Extraction de PCFG et analyse de phrases pré-typées

Noémie-Fleur Sandillon-Rezer
CNRS, Esplanade des Arts et Métiers, 33402 Talence
LaBRI, 351 Cours de la Libération, 33405 Talence
nfsr@labri.fr

RÉSUMÉ

Cet article explique la chaîne de traitement suivie pour extraire une grammaire PCFG à partir du corpus de Paris VII. Dans un premier temps cela nécessite de transformer les arbres syntaxiques du corpus en arbres de dérivation d'une grammaire AB, ce que nous effectuons en utilisant un transducteur d'arbres généralisé ; il faut ensuite extraire de ces arbres une PCFG. Le transducteur d'arbres généralisé est une variation des transducteurs d'arbres classiques et c'est l'extraction de la grammaire à partir des arbres de dérivation qui donnera l'aspect probabiliste à la grammaire. La PCFG extraite est utilisée via l'algorithme CYK pour l'analyse de phrases.

ABSTRACT

PCFG Extraction and Pre-typed Sentences Analysis

This article explains the way we extract a PCFG from the Paris VII treebank. Firslty, we need to transform the syntactic trees of the corpus into derivation trees. The transformation is done with a generalized tree transducer, a variation of the usual top-down tree transducers, and gives as result some derivation trees for an AB grammar. Secondely, we have to extract a PCFG from the derivation trees. For this, we assume that the derivation trees are representative of the grammar. The extracted grammar is used, via the CYK algorithm, for sentence analysis.

MOTS-CLÉS : Extraction de grammaire, grammaire de Lambek, PCFG, transducteur d'arbre, algorithme CYK.

KEYWORDS: Grammar Extraction, Lambek grammar, PCFG, tree transducer, CYK Algorithm.

Actes de la conférence conjointe JEP-TALN-RECITAL 2012, volume 3: RECITAL, pages 205–218,
Grenoble, 4 au 8 juin 2012.

1 Introduction

Cet article décrit les méthodes que nous employons pour transformer les arbres syntaxiques du corpus de Paris VII en arbres de dérivation d'une grammaire AB (Lambek, 1958), pour l'extraction de cette grammaire et son utilisation pour l'analyse de phrases. Les grammaires AB sont utilisées dans des algorithmes d'apprentissage tels que celui de Buskowsky et Penn (Buszkowski et Penn, 1990), qui permet d'apprendre une grammaire AB rigide [1], ou celui de Kanazawa (Kanazawa, 1998), permettant d'apprendre une grammaire k-valuée [2] ; c'est pour cela que nous avons souhaité, dans un premier temps, utiliser de telles grammaires. Les grammaires AB représentent un fragment des grammaires de Lambek, comprenant uniquement des règles de dérivation de type $a \rightarrow a/b\ b$ et $a \rightarrow b\ b\backslash a$. Le corpus de Paris VII (Abeillé *et al.*, 2003) est composé de 12855 phrases tirées du journal *Le Monde*, annotées et analysées par le laboratoire de Paris VII. Les arbres syntaxiques sont planaires, le nombre de fils par noeud et la profondeur ne sont pas fixés. Cela rend l'application d'algorithmes d'apprentissage usuels impossible ; nous avons donc pris le parti d'utiliser un transducteur d'arbres.

Nous avons utilisé, pour notre travail, une sous-partie du corpus, présentée sous forme parenthésée de 12351 phrases, alors que le corpus complet est au format **XML**. Les 504 phrases laissées de côté forment un corpus annexe dont nous nous servons pour l'évaluation.

En premier lieu, nous présenterons le transducteur, utilisé pour transformer les arbres syntaxiques en arbres de dérivation, puis nous nous pencherons sur l'extraction d'une grammaire PCFG. La troisième partie détaillera l'analyse du placement des syntagmes prépositionnels dans la phrase ; tandis que la quatrième présentera les résultats expérimentaux obtenus en utilisant notre grammaire PCFG pour trouver la meilleure analyse possible pour une phrase via l'algorithme CYK (Younger, 1967).

2 Transducteur d'arbres généralisé

Ce n'est pas la première fois que les transducteurs sont utilisés dans le cadre de la linguistique computationnelle ; on peut citer Knight et Graehl (Knight et Graehl, 2005), qui utilisent des transducteurs d'arbres à états finis, dont hélas l'utilisation ne correspondait pas à notre problématique.

Des travaux de recherche plus appliqués, tels que (Hockenmaier et Steedman, 2007; Moot, 2010a,b; Moortgat et Moot, 2001), utilisent des algorithmes spécialisés qui s'appliquent uniquement à un corpus donné, avec un espoir faible de réutilisation. Etant donné les différences d'annotations d'un corpus à l'autre, et les variations grammaticales que l'on peut trouver entre deux langues, adapter un outil pour le corpus de Paris VII est toujours particulièrement laborieux. Etant donné que nous avons totalement séparé, lors de l'implémentation, le fonctionnement du transducteur de l'ensemble des données qui lui sont passées en entrée (telles que les fichiers de règles et le corpus sous forme parenthésée), nous pensons qu'un lissage des données suffit à appliquer notre transducteur à d'autres ensembles d'arbres.

