L'apport du faisceau dans l'analyse syntaxique en dépendances par transitions : études de cas avec l'analyseur Talismane

Assaf Urieli et Ludovic Tanguy
(1) CLLE-ERSS : CNRS & Université de Toulouse 2
assaf.urieli@univ-tlse2.fr, ludovic.tanguy@univ-tlse2.fr

RÉSUMÉ

L'analyse syntaxique (ou parsing) en dépendances par transitions se fait souvent de façon déterministe, où chaque étape du parsing propose une seule solution comme entrée de l'étape suivante. Il en va de même pour la chaîne complète d'analyse qui transforme un texte brut en graphe de dépendances, généralement décomposé en quatre modules (segmentation en phrases, en mots, étiquetage et parsing) : chaque module ne fournit qu'une seule solution au module suivant. On sait cependant que certaines ambiguïtés ne peuvent pas être levées sans prendre en considération le niveau supérieur. Dans cet article, nous présentons l'analyseur Talismane, outil libre et complet d'analyse syntaxique probabiliste du français, et nous étudions plus précisément l'apport d'une recherche par faisceau (*beam search*) à l'analyse syntaxique. Les résultats nous permettent à la fois de dégager la taille de faisceau la plus adaptée (qui permet d'atteindre un score de 88,5 % d'exactitude, légèrement supérieur aux outils comparables), ainsi que les meilleures stratégies concernant sa propagation.

ABSTRACT

APPLYING A BEAM SEARCH TO TRANSITION-BASED DEPENDENCY PARSING: A CASE STUDY FOR FRENCH WITH THE TALISMANE SUITE

Transition-based dependency parsing often uses deterministic techniques, where each parse step provides a single solution as the input to the next step. The same is true for the entire analysis chain which transforms raw text into a dependency graph, generally composed of four modules (sentence detection, tokenising, pos-tagging and parsing): each module provides only a single solution to the following module. However, some ambiguities cannot be resolved without taking the next level into consideration. In this article, we present Talismane, an open-source suite of tools providing a complete statistical parser of French. More specifically, we study the contribution of a beam search to syntax parsing. Our analysis allows us to conclude on the most appropriate beam width (enabling us to attain an accuracy of 88.5%, slightly higher than comparable tools), and on the best strategies concerning beam propagation from one level of analysis to the next.

MOTS-CLÉS : Analyse syntaxique en dépendances, ambiguïtés, évaluation, beam search
KEYWORDS: Dependency parsing, ambiguities, evaluation, beam search

1 Introduction

L'analyse syntaxique par dépendances s'inspire de l'œuvre de Tesnières (1959), et connaît un très grand engouement pour le développement d'analyseurs syntaxiques automatiques. Les avantages les plus connus sont, sur le plan linguistique, la possibilité de créer des dépendances croisées (arbres non projectifs) et l'expression efficace des structures argumentales des verbes. Sur le plan informatique, ce mode de représentation se prête très

facilement aux méthodes d'apprentissage automatique supervisé, puisque la détection d'un lien de dépendance entre deux mots et l'étiquetage de ce lien par une relation syntaxique peuvent se ramener à des opérations de classification.

Il existe deux principales techniques pour l'analyse syntaxique statistique en dépendances : l'analyse par transitions (Nivre, 2008) et l'analyse par graphes (McDonald, 2006). L'analyse par transitions présente l'intérêt d'une complexité de calcul linéaire, en transformant le problème d'analyse de syntaxe en un algorithme de type *Shift-Reduce*. Au cours de tests effectués par Candito et al (2010) et McDonald et Nivre (2007), il a été démontré que, comparée à l'analyse par graphes, l'analyse par transitions a des performances dégradées pour une distance de rattachement supérieure à deux mots. Une façon de corriger cette dégradation est d'y introduire une recherche par faisceau (*beam search*, cf. section 3.1). Cette méthode a déjà été appliquée par Sagae et Lavie (2006), Johanssen et Nugues (2006) et Johanssen et Nugues (2007), avec des résultats prometteurs pour une dizaine de langues, mais parmi lesquelles le français ne figure malheureusement pas.

Dans cet article, nous présentons tout d'abord un nouvel analyseur syntaxique en dépendances, Talismane (section 2), qui implémente de nouvelles fonctionnalités au niveau du faisceau et une syntaxe très expressive pour décrire les informations utilisées pour l'analyse. Cet outil est disponible librement et est directement opérationnel pour le français.

Dans la section 3, nous nous intéressons au mécanisme de la recherche par faisceau, à la fois au niveau quantitatif et qualitatif. Nous apportons des précisions sur la façon d'appliquer le faisceau à des problèmes où la comparaison des solutions intermédiaires n'est pas triviale.

Dans la section 4, nous testons l'hypothèse selon laquelle, si on propage le faisceau à travers les différents modules de l'analyse, un module de niveau plus élevé peut corriger les erreurs d'un module de niveau plus bas. Plus précisément, nous nous intéressons aux questions suivantes : le parseur est-il capable de corriger des erreurs de segmentation en mots (notamment en ce qui concerne l'identification des locutions) et des erreurs d'étiquetage morphosyntaxique ?

Dans la section 5, nous présentons une comparaison avec d'autres études similaires, et notamment une mesure des performances globales de Talismane.

