Université Paris VII – Denis Diderot

UFR de Linguistique
Laboratoire de Linguistique Formelle
Ecole doctorale de Science du langage

N° attribué par la bibliothèque :
| | | | | | | | | | |
Année : 2005 - 2006

Extraction automatique d'une grammaire d'arbres adjoints à partir d'un corpus arboré pour le coréen.

Jungyeul Park

Thèse pour obtenir le grade de docteur de l'Université Paris 7 – Denis Diderot,
en Linguistique Théorique, Formelle et Automatique

soutenue publiquement le 29 mai 2006

Directrice :	Mme. Anne Abeillé	
Jury :	M. Eric Laporte	Rapporteur
	M. Alexis Nasr	
	M. Jean-Marie Pierrel	
	M. Owen Rambow	Rapporteur

Extraction automatique d'une grammaire d'arbres adjoints à partir d'un corpus arboré pour le coréen.

Jungyeul Park

Remerciements

J'adresse mes sincères remerciements à Anne Abeillé, ma directrice de recherche, pour ses compétences et sa rigueur, ainsi que pour son enthousiasme vis-à-vis de mon travail. Je lui suis reconnaissant de m'avoir accueilli au sein du laboratoire de linguistique formelle (LLF), où j'ai fait des rencontres enrichissantes.

Cette thèse n'aurait pas vu le jour sans l'aide constante d'Eric Laporte et d'Alexis Nasr. Je les remercie chaleureusement pour leurs lectures attentives et exigeantes, leurs remarques, leurs conseils, leur curiosité. Ils m'ont accompagné sans relâche durant toutes les années de mon doctorat.

Je tiens à exprimer ma sincère gratitude envers Owen Rambow qui a accepté d'être mon rapporteur, malgré les contraintes de temps que je lui ai fait subir. Je voudrais remercier chaleureusement Jean-Marie Pierrel pour avoir bien voulu faire partie de mon jury et avoir lu attentivement ma thèse dans le peu de temps dont il disposait.

Je tiens également à remercier Aravind Joshi, Martha Palmer et Chunghye Han pour leur accueil durant mes séjours à l'Institut de recherché en Sciences cognitives à Philadelphie et pour leur générosité de me permettre d'utiliser le Penn Korean Treebank.

A Nancy, les discussions que j'ai pu avoir avec les membres du Loria m'ont aidé beaucoup. Je tiens particulièrement à remercier Azim Roussanaly pour son accueil chaleureux et pour sa gentillesse de m'avoir installé le parseur LLP2.

J'avais toutes les chances d'avoir deux mentors : Cha Jungwon en informatique, Park Chulwoo en linguistique. Je voudrais les remercier pour ses précieux conseils qui m'ont beaucoup aidé tout au long de mes études.

Je tiens à remercier mes bons amis Khun Nghia et Nor-el-houda Arbaoui (Nora) d'avoir lu plusieurs chapitres de ma thèse en préparation et de m'avoir livré leurs réflexions sur mon travail. Je tiens également à remercier Ko Killsoo qui était toujours prêt à discuter de linguistique coréenne avec moi.

Je voudrais remercier à mes amis en France. Je pense en particulier Evi, Guillaume, François, Dina et Benjamine à l'Université de Paris VII, et aussi Xavier, Arnaud, Catherine, Stéphane, Michel, David à l'Université de Marne-la-Vallée. Leur présence m'a beaucoup apporté.

Mes remerciements vont aussi aux les coréens en France pour leur soutient sans faille, Park Yonghyang, Choi Kyungil, Kim Sejong, Choi Yujin, Chang Kiwon, Kim Hyunjong, Kim Dohyung, Lee Kyungja, Kim Jaewan, Song Taejoon, Woo Ik, Choi Sujin, Bae Chulmin, et Kim Myungsuk, et à mes amis en Corée, notamment Kim Hongseok, Jung Jiyoung, Jang Seokwoo, Jang Seokju, Ryu Changsung, Oh Youshin et Shim Heekyun pour leur amitié.

Merci à ma famille qui depuis tant d'années m'encourage et dont la chaleur, l'enthousiasme et le soutien m'ont permis de garder le moral.

Je dois aussi dire que ma fille Emma et mon fils Daniel qui m'ont apporté beaucoup de joie de vivre. Enfin, merci à ma femme Youjeong.

Introduction

La grammaire électronique est une interface entre la complexité et la diversité de la langue naturelle et la régularité et l'efficacité du traitement automatique de la langue (TAL). Elle est une de ressource très importante pour le traitement automatique des langues naturelles (Abeillé & Blache 2000, Abeillé 2002). Une grammaire à grande échelle peut incorporer beaucoup d'informations concernant le lexique, la syntaxe et la sémantique d'une langue humaine et un énorme effort humain est nécessaire pour la construire et la maintenir

Le développement manuel de la grammaire électronique à grande échelle pose enfin des problèmes non triviaux de partage des tâches, de maintenance et de mise à jour, d'interfaçage avec d'autres ressources, notamment lexicales (Abeillé & Blache 2000). Ils signalent que « Pendant longtemps, les seules grammaires électroniques de qualité représentaient le travail d'un ou deux individus pendant 10 ans et étaient étroitement dépendantes de l'application (par exemple la traduction automatique) et des programmes de traitement, ce qui obligeait à modifier les grammaires chaque fois qu'on modifiait les programmes» (Abeillé & Blache 2000). Ils citent les grammaires en chaîne de Salkoff (1973) et Sager (1981), et pour l'anglais, les grammaires industrielles de Logos, d'IBM ou de Metal.

Abeillé & Blache (2000) mentionnent aussi que « La stabilisation des modèles syntaxiques basés sur l'unification a permis le développement de plates-formes d'implantation (workbench) multilingues, librement accessibles, telles que XTAG (Paroubek et al. 1992), ALE (Carpenter 1992), LKB (Copestake 2002) etc., qui incorporent des outils de test et de mise à jour et permettent le développement de grammaires neutres quant à leur application ». Pour l'anglais, les projets XTAG (Doran *et al.* 2000), ou les projets Lingo et Verbmobil (Copestake 2002, Kay *et al.* 1994) et pour l'allemand, le projet Babel (Müller 1999). Pour le

français, on peut citer les lexiques-grammaires de l'équipe de M. Gross (Gross 1975) qui ne sont pas directement formalisés pour l'analyse syntaxique, mais constituent une base de connaissances lexico-syntaxiques irremplaçable, la grammaire FTAG développée à Paris 7 (Abeillé 1991, Candito 1999, Abeillé et Candito 2000, Abeillé 2002) et la grammaire HPSG de Tseng (2003).

Il reste que le coût de développement de telles grammaires est élevé. A mesure que la taille des grammaires s'accroît, le développement manuel fait face à une série de problèmes. La thèse de Xia (2001) mentionne ces problèmes de la façon suivante :

- L'effort humain : le processus de construction de la grammaire est un travail très intensif qui prend beaucoup de temps.

- Flexibilité : puisque la construction de la grammaire nécessite beaucoup d'efforts humains, il est presque impossible pour les développeurs de grammaires de fournir un ensemble de grammaires différentes pour une même langue humaine, de sorte que les utilisateurs `de grammaires puissent choisir celles qui sont les plus adaptées à leurs applications.

- Couverture : Il est difficile d'évaluer la couverture d'une grammaire construite manuellement sur des données naturelles. La technique la plus courante pour évaluer une grammaire consiste à créer une série de tests et à vérifier si la grammaire peut analyser des phrases grammaticales et rejeter des phrases non grammaticales dans cette série de tests. Il est difficile d'étendre cette méthode d'évaluation à un vaste ensemble de données naturelles[1].

[1] Doran *et al.* (1994) et Srinivas *et al.* (1998) prennent les données brutes, c'est-à-dire un ensemble de phrases sans annotation syntaxique pour évaluer la grammaire XTAG, une grammaire construite manuellement. Les données sont analysées par un parseur de la grammaire TAG lexicalisée et la couverture de la grammaire est mesurée comme le pourcentage des phrases dans les données qui a au moins une analyse, ce qui est nécessairement une analyse correcte (Prasad et Sarkar 2000).Parmi les tests suites, on peut citer celle de TSNLP (voir le site de TSNLP : http://cl-www.dfki.uni-sb.de/tsnlp/), et celle de CSLI LKB (Copestake 2002).

- Information statistique : Dans la grammaire construite à la main, aucun poids n'est associé aux éléments primitifs. Pour utiliser la grammaire à des fins de parsing, d'autres ressources telles les règles heuristiques doivent être trouvées pour nous aider à sélectionner les arbres de dérivation les plus probables.

- Cohérence : Les éléments primitifs d'une grammaire partagent souvent des structures communes. Par exemple, les éléments primitifs d'une grammaire TAG lexicalisée sont appelés les arbres élémentaires. Les structures pour différentes constructions comme le *wh*-mouvement en anglais apparaissent dans plusieurs arbres élémentaires. Pour garantir des changements dans la grammaire, tous les éléments primitifs concernés doivent être vérifiés manuellement. Ce processus est inefficace et ne peut pas garantir la cohérence.

- Généralisation : Souvent, l'information linguistique sous-jacente (telles que la description du *wh*-mouvement) n'est pas exprimée explicitement. En conséquence, à partir de la grammaire elle-même, il est difficile de saisir les caractéristiques d'une langue particulière, de comparer des langues et de construire la grammaire d'une nouvelle langue à partir des grammaires préexistantes d'autres langues.

Parce que le développement manuel d'une grammaire est une tâche coûteuse qui prend beaucoup de temps, beaucoup d'efforts pour le développement semi-automatique et automatique de la grammaire ont été fournis pendant la décennie dernière.

Le développement semi-automatique d'une grammaire signifie qu'un système génère les structures finales en utilisant la description des variations syntaxiques (ou linguistiques) et des contraintes d'une langue spécifique. La métagrammaire de Candito (1999) et Crabbé (2005) et la description d'arbres de Xia (2001) sont les bons exemples du développement semi-automatique d'une grammaire TAG. Les métagrammaires qui sont des hiérarchies de descriptions partielles, permettent une génération semi-automatique de grammaires TAG, puisque c'est un compilateur qui combine ces descriptions en autant d'arbres élémentaires (Candito 1999, Abeillé 2000, Xia 2001, Duchier *et al.* 2005, Thomasset et de la Clergerie 2005). Même avec le développement semi-automatique, nous avons besoin de bonnes descriptions des phénomènes linguistiques pour chaque langue, ce qui nécessite une

connaissance de haut niveau de la linguistique. En plus, avec la grammaire générée semi-automatiquement on a facilement un problème de débordement

Le développement automatique d'une grammaire signifie qu'un système extrait une grammaire à partir d'un corpus arboré qui a une grammaire implicite. Le système d'extraction d'une grammaire prend les phrases analysées syntaxiquement comme entrée et produit une grammaire cible. La grammaire extraite est soit, la même que la grammaire du corpus arboré, soit elle est différente et ce selon le but particulier des utilisateurs. Parmi les grammaires extraites automatiquement, on peut cite celle de Chen (2001) qui extrait une grammaire TAG lexicalisée à partir du corpus Penn Treebank et celles des autres travaux basés sur la procédure de Chen (2001) telles que Nasr (2004) et Johansen (2004) pour le français et Habash et Rambow (2004) pour l'arabe. On peut aussi citer celle de Chiang (2000) qui extrait une variation de TAG telle qu'une grammaire d'insertion d'arbres, celles de Xia (2000) qui développe un système d'extraction des grammaires TAG lexicalisée pour l'anglais, le chinois et le coréen, celles de Neumann (1998 et 2003) qui extrait une grammaire à partir du corpus Penn Treebank pour l'anglais et à partir du corpus NEGRA pour l'allemand, et celle de Frank (2001) qui applique sa méthode d'extraction au corpus allemand NEGRA. On peut aussi citer les travaux d'extraction pour LFG (Cahill et al. 2003) et pour HPSG (Simov 2001).

La grammaire extraite automatiquement a l'avantage de la cohérence et de la rapidité de son développement. Cependant, comme il est dépendant du corpus arboré où le système d'extraction est employé, la couverture du lexique est limitée à l'échelle du corpus arboré. De plus, le corpus arboré fiable n'est guère trouvé, surtout dans un domaine public.

La différence entre le développement manuel, semi-automatique et automatique d'une grammaire est résumée dans le Tableau 1. A partir de ce tableau, il est clair que le développement semi-automatique et/ou automatique a plus d'avantages que le développement manuel.

	Développement manuel	Développement semi-automatique	Développement automatique
Effort humain	énorme	moyen	petit
Flexibilité	très faible	moyen	faible
Couverture	difficile à évaluer	paramétrable	couvre la source du corpus arboré
Information statistique	non disponible	non disponible	disponible
Cohérence	non garantie	cohérent	cohérent
Généralisation	cachée dans les arbres élémentaires ou les règles	exprimée explicitement	cachée dans les arbres élémentaires ou les règles
Famille d'arbres et traits	oui	oui	non ?

Tableau 1. Comparaison entre l'approche traditionnelle et le développement semi-automatique et automatique

Puisque nous pouvons extraire une grammaire automatiquement sans beaucoup d'efforts si le corpus arboré fiable est fourni, nous réalisons un système qui extrait une grammaire d'arbres adjoints lexicalisée et une grammaire d'arbres adjoints lexicalisée avec traits à partir du corpus arboré *Sejong* (SJTree) pour le coréen dans cette thèse. Le corpus SJTree contient 32.054 *eojeols* (l'unité de la segmentation dans la phrase coréenne), c'est-à-dire 2.526 phrases avec 69.240 morphèmes (Sejong Project 2003). Le corpus SJTree emploie 42 étiquettes de partie de discours et 55 étiquettes syntaxiques.