Le transducteur que nous avons créé est le pivot central du processus d'extraction de grammaire.

1. Chaque mot du lexique n'a le droit d'avoir qu'un seul type.
2. Chaque mot du lexique peut avoir jusqu'à k types.

En effet, c'est la binarisation des arbres syntaxiques, fondée sur les règles usuelles de dérivation d'une grammaire AB [3] et les annotations morpho-syntaxiques du corpus (Abeillé et Clément, 2003), qui paramétrise la grammaire extraite.

Nous avons d'abord mis au point une version théorique de notre G-transducteur (G pour généralisé) avant de l'implémenter pour le tester sur le corpus de Paris VII.

La création du transducteur d'arbres est décrite en détail dans (Sandillon-Rezer et Moot, 2011).

En nous fondant sur les transducteurs d'arbres *top-down* décrits dans TATA (Comon *et al.*, 2007), nous avons généralisé les règles de transduction de manière à créer un outil plus adapté au corpus de Paris VII. Ainsi, on peut dire que les trois principales différences entre un transducteur *top-down* classique et notre G-transducteur sont : sa récursivité, sa paramétrisation et son système de règles de priorité.

La récursivité permet d'appliquer un ensemble de règles à un nœud, jusqu'à ce qu'il soit traité en entier, sans pour autant changer l'état du transducteur ni utiliser un nouvel ensemble de règles de transduction (voir figure 1).

La paramétrisation permet de définir des règles avec variables. Ainsi, on peut donner une transduction générale pour les adverbes et les modificateurs (voir figure 2).

Les règles de priorité : assurent le déterminisme de notre transducteur. Ainsi, lorsque deux règles peuvent s'appliquer, on leur donne un ordre d'application qui permet d'avoir toujours les mêmes arbres de sortie (voir figure 3).

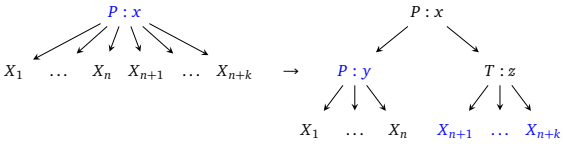

FIGURE 1 – Le nouveau nœud $P : y$ a moins de fils qu'avant la transduction et pour le transducteur, il restera dans le même état et avec le même label que le nœud parent $P : x$. Généralement, la règle sera écrite de manière à ce que le sous arbre $T : z$ soit binaire et les types y et z doivent obligatoirement se combiner pour donner le type x.

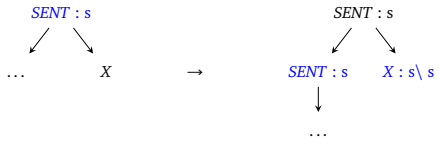

FIGURE 2 – La même règle sera appliquée pour $X \in \{ADV, PP-MOD, AdP-MOD, AP-MOD, ...\}$

3. les groupes nominaux auront le type *np* etc.

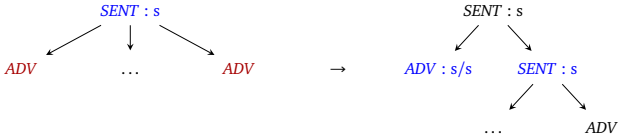

FIGURE 3 – Lorsque plus d'une règle peut s'appliquer à un arbre, le fait de suivre un ordre prédéfini permet d'éviter le non-déterminisme du transducteur.

Les règles ont été déduites d'une analyse systématique des formes présentes dans le corpus. Un exemple de règle est donné dans la figure 4 et un de résultat dans la figure 5. Une fois le corpus transformé en forêt d'arbres de dérivations, nous n'utilisons plus le transducteur, que ce soit pour l'extraction de grammaire ou l'analyse de phrases.

```
(rule
      (SENT:* NP-SUJ (VN tree VPP) PP-OBJ)
      ("SENT:*" "NP-SUJ:np"
            ("VN:np\\*" "VN:(np\\*)/(np\\s_p)"
                  (":np\\s_p" "VPP:(np\\s_)/pp"
            PP-OBJ:pp))))
```

FIGURE 4 – Exemple de règle telle que donnée au transducteur. On note deux points importants, directement dérivés des spécifications de notre G-transducteur : le mot clef *tree*, qui permet de remplacer "un certain nombre de nœuds", qui peut apparaître plusieurs fois dans le motif de départ mais pas dans le motif de remplacement ; et le type *, qui remplace n'importe quel type hérité des étapes précédentes.

3 Extraction de grammaire

Même si le lexique, récupéré à partir des feuilles des arbres de dérivation, suffirait à représenter la grammaire AB, il nous limite aux mots présents dans le corpus. Or, bien que nous puissions avoir des probabilités sur les types des mots, nous voulions une grammaire PCFG. Par conséquent, nous avons pris le parti d'extraire une grammaire probabiliste à partir des arbres.