2 L'analyseur Talismane

L'outil Talismane[1] est un analyseur syntaxique développé par Assaf Urieli dans le cadre de sa thèse au sein du laboratoire CLLE-ERSS, sous la direction de Ludovic Tanguy. Il est écrit intégralement en Java : il fonctionne donc sur tous les systèmes d'exploitation et est facilement intégrable à d'autres applications.

Pour passer d'un texte brut à un réseau de dépendances syntaxiques, Talismane utilise une analyse en cascade avec quatre étapes classiques pour ce type de tâche : le découpage en phrases (non traité ici), la segmentation en mots, l'étiquetage (attribution d'une catégorie morphosyntaxique), et le parsing (repérage et étiquetage des dépendances syntaxiques entre les mots).

[1] Disponible sous licence GPL à cette adresse : `http://redac.univ-tlse2.fr/talismane`

La tâche de chacun des modules est définie comme un problème de classification, et résolue de façon statistique, en entraînant un modèle probabiliste sur un corpus annoté.

Chacun des modules est configurable à la fois au niveau des *traits* et des *règles*. Les traits sont les informations sur les configurations rencontrées dont dispose l'algorithme pour prendre chacune des décisions, alors que les règles sont des contraintes qui forcent (ou interdisent) des décisions locales.

Le modèle par défaut proposé par Talismane utilise des traits classiques pour chacune des opérations. Pour l'étiquetage, par exemple, sont calculés pour chaque mot des traits liés à sa forme, aux étiquettes indiquées dans un lexique de référence, aux catégories des mots qui l'entourent, etc. La syntaxe de définition des traits est suffisamment expressive pour définir des traits plus complexes, par exemple le fait que le mot précédent soit situé entre parenthèses.

Les règles, qui ne sont appliquées qu'au moment de l'analyse (et pas lors de l'apprentissage), permettent de remplacer ou de contraindre les réponses fournies par le classifieur probabiliste, quand un critère est rempli. Des règles définissables suivant une syntaxe souple permettent d'éviter des résultats aberrants (comme l'attribution d'une classe fermée à un mot inconnu du lexique, l'attribution de deux sujets à un verbe, etc.) soit de respecter des contraintes propres à un corpus spécifique (en attribuant une catégorie fixe à un mot donné, par exemple).

Pour le parsing, Talismane se base sur l'algorithme décrit par (Nivre 2008) avec certaines modifications pour rendre possible la recherche par faisceau. Nous avons testé deux algorithmes présentés par Nivre : l'algorithme « classique » et l'algorithme dit « *arc eager* ». Le deuxième algorithme a fourni de meilleurs résultats globaux, et est le seul utilisé pour les expérimentations présentées dans cet article.

2.1 Classifieurs

Les algorithmes de classification utilisables par chaque module sont interchangeables, et trois classifieurs différents sont disponibles dans Talismane : un classifieur par entropie maximale[2], basé sur (Ratnaparkhi, 1998), un SVM linéaire[3] (Ho et Lin, 2012), et un classifieur par perceptrons multicouches (Attardi et al, 2009). Nous avons comparé les résultats de ces trois classifieurs avec le même jeu de traits et en testant différentes configurations pour leurs paramètres spécifiques. Le classifieur par entropie maximale donne des résultats supérieurs où égaux à ceux du SVM linéaire, avec l'avantage d'un algorithme d'entraînement plus rapide et une interprétation plus aisée des coefficients de chaque paramètre. Nous avons donc opté pour cette option dans les expériences présentées ici ainsi que pour le comportement par défaut de Talismane, et ce pour chacun des quatre modules de la chaîne de traitement.

2.2 Corpus d'entraînement et ressources externes

Le corpus d'entraînement pour les modules de segmentation et d'étiquetage est le French

[2] http://opennlp.apache.org/
[3] http://liblinear.bwaldvogel.de/

Treebank (Abeillé et al, 2003). Pour la segmentation, nous avons retenu les mots composés des catégories fermées (déterminants, pronoms, prépositions et conjonctions) ainsi que les adverbes qui ne sont pas par ailleurs des syntagmes prépositionnels bien formés. Pour l'étiquetage, le jeu de tags utilisé est celui de Crabbé et Candito (2008). Pour le parsing, nous avons utilisé le French Treebank converti automatiquement en dépendances par Candito et al (2010). Nous avons retenu leur division en corpus d'apprentissage, de développement (*dev*, 10 % du total) et de test (10 % du total) pour pouvoir comparer nos résultats directement.

A la différence des autres études, et grâce à l'expressivité syntaxique de Talismane, nous avons utilisé un jeu de traits complexe et parfois spécifique au français. Du coup, notre système n'est pas directement applicable à d'autres langues sans la création d'un nouveau jeu de traits, qui serait construit sur la base de la connaissance des mécanismes de la langue, de la disponibilité de ressources lexicales ou sémantiques, et des spécificités des corpus d'entraînement et d'évaluation.

Nous faisons un usage massif, dans les traits, du lexique LEFFF (Sagot et al, 2006) à la fois au niveau du segmenteur, de l'étiqueteur et du parseur. Comme dans Denis et Sagot (2009), nous utilisons les catégories grammaticales du lexique LEFFF comme traits de l'étiqueteur, en y ajoutant quelques contraintes (surtout au niveau des classes fermées). La liste complète des traits utilisés pour construire le modèle proposé par défaut est consultable en ligne[4].