Le système d'extraction d'une grammaire qui va être présenté dans cette thèse, a plusieurs avantages. D'abord, le système est totalement indépendant du corpus et de la langue. Etant donné un nouveau corpus arboré, un expert en linguistique n'a besoin que de peu de temps pour fournir l'information spécifique pour chaque corpus et chaque langue, par exemple : un ensemble d'étiquettes, la percolation de la tête, et les arguments de la tête. Deuxièmement, le système donne à ses utilisateurs la possibilité d'un contrôle sur le type de la grammaire de corpus arboré extraite. Les utilisateurs peuvent exécuter le système avec des réglages différents afin d'obtenir plusieurs grammaires de corpus arboré différentes, puis choisir la plus convenable. Troisièmement, le système ne produit pas seulement une grammaire de corpus arboré, mais aussi les informations concernant la fréquence de la combinaison de certaines structures élémentaires pour former des structures syntaxiques. Cette information peut être mise à profit pour concevoir des parseurs statistiques.

Ensuite, nous choisissons un formalisme avec lequel le système d'extraction produit une grammaire cible. Plusieurs formalismes ont été proposés pour les langues naturelles, par exemple : la grammaire lexicale fonctionnelle ('Lexical Functional Grammar' ou 'LFG') (Bresnan 2001, Dalrymple 2001), la grammaire syntagmatique guidée par les têtes ('Head-Driven Phrase Structure Grammar' ou 'HPSG') (Sag et Pollard 1987, Sag et Pollard 1989, Sag *et al.* 2003), et la grammaire catégorielle combinatoire ('Combinatory Categorial Grammar' ou 'CCG') (Steedman 1996, Steedman 2000, Steedman et Baldridge 2003). Dans cette thèse, nous choisissons le formalisme de la grammaire d'arbres adjoints lexicalisée ('Lexicalized Tree-Adjoining Grammar' ou 'LTAG') (Joshi *et al.* 1975, Schabes 1990, Abeillé et Rambow 2000) car ses propriétés linguistiques et informatiques sont attirantes pour représenter divers phénomènes dans les langues naturelles et que ce formalisme a été utilisé dans plusieurs aspects de la compréhension des langues naturelles, par exemple : le parsing (Schabes 1990; Srinivas 1997), la syntaxe (Frank 2002, Abeillé 2002), la sémantique (Joshi et

Vijay-Shanker 1999; Kallmeyer et Joshi 1999), la sémantique lexicale (Palmer et al. 1998, Kipper et al. 2000), le discours (Webber et Joshi 1998; Webber et al. 1998)), et un certain nombre d'applications du TAL, par exemple : la traduction automatique (Palmer et al.1998), la recherche d'informations (Chandrasekar et Srinivas 1997), la génération (Stone et Doran 1997; McCoy et al. 1992), et le résumé (Baldwin et al 1997).

Le formalisme des TAG lexicalisées pourrait générer une langue formelle comme $\{a^n b^n c^n\}$ en plus des langues naturelles. Puisque son usage n'est pas limité aux langues naturelles, le formalisme lui-même n'impose aucune contrainte sur les arbres élémentaires dans la grammaire TAG lexicalisée à l'exception des prérequis fondamentaux indiquant que chaque arbre élémentaire doit être ancré par un item lexical et qu'un arbre auxiliaire doit avoir exactement un nœud pied de même catégorie que le nœud racine. Dans cette thèse, nous nous intéressons seulement aux grammaires du coréen.

Dans le chapitre 1, nous commençons par une présentation brève du formalisme TAG et puis nous présentons les travaux d'extraction. Plusieurs travaux ont été faits pour extraire une grammaire d'arbres adjoints lexicalisée à partir d'un corpus arboré.

Le chapitre 2 présente le corpus *Sejong* Korean Treebank (le corpus SJTree) à partir duquel une grammaire TAG lexicalisée est extraite. C'est un corpus arboré, annoté syntaxiquement pour le coréen. Nous en décrivons en particulier le schéma d'annotation, qui est basé sur l'analyse linguistique. Nous présentons également le schéma d'annotation du Penn Treebank pour le coréen (Penn Korean Treebank) et comparons les schémas des deux corpus. La comparaison nous permet de relever les différences entre les deux corpus susceptibles d'impliquer des modifications dans la procédure d'extraction.

Nous proposons un algorithme d'extraction d'une grammaire TAG lexicalisée à partir d'un corpus arboré dans le chapitre 3. L'algorithme est implémenté et testé sur le corpus SJTree. Nous présentons les détails de l'algorithme et le résultat des expériences d'extraction.

Dans le chapitre 4, nous extrayons les traits. L'extraction automatique des traits n'est pas encore réalisée dans les autres travaux d'extraction. Ici, nous proposons une procédure d'extraction de traits en deux étapes qui permet d'établir une grammaire TAG lexicalisée avec traits pour le coréen : la première est la conversion d'étiquettes syntaxiques du corpus en

catégories plus traits, et la deuxième est l'ajout d'équations plus générales sur ces mêmes traits en se basant sur la notion d'épine dorsale.

Dans le chapitre 5, les grammaires extraites sont évaluées par leur taille, leur couverture et leur ambiguïté moyenne. La taille des grammaires est le nombre d'arbres lexicalisés et le nombre de catégories (les schémas d'arbre). Le nombre de schémas apparaissant au moins deux fois dans le corpus est quasiment stabilisé à l'issue de l'extraction et le nombre de schémas des grammaires supérieures (celles qui sont extraites après la modification du corpus) est aussi plus stabilisé que les grammaires inférieurs. La couverture des grammaires, étant donné un corpus, est le rapport du nombre d'occurrences de catégories inconnues dans le corpus à la taille du corpus (le nombre total d'occurrences de mots dans le corpus). L'ambiguïté moyenne des grammaires est la moyenne du nombre d'analyses différentes que la grammaire associe aux phrases du corpus. En plus, nous évaluons notre programme d'extraction en l'appliquant à un autre corpus arboré (le Penn Korean Treebank).

Nous présentons la sous-catégorisation des verbes et des adjectifs en coréen basée sur la grammaire extraite dans le chapitre 6. Nous ne donnons pas seulement la sous-catégorisation initiale, mais aussi les variations syntaxiques que nous avons extraite à partir du corpus arboré.

Dans la conclusion, nous résumons la thèse et signalons les indications de travaux futurs.

Enfin, cette thèse inclut cinq annexes. L'annexe A présente une statistique du corpus SJTree, c'est-à-dire la distribution des étiquettes POS et des étiquettes syntaxiques. L'annexe B résume un algorithme d'extraction dans cette thèse. L'annexe C fourni un tableau complet de conversion d'étiquettes syntaxiques. L'annexe D est un exemple d'une entrée et d'une sortie de LLP2. L'annexe E montre un *chunker* (ou un *shallow* parseur) qui prend une phrase annotée morphologiquement comme entrée et produit une structure plate comme sortie. Le *chunker* est considéré comme un prétraitement pour diminuer la longueur des entrées et améliorer la vitesse de l'analyseur syntaxique.

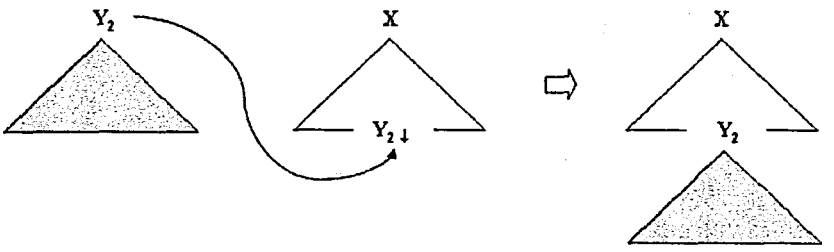
Chapitre 1. Grammaire TAG lexicalisée et travaux d'extraction

Dans ce chapitre nous allons présenter les travaux d'extraction de grammaries. Plusieurs travaux ont été faits pour extraire une grammaire d'Arbres Adjoints (en anglais 'Tree Adjoining Grammar' ou TAG) à partir d'un corpus arboré. Le but de l'extraction automatique est premièrement, un développement des grammaires TAG et deuxièmement, une estimation des paramètres probabilistes à partir des dérivations reconstruites (Johansen 2004). Nous présentons d'abord, brièvement le formalisme TAG et puis nous décrirons les travaux d'extraction.

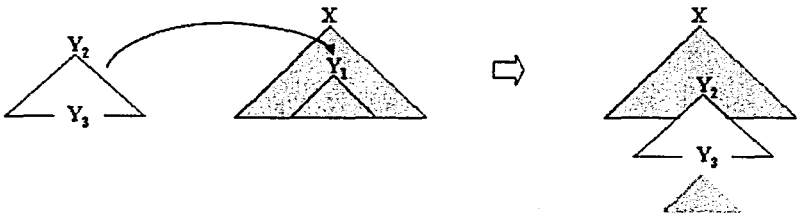
1. Grammaire TAG lexicalisée

Les éléments primitifs de la grammaire TAG lexicalisée sont les arbres élémentaires. Chaque arbre élémentaire est associé, sur son nœud frontière, à un élément du lexique que l'on appelle l'ancre de l'arbre. Nous avons choisi une TAG lexicalisée comme une grammaire cible, c'est-à-dire une grammaire extraite, parce que la grammaire TAG lexicalisée est un formalisme qui possède beaucoup de propriétés désirables pour modéliser les langues naturelles. Par exemple, la grammaire TAG permet aux dépendances d'être localisées et à la récursivité d'être factorisée. De plus, la grammaire TAG est lexicalisée, donc les contraintes syntaxiques sont associées directement aux éléments lexicaux. Aussi, la grammaire TAG est une grammaire « légèrement contextuelle ('mildly context sensitive'), » c'est-à-dire qu'elle est assez expressive pour capturer les phénomènes syntaxiques qui se trouvent dans plusieurs langues naturelles.

Il y a deux types d'arbres élémentaires : l'arbre initial et l'arbre auxiliaire. Celui-ci représente une structure récursive et a un nœud non-terminal, qui s'appelle le nœud pied et qui est du même type syntaxique que le nœud de la racine (voir les arbres $\beta_{1\text{-}3}$ dans la Figure (2a)). Les nœuds autres que le nœud ancré et le nœud pied sont les nœuds de substitution (voir aussi les arbres α_3 dans la Figure (2a)). Les arbres élémentaires se combinent par deux opérations décrites dans la Figure 1. Cette combinaison des arbres élémentaires nous donne un arbre dérivé. Le processus de la combinaison est montré par un arbre de dérivation.



a. Opération de substitution



b. Opération d'adjonction

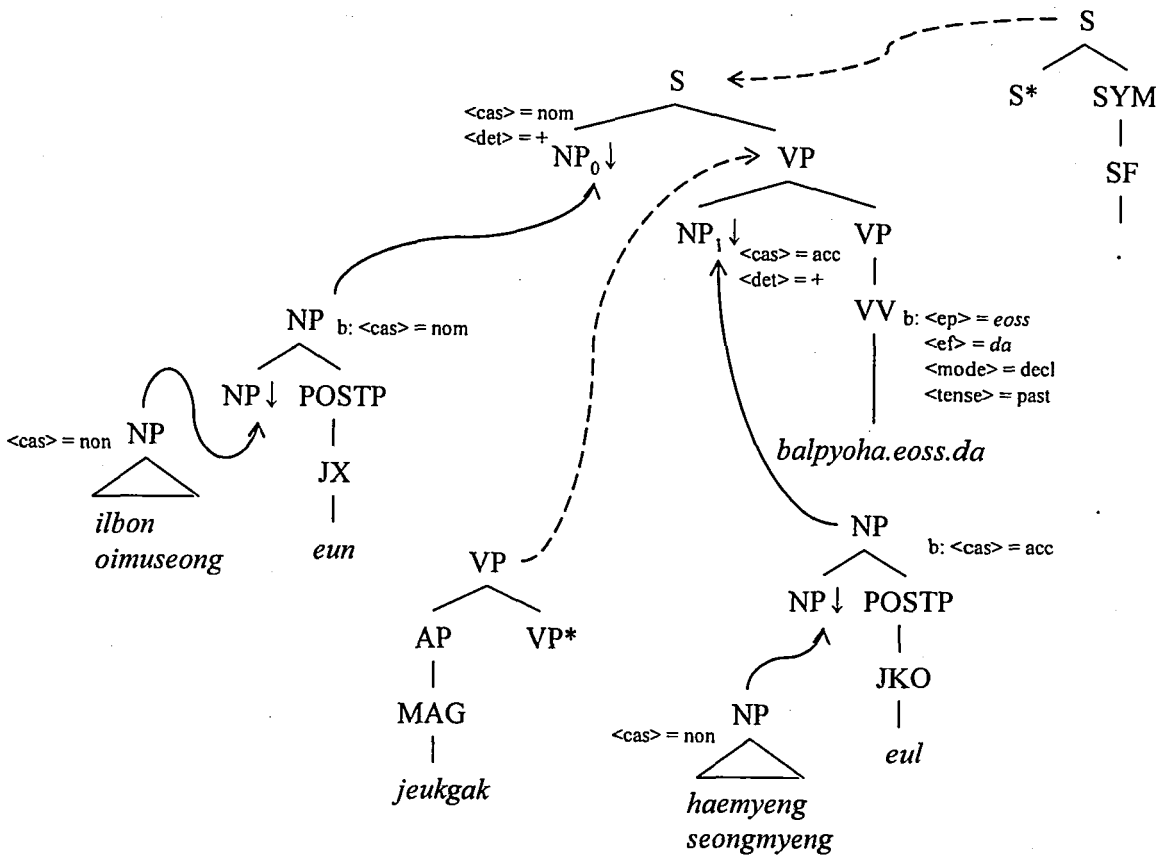
Figure 1. Opération de substitution et d'adjonction dans le formalisme TAG

La Figure 2 nous montre les arbres élémentaires, l'arbre dérivé et l'arbre de dérivation pour la phrase en (1) *ilbon oimuseong.eun jeukgak haemyeng seongmyeng.eul balpyoha.eoss.da* ('Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation'). Ici, la substitution et l'adjonction sont marquées, respectivement, sur les nœuds de substitution et pieds par une flèche vers le bas (↓) et un astérisque (*) dans l'arbre dérivé. La ligne continue et la ligne pointillée dans l'arbre de dérivation sont, respectivement, pour l'opération de

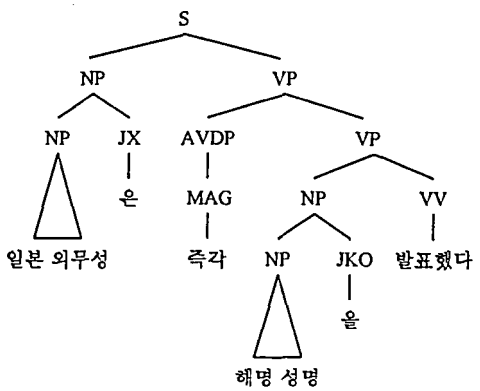
substitution et d'adjonction. Dans la Figure 2, les arbres initiaux $\alpha 1$ et $\alpha 2$ sont pour les noms. Cette thèse n'analyse pas la structure du syntagme nominal. Pour cette raison nous montrons ici une structure plate pour le syntagme nominal. Pour les détails sur le syntagme nominal, voir le chapitre 3 : la procédure d'extraction.