Les arbres en sortie du transducteur donnent des informations à la fois syntaxiques, car nous gardons les labels donnés par le corpus et, bien sûr, des informations structurelles. Nous avons pris le parti de laisser le choix des informations que nous souhaitons garder, en effectuant une passe de prétraitement, sachant bien sûr que les types sont, de toute façon, obligatoirement conservés. La grammaire extraite sera de toute façon une grammaire hors contexte, avec une probabilité calculée sur les règles en fonction de leur racine. Pour plus de simplicité, on rappelle que les grammaires sont de la forme $\{N, T, S, R\}$:

N l'ensemble des symboles non terminaux, correspondant aux nœuds internes de l'arbre.

T l'ensemble des symboles terminaux, correspondant à l'ensemble des mots typés.

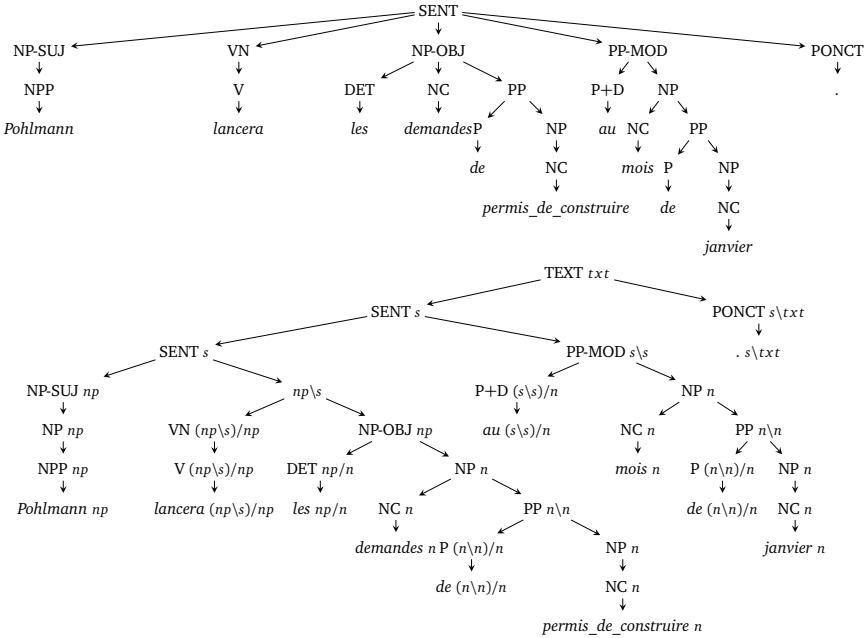

FIGURE 5 – Arbre d'entrée et de sortie du transducteur correspondant à la phrase "Pholmann lancera les demandes de permis de construire au mois de janvier.".

S le symbole initial. On choisira, en fonction de la passe de pré-traitement, *TXT :txt* ou *txt*.

R l'ensemble des règles.

L'algorithme utilisé pour extraire la grammaire consiste à parcourir les arbres donnés en paramètre et stocker les règles de dérivation que l'on rencontre. On considère qu'une règle de dérivation est constituée d'une racine et d'un ou deux fils :

La racine a deux fils : On est dans le cas de figure classique d'une règle d'élimination à droite ou à gauche ($a \rightarrow a/b \quad b$ ou $a \rightarrow b \quad a\backslash b$).

La racine a un seul fils : Il y a simplement transmission de type au fils. Ce cas de figure apparaît, par exemple, lorsqu'un groupe nominal est composé uniquement d'un nom propre, ou encore lorsqu'on est au niveau du nœud pré-terminal, c'est à dire l'étiquette de partie du discours (POS-tag) de la feuille. Dans ce cas, la feuille héritera directement du type du POS-tag.

Chaque règle est accompagnée d'un compteur et les probabilités sur les règles sont calculées par groupe ayant la même racine. On récupère aussi des informations de profondeur minimale et maximale d'apparition de la règle, cependant elles ne sont pas utilisées pour l'instant.

Ainsi, on résume dans le tableau 1 les différentes grammaires que peut générer l'extracteur [4] .Chacune des versions montre un intérêt : autant la première, extraite des arbres juste après transduction, garde les informations syntaxiques données par le corpus ; autant les suivantes sont plus utiles pour appliquer un algorithme d'analyse de phrases, tel que CYK (voir section 4), sur des phrases non typées. Le tableau 2 montre des extraits des différentes grammaires en fonction des arbres donnés en entrée.

Forme des arbres	**Règles extraites**	**Spécifications**	**Nombre de règles**
Arbres de dérivation bruts	$n_1 \rightarrow n_2 \quad n_3$ $n_1 \rightarrow n_2$ $n_1 \rightarrow t_1$	Facilement normalisable en FNC : il suffit d'enlever les chaines unaires	63368
Retrait des chaines unaires et des labels sauf les POS-tag	$n_1 \rightarrow n_2 \quad n_3$ $n_1 \rightarrow t_1$	La grammaire est en FNC.	59505
Retrait de tous les labels et des chaines unaires. Il n'y a plus de différence entre N et T.	$n_1 \rightarrow n_2 \quad n_3$	Les mots n'apparaissent plus, ce qui laisse uniquement le squelette des arbres.	3494

TABLE 1 – Grammaires extraites en fonction des arbres de dérivation donnés en entrée. On précise que $n_i \in N$ et $t_i \in T$.