3 Le principe du faisceau dans Talismane

Nous présentons ici les détails techniques de la recherche par faisceau dans Talismane.

3.1 Fonctionnement général

Que ce soit pour la segmentation ou le parsing, un analyseur probabiliste doit envisager un très grand nombre de configurations possibles pour une même phrase, en considérant toutes les combinaisons de catégories que l'on peut affecter à chaque élément (caractère pour la segmentation, mot pour l'étiquetage, paire de mots ou relation de dépendance pour le parsing). Afin de trouver la séquence (de frontières de mots, d'étiquettes, ou de liens syntaxiques) la plus probable, le système doit comparer théoriquement un très grand nombre de cas possibles ; pour limiter cette explosion combinatoire seules les k configurations les plus probables sont considérées à chaque étape du calcul. Le faisceau (de largeur k) est donc la liste de ces configurations partielles. Un faisceau de grande largeur a donc plus de chances de trouver la meilleure configuration, mais consommera également plus de ressource, en nombre de traits à calculer et de comparaisons à effectuer.

Pour l'étiquetage par exemple, les mots sont traités dans l'ordre de la phrase, et à chaque étape du calcul le faisceau contient les k séquences d'étiquettes les plus probables. Un faisceau de largeur 1 devra alors attribuer définitivement la catégorie d'un mot au moment où celui-ci est traité (ce qui ne veut pas dire qu'il le fait indépendamment des mots qui le suivent, puisque ceux-ci sont pris en compte via des traits). Dans tous les cas, le faisceau contient, à la fin de l'analyse d'une phrase, les k sorties les plus probables pour cette phrase. Des exemples plus détaillés de ce mécanisme sont présentés en section 4.1.

[4] http://redac.univ-tlse2.fr/talismane/features

3.2 Spécificités du faisceau dans le parseur

Dans le parsing par transitions, la situation est nettement plus complexe. Une « configuration » (Nivre, 2008) est une structure qui contient une pile de mots partiellement traités, un *buffer* contenant les mots non encore traités, et un jeu de dépendances déjà générées. A cela on peut ajouter une liste de transitions qui ont permis d'arriver à cette configuration à partir de la configuration initiale. La liste des transitions possibles est un petit ensemble fermé. Par exemple, la transition « Shift » enlève le mot en tête du buffer et le met en tête de pile, sans créer de dépendance entre les deux. L'entraînement consiste donc à apprendre quelle transition il faut appliquer étant donné une configuration. La configuration est considérée comme terminale quand le *buffer* est vide.

Appliquer un faisceau au parseur n'est pas trivial, dans la mesure où il est difficile de comparer des configurations qui ont créé un nombre différent de dépendances dans un ordre différent. Sagae et Lavie (2006) ont utilisé une stratégie particulière qui implique un certain nombre de biais, et Johanssen et Nugues (2007) ne donnent pas de précisions sur la façon d'appliquer le faisceau. Nous avons fait le choix d'utiliser une moyenne harmonique des probabilités individuelles, afin d'éviter de privilégier le chemin le plus court à une solution, et de comparer entre elles les configurations ayant traité un même nombre de mots.

3.3 Impact de la largeur du faisceau sur les performances globales

Nous avons tout d'abord évalué différentes largeurs de faisceau à l'intérieur de chaque module, sans considérer leur enchaînement. Les mesures ont été faites sur le corpus de test du French Treebank (10% du corpus, soit 32000 mots) en fournissant en entrée à chaque module les données annotées qui s'y trouvent.

Pour le **segmenteur en mots**, vu qu'il n'y a pas de dépendances contextuelles entre les décisions locales à différents endroits de la phrase, la solution la plus probable localement reste toujours en tête de liste, si bien que la largeur de faisceau n'a aucun effet sur les performances. A ce stade, le faisceau sert uniquement à fournir plusieurs segmentations possibles aux modules suivants (voir section 4.1).

Pour l'**étiqueteur morphosyntaxique**, le faisceau apporte un gain non significatif. Sur le sous corpus « test », on passe d'une exactitude de 97,81 % pour un faisceau de largeur 1 à une exactitude de 97,83 % pour un faisceau de 20. On voit donc que, même sans recherche par faisceau, le module d'étiquetage de Talismane se situe au niveau actuellement atteint par d'autres outils pour le français (Denis et Sagot, 2009).

Pour le **parseur**, nous avons mesuré la f-mesure pour chaque étiquette de dépendance (« sujet », « objet », …) à différentes largeurs de faisceau. Pour cette f-mesure, on considère une réponse comme correcte uniquement si l'arc est correct (bon gouverneur) et bien étiqueté (bonne relation). La TABLE 1 donne, pour le sous corpus « test », les f-mesures de certaines étiquettes. Les f-mesures pour l'ensemble des relations syntaxiques augmentent avec la largeur du faisceau, avec une relative stabilité à partir du faisceau 5. Notons au passage que cette dernière information est très utile : le parseur consomme un temps linéairement proportionnel à la largeur du faisceau, et ces données permettent donc de voir qu'un faisceau large (plus de 5) n'est pas rentable. On observe généralement un gain de précision minime, voir une perte légère, mais un gain de rappel important (pour la relation

« racine », par exemple, on observe un gain de rappel de 5,59 % entre les faisceaux 1 et 20).