(1) 일본 외부성은 즉각 해명 성명을 발표했다.

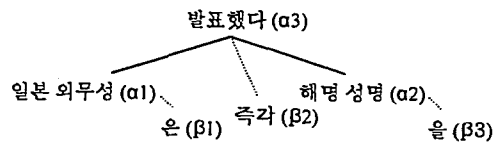
ilbon *oinuseong.eun* *jeukgak* *haemyeng*
Japon ministre_des_affaires_étrangères.Nom immédiatement élucidation
seongmyeng.eul *balpyoha.eoss.da*
déclaration.Acc annoncer.Pass.Ter
'Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation.'



a. Arbres élémentaires



b. Arbre dérivé



c. Arbre de dérivation

Figure 2. Arbres élémentaires, arbre dérivé et arbre de dérivation pour la phrase en (1)

2. Travaux d'extraction d'une grammaire TAG lexicalisée

Dans cette section nous allons situer les principaux travaux d'extraction préexistants pour la grammaire TAG lexicalisée. Plusieurs travaux ont été faits pour extraire une grammaire TAG à partir d'un corpus arboré ou d'un corpus annoté. On peut citer ceux de :

- Chen (2001) qui extrait une grammaire TAG lexicalisée à partir du corpus Penn Treebank et les autres travaux qui sont basés sur la procédure de Chen (2001) tels que Nasr (2004) et Johansen (2004) pour le français et Habash et Rambow (2004) pour l'arabe,

- Chiang (2000) qui extrait une variation de TAG telle qu'une grammaire d'insertion d'arbres,

- Xia (2001) qui développe un système d'extraction des grammaires TAG lexicalisées pour l'anglais, le chinois et le coréen,

- Neumann (2003) qui extrait une grammaire à partir du corpus Penn Treebank pour l'anglais et le corpus NEGRA pour l'allemand.

Après cette présentation des travaux précédents d'extraction de la grammaire TAG lexicalisée, nous élaborons un résumé de comparaison de ces travaux.

2.1 Grammaire TAG lexicalisée à partir d'un corpus arboré - Chen (2001)

Chen (2001) et Chen et Vijay-Shanker (2000) ont proposé un algorithme pour extraire une grammaire TAG lexicalisée à partir d'un corpus arboré. Cet algorithme a été appliqué sur le Penn Treebank (Taylor et al. 2003). L'arbre lexicalisé γ par le mot $\omega \in S$, où S est une phrase annotée du corpus Penn Treebank, est extrait comme suit.

D'abord, « un tableau de percolation de la tête » est utilisé pour déterminer le tronc de γ. Le tableau de percolation de la tête, qui est introduit dans Magerman (1995), assigne un mot de la tête (*'headword'*) à chaque nœud de S en utilisant l'information structurelle locale. Le tronc de γ est défini comme le parcours à travers S dont les nœuds sont étiquetés avec le mot de tête

w, un exemple de ce procédé est décrit dans la Figure (3b). Chaque nœud η' qui est immédiatement dominé par un noeud η du tronc, est soit lui même dans le tronc, soit il est un complément de la tête du mot dans le tronc – auquel cas il appartient à γ, soit il est un adjoint du mot de tête du tronc – auquel cas il appartient à un autre arbre (auxiliaire) β qui modifie γ.

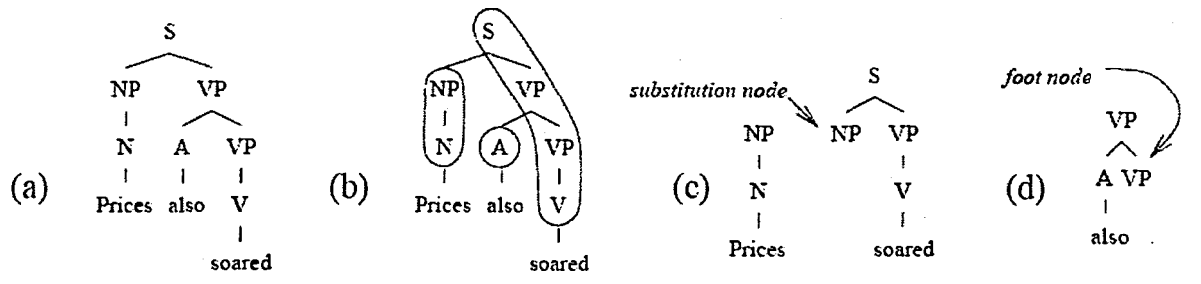


Figure 3. a. Structure phrastique, b. Structure phrastique où les non-terminaux appartenant au même tronc ont été entourés, c. Localisation de la dépendance des arguments dans le même arbre élémentaire, d. Chaque instance récursive est factorisée dans un arbre élémentaire séparé[2].

Donc, il est nécessaire de déterminer le statut d'un nœud en tant que complément ou adjoint. Collins (1997) introduit une procédure qui détermine cela à partir du corpus arboré en se basant sur les étiquettes du nœud, les étiquettes sémantiques et l'information structurelle locale. Comme décrit dans Marcus *et al.* (1994), les étiquettes sémantiques d'un nœud fournissent des informations très utiles pour déterminer le statut du nœud, par exemple sa fonction grammaticale et son rôle sémantique.

La procédure de Chen (2001) pour identifier le complément ou l'adjoint, effectivement, suit dans une large mesure la méthode de Collins (1997). La différence principale entre le travail de Chen (2001) et celui de Collins (1997) se trouve dans le fait que Chen essaye de traiter les nœuds comme des compléments qui sont typiquement localisés dans les arbres de TAG lexicalisée. Une des principales différences par rapport à Collins (1997) est dans le traitement de la cible (en anglais '*landing site*') du *wh*-mouvement. La procédure de Collins (1997) traite la cible du syntagme nominal comme étant la tête et sa sœur (typiquement étiqueté en *S*) comme étant un complément. Dans la procédure de Chen (2001) pour l'extraction d'arbres de

[2] Les Figures de 3 à 5 sont de Chen et Vijay-Shanker (2000).

TAG lexicalisés, il projette un élément lexical au parcours de têtes. Alors, en adoptant la méthode de Collins (1997), la cible serait sur le parcours de la projection, et conformément à la procédure d'extraction de Chen (2001) le *wh*-mouvement ne serait pas localisé. Par conséquent, Chen (2001) traite la cible de la sœur (le nœud S) comme étant le fils de tête et celui du syntagme nominal comme étant un complément. La Figure (5c) montre un exemple d'arbre lexicalisé extrait par cette méthode qui localise un mouvement de longue distance.

Chen (2001) a conduit des travaux expérimentaux sur deux procédures pour déterminer le statut d'un nœud en tant que complément ou adjoint. La première procédure, notée « CA1 », utilise les étiquettes fonctionnelles et sémantiques du noeud η et du parent de η dans une procédure en deux étapes. Dans la première étape, les étiquettes fonctionnelles et sémantiques sont utilisées telles que elles sous la forme d'un index qui détermine le statut de complément ou d'adjoint. 'Si le nœud actuel est PP-DIR et que le noeud parent est VP, alors assigner l'adjoint au noeud actuel' est un exemple d'entrée du tableau car ce tableau est clairsemé, l'index ne se trouve pas dans le tableau, la deuxième procédure est invoquée :

1. Le noeud non-terminal PRN est un adjoint.

2. Les noeuds non-terminaux avec l'étiquette fonctionnelle NOM, DTV, LGS, PRD, PUT, SBJ sont des compléments.

3. Les noeuds non-terminaux avec l'étiquette sémantique ADV, VOC, LOC, PRP sont des adjoints.

4. Si aucune autre condition ne s'applique, le nœud non-terminal est un adjoint.

Alors que « CA1 » utilise les étiquettes fonctionnelles et sémantiques d'un nœud η et de son parent η', « CA2, » la procédure décrite dans Xia (2001) utilise les étiquettes fonctionnelles et sémantiques d'un nœud η, sa sœur de tête η_h, et la distance entre η et η_h pour déterminer le statut de complément ou d'adjoint du nœud η. « CA2 » s'appuie sur deux tableaux manuellement construits : un tableau d'arguments et un tableau d'ensemble d'étiquettes.

Le tableau d'arguments suppose que η est un complément en fonction des étiquettes fonctionnelles et sémantiques de η et de η_h, et de la distance les séparant. Par exemple, si η est étiqueté par NP, alors le tableau d'arguments suppose que η si η_h est étiqueté par VB et qu'il y

a moins de quatre nœuds séparant η *de* η_h. Le tableau d'ensemble d'étiquettes suppose que η est un complément en se basant sur la seule étiquette sémantique de η. Si le tableau d'arguments et le tableau d'ensemble d'étiquettes supposent tous deux que η doit être un complément, il est étiqueté en tant que tel. Sinon, il est étiqueté en tant qu'adjoint.

Une procédure récursive est utilisée pour extraire les arbres par un algorithme ascendant (*'bottom-up algorithm'*) étant donné un corpus arboré. La Figure (4a) montre une étape de cette procédure. Parmi les filles du nœud η_2, une fille η_1 est sélectionnée en utilisant un tableau de percolation de la tête de sorte que le tronc φ associé avec η_1 soit étendu jusqu'à η_2. Les sœurs de η_1 sont ensuite étiquetées comme étant complément ou adjoint. Les nœuds compléments sont attachés au tronc φ et les arbres qu'ils dominent deviennent les arbres initiaux. Les nœuds adjoints sont factorisés sous forme d'arbres auxiliaires, de sorte que les plus éloignés de η_1 s'adjoignent à η_2 et que les plus proches de η_1 s'adjoignent à η_1, ainsi que décrit dans la Figure (4b). Pour Chen (2001), ce sont des arbres auxiliaires modifieurs. Bien que la structure définie par la grammaire résultante puisse différer du corpus original (voir la Figure (4c)), aucune annotation modifiée ne contredit celles de la structure du corpus arboré original. Ceci est important pour Chen (2001) dont le but est de comparer le parseur basé sur cette grammaire avec les autres parseurs testés sur le corpus Penn Treebank. Cette factorisation tend à réduire le nombre d'arbres dans la grammaire résultat. Par exemple, les arbres extraits dans la Figure (4b) peuvent être utilisés pour représenter non seulement la phrase *Later prices drastically fell*, mais aussi d'autres phrases telles que *Prices fell* et *Later prices fell* (Chen 2000).

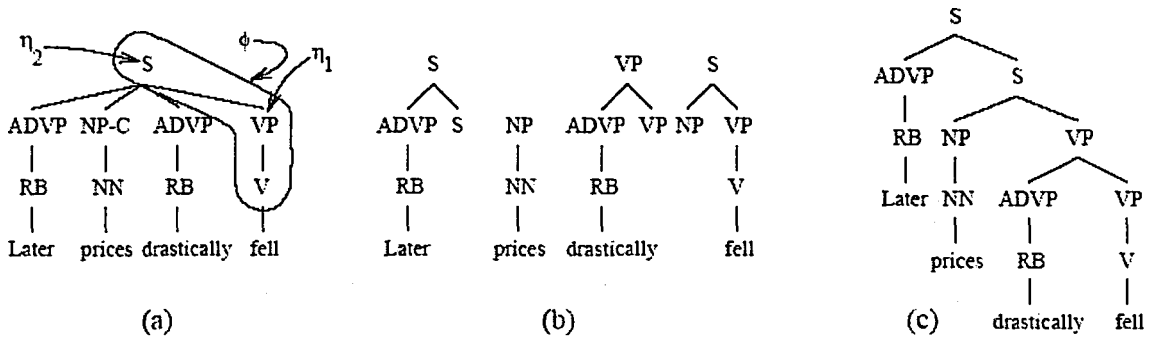


Figure 4. a. Corpus arboré original avec la sœur de la tête η_1 et son parent η_2 les deux sur le tronc du mot de la tête *fell* et les sœurs de l'étiqueté η_1 par « -C » pour le complément et pas d'annotation pour l'adjoint, b. Arbres extraits, c. Annotation avec les arbres extraits.

Le corpus Penn Treebank comporte des nombreux éléments vides qui sont utilisés pour définir des phénomènes qui ne sont pas habituellement exprimés dans la grammaire TAG lexicalisée. Cette observation l'amène à essayer deux stratégies différentes pour traiter les éléments vides. La première stratégie « ALL » est d'inclure tous les éléments vides dans la grammaire. La seconde stratégie « SOME » est d'inclure seulement les éléments vides représentant des complément nul (0), un PRO vide et une trace de NP passif (*), et les traces de mouvement syntaxique (*T*) ; ce sont les éléments vides qui se trouvent habituellement dans les grammaires TAG lexicalisées pour l'anglais.

