</GN> <GNb>homme</GNb>`, avec *jeune* comme nom. Ces dernières erreurs produisant deux erreurs de précision, l'ambiguïté adjectif/nom est responsable de plus de 28% du bruit.

Signalons également que la faible précision obtenue pour les GR est due principalement à l'ambiguïté de *comme* et des interrogatifs, la grammaire ne traitant pas les conjonctions et autres introducteurs de propositions.

[10] Les 100% de rappel obtenus pour GP0 en témoignent également. On notera que la seule erreur observée pour les GP0 l'est dans un cas limite, où le découpage en syntagmes noyau sans enchâssement n'est plus possible: *avec, quand même, un problème* (cf. (Gendner & Vilnat, 2004, alinéa B.3)).

5 Conclusion

Le travail que nous avons présenté ici a mis en lumière la difficulté de la reconnaissance de patrons sur une entrée ambigüe, tout en proposant une voie d'étude alternative à l'approche paradigmatique déterministe et incrémentale de l'analyse de surface. On obtient un outil pour l'analyse linguistique dont on espère qu'il permettra de poser un regard neuf sur l'ambiguïté catégorielle en plaçant l'analyse au niveau des syntagmes noyau plutôt qu'au niveau des unités lexicales.

Au-delà, ce travail suggère aussi une piste d'amélioration des logiciels. Nous avons vu que la reconnaissance de la plus longue chaine était adaptée pour les mots fonctionnels (cf. les exemples 3 et 4), tandis que la reconnaissance de toutes les correspondances serait souhaitable au niveau des mots lexicaux (cf. *la petite brise la glace*). On pourrait alors imaginer un nouveau mode de reconnaissance de patrons, hybride, caractérisable comme la reconnaissance de toutes les correspondances *en privilégiant le rattachement des mots fonctionnels*. Ces derniers pouvant être spécifiés déclarativement comme une liste de catégories particulières, on aurait alors un algorithme qui prendrait mieux en compte les propriétés de l'objet qu'il vise à analyser.

Références

ABNEY S. (1991). Parsing by chunks. In R. BERWICK, S. ABNEY & C. TENNY, Eds., *Principle-Based Parsing*. Dordrecht: Kluwer Academic Publishers.

AÏT-MOKHTAR S., CHANOD J.-P. & ROUX C. (2002). Robustness beyond shallowness: incremental dependency parsing. *Natural Language Engineering*, **8**, 121–144.

BIRD S., KLEIN E. & LOPER E. (2009). *Analyzing Text with Python and the Natural Language Toolkit*. http://www.nltk.org/book.

BOURIGAULT D., FABRE C., FRÉROT C., JACQUES M.-P. & OZDOWSKA S. (2005). Syntex, analyseur syntaxique de corpus. In (Jardino, 2005).

CORI M. (2008). Des méthodes de traitement automatique aux linguistiques fondées sur les corpus. *Langages*, **171**.

GENDNER V. & VILNAT A. (2004). Les annotations syntaxiques de référence PEAS. `http://www.limsi.fr/Recherche/CORVAL/easy/`.

HINDLE D. (1994). A parser for text corpora. In B. ATKINS & A. ZAMPOLLI, Eds., *Computational Approaches to the Lexicon*, p. 103–151. Oxford: Clarendon Press.

M. JARDINO, Ed. (2005). *Actes de TALN 2005 (Traitement automatique des langues naturelles)*, Dourdan. ATALA, LIMSI.

KINYON A. (2001). A language-independent shallow-parser compiler. In *ACL*, p. 322–329.

ROUSSANALY A., CRABBÉ B. & PERRIN J. (2005). Premier bilan de la participation du LORIA à la campagne d'évaluation EASY. In (Jardino, 2005), p. 49–52.

SILBERZTEIN M. (2004). Nooj : an oriented object approach. In J. ROYAUTÉ & M. SILBERZTEIN, Eds., *INTEX pour la Linguistique et le Traitement Automatique des Langues*. Presses Universitaires de Franche-Comté.

VERGNE J. (1999). Etude et modélisation de la syntaxe des langues à l'aide de l'ordinateur. Analyse syntaxique automatique non combinatoire. Dossier d'habilitation à diriger des recherches. Université de Caen.