Etiquette	Nombre de cas	Largeur de faisceau :					Gain (f20-f1)		
		1	2	5	10	20	Préc.	Rap	F-mes
total[5]	**31 703**	**87,7**	**88,5**	**88,8**	**89,0**	**89,1**	**+1,38**		
sujet	2 132	92,8	93,7	94,3	94,5	94,6	-0,25	+3,51	+1,82
racine[6]	1 235	91,9	93,5	94,5	94,8	95,1	-0,06	+5,59	+3,12
coordonné[7]	939	89,4	90,5	91,2	91,4	91,4	-0,25	+3,41	+1,94
coordonnant[8]	819	68,3	69,5	70,4	70,5	70,4	+0,32	+2,45	+2,12
relative[9]	379	78,4	79,9	80,5	80,9	81,1	+1,96	+2,91	+2,69

TABLE 1 : F-score par largeur de faisceau : valeur globale et détails pour certaines étiquettes choisies

Pour le score total, les différences entre les différentes largeurs de faisceau sont toutes significatives (test de McNemar, $p<0,05$). Pour les relations individuelles, on observe globalement un gain non-significatif lorsque le faisceau dépasse une largeur de 5.

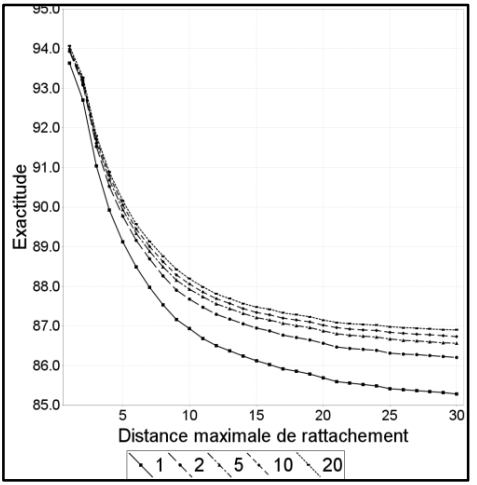

FIGURE 1 : Exactitude par faisceau et par distance maximale de rattachement

[5] A l'instar d'autres études similaires, nous donnons l'exactitude totale hors ponctuation
[6] Relation dont le dépendant est le verbe principal de la phrase, et le gouverneur et une « racine » artificielle
[7] Relation dont le dépendant est un mot coordonné et le gouverneur est le coordonnant qui le précède
[8] Relation dont le dépendant est une conjonction ou une virgule et el gouverneur est le coordonné qui la précède
[9] Relation dont le dépendant est le verbe d'une subordonnée relative , et le gouverneur est l'antécédent du pronom relatif qui introduit cette subordonnée (le pronom relatif lui-même sera rattaché au verbe par les relations « suj », « obj », ...)

La FIGURE 1 donne l'exactitude en fonction des distances maximales de rattachement (en nombre de mots séparant les mots reliés syntaxiquement). Chaque point de la courbe représente donc l'exactitude pour tous les liens de dépendances dont la distance entre le gouverneur et le dépendant est inférieure ou égale à une distance donnée. Alors que l'exactitude baisse avec la distance maximale pour tous les faisceaux, l'écart entre les faisceaux s'accroît : plus le faisceau est large, plus le parseur parvient à traiter correctement les relations à longue distance.

4 Le faisceau entre les modules

Dans ce paragraphe, nous nous intéressons à la propagation du faisceau *entre* les modules. Sans propagation, chaque module du début de la chaîne (segmentation ou étiquetage) choisit la meilleure configuration possible et la transmet au module suivant (étiquetage ou parsing). Si l'on active la propagation avec un faisceau de largeur *k*, le module fournit alors *k* propositions qui vont être prises en considération (avec une probabilité associée). Au fur et à mesure de l'analyse, certains choix du module précédent seront abandonnés (largeur de faisceau oblige), alors que d'autres seront retenus, voire ramenés en haut de la pile.

Nous avons utilisé deux corpus d'évaluation : le premier est le corpus de test issu du French Treebank, et permet d'avoir un aperçu quantitatif en comparant les résultats avec l'annotation manuelle. Nous y avons ajouté un extrait du corpus Leximedia 2007[10] qui contient des articles de presse de plusieurs quotidiens français relatifs à la précédente campagne présidentielle. Dans ce corpus, nous avons analysé manuellement les 100 premières différences de traitement obtenues avec et sans propagation du faisceau, et ce pour plusieurs largeurs, afin d'avoir une vision qualitative des phénomènes mis en jeu.

4.1 Impact de l'étiquetage et du parsing sur la segmentation : le traitement des unités polylexicales ambiguës

Notre hypothèse est que le repérage des unités polylexicales peut être amélioré en prenant en considération les informations morphosyntaxiques et syntaxiques. A notre connaissance, tous les systèmes d'étiquetage effectuent un traitement systématique des locutions et expressions figées (lorsqu'ils traitent ces cas) en projetant un lexique sans condition. Si certaines locutions sont totalement non ambiguës (« *parce_que* », « *d'ores_et_déjà* » etc.) certaines occurrences peuvent correspondre à des configurations syntaxiques particulières comme dans « Jean-Claude Brialy, qui nous *quitte à* 74 ans, avait été un jeune premier éblouissant. ». Dans cet exemple extrait de notre corpus d'évaluation (et correctement traité grâce à cette méthode), il est clair que « *quitte_à* » n'est pas une préposition (mais un verbe suivi d'une préposition) quand on considère la configuration globale de la phrase. Dans le cas d'une propagation, les deux solutions de segmentation envisageables vont donc être soumises à l'étiqueteur qui pourra soit décider, soit transmettre l'ambiguïté à son tour au parseur (en fonction des priorités et des autres ambiguïtés qu'il ordonnancera dans son faisceau). La décision finale de segmenter ou non sera prise à la toute fin du processus.