L'ensemble des étiquettes non-terminales et terminales dans le corpus Penn Treebank est plutôt vaste. Un ensemble vaste signifie généralement qu'un grand nombre d'arbres est extrait du corpus arboré ; ces arbres pourraient manquer certaines généralisations et exacerber le problème d'éparpillement des données de tout modèle statistique les prenant pour base. De plus, certaines étiquettes non- terminales sont superflues parce qu'ells indiquent les configurations structurelles. Par exemple, NX est utilisé pour étiqueter les nœuds dans la structure interne des conjonctions de NP composites dans un NP englobant. Si NX était remplacé par NP, la procédure d'extraction de l'arbre pourrait encore déterminer qu'une instance de conjonction existe et prendre une action appropriée. D'autre part, les distinctions faites dans un ensemble plus vaste d'étiquettes, comme celui du corpus Penn Treebank, peuvent aider le modèle statistique. Pour cette raison, Chen (2001) évalue deux stratégies différentes. La première stratégie « FULL » utilise l'ensemble des étiquettes originales du corpus Penn Treebank. L'autre stratégie « MERGED » utilise l'ensemble des étiquettes réduit. Dans la dernière approche, l'ensemble original est projeté sur un ensemble d'étiquettes similaire à celui utilisé dans la grammaire XTAG (XTAG Research Group 1999). Dans l'approche de Chen (2001), le fait d'être une tête et le statut de complément ou d'adjoint est en premier lieu déterminé en fonction de l'ensemble complet des étiquettes avant que les arbres ne soient re-étiquetés conformément à l'ensemble d'étiquettes réduit.

En plus des arbres auxiliaires modifieurs, il y a des arbres auxiliaires prédicatifs qui sont générés comme suit : Pendant l'extraction des arbres, supposons que le tronc φ ait un nœud η qui partage la même étiquette qu'un autre nœud η' où η' est le complément, pas sur φ, mais est immédiatement dominé par un nœud sur φ. Dans ce cas, l'arbre auxiliaire prédicatif est extrait où η est sa racine, η' est son nœud pied et avec φ lui servant de tronc. Par la suite, le parcours φ' dominé par η' devient un candidat pour être agrandi de nouveau. Voir la Figure (5a). Ce

mécanisme fonctionne de concert avec d'autres parties de la procédure d'extraction de l'arbre (notamment l'identification des compléments et adjoints, la fusion des labels non-terminaux (de SBAR à S), et la politique de gestion des éléments vides) afin de produire des arbres localisant des mouvements de longue distance, ainsi qu'illustré dans la Figure (5c).

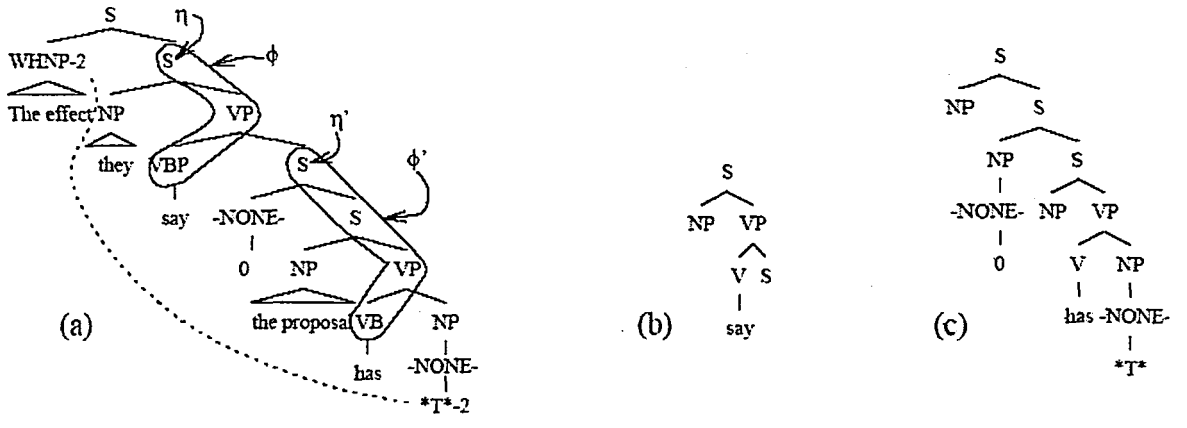


Figure 5. a. L'arbre auxiliaire prédicatif associé avec φ est sorti parce que les nœuds η et η' ont les mêmes étiquettes en laissant le tronc φ' pour être agrandi jusqu'à la capture du mouvement longue de distance, b. L'arbre auxiliaire prédicatif extrait, c. L'arbre montre le mouvement à long distance.

Chen (2001) extrait une grammaire dans la Section 02-21 du corpus Penn Treebank. La taille de la grammaire TAG extraite est montrée dans le Tableau 1 (pour le détail de l'évaluation, voir Chen (2001)).

Complément - Adjoint	Eléments nuls	Ensemble des étiquettes	Taille de la grammaire TAG	
			Schémas d'arbres	Arbres lexicalisés
CA1	ALL	FULL	8 996	118 333
CA1	ALL	MERGED	5 165	111 220
CA1	SOME	FULL	8 623	117 527
CA1	SOME	MERGED	4 911	110 428
CA2	ALL	FULL	5 354	116 326
CA2	ALL	MERGED	2 632	108 370
CA2	SOME	FULL	4 936	115 335
CA2	SOME	MERGED	2 366	107 387

Tableau 1. Taille de la grammaire extraite.

2.2 Autres travaux basés sur l'algorithme de Chen (2001)

On peux également citer les travaux d'extraction de Nasr (2004) et Johansen (2004) pour le français et de Habash et Rambow (2004) pour l'arabe. Ce sont des travaux basés sur l'algorithme de Chen (2001) et appliqués à chacune de ces langues.

Nasr (2004) et Johansen (2004) présente les trois stratégies implémentées pour le traitement des groupes prépositionnels (tels que G_1, G_2, et G_3) à partir du corpus arboré Paris 7 (Abeillé et al. 2003) : la grammaire G_1 revient à considérer tous les groupes prépositionnels comme compléments, tandis que G_3 considère, quant à elle, tous les groupes prépositionnels comme adjoints. G_2 constitue une position intermédiaire entre G_1 et G_3. Elle repose sur une liste de 14 prépositions établie manuellement pour le français dont on considère qu'elles introduisent des compléments. Les 103 autres prépositions (principalement des locutions prépositionnelles) introduisent des ajouts.

Grammaire	Taille	Couverture	Ambiguïté moyenne	Temps d'analyse moyen
G_1	3 808	0,993	11 003	2,22 ms
G_2	5 910	0,988	750	1,39 ms
G_3	7 123	0,985	346	0,13 ms

Tableau 2. Taille des trois grammaires G_1, G_2 et G_3 ainsi que leur couverture, leur ambiguïté moyenne et le temps moyen d'analyse par phrase sur le corpus de test (Nasr 2004).

« Les résultats de Tableau 2 confirment et quantifient les prédictions théoriques. G_3 définit approximativement deux fois plus de catégories que G_1, et G_2 se situe entre les deux. De plus, la taille de la grammaire influence directement l'ambiguïté moyenne et le temps d'analyse. La comparaison de l'ambiguïté moyenne pour G_1 et pour G_3 permet en outre de quantifier la part d'ambiguïté provenant des rattachements prépositionnels. En effet, ces deux grammaires ne se distinguent que par leur traitement des rattachements prépositionnels, qui ne sont jamais ambigus pour G_3. La différence de l'ambiguïté moyenne pour G_1 et G_3 est très largement dûe aux rattachements prépositionnels. Parmi les 11 003 analyses construites en moyenne par phrase pour G_1, à peu près 10 600 proviennent de rattachements prépositionnels, soit une

proportion de l'ordre de 95 %. Le Tableau 3 permet aussi d'observer que les trois heuristiques exercent une faible influence sur les couvertures des grammaires qu'elles définissent : le rapport des tailles des grammaires G_1 et G_3 vaut 0,53 alors que le rapport de leur couverture n'est que de 1,008. Un accroissement considérable du nombre de catégories ne diminue donc que très faiblement la couverture des grammaires » (Nasr 2004, pp.141-142).

Habash et Rambow (2004) présentent seulement une introduction courte de l'état actuel du projet qui extrait une grammaire TAG lexicalisée à partir du corpus Penn Arabic Treebank (PATB) qui contient environ 160 000 mots.

2.3 Extraction des grammaires d'insertion d'arbres probabilistes - Chiang (2000)

Chiang (2000) décrit une méthode permettant d'extraire une variante d'une grammaire TAG et ses paramètres probabilistes afin de les exploiter pour l'analyse syntaxique. La méthode de Chiang est très proche de celle de Chen (2001) : étant donné un noeud *n*, Chiang sélectionne la fille tête à l'aide d'un tableau de percolation de la tête. Les nœuds qui ne sont pas la tête sont catégorisés comme adjoint ou complément avec la méthode de Collins (1997).

La grammaire extraite par Chiang est une grammaire d'insertion d'arbres lexicalisée ('*Lexicalized Tree Insertion Grammar*' ou LTIG). LTIG, un formalisme introduit par Schabes et Waters (1995), est une restriction de TAG lexicalisée dans laquelle l'adjonction n'est pas sensible au contexte. De ce fait le langage dérivé par une grammaire LTIG est un langage hors contexte. La principale restriction par rapport à TAG lexicalisée est l'interdiction des arbres auxiliaires enveloppants. En plus, TIG ne permet pas à un arbre auxiliaire droit (ou gauche) d'être adjoint sur un nœud qui est sur le *spine* d'un arbre auxiliaire gauche (ou droit) et permet l'adjonction simultanée sur un seul nœud qui s'appelle « Adjonction-sœur » (Schabes et Waters 1995).

Chiang (2000) extrait une grammaire LTIG aussi à partir des sections 02-21 du Penn Treebank et obtient une grammaire contenant 3626 schémas d'arbres. La grammaire de Chiang est donc relativement petite en comparaison avec les grammaires extraites par Chen (2001). Chen explique cette différence par les différences dans le tableau de percolation de la tête et par le fait que Chiang (2000) permet l'adjonction entre deux compléments. Pour chaque

nœud η, les règles classifient exactement une fille de η comme la tête et le reste comme arguments ou adjoints. Ayant utilisé cette classification, Chiang (2000) peut construire la dérivation de la grammaire TAG (y compris les arbres élémentaires) à partir de l'arbre dérivé comme suit :

1. Si η est un adjoint, extraire le sous-arbre raciné sur η pour former un arbre auxiliaire modifieur.

2. Si η est un argument, extraire le sous-arbre raciné sur η pour former un arbre initial en laissant le nœud de substitution.

3. Si η a le coin droit θ lequel est un argument avec la même étiquette comme η (et tous les nœuds intervenant sont les têtes), extraire le segment du nœud η au nœud θ pour former l'arbre auxiliaire.

Les règles (1) et (2) produisent le résultat désiré, la règle (3) change l'analyse en faisant des sous-arbres avec les arguments récursifs des arbres auxiliaires prédicatifs. Ceci produit, entre autres, l'analyse des verbes auxiliaires. Il est appliqué par un algorithme exhaustif avec ηs potentiel considéré comme un algorithme descendant et avec θs potentiel considéré comme un algorithme ascendant. Les restrictions compliquées sur θ servent à assurer que la dérivation du TIG bien formée est produite.

La Figure 6 montre trois paramètres de TAG stochastique (Resnik 1992, Schabes et Shieber 1992) et un paramètre de TIG, tel que l'opération d'adjonction-sœur,

$$\sum_{\alpha} P_I(\alpha) = 1$$

$$\sum_{\alpha} P_S(\alpha \mid \eta) = 1$$

$$\sum_{\beta} P_A(\beta \mid \eta) + P_A(NONE \mid \eta) = 1$$

$$\sum_{\alpha} P_{SA}(\gamma \mid \eta, i, f) + P_{SA}(STOP \mid \eta, i, f) = 1$$

Figure 6. Paramètre de TIG stochastique

où α a une portée sur les arbres initiaux, β sur les arbres auxiliaires, γ sur les arbres auxiliaires modifiants, et η sur les nœuds. $P_i(\alpha)$ est une probabilité d'un commencement de la dérivation avec α ; $P_s(\alpha \mid \eta)$ est une probabilité de substitution α sur le nœud η ; $P_a(NONE \mid \eta)$ est une probabilité d'adjonction β sur le nœud η, finalement $P_{sa}(\gamma \mid \eta, i, f) + P_{sa}(STOP \mid \eta, i, f)$ est une probabilité d'adjonction-sœurs entre *i*ème et *i*+1ème filles de η.

Chiang (2000) applique l'algorithme d'extraction à la Section 4.1 sur les sections 02-21 du corpus Penn Treebank. La taille de la grammaire extraite est d'environ 73 000 arbres lexicalisés, avec les mots qui apparaissent moins de quatre fois remplacés par le symbole *UNKNOWN*. S'il considère seulement les gabarits d'arbre, la grammaire extraite a 3 626 schémas d'arbre dans lesquelles 2 039 schémas apparaissent plus d'une fois.

2.4 Le système LexTract : extraction des grammaires TAG lexicalisées - Xia (2001)

Xia *et al.* (2000), Xia *et al.* (2001) et Xia (2001) développent un système d'extraction des grammaires TAG lexicalisées à partir des corpus annotés tel que le corpus Penn Treebank pour l'anglais, le chinois et le coréen. Les grammaires extraites sont utilisées pour entraîner des analyseurs syntaxiques et pour estimer la couverture des grammaires manuellement construites. La méthode de Xia a été exploitée pour extraire des grammaires TAG lexicalisée pour l'anglais, le chinois et le coréen.

Le processus d'extraction de Xia et al. (2000) a trois étapes : D'abord, LexTract met entre parenthèses (*ou* annote) chaque *ttree* (les arbres dans le Treebank) ; Deuxièmement, LexTract décompose *ttree* en un ensemble d'*etrees* (les arbres élémentaires) ; Troisièmement, LexTract construit l'arbre de dérivation pour *ttree*.