Arbres de dérivation bruts		
Exemple de règles	$NP:np \rightarrow NPP:np$	$1.01x10^{-1}$
	$NP:np \rightarrow DET:np/n\ NC:n$	$2.02x10^{-1}$
	...	
Retrait des chaines unaires et des labels sauf les POS-tag		
Exemple de règles	$(np\backslash s_i)/(np\backslash s_p) \rightarrow VINF:(np\backslash s_i)/(np\backslash s_p)$	$9.53x10^{-1}$
	$(np\backslash s)/(np\backslash s_p) \rightarrow CLR:cl_r \quad cl_r\backslash((np\backslash s)/(np\backslash s_p))$	$2.88x10^{-2}$
	...	
Retrait de tous les labels et des chaines unaires		
Exemple de règles	$s \rightarrow np\ np\backslash s$	$3,81x10^{-1}$
	$s \rightarrow s\ s\backslash s$	$2,65x10^{-1}$
	$s \rightarrow np\backslash s_p\ (np\backslash s_p)\backslash s$	$1,13x10^{-3}$
	$n \rightarrow n\ n\backslash n$	$7,97x10^{-1}$
	$np \rightarrow np/n\ n$	$8,02x10^{-1}$
	...	

TABLE 2 – Exemples des différentes règles que l'on peut extraire des arbres.

4 Analyse de phrases

La question de l'analyse de phrases en fonction d'une grammaire PCFG se subdivise en deux problèmes. En effet, il faut d'une part trouver les types des mots et d'autre part que les règles

4. Les *POS-tags* sont les étiquettes de parties du discours.

existent dans la grammaire passée en paramètre à l'analyseur.

4.1 Typages des mots

En réunissant les feuilles des arbres de dérivation, nous pouvons collecter un lexique contenant les mots, leur occurrence, les types de ceux-ci et la probabilité du type (*nb_occurrences_du_type*/ *nb_occurrences_du_mot*). Cependant, nous n'utilisons pas encore le lexique pour typer les phrases que nous analysons. Il faudrait pourtant sélectionner les types apparaissant le plus souvent et les lier aux mots. Cette technique n'assurerait pas l'analyse systématique de la phrase, car si le type nécessaire fait partie de ceux écartés, une phrase juste pourrait ne pas avoir d'analyse. Nous avons pris le parti de typer les mots soit en utilisant le Supertagger ((Moot, 2010a,b)), soit en utilisant les phrases typées à la sortie du transducteur. La première méthode nous permet à la fois de valider les types donnés aux mots par le Supertagger et d'analyser des phrases dont les mots n'apparaissent pas dans le corpus de Paris VII, tandis que la seconde méthode nous permet de tester nos différentes grammaires en fonction des arbres de dérivation. On peut aussi utiliser un typage plus manuel, qui utilise le Supertagger pour effectuer une première passe de typage et qui permet ensuite à l'utilisateur de modifier à loisir les types proposés.

4.2 Analyse des phrases typées

Pour l'algorithme de reconstruction des phrases, nous avons décidé d'utiliser l'algorithme CYK (Younger, 1967; Knuth, 1997; Hopcroft et Ullman, 1979) et d'en implémenter une version probabiliste : en effet, étant donné que cet algorithme a déjà été testé et est une référence, il nous a permis de tester l'efficacité de notre grammaire sans avoir à s'inquiéter de l'efficacité de l'algorithme. D'autres algorithmes auraient pu être utilisés, tel que celui d'Earley (Earley, 1973), cependant CYK demandait en entrée une grammaire très proche de celle que nous obtenions après extraction. De plus, l'ajout de l'aspect probabiliste était trivial sur cet algorithme. La seule modification que nous avons effectuée était de retirer la phase de typage des mots, initialement effectuée par CYK grâce aux règles de type $n_1 \rightarrow t_t$. Nous avons donc pu donc utiliser la grammaire la plus simple, de 3494 règles, pour analyser les phrases. Le premier test effectué, pour savoir si l'algorithme fonctionnait correctement, a été d'analyser les phrases extraites des arbres de dérivation avec les règles provenant de ces mêmes arbres. Nous avons ensuite pu tester l'analyse avec des phrases typées par le Supertagger ou notre transducteur et des grammaires extraites soit du corpus de 12351 phrases, soit du corpus de phrases laissées de côté (cf section 6).

Les arbres de dérivation correspondants aux phrases "Pourtant tout n'est pas gagné." et "Ce procès gagné donne au Crédit Lyonnais les coudées franches pour gérer MGM" sont montrés dans la figure 6 et 7. A chaque fois, on a pris les deux arbres les plus probables, typés par le Supertagger et les phrases ont été analysées avec la même grammaire et l'algorithme CYK. Deux informations sont intéressantes pour choisir quel est le meilleur arbre de dérivation sur les phrases : on regarde à la fois la complexité des types et la probabilité. Cependant, nous sommes conscient qu'il est complexe de comparer deux arbres qui n'ont ni la même structure, ni les mêmes feuilles. La préférence que l'on porte à un résultat sera fortement dépendante des critères de sélection donnés. Ainsi, sur la figure 6, on remarque que les deux arbres ont la même probabilité, cependant nous sélectionnons celui qui a l'indexation la plus faible pendant

l'exécution de CYK. Sur la seconde phrase (figure 7), c'est majoritairement l'attachement du groupe prépositionnel final qui modifie la forme de l'arbre. L'attachement de la préposition à un groupe nominal est plus représentatif du corpus d'origine (voir section 5) .