Pour comprendre le mécanisme interne, prenons la phrase « Elle pourrait *même s'*ennuyer. » Au niveau de la segmentation, il y a une ambiguïté entre « *même_si* » (conjonction de

[10] http://redac.univ-tlse2.fr/Leximedia2007/

subordination) et les deux mots « *même* » (adverbe) et « *se* » (pronom clitique réfléchi). On a appliqué ici une analyse avec un faisceau de largeur 2. La TABLE 2 ci-dessous montre le faisceau terminal du segmenteur pour cette phrase, où la proposition erronée d'un seul mot composé « *même si* » est privilégiée (la probabilité globale étant supérieure).

Elle	*pourrait*	*même s'*		*ennuyer*	.	Score : 66%
Elle	*pourrait*	*même*	*s'*	*ennuyer*	.	Score : 34%

TABLE 2 : Faisceau final du segmenteur pour la phrase « *Elle pourrait même s'ennuyer*. »

Sans la propagation, la proposition erronée est donc la seule transmise à l'étiqueteur. Avec la propagation, les deux propositions sont transmises et analysées, tel qu'on le voit dans la TABLE 3. L'étiqueteur arrive donc à trancher pour la bonne solution, car la séquence [*verbe indicatif, conjonction de subordination, verbe infinitif*] est très peu probable dans le corpus d'entraînement. Le parseur (détails non fournis) ne fera ici que confirmer ce choix.

Elle CLS[11]	*pourrait* V	*même* ADV	*s'* CLR	*ennuyer* VINF	. PONCT	Score d'étiquetage[12]	Score de segmentation	Score total
96 %	99 %	99 %	88 %	94 %	94 %	**95 %**	34 %	**32 %**
Elle CLS	*pourrait* V	*même s'* CS		*ennuyer* VINF	. PONCT	Score d'étiquetage	Score de segmentation	Score total
96 %	99 %	8 %		24 %	83 %	43 %	**66 %**	29 %

TABLE 3 : Faisceau final de l'étiqueteur pour la phrase « *Elle pourrait même s'ennuyer*. »

Notons que le score associé à chaque étiquette représente sa probabilité dans une distribution couvrant toutes les étiquettes morphosyntaxiques possibles. L'étiquette choisie est celle dont la probabilité est la plus élevée dans cette distribution, et dont le choix n'est pas interdit par les règles que l'utilisateur aura configurés.

Prenons un autre exemple : « *Il y a* plus grave. » L'expression « il y a » est de segmentation ambiguë, car considérée comme une préposition (ex. « Je suis venu *il y a* trois ans. ») ou comme une séquence de trois mots. La TABLE 4 ci-dessous montre le faisceau terminal du segmenteur pour cette phrase, qui privilégie donc la locution prépositionnelle.

Il y a			*plus*	*grave*	.	Score : **55%**
Il	*y*	*a*	*plus*	*grave*	.	Score : 45%

TABLE 4 : Faisceau final du segmenteur dans la phrase « Il y a plus grave. »

Le faisceau terminal de l'étiqueteur, montré dans la TABLE 5 ci-dessous, rapproche les probabilités des deux solutions, sans pour autant en changer l'ordre.

[11] Etiquettes morphosyntaxiques de Crabbé et Candito (2008) : ADV = adverbe. CLO = clitique objet. CLR = clitique réfléchi. CLS = clitique sujet. P = préposition. V = verbe indicatif. VINF = verbe infinitif.
[12] Moyenne harmonique des probabilités individuelles.

Il y a P			*plus* ADV	*grave* ADJ	. PONCT	Score d'étiquetage	Score de segmentation	Score total
67 %			99 %	94 %	98 %	88 %	**55 %**	**49 %**
Il CLS	*y* CLO	*a* V	*plus* ADV	*grave* ADJ	. PONCT	Score d'étiquetage	Score de segmentation	Score total
97 %	95 %	99 %	98 %	94 %	98 %	**97 %**	45 %	44 %

TABLE 5 : Faisceau final de l'étiqueteur dans la phrase « *Il y a plus grave*. »

Ce sera ici le parseur qui permettra de corriger l'erreur. La FIGURE 2 montre le faisceau terminal du parseur. Pour simplifier, nous avons attribué des probabilités aux arcs de dépendance. En réalité, il y a une probabilité pour chaque transition, même celles qui n'ont pas généré des arcs (ex. Shift). Nous avons intégré ces probabilités dans celles des arcs.