Les arbres *ttrees* dans le corpus Penn Treebank ne distinguent pas explicitement les arguments et les modifieurs, alors que *etrees* le font. Pour justifier cette différence, Xia et al. (2000), annote d'abord *ttrees* en ajoutant les nœuds intermédiaires pour qu'à chaque niveau, une des relations suivantes applique entre la tête et les sœurs : (1) la relation de tête-argument, (2) la relation de modification, et (3) la relation de coordination. LexTract obtient ceci en choisissant d'abord la tête et le fils à chaque niveau et en distinguant les arguments des adjoinst à l'aide des trois tableaux, ensuite, en ajoutant les nœuds intermédiaires pour que les

modifieurs et les arguments de la tête s'attachent aux niveaux différents. La Figure 7 montre l'arbre *ttree* entièrement annoté. Les nœuds insérés par LexTract sont en gras.

```
((S (PP-LOC (IN at)
            (NP (NNP FNX)))
    (S  (NP-SBJ-1 (NNS underwriters))
        (VP  (ADVP (RB still))
             (VP  (VP  (VBP draft)
                       (NP (NNS policies)))
                  (S-MNR
                    (NP-SBJ (-NONE- *-1))
                    (VP (VBG using)
                        (NP (NP (NN fountain)
                                (NP  (NNS pens)))
                            (CC and)
                            (NP (VBG blotting)
                                (NP  (NN papers))))))))) )))
```

Figure 7. Arbre *ttree* entièrement annoté

A l'étape de la construction des *etrees*, LexTract enlève les structures récursives, qui deviendront *mod-etrees* ou *conj-etrees*, de l'arbre *ttree* entièrement annoté et construit *spine-etrees* pour les structures non-récursives. A partir de la racine de l'arbre *ttree* entièrement annoté, LexTract trouve d'abord un chemin unique de la racine à sa tête. Ensuite, le système vérifie chaque nœud *e* sur le chemin. Si le fils d'*e* dans *ttree* est marqué comme modifieur, LexTract marque *e* et le parent d'*e* et construit *mod-etree* (ou *conj-etree* si *e* a l'autre fils qui est une conjonction) avec le parent d'*e* comme le nœud de la racine, et le fils d'*e* comme modifieur. Après, LexTract crée *spine-etree* avec les nœuds restants non marqués sur le chemin et leur fils. Finalement, LexTract répète ce processus pour les nœuds qui ne sont pas sur le chemin. Dans la Figure 8, qui est la même que celle du 7 à part les nœuds qui sont numérotés et divisés en deux parties au-dessus et en dessous, le chemin de la racine S_1 à la tête VBP est : $S_1 \rightarrow S_2 \rightarrow VP_1 \rightarrow VP_2 \rightarrow VP_{13} \rightarrow VBP$. Tout au long du chemin, PP – *at FNX* – est un modifieur de S_2 ; par conséquent, $S_1.b$, $S_2.t$, et *spine-etree* qui a *PP* pour sa racine, forment *mod-etree* #1. De la même manière, *ADVP* est toujours un modifieur de VP_2, et S_3 est un modifieur de VP_3, et les structures correspondantes forment *mod-etrees* #4 et #7. Sur le chemin de la racine à *VBP*, $S_1.t$ et $S_2.b$ sont fusionnés (et aussi pour $VP_1.t$ et $VP_3.b$) pour former *spine-etree* #5. La répétition de ce processus pour les autres nœuds va générer les autres arbres tels que les arbres #2, #3 et #6. L'arbre *ttree* entier produit 12 *etrees* comme décrit dans la Figure 9.

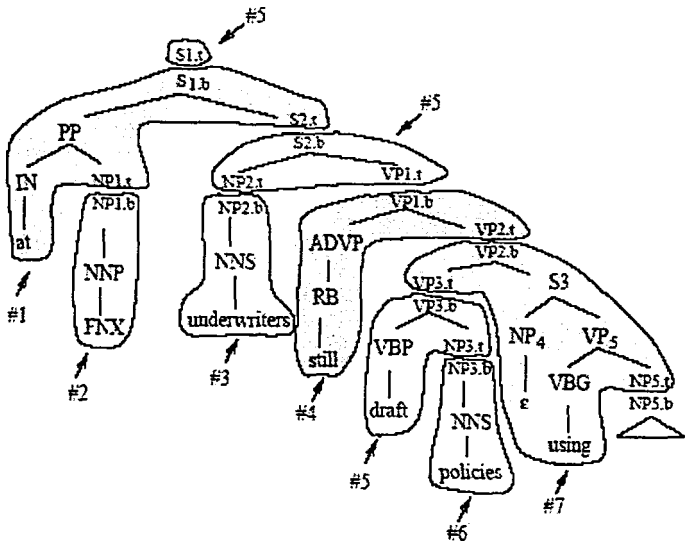


Figure 8. Arbres *etrees* extraits peuvent être vus comme la décomposition de l'arbre *ttree* entièrement annoté

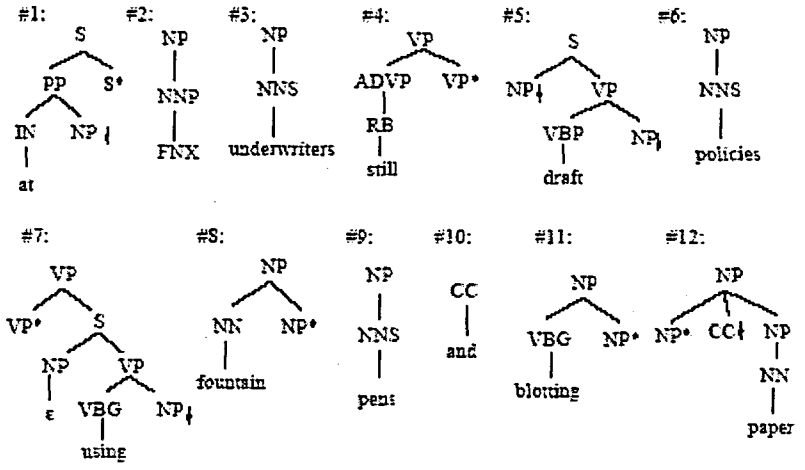


Figure 9. Arbres *etrees* extraits à partir de l'arbre *ttree* entièrement annoté

Xia (2000) fait une extraction avec LexTract sur un million de mots d'anglais du corpus Penn Treebank et extrait deux grammaires. La première G_1, utilise l'ensemble d'étiquettes du corpus, tandis que la deuxième utilise un ensemble réduit où quelques étiquettes du corpus sont fusionnées dans une seule étiquette. L'ensemble réduit d'étiquettes est basé sur celui qui est utilisé dans la grammaire XTAG (XTAG Group 1999). La taille de la grammaire extraite est montrée dans le Tableau 3.

	Nombre de schémas d'arbre	Nombre d'arbres élémentaires	Nombre de mots du corpus	Nombre d'arbre élémentaire par type de mots	Nombre d'arbre élémentaire par le *token*	Règles CFG (non-lexicalisées)
Eng G_1	6 926	131 397	49 206	2,67	34,68	1 524
Eng G_2	2 920	117 356	49 206	2,38	27,70	675
Ch G_3	1 140	21 125	10 772	1,96	9,13	515
Kor G_4	634	9 787	6 747	1,45	2,76	177

Tableau 3. Grammaires extraites à partir des corpus Penn English, Chinese et Korean Treebank.

2.5 Extraction des grammaires d'arbres lexicalisées probabilistes - Neumann (2003)

Neumann (1998) développe une méthode pour extraire des grammaires d'arbre lexicalisées stochastiques à partir d'un corpus annoté. Neumann applique sa méthode à deux corpus : le Penn Treebank (Marcus et al. 1993, Taylor et al. 2003) pour l'anglais et le corpus NEGRA (Skut et al. 1997) pour l'allemand. Dans le corpus NEGRA les éléments tête et modifieurs sont annotés explicitement.

Neumann (1998) produit une grammaire d'arbres, qui comprend, de ce fait, des arbres initiaux et des arbres auxiliaires. Pour produire une grammaire d'arbres adjoints, Neuman (1998) localise une relation de tête-dépendance dans les arbres élémentaires en prenant un tableau de percolation de la tête pour déterminer la tête des noeuds intérieurs dans l'arbre syntaxique et ensuite en stipulant que les arbres élémentaires ont un tronc dont les noeuds ont une tête commune.

Selon Neumann (1998) avec un Treebank, l'extraction d'une grammaire est un processus de décomposition des arbres syntaxiques en petites unités qui sont appelées sous-arbres. Dans l'approche de Neumann (1998) l'opération de décomposition :

1. doit produire des sous-arbres ancrés dans le lexique,

2. doit être guidée par des principes linguistiques.

La motivation derrière (1) '*doit produire des sous-arbres ancrés dans le lexique*' tient à l'observation qu'en pratique, la grammaire CFG stochastique fonctionne moins bien que les approches non hiérarchiques, et que la grammaire d'arbres lexicalisée peut être capable de capturer à la fois l'information distributionnelle et l'information hiérarchique (Schabes et Waters 1995). En ce qui concerne (2) '*doit être guidée par des principes linguistiques*' Neumann veut profiter des principes linguistiques qui définissent le corpus Treebank. Ceci est motivé par l'hypothèse selon laquelle le Treebank supportera mieux le développement de stratégies d'apprentissage en ligne ou incrémentale (les critères d'extraction sont moins dépendants de la taille et de la qualité du Treebank existant que des approches basées sur la statistique pure) et qu'il rend possible la comparaison de la grammaire extraite avec une grammaire basée sur la linguistique. Ces deux aspects (mais surtout le dernier) sont importants parce qu'il est possible d'appliquer la même stratégie d'apprentissage à un Treebank construit par une grammaire basée sur la linguistique, et de chercher des méthodes pour combiner le Treebank et les grammaires basées sur la linguistique.

Neumann (1998) décompose les arbres du corpus annoté en arbres élémentaires. Un arbre élémentaire qui est extrait par la méthode de Neumann est lexicalisé et les noeuds sur le chemin entre l'ancre lexicale et la racine de l'arbre élémentaire sont les nœuds têtes. Pour le corpus NEGRA, l'information sur les têtes est directement disponible et pour le Penn Treebank, Neumann se sert d'un tableau de percolation de la tête. Dans le cas où il n'est pas possible d'identifier une tête, il y a un paramètre appelé la DIRECTION qui spécifie si le candidat gauche ou droit doit être sélectionné. Au moyen de ce paramètre, Neumann (1998) peut aussi spécifier si la grammaire de résultat doit préférer la branche gauche ou droite. Tout noeud non-tête est détaché de l'arbre en laissant des noeuds de substitution.

En utilisant l'information sur les têtes en tant que principe de décomposition guidée par les têtes, chaque arbre du Treebank est décomposé de haut en bas en un ensemble de sous-arbres, tel que chaque sous-arbre non-terminal et sans tête est extrait et le point de bouture est marqué pour l'opération de substitution. Le même processus est appliqué récursivement à chaque sous-arbre extrait. Etant donné la notion de tête retenue par l'auteur, chaque arbre extrait est ancré dans le lexique (et le parcours de l'ancrage lexical à la racine peut être vu comme la 'chaîne de tête'). De plus, tous les éléments terminaux qui sont sœurs du nœud de la chaîne de tête resteront aussi dans l'arbre extrait. Par conséquent, le résultat de l'arbre extrait peut contenir plusieurs sous-chaînes terminales. Pour chaque arbre extrait, une indication de fréquence est utilisée pour calculer la probabilité $p(t)$ de l'arbre t, après que le Treebank entier ait été traité. Ainsi,

$\sum_{t:root(t)=\alpha} p(t) = 1$, où α représente l'étiquette de la racine de l'arbre t.

Après que l'arbre est décomposé complètement, le système de Neumann (1998) obtient un ensemble d'arbres élémentaires lexicalisés où chaque non-terminal du résultat est marqué pour l'opération de substitution. Dans l'étape suivante, l'ensemble des arbres élémentaires est divisé en un ensemble d'arbres initiaux et auxiliaires.

L'ensemble des arbres auxiliaires est encore subdivisé en un ensemble d'arbres auxiliaires gauche, droit et enveloppant (en utilisant des étiquettes spéciales du nœud pied telles que *:lfoot*, *:rfoot*, et *:wfoot* (Schabes et Waters 1995)). L'identification d'arbres auxiliaires possibles est fortement déterminée par le corpus, c'est-à-dire, l'utilisation d'étiquettes spéciales du nœud pied permet de déclencher les règles d'inférence correspondantes. Par exemple, il est possible de traiter l'étiquette *:wfoot* comme une étiquette pour la substitution si la grammaire extraite est considérée comme une grammaire LTIG où seulement les arbres auxiliaires enveloppants très fréquentés seront considérés. Il est aussi possible de traiter toutes les étiquettes du nœud pied comme des étiquettes de substitution si la grammaire extraite permet seulement l'opération de substitution.

Neumann (1998) fait brièvement un rapport sur le résultat de sa méthode d'extraction en utilisant le corpus Negra Treebank (4270 phrases) et les sections 02, 03 et 04 du corpus Penn Treebank (les premières 4270 phrases). Dans les deux cas, il extrait trois versions différentes de la grammaire TAG stochastique : (a) les ancres sont des mots, (b) les ancres sont des

parties du discours, et (c) tous les éléments terminaux sont substitués par une constante :*term*, c'est-à-dire que l'information lexical est ignorée. Le Tableau 4 résume les résultats.

Corpus	Arbres lexicalisés	Arbres des POS	Arbres de :Terme
Negra	26 553	10 384	6 515
Penn Treebank	31 944	11 979	8 132

Tableau 4. Résumé des résultats de Neumann (1988)

2.6 Résumé sur les travaux d'extraction et conclusion

Les travaux d'extraction expliqués ci-dessus sont résumés dans le Tableau 5.

	Chen (2001)	Chiang (2000)	Xia (2001)	Neumann (2003)
Identification du complément ou de l'adjoint	CA1 : étiquettes fonctionnelles et sémantiques d'un nœud η et de son parent η' CA2 : celles d'un nœud η sa sœur de tête η_k et la distance entre η et η_k	Règles de percolation de la tête	Trois tableaux tels que, le tableau de percolation de la tête, le tableau des arguments, et le tableau de *Tagset*.	Etiquettes spéciales du nœud pied qui permettent de déclencher les règles d'inférence après avoir marqué pour la substitution.
Procédure récursive	Ascendant	Ascendant	Descendant	Descendant
Corpus utilisé	Penn Treebank	Penn Treebank	Penn Treebank[3], Penn Chinese Treebank et Penn Korean Treebank	Penn Treebank pour l'anglais et Negra pour l'allemand
Grammaire extraite	TAG lexicalisée sans traits	TIG lexicalisée sans traits	TAG lexicalisée sans traits	Grammaire d'arbres lexicalisée sans traits

Tableau 5. Résumé des travaux d'extraction

[3] Elle fait aussi une extraction pour le chinois et le coréen. Voir l'Annexe de Xia (2001) pour les détails d'extraction pour ces langues.