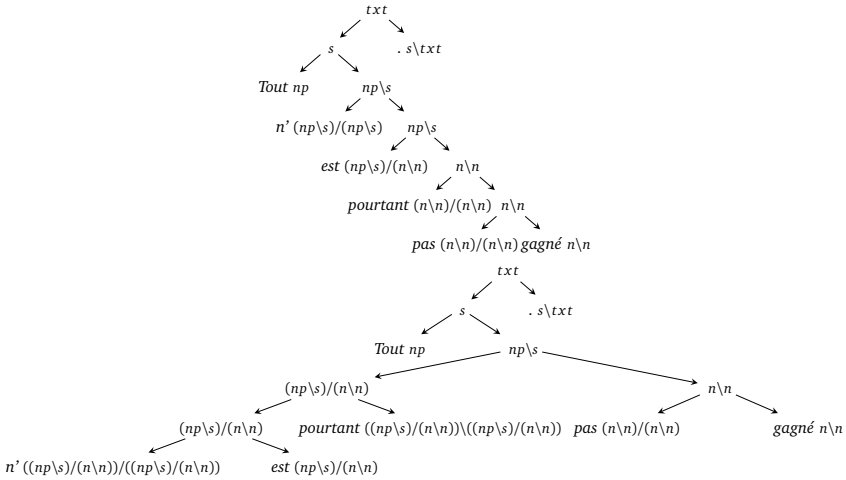

FIGURE 6 – Le premier arbre est généré avec le typage du transducteur et a une probabilité de $9,6x10^{-05}$ et le second est typé avec le Supertagger, avec une probabilité de $1,9x10^{-05}$

5 Analyse des prépositions

Nous allons nous focaliser sur l'analyse des syntagmes prépositionnels (*PP*, *PP-MOD*, *PP-OBJ* etc.) et de l'attachement par rapport à la phrase. Dans un premier temps, nous étudierons l'attachement des groupes prépositionnels dans le corpus d'origine, puis nous nous focaliserons sur les types des prépositions, via le transducteur et le Supertagger pour enfin nous pencher sur l'attachement dans les arbres de dérivation générés via l'algorithme CYK.

5.1 Attachement dans le corpus

Les groupes prépositionnels sont particulièrement nombreux dans le corpus (49039 occurrences). Comme nous pouvons le voir dans le tableau 3, ils sont majoritairement étiquetés *PP*. Leur attachement de départ dans le corpus est aussi particulièrement important, car c'est celui-ci qui définira le type de la préposition. Le tableau 4 résume la répartition des syntagmes prépositionnels dans le corpus, en fonction de leur parent. En effet, la transduction aura tendance à donner un type aux syntagmes prépositionnels qui correspond à leur place dans la structure de la phrase.

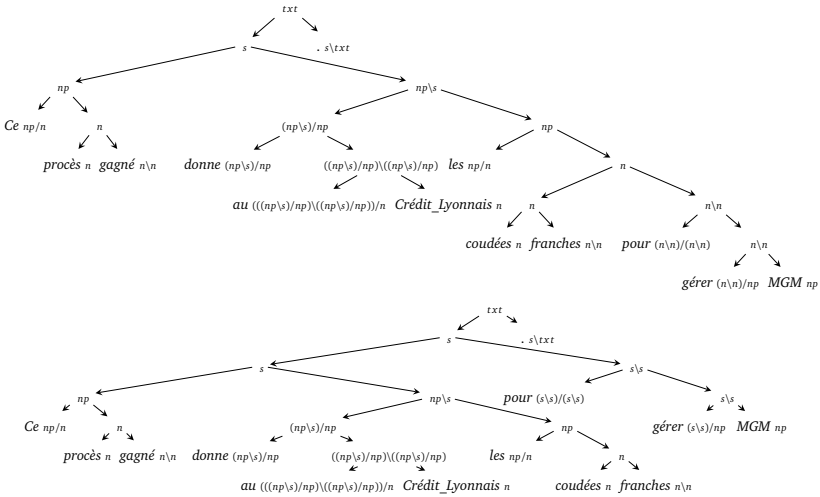

FIGURE 7 – Probabilité du premier arbre : $2,2x10^{-09}$. Probabilité du second arbre : $1,5x10^{-09}$.

Ainsi, dans un groupe nominal, le *PP* aura plus souvent le type $n\backslash n$, alors qu'au milieu d'une phrase le typage sera plus complexe. Lors de la transduction, on ne change pas l'ordre des mots, mais quelques fois leur attachement au sein de la structure. Cependant, on peut dire que les groupes prépositionnels ne bougent pas, sauf s'ils sont à l'extérieur d'un noyau verbal et que celui-ci se termine par un *VPP*, auquel cas on lie plus spécifiquement le participe passé au groupe prépositionnel, comme on peut voir figure 4.