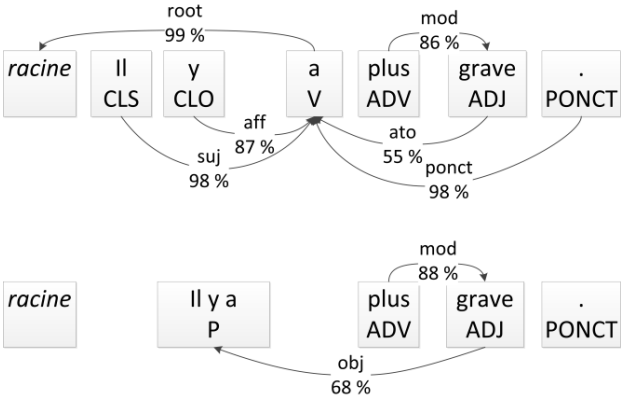

FIGURE 2 : Faisceau final du parseur pour la phrase « *Il y a plus grave.* »

Notons que dans le cas de « *il y a* » comme préposition composée, le parseur n'a pas trouvé de racine (la phrase n'ayant pas de verbe), et du coup n'a pas rattaché la ponctuation non plus (classiquement rattachée au verbe central). La TABLE 6 ci-dessous montre ce même faisceau final du parseur avec les scores. Pour chaque mot, on a indiqué la probabilité de l'arc qui gouverne ce mot. Le score total est la moyenne harmonique des probabilités de chaque arc (ou plutôt, de chaque transition), multipliée par le score d'étiquetage. Le parseur arrive donc à trancher pour la bonne réponse, quoiqu'avec une faible marge.

Nous passons maintenant aux évaluations globales. Pour la segmentation, la question ne se pose que pour un ensemble de locutions prédéfinies. Sans propagation, le segmenteur atteint déjà une exactitude de 94,9 % pour les séquences de mots correspondantes sur le sous corpus de test (à peu près 2 000 bonnes réponses sur 2 100). Une étude des erreurs montre que, sur les 50 premières erreurs, 60 % se révèlent en fait être des erreurs d'annotation. Dans ce contexte, la propagation a très peu d'effet sur le score total. Entre les faisceaux 1 et 2 il n'y a que 13 cas de différence (sur 2 100), dont 5 corrections et 8 erreurs introduites. Les faisceaux plus larges ont le même comportement.

Il CLS suj[13]	*y* CLO aff	*a* V root	*plus* ADV mod	*grave* ADJ ato	. PONCT ponct	Score de parsing	Score d'étiquetage	Score total
98 %	87 %	99 %	86 %	55 %	98 %	**85 %**	44 %	**38 %**
Il y a P NA			*plus* ADV mod	*grave* ADJ obj	. PONCT NA	Score de parsing	Score d'étiquetage	Score total
NA			89 %	68 %	NA	75 %	**49 %**	37 %

TABLE 6 : Faisceau final du parseur dans la phrase « Il y a plus grave. »

De cette évaluation peu convaincante, nous passons au corpus non annoté Leximedia2007. Ici , nous avons appliqué la segmentation, l'étiquetage et le parsing à un texte brut à différentes largeurs de faisceau avec et sans propagation. Nous avons par la suite comparé les segmentations de différents runs, et annoté manuellement les 109 premiers cas de différence (on a observé 300 différences pour 1 million de mots). Comme attendu, dans les essais sans propagation, la segmentation est restée identique (voir paragraphe 3.1 ci-dessus). La TABLE 7 ci-dessous donne le nombre de bonnes réponses au niveau de la segmentation par faisceau, quand la propagation est appliquée.

Faisceau	**1**	**2**	**5**	**10**	**20**
Bonnes réponses	**69**	46	50	45	49

TABLE 7 : Bonnes réponses de la segmentation avec propagation sur le corpus Leximedia, pour les 109 premiers cas de différence entre les faisceaux

En règle générale, les faisceaux à partir de 2 dégradent les résultats, en séparant à tort des locutions (45 cas pour le faisceau 2). On observe toutefois plusieurs cas (22 pour le faisceau 2) où la segmentation est effectivement corrigée, comme par exemple :

- « Villepin précise *encore que*, bien évidemment, il a fait procéder…»
- « Elle pourrait *même s'*être retournée contre les amis de M. Strauss-Kahn, soupçonnés de l'avoir diffusée. »
- « Jean-Claude Brialy, qui nous *quitte à* 74 ans, avait été un jeune premier éblouissant. »

Au vu de ce bilan global, il apparaît que notre hypothèse sur l'utilité de la propagation du faisceau pour la segmentation est à rejeter en l'état.

4.2 Impact du parsing sur l'étiquetage

Pour cette seconde articulation entre deux modules, notre hypothèse est que certaines

[13] Les étiquettes des arcs suivent le guide d'annotation de Candito, Crabbé et Falco : aff = clitique figé, ato = attribut de l'objet, mod = modifieur, obj = objet de préposition ou objet direct du verbe, suj = sujet, ponct = ponctuation, root = relation réliant le verbe central à une « racine » artificiel

ambiguïtés catégorielles ne peuvent être efficacement traitées qu'en considérant le niveau syntaxique. Nous avons donc comparé, pour une même segmentation des deux corpus d'évaluation, une analyse avec et sans propagation pour la même largeur de faisceau, de façon à pouvoir isoler le gain apporté par la parseur à l'étiquetage morphosyntaxique.

Pour le corpus de test du French Treebank, comme vu précédemment, la largeur de faisceau a très peu d'effet sur l'exactitude totale *sans* propagation. Le gain est bien plus perceptible avec propagation, comme on le voit dans la TABLE 8 ci-dessous.