Bien qu'il y ait des travaux d'extraction d'une grammaire lexicalisée, l'extraction des traits n'est pas encore réalisée. Dans cette thèse nous présentons une implémentation d'un système qui n'extrait pas seulement une grammaire lexicalisée (LTAG), mais aussi une grammaire LTAG avec traits (FB-LTAG) à partir d'un corpus arboré pour le coréen.

Chapitre 2. La description du corpus coréen

Ce chapitre présente le corpus *Sejong* Korean Treebank (SJTree), à partir duquel une grammaire LTAG est extraite. C'est un corpus arboré, annoté syntaxiquement pour le coréen. Nous en décrivons en particulier le schéma d'annotation, qui s'est basé sur l'analyse linguistique. Nous présentons également le schéma d'annotation du Penn Treebank pour le coréen (Penn Korean Treebank) et comparons les schémas des deux corpus. La comparaison nous permet de relever les différences entre les deux corpus susceptibles d'impliquer des modifications dans la procédure d'extraction.

1. Origine du corpus SJTree

Le corpus SJTree est construit comme une partie du « Projet de Sejong du 21ème siècle » qui se déroule de 1998 à 2007[4] (Sejong Project 2003). Ce projet est organisé par le Ministère de la culture et de tourisme de Corée, et il est effectué par l'Institut national de la langue coréenne. Le corpus SJTree est une partie du corpus Sejong 21 qui est construit avec des extraits des journaux et des textes littéraires couvrant de nombreux auteurs et de nombreux domaines. En général, le corpus Sejong 21 est composé d'environs 5 millions *eojeol*s avec l'annotation

[4] Le projet construit (i) un corpus brut, morphologiquement analysé, syntaxiquement analysé et sémantiquement analysé, et les outils qui nous permettent de le gérer, (ii) un corpus pour les langues spéciales, telles que la langue nord-coréenne et la langue coréenne à l'étranger, (iii) un dictionnaire comparé des langues sud-coréenne et nord-coréenne, et celui des dialectes, (iv) un dictionnaire électronique du coréen, (v) un dictionnaire pour les termes techniques, (vi) une base de données pour le caractère du sino-coréen et l'alphabet du coréen ancien, et (vii) des nouvelles polices de caractères du coréen pour l'ordinateur.

morphologique[5]. Une partie a été effectuée qu'est le corpus SJTree. Le corpus SJTree est de 30 853 *eojeol*s, donc 52 234 mots pour 2 512 phrases à partir duquel nous extrayons une grammaire TAG lexicalisée.

2. Annotation morphosyntaxique du corpus SJTree

L'annotation morphologique s'effectue en segmentant chaque *eojeol* dans le corpus SJTree. L'*eojeol* se compose d'au moins un mot et il fonctionne comme unité constituant une phrase.

L'étiquetage pour l'annotation morphologique est basé sur le « Critère de l'étiquette du mot pour l'analyse morphologique » et le « Critère du Tag Set » de l'Electronics and Telecommunications Research Institute (ETRI 1999). Le corpus SJTree s'est également référé au critère du Laboratoire du Traitement Automatique des Langues Naturelles à l'Université de Korea (Université de Korea 1999). Une étiquette comporte trois informations principales décrites ci-dessous et spécifiées dans le Tableau 1 :

1. La partie du discours (POS). Catégorie « classique »,

2. Une sous-catégorisation. Par exemple le nom commun (NNG) ou le nom propre (NNP) pour les noms (NN). Egalement les symboles, les mots étrangers et les autres caractères spéciaux sont notés par S*x* où *x* exprime la sous-catégorisation du symbole (S).

3. Le lemme. La forme non fléchie pour les prédicats, donc les verbes et les adjectifs.

[5] *eojeol* est une unité de la segmentation dans la phrase coréenne et il est composée par plus d'un mot. Prenons, par exemple, la phrase en (1) *eodi ga.neunya.ga geu.ui muleum.i eoss.da* ('Sa question était où je serais') : elle contient 5 *eojeol*s, mais 10 mots. Donc, chacun des eojeols comportent 2-3 mots sauf le premier qui n'en comporte qu'un. Les *eojeol*s sont séparés par le blanc selon la norme de l'écriture, tandis que l'on ajoute les points à la transcription pour articuler et expliciter les mots de chaque *eojeol*.

Catégorie	Sous-catégorie	Description	Exemple
NN	NNG (Noms commun), NNP (Nom propre), NNB (Nom dépendant)	Nom	*gajok.eul* : *gajok*('famille')/NNG + *eul*/JKO)
NPR	-	Pronom	*geugeos.eun* : *geugeos*('ceci')/NPR + *eun*/TOC
NR	-	Numéral	*man* : *man*('dix mils')/NR
VV	-	Verbe	*bonay.eoss.da* : *bonay*('envoyer')/VV + *eoss*/EP + *da*/EF
VA	-	Adjectif	*jeolm.eun* : *jeolm*('jeune')/VA + *eun*/ETM
VX	-	Auxiliaire	*doi.n* : *doi*('devenir')/VX + *n*/ETM
VC	VCP (Copule positive), VCN (Copule négative)	Copule	*seolmyeong.i.da* : *seolmyeong*('explication')/ NNG + *i*/VCP + *da*/EF
MM	-	Déterminant	*dareun* : *dareun*('autre')/MM
MA	MAG (Adverbe général), MAJ (Conjonctif)	Adverbe	*dasi* : *dasi*('encore')/MAG

IC	-	Interjection	*anya* : *anya*('non')/IC
JK	JKS (Postposition nominative), JKC (Postposition d'attribut), JKG (Postposition de génitif), JKO (Postposition accusative), JKB (Postposition adverbiale), JKV (Postposition vocative), JKQ (Postposition de citation)	Postposition	*saram.i* : saram('homme')/NNG + *i*/JKS
JX	-	Postposition auxiliaire	*nalssi.ey.do* : *nalssi*('temps')/NNG + *ey*/JKB + *do*('aussi')/JX
JC	-	Postposition conjonctive	*panmae.wa* : *panmae*('vente')/NNG + *wa*/JC
E	EP (Pré-terminaison), EF (Terminaison finale), EC (Terminaison conjonctive), ETN (Terminaison de dérivation nominale), ETM (Terminaison de dérivation adnominale)	Terminaison	*bonay.eoss.da* : *bonay*('envoyer')/VV + *eoss*/EP + *da*/EF
XP	XPN (Préfixe nominal)	Préfixe	*dae gyumo* : dae('grand')/XPN + *gyumo*('échelle')/NNG
XS	XSN (suffixe de dérivation nominale), XSV (suffixe de dérivation verbale), XSA (suffixe de dérivation	Suffixe	*il.ha.l* : *il*('travail')/NNG + *ha*('faire')/XSV + *l*/ETM

	adjectivale)		
XR	-	Racine du mot	*ganeung.seong* ('possibilité'): *ganeung*('possible')/XR + *seong*/XSN
S	SF, SP, SS, SE, SO, SH, SL, SW, SN	Symbole	Voir le Tableau 2 ci-dessous
N	NF, FV, NA	Catégorie inanalysable	Voir ci-dessous.

Tableau 1. Jeu d'étiquette pour l'annotation morphologique du corpus SJTree

Le corpus SJTree a traité en détail des symboles, des mots étrangers et des chiffres qui sont représentés par *S*.

. ? !	SF
, · : ; /	SP
" " ( ) {} <> –	SS
…	SE
~	SO
$, @, #, ou les autres symboles	SW
les mots chinois	SH
les autres mots étrangers	SL
les chiffres	SN

Tableau 2. Jeu d'étiquette pour les symboles du corpus SJTree

Si l'on trouve un mot inanalysable, le corpus SJTree le marque par la catégorie N. Et puis on ajoute la notation pour sa sous-catégorisation si on peut supposer à quelle catégorie il appartient. Ainsi, il est noté par NF (catégorie assumée comme nom) ou NV (catégorie assumée comme verbe). D'autre part, on le note par NA, si on ne peut pas trouver une catégorie supposée. En fait, on ne trouve aucun mot de ces catégories dans le corpus SJTree, tandis que pas mal de mots sont classés en ces catégories dans le corpus original qui ne contient que l'annotation morphologique.

Nous présentons ici l'annotation morphologique du corpus en montrant les exemples des noms. Pour les exemples des autres catégories, voir Sejong (2002).

- On analyse une forme fléchie du verbe, même si elle fonctionne comme un nom sans marqueur de citation :

(1) 어디 가느냐가 그의 물음이었다.

[가/VV+느냐/EC+가/JKS]

eodi	***ga.neunya.ga***	*geu.ui*	*muleum.i eoss.da*
où	aller.Nom	lui.Gen	question.Cop.Pass.Ter

'Sa question était où je serais.'

- On analyse un syntagme comme sa propre partie du discours même s'il est cité par les marqueurs et qu'il fonctionne comme un nom :

(2) 그것은 "는"이 아니라 "를"이다.

["/SS+는/JX+"/SS+이/JKC]

geugeos.eun	***"neun".i***	*ani.la*	*"reul".i.da*
ceci.Toc	"Nom".Nom	ne_pas.la	"Acc".Cop.Ter

'Ceci n'est pas « *neun,* » c'est « *leul.* »'

- On analyse un syntagme adverbial comme un adverbe même s'il a un fonctionnement nominal avec une postposition casuelle :

(3) 가족을 멀리에 보냈다.

[멀리/MAG+에/JKB]

gajok.eul	***meolli.e***	*bonay.eoss.da*
famille.Acc	loin.*e*	envoyer.Pass.Ter

'Il a envoyé sa famille loin d'ici.'

3. L'annotation syntaxique : constituants et fonctions

Il a été proposé plusieurs types de Treebanks (voir Abeillé 2003) : certains sont orientés pour les constituants (Penn Treebank) et d'autres sont annotés pour les fonctions (Prague Dependency Treebank). Ici, nous nous servirons plutôt d'une méthode mixte comme pour le français, mais plus détaillé pour l'annotation. Pour ainsi dire, l'étiquette syntaxique d'un

constituant dans le corpus SJTree donne deux informations décrites : la catégorie majeure telle que S, PP et NP et la fonction de surface comme sujet ou objet, ce qui n'est pas donné dans le corpus pour le français.

L'annotation « de surface » des constituants dans le corpus SJTree pour le coréen veut être compatible avec plusieurs théories linguistiques. C'est un corpus basé sur la structure syntagmatique (en anglais 'phrase structure') qui nous permet de représenter l'information linguistique en détail. Le corpus SJTree ne contient pas de catégorie vide qui correspond à un argument omis ou à une trace laissé par le déplacement. En général, la catégorie vide sert à expliciter la relation entre le prédicat et ses arguments. Cependant les règles de réécriture peuvent représenter le déplacement des arguments sans catégorie vide. Par exemple, l'antéposition d'un objet à gauche se décrit par une règle « S → NP_OBJ S ». Cette règle peut être interprétée comme s'il y a l'antéposition de l'objet, c'est-à-dire que la phrase n'a pas d'objet. Si nous nous en servons, nous n'avons pas besoin d'une représentation de la catégorie vide dans le corpus SJTree (Sejong Project 2003).

Egalement, nous allons faire face à un autre problème avec les arguments omis. L'introduction des arguments omis sert à représenter les arguments nécessaires. Pour cela on doit avoir une base de données sur la structure des arguments ou sur la grille de rôle thématique (en anglais 'thematic role grid') des prédicats à laquelle on peut se référer. Mais, ce genre d'information pour le coréen n'est pas encore établi parfaitement. Le corpus SJTree, donc, n'introduit pas de catégorie vide pour les arguments omis (c'est-à-dire les pronoms nuls).

L'annotation syntaxique dans le corpus SJTree suit les principes décrits ci-dessous :

1. On privilégie la cohérence et l'efficacité qui sont importants dans le domaine du traitement automatique de la langue, mais le point de vue en linguistique n'est pas ignoré pour autant.

2. On analyse la structure de la phrase en surface.

3. On adopte la théorie de la bifurcation, donc la structure binaire et n'adopte pas la structure plate.

4. On n'a pas de catégories vides.

5. L'unité de l'analyse est un *eojeol* (voir la note en bas 2).

6. On distingue le complément et l'adjout en limitant strictement le premier.

7. On ne distingue pas la subordination et la conjonction. Toutes les phrases conjonctives sont analysées comme des phrases adverbiales.

8. Si le sujet est lié à la phrase principale et la phrase subordonnée en même temps, il est analysé comme le sujet de la phrase principale.

3.1 Les types d'étiquettes

Les sections suivantes donnent le type de l'étiquette pour l'annotation syntaxique dans le corpus SJTree que nous présentons ici l'étiquette pour les catégories grammaticales et les catégories fonctionnelles. Il y a aussi les autres catégories pour les syntagmes cités, comme PRN (syntagme cité), X (pseudo syntagme), L ou R (symboles pour les phrases citées).