Label	**occurrence**	**Label**	**occurrence**	**Label**	**occurrence**
PP	32023	*PP-MOD*	11899	*PP-DE_OBJ*	1668
PP-A_OBJ	1565	*PP-P_OBJ*	1389	*PP-ATS*	323
PP-OBJ	130	*PP-ATO*	30	*PP-SUJ*	12

TABLE 3 – Distribution des groupes prépositionnels en fonction de leur label.

5.2 Typage des syntagmes prépositionnels

Pour étudier le typage, nous nous sommes focalisés sur les groupes prépositionnels dont, bien sûr, la transduction avait réussi. Cela fait tomber le nombre de syntagmes prépositionnels à 45351 (92,5% du total). Les quatre familles de types les plus donnés (au dessus de 2000 fois) par le transducteur sont résumés dans le tableau 5. Ils couvrent 92,2% des types que l'on peut trouver pour des prépositions. Les types restants, marginaux, correspondent, par exemple, à un syntagme prépositionnel contenant uniquement un pronom relatif, qui prend en argument une

Syntagme parent	occurrence	Label le plus courant	pourcentage
Syntagme Nominal	24817	*PP*	99,2%
Phrase complète	8478	*PP-MOD*	72,2%
Proposition rel. ou sub.	3833	*PP-MOD*	57,9%
Proposition participiale	3552	*PP*	80,1%
Proposition infinitive	3190	*PP-MOD*	63,5%
Syntagme prépositionnel	843	*PP*	89,2%
Noyau verbal	45	*PP*	88,9%

TABLE 4 – Distribution des groupes prépositionnels en fonction de leurs parents. On remarque que les groupes nominaux sont ceux qui regroupent le plus de *PP* , c'est à dire presque la moitié.

subordonnée.

Le typage effectué avant l'analyse via CYK, avec le Supertagger, nous permet de régler la précision que l'on souhaite sur les types : en effet, on peut régler le paramètre β, qui déterminera le nombre de types possibles autorisés par mot. On gardera alors les types ayant une probabilité supérieure ou égale à β fois la plus grande probabilité trouvée [5]. Le tableau 6 résume la justesse des types donnés aux prépositions en fonction de β. On remarque que ce sont des mots difficiles à typer, étant donné que les résultats sont inférieurs aux résultats globaux, bien que les adverbes et les verbes soient encore plus complexes à typer de manière exacte.

Il faut cependant noter qu'il n'est pas nécessaire d'avoir une formule correcte pour que l'attachement du syntagme prépositionnel dans la phrase soit correct.

Famille de type	occurrence
n\n ou np\np ou n\np	23901
a\a (ex. s\s)	8548
pp ou pp_a ou $pp_d e$	6486
a/a (ex. s/s)	2882

TABLE 5 – Les quatre familles de types les plus courants correspondent à un modificateur de groupe nominal, un groupe prépositionnel généralement argument d'un groupe verbal et des modificateurs de phrase, placés au début ou à la fin de la phrase.

β	pertinence des types	pertinence globale
1.0	61,0%	76,9%
0.1	83,1%	87,0%
0.05	86,2%	88,9%
0.01	90,2%	91,7%

TABLE 6 – Justesse du typage via le Supertagger.

5. Plus β est petit, plus il y a de types proposés et plus on a de chance de trouver le type qui se combinera avec ceux des autres mots.

5.3 Attachement des syntagmes prépositionnels dans les arbres reconstitués

Pour cette partie, nous nous sommes focalisés sur 55 *PP*, que nous avons sélectionnés dans le corpus d'origine, de manière à respecter le ratio présenté dans le tableau 3. Cela correspond à 21 phrases, dont l'analyse a réussi. Nous avons généré les types possibles avec $\beta = 0.05$. Ensuite, nous avons étudié la différence de types donnés aux prépositions ainsi que leur attachement. On remarque, dans le tableau 7, que les syntagmes prépositionnels liés aux groupes nominaux sont attachés sensiblement au même endroit. On note une différence faible entre les groupes prépositionnels qui seront arguments d'un verbe, un peu plus importante entre les modificateurs globaux qui agissent sur toute la phrase. Il y a 4 cas, dans les arbres régénérés via CYK, où l'algorithme a jugé plus pertinent de préférer le type *n* ou *np* pour le syntagme prépositionnel ("On ne porte pas impunément atteinte *à des tabous*."), alors qu'on s'attend plutôt à une analyse qui lierait "atteinte" et "à" et qui prendrait en argument le groupe nominal "des tabous" [6].

On peut dire que le typage et l'attachement des syntagmes prépositionnels semblent cohérents avec l'attachement présent dans le corpus d'origine, ainsi que le typage effectué par le transducteur. Cependant, pour pouvoir l'affirmer, il faudrait faire des tests plus poussés, qui prendraient en compte la totalité du corpus.