Faisceau	1	2	5	10	20
Sans propagation	97,81	97,82	97,83	97,83	97,83
Avec propagation	97,81	97,87	97,92	97,94	97,95

TABLE 8 : Exactitude total de l'étiqueteur morphosyntaxique, avec et sans propagation vers le parseur, pour 5 largeurs de faisceau

En terme de significativité statistique (test de McNemar, $p<0,05$), les gains apportés par l'élargissement du faisceau sans activer la propagation ne sont pas significatifs (première ligne du tableau). Ils le sont par contre pour chaque largeur de faisceau lorsque l'on active la propagation (pour chaque colonne du tableau) et également lorsque l'on compare les différentes largeurs avec propagation (seconde ligne du tableau).

Dans les détails, les gains sont concentrés sur certaines catégories grammaticales (adjectif, conjonction de subordination, déterminant, pronom, pronom relatif).

Pour le corpus non annoté de Leximedia2007, nous avons examiné 132 cas de différences entre les configurations envisagées (on a observé globalement une différence tous les 100 mots), en identifiant manuellement la bonne réponse à chaque fois. La TABLE 9 donne le nombre de bonnes réponses pour chaque largeur de faisceau, avec et sans propagation. Nous observons ici un gain très net avec l'application de la propagation. Les erreurs ont par contre tendance à croître légèrement à partir d'un faisceau de largeur 10.

Faisceau	1	2	5	10	20
Sans propagation	52	58	58	53	52
Avec propagation	52	71	**72**	71	69

TABLE 9 : Nombre de bonnes réponses de l'étiqueteur morphosyntaxique pour le corpus Leximedia2007 avec et sans propagation (132 premières différences)

Nous n'avons pas pu isoler de régularités dans les types d'erreurs ainsi corrigées, qui semblent couvrir les cas classiques d'ambiguïté catégorielle. Les cas suivants sont corrigés avec un faisceau de 5 (et au-delà) avec propagation :

- « … a estimé "vraisemblable" qu'après l'élection de M. Sarkozy, un nouveau traité *soit* achevé "au plus tard en décembre". » (conjonction de coordination → **verbe subjonctif**)
- « … en soulignant "l'émotion" *qu'*il ressentait au cours de cette première visite d'Etat … » (conjonction de subordination → **pronom relatif**)

– « Evoquant sous les applaudissements cette "place de France *que* je voudrais aussi place de la paix", … » (conjonction de subordination → **pronom relatif**)

Pour le faisceau 2, on a observé 43 cas de correction, contre 24 cas de dégradation, comme celui-ci-dessous :

– « *Qui* mieux que le peuple corse peut choisir librement son développement ? » (**pronom interrogatif** → pronom relatif)

Au vu de ces résultats, il semble donc que les modifications apportées à l'étiquetage par propagation du faisceau vers le parseur soient des améliorations.

5 Comparaison avec d'autres études

La TABLE 10 montre les exactitudes atteintes par Talismane par comparaison avec Candito et al (2010). Pour pouvoir comparer nos résultats, nous donnons ici l'exactitude pour un texte pré-segmenté en mots (les sous-corpus d'évaluation « *dev* » et « *test* » du French Treebank), auquel on a appliqué l'étiqueteur morphosyntaxique et le parseur (avec propagation du faisceau). Les trois premières lignes sont celles fournies par Candito et al, (2010), pour leur meilleur jeu de traits. Pour le temps de calcul, Talismane a été évalué avec une architecture semblable[14].

Parseur	LAS[15] Dev	UAS[16] Dev	LAS Test	UAS Test	Temps de calcul
Berkeley	86,5	90,8	86,8	91,0	12m46s
MSTParser	87,5	90,3	88,2	90,9	14m39s
MaltParser	86,9	89,4	87,3	89,7	1m25s
Talismane (faisc 1)	86,8	90,2	87,2	90,6	7m56s
Talismane (faisc. 2)	87,3	90,4	88,0	91,0	14m51s
Talismane (faisc. 5)	87,8	90,7	88,3	91,1	38m26s
Talismane (faisc. 10)	88,0	90,8	88,4	91,1	80m36s
Talismane (faisc. 20)	88,1	90,8	88,5	91,1	157m53s

TABLE 10 : Exactitude et temps de calcul par parseur

Du point de vue de son architecture, Talismane se rapproche surtout du MaltParser, qui est lui aussi un parseur en dépendances par transitions. Avec un faisceau de 1, les scores sont effectivement proches pour le score avec étiquettes (LAS), et Talismane est légèrement meilleur pour les seuls gouverneurs (UAS). Par contre, le MaltParser est bien plus rapide. Avec un faisceau de 2, Talismane est très proche des scores du MSTParser (parseur par

[14] Intel i5 CPU 2.40 GHz

[15] LAS : *Labeled Attachment Score* = l'exactitude en considérant à la fois l'identification du gouverneur et l'étiquetage des arcs. La ponctuation n'est pas prise en compte.

[16] UAS : *Unlabeled Attachment Score* = l'exactitude si on prend on compte uniquement les gouverneurs, et non les étiquettes des arcs. La ponctuation n'est pas prise en compte.

graphes) pour le LAS et l'UAS. Les scores pour les faisceaux plus larges sont légèrement meilleurs, mais au prix d'un temps de calcul bien plus élevé (comme dit précédemment, l'impact de la largeur sur le temps est linéaire). Il reste bien entendu à comparer Talismane avec des analyseurs basés sur une grammaire, notamment FRMG (Villemonte de la Clergerie et al. 2009).