Catégorie		Tête	Exemple
S	Phrase	Syntagme verbal (VP)	Voir la Figure (1a)
NP	Syntagme nominal	Nom, pronom et chiffre	Voir la Figure (1a)
VP	Syntagme verbal et adjectival	Verbe et adjectif	Voir la Figure (1a)
VNP	Syntagme copule	Copule avec *ida*	Voir la Figure 29 dans le chapitre 3.
AP	Syntagme adverbial	Adverbe	Voir la phrase en (9')
DP	Syntagme déterminant	Déterminant	(DP 어느/MM) : (DP eoneu('quel')/MM)
IP	Syntagme interjectif	Interjection	Voir le cas du « prédicat général supprimé » dans le Tableau 4

Tableau 3. Catégories grammaticales

Catégorie		Usage	Exemple
SBJ	Sujet	NP_SBJ, S_SBJ, VNP_SBJ, VP_SBJ, etc	Voir la Figure (1a) pour l'exemple de NP_SBJ
OBJ	Objet	NP_OBJ, S_OBJ, VNP_OBJ, VP_OBJ, etc	Voir la Figure (1a) pour l'exemple de NP_OBJ
CMP	Attribut	NP_CMP, S_CMP, VNP_CMP, VP_CMP, etc	Voir la Figure (1a) pour l'exemple de NP_CMP
MOD	Modifieur nominal	NP_MOD, S_MOD, VNP_MOD, VP_MOD, etc	Voir la Figure (1a) pour l'exemple de VP_MOD
AJT	Modifieur verbal et/ou complément datif, locatif	AP_AJT, NP_AJT, VP_AJT, S_AJT, etc	Voir la phrase en (9') pour l'exemple de NP_AJT
CNJ	Syntagme nominal conjoint	NP_CNJ, S_CNJ, VNP_CNJ, VP_CNJ, X_CNJ, etc	
INT	Vocatif	NP_INT, etc	

Tableau 4. Catégories fonctionnelles

La Figure 1 donne un exemple de la phrase du corpus SJTree et sa traduction est montrée dans la phrase en (1). L'arbre syntaxique dans la Figure (1b) ne montre pas l'analyse morphologique, il représente seulement l'analyse syntaxique pour simplifier les notations.

(4) 막일을 기피하는 풍조는 건설 공사장뿐만이 아니다.

makil.eul	*gipiha.neun*	*pwungjo.neun geonseol*	
travail_de_manoeuvre.Acc	ne_pas_aimer.*neun*	tendance.Nom	construction
gongsajang.bbunmani	*ani.da.*		
chantier.aussi_bien_que	ne_pas.Ter		

'La tendance de ne pas aimer le travail manuel n'est pas limitée au chantier.'

a. (S (NP_SBJ (VP_MOD (NP_OBJ 막일/NNG+을/JKO)
(VP_MOD 기피/NNG+하/XSV+는/ETM))
(NP_SBJ 풍조/NNG+는/JX))
(VP (NP_CMP (NP 건설/NNG)
(NP_CMP 공사장/NNG+뿐/JX+만/JX+이/JKC))
(VP 아니/VCN+다/EF+./SF))))

b.

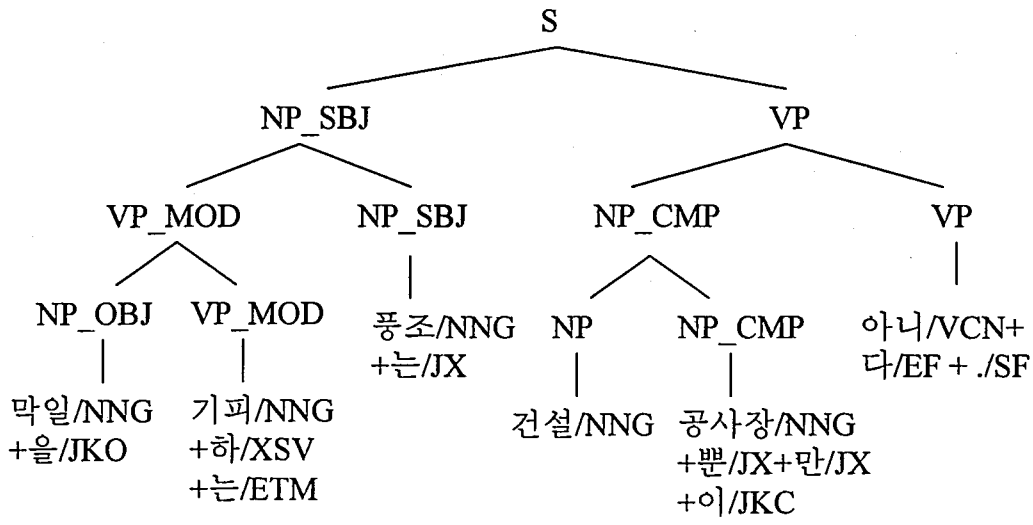


Figure 1. Corpus arboré et son arbre syntaxique du SJTree

4. Le Penn Korean Treebank

Le Penn Korean Treebank (PKT) (Han *et al.* 2001) contient 54 366 *eojeol*, soit 5 078 phrases extraites des textes militaires. Comme le corpus SJTree, le corpus PKT fait l'annotation aux

niveaux morphologique, syntaxique et aussi fonctionnel. Le Penn Treebank pour le coréen admet des catégories vides comme *pro* ou trace à la différence du corpus SJTree.

Dans Le Penn Treebank pour le coréen, chaque mot est doté d'une étiquette mentionnant sa partie du discours. Le jeu d'étiquettes du PKT comporte 29 étiquettes morphosyntaxiques et 5 étiquettes pour les ponctuations et autres symboles.

Le jeu d'étiquettes syntaxiques comporte 27 étiquettes (le nombre concerne seulement les étiquettes qui sont actuellement utilisées dans le corpus PKT). Chaque constituant est obligatoirement doté d'une étiquette syntaxique.

- Etiquette au niveau de la phrase : S
- Etiquette au niveau du syntagme : ADJP, VP, NP, ADVP, ADCP, DANP, INTJ, PRN, X, LST
- Etiquette au niveau de la tête : VV, VJ, VX, LV, ADV, CV
- Etiquette au niveau fonctionnel : -SBJ, -OBJ, -COMP, -ADV, -VOC, -LV

Voici un exemple montrant la représentation du Penn Korean Treebank, avec une phrase dont le prédicat est un verbe transitif 다루다 *daruda* ('traiter') et le sujet est un pronom nul.

(5)

어떤	과목에서	직일병의	근무	요령을
eoddeon	*goamok.eseo*	*jikilbyeong.eui*	*geunmu*	*yoryeong.eul*
quel	matière.*eseo*	soldat.Gen	devoir	art.Acc

다루는가 ?
daru.neun.ga ?
traiter ?
'Dans quelle matière, l'art du devoir des soldats est traité ?'

```
(S      (NP-SBJ *pro*)
        (VP     (NP-ADV     어떤/DAN
                            과목/NNC + 에서/PAD)
                (VP     (NP-OBJ     (NP     직일병/NNC + 의/PCA)
                                    (NP     근무/NNC
                                            요령/NNC + 을/PCA))
                                    다루/VV + 는가/EFN))
        ?/SFN)
```

Figure 2. Exemple du corpus arboré PKT

Comme il a été déjà mentionné, le corpus Penn Korean Treebank est caractérisé par les éléments *nul*, tels que la trace laissée par le déplacement, un élément omis, un opérateur vide, ou un élément effacé (ou omis). Le Tableau 5 montre le détail des éléments *nul* utilisés dans le corpus Penn Korean Treebank.

T	Une trace de déplacement. Cette étiquette est utilisée dans le cas de la dislocation et de la relativisation de l'objet ou du complément.
(NP *pro*)	Exprime le sujet ou l'objet omis. En général le syntagme nominal omis renvoie aux éléments à l'intérieur ou l'extérieur de la phrase à laquelle il s'attache et peut être récupéré à partir du contexte. La phrase est encore grammaticale si la catégorie vide est remplacée par le constituant approprié récupéré à partir du contexte.
(WHNP *op*)	Opérateur vide dans la construction relative.
(VV *?*)	Effacement du verbe.
(VP *?*)	Ellipse du VP.
(XP *?*)	Autres catégories vides inconnues.

Tableau 5. Les éléments nulls dans le corpus Penn Korean Treebank

Pour les phrase complexes (les phrases qui contiennent la subordonnée), Le Penn Treebank pour le coréen a deux types : complément et ajout phrastiques. Le complément phrastique est un argument du verbe et il est annoté comme « S–COMP » : S signifie qu'il est une phrase et –COMP qu'il est un complément du verbe. En général il constitue un noeud relié au verbe avec lequel il est associé. L'ajout phrastique forme un nœud S sans « –COMP » et qui est attaché à S ou VP avec lequel il est associé.

La différence la plus importante entre le corpus SJTree et Le Penn Treebank pour le coréen est celle de leurs critères d'analyse syntaxique. Tout d'abord, Le Penn Treebank pour le coréen a introduit les catégories vides tandis que le corpus SJTree ne les a pas adoptées comme il a déjà été mentionné.

Deuxièmement, les deux corpus ont utilisé différents nombres d'étiquettes pour les schémas d'annotation. En gros, le jeu d'étiquettes du Penn Korean Treebank comporte plus d'étiquettes que celui du corpus SJTree si on ne compte pas les étiquettes exprimant les sous-catégories du corpus SJTree. Le Tableau 6 ci-dessous nous montre le nombre des étiquettes des deux corpus.

	Etiquettes de POS	Etiquettes de ponctuation et symboles	Etiquettes syntagmatiques	Etiquettes fonctionnelles
Corpus SJTree	36	9	55	42
Corpus PKT	29	5	27	8

Tableau 6. Tableau comparatif des jeux d'étiquettes du corpus SJTree et du Penn Korean Treebank[6].

[6] Les étiquettes syntagmatiques contiennent déjà les étiquettes fonctionnelles. Par exemple, parmi 27 étiquettes syntagmatiques du corpus Penn Korean Treebank, 8 sont les étiquettes fonctionnelles telles que NP-OBJ, NP-SBJ, NP-OBJ-LV, S-OBJ, NP-VOC, NP-COMP, NP-ADV, et S-COMP. On trouve aussi 5 étiquettes parmi 27 étiquettes syntagmatiques, qui ne concernent que les catégories antéposées telles que WHNP-1, WHNP-2, NP-OBJ-1, NP-COMP-1, et NP-COMP-2.

A première vue il y a une différence entre le nombre d'étiquettes des deux corpus. En réalité, le corpus SJTree a introduit plusieurs étiquettes pour exprimer une catégorie dans Le Penn Treebank pour le coréen en la partageant en plusieurs sous-catégories. Par exemple, le corpus SJTree a distingué 5 catégories pour la postposition casuelle alors que seulement une catégorie a été utilisée dans Le Penn Treebank pour le coréen.

Troisièmement, les compléments dans une phrase intransitive s'analysent différemment dans les deux corpus, et on en trouve différents schémas d'annotation syntaxique. Parmi ces compléments, la grammaire scolaire du coréen considère comme complément attribut seulement ceux qui précèdent les verbes 되다 *doida* ('devenir') et 아니다 *anida* ('ne pas être'). Les autres sont considérés comme modifieur verbal. Le corpus SJTree a suivi cette analyse en limitant « le verbe intransitif avec un complément attribut NP » seulement à ces deux verbes. Il a marqué aux autres compléments une étiquette « -AJT » qui signifie un modifieur verbal. Par contre, Le Penn Treebank pour le coréen n'a pas distingué le complément du modifieur dans une phrase intransitive. Si on examine le corpus, l'annotation des compléments se compose de celle de la postposition et une étiquette « -CMP ». Donc, presque tous les verbes intransitifs avec un complément NP dans Le Penn Treebank pour le coréen correspondent aux verbes intransitifs avec un modifieur dans le corpus SJTree.

Enfin, la structure PKT n'est pas toujours binaire car le symbole est séparé de l'*eojeol* dans PKT. En plus, dans le corpus SJTree, chaque nœud contient un *eojeol*. Cependant, la structure de PKT est différente du traitement du syntagme nominal. Le corpus PKT considère le syntagme nominal comme un nœud même dans le cas où le syntagme contient plusieurs *eojeol*s.

Nous pouvons encore poser une question, « lequel est le mieux adopté à l'extraction d'une grammaire TAG lexicalisée? » Le corpus PKT contient plus d'*eojeol* que le corpus SJTree (54 366 versus 30 853). Le corpus PKT, cependant, est basé sur les textes militaires tandis que le corpus SJTree se compose des extraits des journaux et des textes littéraires de nombreux auteurs et de nombreux domaines. Cette particularité du corpus PKT pourrait réduire la qualité et la couverture de la grammaire extraite, et pour cette raison, nous choisissons le corpus SJTree pour l'extraction d'une grammaire TAG lexicalisée.

Chapitre 3. La procédure d'extraction de grammaires TAG lexicalisées

Nous proposons ici un algorithme pour extraire une grammaire TAG lexicalisée à partir d'un corpus arboré pour le coréen. L'algorithme est implémenté et testé sur le corpus arboré Sejong (SJTree) pour le coréen (Sejong Project 2003). Nous présentons dans ce chapitre les détails de l'algorithme.

1. Présentation générale

Le but de l'algorithme est de construire automatiquement une grammaire TAG lexicalisée à partir d'un corpus arboré. Un corpus arboré est un ensemble de textes avec les analyses morphosyntaxiques, les constituants et les fonctions. Le corpus se présente donc comme une suite de phrases dotées d'annotations morphosyntaxiques ou en d'autres termes comme une suite d'arbres syntaxiques. Voir la Figure 1 pour la phrase en (1) *makil.eul gipiha.neun pwungjo.neun geonseol gongsajang.bbunmani ani.da* ('La tendance de ne pas aimer le travail de manoeuvre n'est pas limitée au chantier').

(1) 막일을 기피하는 풍조는 건설 공사장뿐만이 아니다

makil.eul *gipiha.neun* *pwungjo.neun*
travail_de_manœuvre.Acc ne_pas_aimer.*neun* tendance.Nom
geonseol *gongsajang.bbunmani* *ani.da*
construction chantier.aussi_bien_que ne_pas.Ter
'La tendance de ne pas aimer le travail de manoeuvre n'est pas limitée au chantier.'