Type	**occurrence après transduction**	**occurrence après CYK**
pp, pp_{de} ou pp_{a}	6	3
Modificateur de *NP*	35	37
Modificateur de *SENT*	9	4
np	0	4
Modificateur autre	5	7

TABLE 7 – Typage des prépositions dans le cadre d'une transduction comparées à celui effectué via le Supertagger avant reconstitution des arbres de dérivation avec CYK. Les modificateurs autres sont des modificateurs de proposition infinitive ou de syntagme adjectivaux.

Le typage, cependant, n'est pas entièrement lié à l'attachement dans la phrase. Nous avons comparé l'attachement des syntagmes prépositionnels et nous pouvons dire que, sur les 55 cas, il y en a 37 placés de manière identique et 18 non, soit 67,3% de ressemblance. Les différences majeures sont au niveau des prépositions qui sont plus souvent attachées aux groupes nominaux et argument des noyaux verbaux (ceux-ci peuvent alors prendre le type *np* plutôt que *pp*).

6 Évaluation et résultats

L'évaluation des différentes méthodes a été effectuée avec différents ensembles de données. Pour tester la totalité de nos travaux, nous avons utilisé le corpus de Paris VII dans son intégralité, c'est à dire :

– Les 12351 phrases parenthésées que nous avons étudiées en profondeur pour fonder l'ensemble de règles de notre transducteur, que nous appellerons *corpus principal*.

6. L'analyse CYK fait ressortir l'aspect idiomatique de "porter atteinte à".

– Les 504 phrases qui n'existaient pas sous forme parenthésée. Ce *corpus annexe* a été adapté pour être sous forme parenthésée et pour que les étiquettes soient celles utilisées par les règles.

Quel que soit le corpus utilisé, on parlera d'un *corpus partiel* pour dénoter le fragment dont la transduction a réussi. Les grammaires extraites des arbres de dérivation, donc des corpus partiels, auront le même nom que le corpus dont elles sont extraites, soit grammaire principale et grammaire annexe. L'évaluation du transducteur et de l'analyseur de phrases se mesure en pourcentage de phrases sur lesquelles l'opération a réussi. Dans le cadre du transducteur, cette notion correspond à la transformation des arbres syntaxiques en arbres de dérivation et dans le cadre de l'analyseur, elle correspond à la réussite de la combinaison des types donnés aux mots par le Supertagger.

6.1 Transducteur

Le transducteur transforme pour l'instant, avec 1671 règles, 92,6% du corpus principal (soit 11447 phrases) et 87,3% du corpus annexe (404 phrases) en arbres de dérivation d'une grammaire AB. On peut résumer l'utilisation des règles, dans le cadre de la transduction du corpus principal, dans le tableau 8. On remarque que, bien qu'il y ait de nombreuses règles qui sont utilisées peu de fois, elles ont un poids faible sur la totalité des transductions effectuées. Les règles les plus importantes sont exprimées dans le tableau 9, sous forme parenthésée telle qu'utilisée dans la syntaxe de *Tregex* (Levy et Andrew, 2006). La dernière règle, gérant la ponctuation finale, n'est pas utilisée autant de fois qu'il y a d'arbres de dérivation. Cela vient du fait que nous avons souhaité traiter différemment les phrases comprenant uniquement un groupe nominal et que certaines phrases, tels les titres d'articles, n'ont pas de ponctuation finale. De même, la règle qui s'occupe du déterminant au début d'un nom commun devrait être employée plus que ça, vu le nombre de groupes nominaux du corpus. Cependant, une règle prioritaire s'occupe du cas où le groupe nominal est composé d'un déterminant et d'un nom commun et est appelée 8892 fois.

Nombre de règles	Occurrence minimale et maximale	nombre d'applications
1148	entre 1 et 20	005818
303	entre 21 et 100	014174
170	entre 101 et 1000	054266
41	entre 1001 et 10000	**125405**
4	supérieur à 10000	060779

TABLE 8 – Récapitulatif de l'utilisation des règles.

motif de départ	motif d'arrivée	nombre d'applications
(*NP :* NC PP*)	(*NP :* NC :n PP :n*)*	17767
(*NP :* DET tree*)	(*NP :* DET :np/n NP :n*)	16232
(*PP :* P NP*)	(*PP :* P :*/np NP :np*)	16037
(*SENT tree PONCT*)	(TEXT :txt *SENT :s PONCT :s\ txt*)	10819
(*NP :* tree* (*COORD CC NP*))	(*NP :* (:* NP :**	
	(*COORD :** CC :(**)/np NP :np*)))	2511
(*SENT :* NP-SUJ VN NP-OBJ*)	(*SENT :* NP-SUJ :np*	
	(*:np* VN :(np*)/np NP-OBJ :np*))	1820

TABLE 9 – Quelques règles du transducteur, dont les quatre règles les plus utilisées.

Les arbres de dérivation des deux corpus nous permettent d'extraire deux grammaires, sur lesquelles les tests d'analyse que nous avons effectués seront détaillés dans la partie suivante 6.2. En addition des grammaires, nous pouvons créer un lexique, contenant les mots et les différents types qui leur sont associés en fonction des transductions. Le lexique correspondant au corpus principal contient 26765 mots sur les 27589 présents, il couvre donc 96,9% du vocabulaire présent dans le corpus de Paris VII.