Pour l'étiqueteur morphosyntaxique, (Denis et Sagot, 2009) signalent un score de 97,7 % sur une partie du French Treebank. Notre score de 97,8 % sans faisceau ni propagation est donc tout à fait comparable. Après les corrections du parseur par propagation du faisceau, le score de 97,9 % est légèrement supérieur.

6 Conclusions

Nous avons présenté l'outil Talismane et la chaîne complète d'analyse syntaxique que cet outil propose, permettant de produire un arbre de dépendances à partir d'un texte brut. D'après notre évaluation, cet un outil atteint (voire dépasse) les autres analyseurs statistiques actuellement disponibles pour le français.

Nous avons étudié de plus près les effets de la recherche par faisceau entre les différents modules d'analyse. Selon nos évaluations, si la propagation des ambiguïtés entre les modules a peu d'intérêt pour la segmentation en mots, elle semble au contraire très intéressante pour l'étiquetage morphosyntaxique, avec un gain significatif. Nous avons modifié la dernière version disponible de Talismane en conséquence.

Nous avons étudié le comportement de chaque module avec différentes largeurs de faisceau. Pour le parseur en particulier, un faisceau de largeur 2 ou 5 semble être un bon compromis entre exactitude des résultats et vitesse d'analyse, une largeur plus grande apportant très peu d'améliorations. Par contre, un faisceau large semble critique pour traiter efficacement les relations syntaxiques à grande distance.

L'analyse qualitative des phénomènes syntaxiques mieux ou moins bien traités par chaque configuration est encore à affiner. Cet aspect est important à plusieurs titres. Tout d'abord, on sait que les évaluations globales d'un analyseur syntaxique ne sont au final pertinentes qu'au vu d'une tâche particulière, qui peut accorder plus ou moins d'importance au traitement efficace de tel ou tel phénomène syntaxique. Ensuite, une caractéristique importante de Talismane est la souplesse de définitions de traits et de règles qui permet précisément de cibler des phénomènes particuliers une fois ceux-ci identifiés, en gardant une porte d'entrée linguistique dans un système statistique opaque par essence (Tanguy, 2012).

Remerciements

Nous tenons à remercier Marjorie Raufast pour son aide précieuse dans l'évaluation détaillée.

Références

ABEILLÉ, A., L. CLÉMENT, ET F. TOUSSENEL (2003). Building a treebank for French, in A. Abeillé (ed) *Treebanks* , Kluwer, Dordrecht.

ATTARDI, G., DELLORLETTA, F., SIMI, ET M., TURIAN J. (2009) Accurate Dependency Parsing with a Stacked Multilayer Perceptron. In *Proceedings of Evalita'09 at AI*IA*, Reggio Emilia, Italy.

CANDITO M.-H., CRABBÉ B., ET DENIS P., (2010) Statistical French dependency parsing: treebank conversion and first results*, Proceedings of LREC'2010*, La Valletta, Malta.

CANDITO M.-H., NIVRE J., DENIS P. ET HENESTROZA ANGUIANO E., (2010) Benchmarking of Statistical Dependency Parsers for French, in *Proceedings of COLING'2010*, Beijing, China

CRABBÉ B. ET CANDITO M.-H. (2008), Expériences d'analyse syntaxique statistique du français, in *Actes de TALN 2008*, Avignon, France.

DENIS P. ET SAGOT B., (2009) Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort, *in Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC), Hong-Kong*.

HO C.-H. ET LIN C.-J. (2012), Large-scale Linear Support Vector Regression, *Journal of Machine Learning Research*, 13, pp. 3323-3348.

JOHANSSON R. ET NUGUES P. (2006). Investigating multilingual dependency parsing. *In proceeding of CoNLL-X*, New York.

JOHANSSON R. ET NUGUES P. (2007). Incremental Dependency Parsing Using Online Learning. *In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, Prague

MCDONALD, R. (2006). *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

MCDONALD, R. ET J. NIVRE. (2007). Characterizing the errors of data-driven dependency parsing models. *In proceedings of EMNLP-CoNLL 2007*, Prague.

NIVRE J. (2008), *Algorithms for Deterministic Incremental Dependency Parsing*, Computational Linguistics, 34(4), 513-553.

RATNAPARKHI, A. (1998) *Maximum entropy models for natural language ambiguity resolution*, PhD Thesis, University of Pennsylvania, 1998.

SAGAE K. ET LAVIE A. (2006), A best-first probabilistic shift-reduce parser*, in Proceedings of the COLING/ACL joint conference*, Sydney.

SAGOT B., CLÉMENT L., DE LA CLERGERIE E. ET BOULLIER P. (2006) The Lefff 2 syntactic lexicon for French: architecture, acquisition, use, *in Proceedings of LREC*, Gênes.

TANGUY, L. (2012). *Complexification des données et des techniques en linguistique : contributions du TAL aux solutions et aux problèmes*. Mémoire d'HDR, Université de Toulouse.

TESNIÈRE, LUCIEN. (1959). *Eléments de syntaxe structurale,* Klincksieck, Paris.

VILLEMONTE DE LA CLERGERIE, E, SAGOT, B., NICOLAS L. ET GUÉNOT, ML. (2009). FRMG : évolutions d'un analyseur syntaxique TAG du français *Journée de l'ATALA sur « Quels analyseurs syntaxiques pour le français ? »*.