6.2 Analyse de phrases

Nous avons effectué de nombreux tests avec notre analyseur de phrases. En effet, nous avons utilisé les deux grammaires différentes et nous avions à disposition des phrases typées par le transducteur ou par le Supertagger, avec $\beta = 0.01$.

Grâce au Supertagger, nous avons pu analyser aussi bien les phrases venant du transducteur que les phrases laissées de côté. Les résultats sont regroupés dans la table 10. On remarque que les résultats sont proportionnellement moins bons, mais que certaines phrases venant de la partie non traitée des différents corpus sont analysées et transformées en arbre de dérivation.

Origine des phrases	**Phrases analysées**	**Grammaire utilisée**	**Résultat**
Transducteur	Corpus principal partiel	Grammaire annexe	54,4%
	Corpus annexe partiel	Grammaire principale	85%
Supertagger	Corpus principal	grammaire principale	89,38%
	Corpus annexe	grammaire principale	83,1%

TABLE 10 – Tableau de résultat.

7 Conclusion et perspectives

Dans cet article, nous avons rapidement rappelé le principe du G-transducteur dont nous nous servons pour transformer les arbres syntaxiques du corpus de Paris VII en arbres de dérivation d'une grammaire AB, puis expliqué la méthode que nous employons pour extraire une PCFG de ces arbres. Les résultats expérimentaux d'analyse de phrase via l'algorithme CYK, en utilisant notre PCFG et des phrases typées au préalable, nous permettent de comparer les annotations produites par le transducteur et la méthode semi-automatique mise en place par Moot.

Cependant, ce travail est loin d'être terminé et nous avons encore plusieurs perspectives à étudier. Bien sûr, nous souhaitons améliorer la couverture du transducteur par rapport au corpus et dépasser les 95% de phrases analysées, bien qu'il ne reste plus que des cas complexes à traiter. Etant donné que les grammaires AB peuvent sembler limitatives lorsque l'on souhaite traiter d'une langue complexe, nous souhaiterions transformer notre transducteur en un transducteur d'arbres vers les graphes. Cela nous permettrait d'utiliser l'ensemble des règles de Lambek et de nous rapprocher de travaux plus modernes sur la question. Par rapport à l'analyseur de phrase, il manque cruellement d'un typage relatif au lexique que nous extrayons des arbres de dérivation. Cette méthode de typage devrait être implémentée rapidement. De même, il pourrait être intéressant d'utiliser d'autres algorithmes que CYK, tel que l'algorithme d'Earley, ou de typer les phrases en utilisant un système tel que SYGFRAN (Chauché, 2011).

Notre travail est disponible à (Sandillon-Rezer, 2012), sous licence *GNU General Public Licence*.

Références

Abeillé, A. et Clément, L. (2003). Annotation morpho-syntaxique.

Abeillé, A., Clément, L. et Toussenel, F. (2003). Building a treebank for french. *Treebanks, Kluwer, Dordrecht*.

Buszkowski, W. et Penn, G. (1990). Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49(4):431–454.

Chauché, J. (2011). Une application de la grammaire structurelle : L'analyseur syntaxique du francais sygfran.

Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S. et Tommasi, M. (2007). Tree automata techniques and applications. Available on : `http://www.grappa.univ-lille3.fr/tata`. release October, 12th 2007.

Earley, J. (1973). An efficient context-free parsing algorithm.

Hockenmaier, J. et Steedman, M. (2007). CCGbank : a corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, page 355–396.

Hopcroft, J. E. et Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley Publishing Company, Reading, Massachusets, USA.

Kanazawa, M. (1998). *Learnable Classes of Categorial Grammars*. Center for the Study of Language and Information, Stanford University.

Knight, K. et Graehl, J. (2005). An overview of probabilistic tree transducers for natural language processing.

Knuth, D. E. (1997). *The Art of Computer Programming Volume 2 : Seminumerical Algorithms (3rd ed.)*. Adison-Wesley Professional.

Lambek, J. (1958). The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3).

Levy, R. et Andrew, G. (2006). Tregex and tsurgeon : tools for querying and manipulating tree data structures.

Moortgat, M. et Moot, R. (2001). CGN to Grail : Extracting a type-logical lexicon from the CGN annotation. *In* Daelemans, W., éditeur : *Proceedings of Computational Linguistics in the Netherlands CLIN 2000*.

Moot, R. (2010a). Automated extraction of type-logical supertags from the spoken dutch corpus. *Complexity of Lexical Descriptions and its Relevance to Natural Language Processing : A Supertagging Approach*.

Moot, R. (2010b). Semi-automated extraction of a wide-coverage type-logical grammar for french. *Proceedings TALN 2010, Monreal*.

Sandillon-Rezer, N. (2012). Syntab : http ://www.labri.fr/perso/nfsr/.

Sandillon-Rezer, N.-F. et Moot, R. (2011). Using tree tranducers for grammatical inference. *Proceedings of Logical Aspects of Computational Linguistics 2011*.

Younger, D. (1967). Context free grammar processing in n^3.