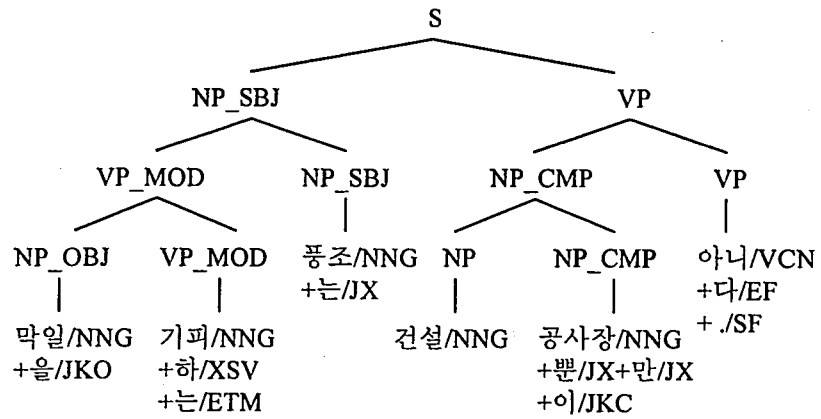


Figure 1. Arbre du corpus SJTree

A partir d'un tel arbre syntaxique, dont la Figure 1 offre un exemple, nous souhaitons extraire une analyse TAG lexicalisée correspondante. La grammaire extraite doit suivre les contraintes formelles et les contraintes linguistiques qui sont normalement vérifiées par une grammaire TAG lexicalisée. Abeillé et Candito (2000) et Abeillé (2002) a proposé les principes de bonne formation des arbres élémentaires. Ces Principes proprement linguistiques auxquels doivent obéir les arbres élémentaires ont été ajoutés :

- principe d'ancrage lexical : tout arbre élémentaire a au moins une tête lexicale non vide ;

- principe de cooccurrence prédicat-argument : tout prédicat contient dans sa structure élémentaire au moins un nœud pour chacun des arguments réalisés qu'il sous-catégorise (sous forme de nœud à substitution ou de nœud pied)[7] ;

- principe d'ancrage sémantique : tout arbre syntaxique élémentaire a un correspondent sémantique non vide. Ceci exclut la plupart des éléments fonctionnels (prépositions « vides », complémenteurs, certains pronoms relatifs) en tant qu'entités autonomes de

[7] Dans la grammaire extraite pour le coréen dans ce chapitre, certains arbres élémentaires des prédicats ne contiennent pas les nœuds à substitution ou les nœuds pied ayant les pronoms nuls. Comme il a déjà été expliqué, le corpus SJTree ne contient pas de catégories vides qui correspondent à un argument omis ou à une trace laissé par le déplacement. Pour cette raison, les arbres extraits des prédicats peuvent ne pas contenir les nœuds à substitution ou les nœuds pied.

la syntaxe : ces éléments apparaissent comme co-têtes lexicales dans un arbre élémentaire ayant une tête lexicale non vide ;

- principe de compositionnalité : un arbre élémentaire correspond à une et une seule unité sémantique.

Il s'agit donc de décomposer l'arbre syntaxique du corpus en un ensemble d'arbres élémentaires de manière à ce que chaque arbre élémentaire encode la structure argumentale de son ancre lexicale et que tout modifieur soit représenté par un arbre auxiliaire.

Les arbres élémentaires (les arbres initiaux et les arbres auxiliaires) extraits possibles à partir de la phrase en (1) sont représentés dans la Figure 2. Par convention, les arbres notés par α sont les arbres initiaux, ceux qui sont notés par β sont les arbres auxiliaires[8].

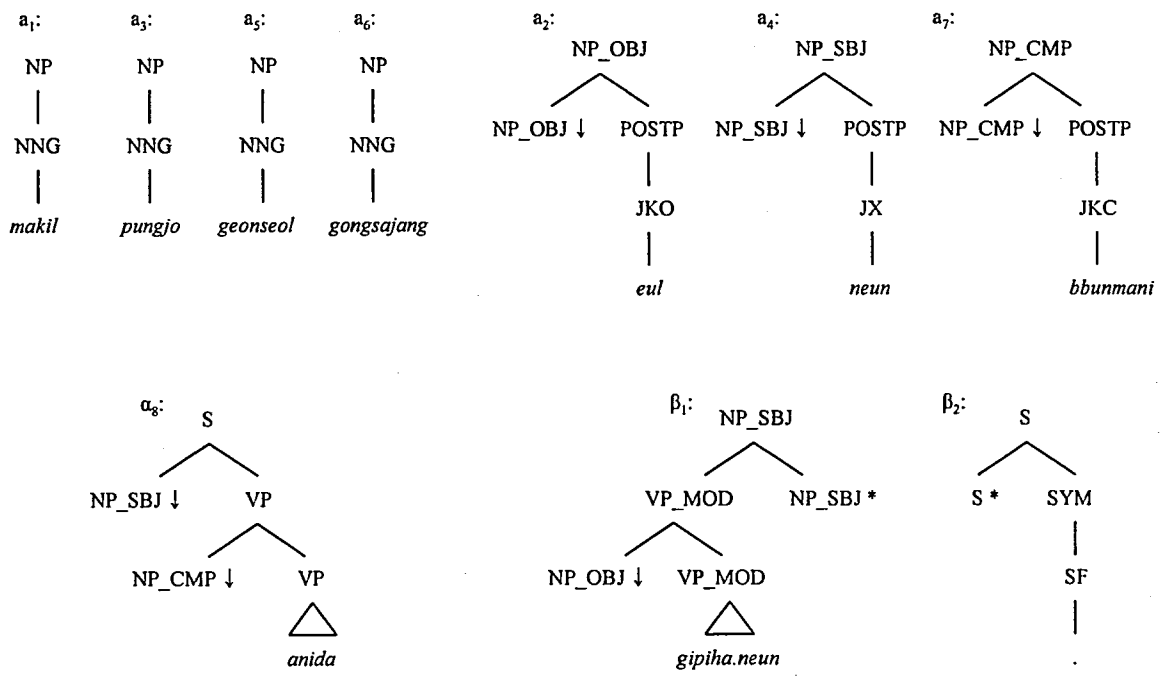


Figure 2. Grammaire TAG lexicalisée extraite pour la phrase donnée dans la Figure 1.

[8] Nous préférons une forme comme (NP (NNG *geonseol* 'construction') pour la grammaire extraite, alors que (NP *geonseol* 'construction'/NNG) pour la représentation de la structure SJTree.

Dans ce chapitre les arbres élémentaires sont construits par un algorithme descendant, donc de la racine vers les feuilles de l'arbre à partir du corpus SJTree. Comme déjà décrit dans le Chapitre 1 sur les « Travaux d'extraction », Chen (2001) et ses variations telles que Johansen (2004), Nasr (2004) et Habash et Rambow (2004) construisent les arbres élémentaires par un algorithme ascendant. Xia (2001) utilise un algorithme descendant, mais celui-ci diffère de celui que nous proposons dans ce chapitre. Xia (2001) extrait d'abord tous les arbres possibles (il y a 2^n arbres possibles où n est le nombre de nœud dans l'arbre morphosyntaxique) et enlève les arbres inappropriés avec les contraintes linguistiques et les contraintes du formalisme TAG lexicalisée tandis que le nôtre n'extrait que les arbres pertinents directement en appliquant ces contraintes (Voir le Chapitre 1 pour les détails sur les travaux d'extraction). L'autre différence entre le travail de Xia et le nôtre est que Xia (2001) modifie le corpus avant d'extraire les arbres élémentaires alors que le nôtre fait une modification en appliquant l'algorithme (ou après l'extraction). Cette modification concerne l'extraction de certains arbres auxiliaires comme les postpositions et les symboles. Nous expliquerons cette modification du corpus dans la section concernée.

Avant l'extraction automatique d'une grammaire TAG lexicalisée, nous transformons la structure des phrases annotées par les parenthèses dans le corpus SJTree en arbres. Ensuite, pour la construction des arbres élémentaires par méthode descendante, nous prenons un algorithme pour le parcours d'arbre dans la structure des données. Il existe deux techniques pour les algorithmes descendants, appelées en « largeur d'abord » et « profondeur d'abord ». Avec la méthode « largeur d'abord », on descend le moins possible, tandis qu'avec la méthode « profondeur d'abord », on descend le plus possible. L'algorithme développé dans ce chapitre choisit le parcours en profondeur d'abord car il extrait les arbres élémentaires ensuivant l'ordre de la phrase, mais théoriquement les deux techniques de parcours d'arbre ne présentent pas de différence de performance.

Le mouvement du point dans l'arbre syntaxique pour le parcours en profondeur d'abord est représenté dans la Figure 3.

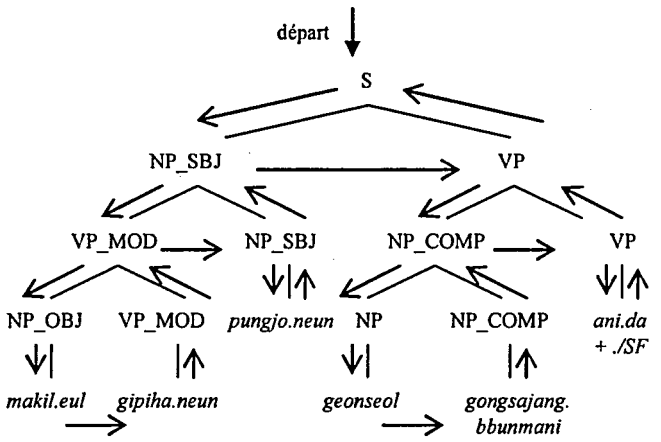


Figure 3. Parcours en profondeur d'abord dans l'arbre syntaxique

A partir de la structure de l'arbre syntaxique de la Figure 1, nous déterminons la tête et le type de l'opération (substitution ou adjonction) pour les nœuds des fils si le nœud donné est une noeud non-terminal. L'extraction des arbres élémentaires de TAG fonctionne comme un parcours d'arbre, de la racine vers les feuilles de l'arbre en substituant les nœuds de substitution et en supprimant les nœuds d'adjonction. Les nœuds supprimés ou substitués pendant le parcours, sont aussi récursivement concernés par l'algorithme, afin de réduire à un arbre élémentaire la structure de l'arbre syntaxique qui reste. Pour cela nous disposons de trois opérations :

1. supprimer les nœuds d'adjonction

2. substituer les nœuds de substitution

3. réduire le tronc (le chemin entre la racine et l'ancre) de l'arbre syntaxique.

Dans la section suivante, nous explorons le détail de l'algorithme d'extraction.

2. Détail de l'algorithme

2.1 Détermination de la tête

Avant de décider si un nœud particulier est un nœud de substitution ou bien un nœud d'adjonction, la tête du syntagme doit être déterminée. Pour la détermination de la tête, nous

supposons que le nœud le plus à droite est une tête parmi les autres frères dans les langues à tête finale comme le coréen. Dans le corpus SJTree la tête du syntagme est facile à trouver, contrairement à d'autres corpora tels que le Penn Treebank (Taylor et al. 2003) ou le corpus de Paris 7 (Abeillé et al. 2003), car le nœud le plus à droite est une tête. Pour cette raison, nous n'utilisons pas le tableau de percolation de la tête comme les autres travaux d'extraction.

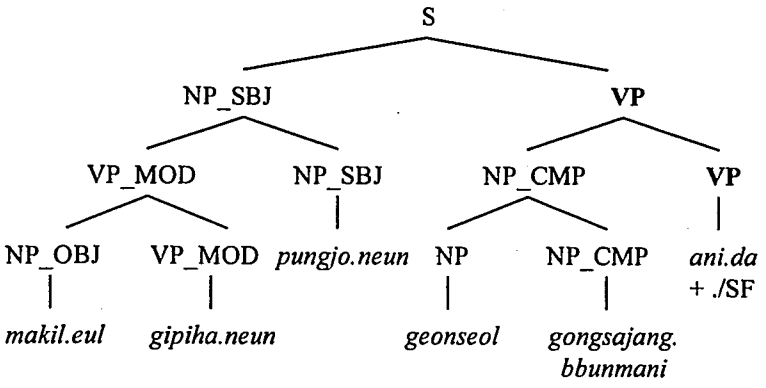


Figure 4. Détermination de la tête dans le corpus SJTree (en gras)

Par exemple, dans la composition [NP NP], le deuxième NP est une tête alors que le premier NP est marqué comme une opération d'adjonction et extrait comme un arbre auxiliaire dans la grammaire G_1 qui utilise les *eojeol*s directement sans modification du corpus SJTree (voir la section 3 pour les détails de l'expérimentation d'extraction). Cependant, il y a une exception pour la détermination de la tête. Dans la composition [VP@VV VP@VX] où le premier VP a une ancre VV (verbe) et le deuxième VP a une ancre VX (verbe auxiliaire), le verbe auxiliaire (VX) est systématiquement marqué comme une opération d'adjonction et extrait comme un arbre auxiliaire. Le verbe (VV) est une tête et il est extrait comme un arbre initial qui contient tous les arguments de la phrase.

2.2 Distinction entre arbre initial et arbres auxiliaire

Ensuite, nous devons distinguer les nœuds de substitution et d'adjonction, ce qui équivaut à une distinction d'étiquetage entre complément et ajout dans la phrase. Différemment des autre corpus arboré comme le Penn Treebank et le Paris 7 Treebank pour le français on pourrait dire que les nœuds de substitution sont distingués explicitement car les étiquettes fonctionnelles parmi 55 étiquettes syntaxiques du corpus SJTree peuvent être considérées comme des

marqueurs de cette distinction. Le Tableau 1 indique sur quel nœud nous pouvons marquer comme une opération de substitution

NP_CMP	NP_OBJ	NP_SBJ
VNP_CMP	VNP_OBJ	VNP_SBJ
VP_CMP	VP_OBJ	VP_SBJ
S_CMP	S_OBJ	S_SBJ

Tableau 1. Etiquettes fonctionnelles pour l'opération de substitution

Parmi 55 étiquettes syntaxique, les syntagmes NP (le syntagme nominal), VNP (le syntagme de copule), VP (le syntagme verbal), et S (la phrase) qui finissent par _CMP (l'attribut), _OBJ (l'objet), et _SBJ (le sujet) sont marqués comme une opération de substitution. Les nœuds étiquetés par les autres étiquettes, sauf la tête, sont marqués comme une opération d'adjonction.

Dans cette distinction, il est possible que les syntagmes VNP et VP soient marqués comme une opération de substitution, ce qui veut dire que les syntagmes VNP et VP sont les arguments de la tête du syntagme car SJTree distribue les étiquettes VNP et VP au lieu de NP pour les formes de nominalisation de VNP et de VP.

Dans la Figure 4, NP_SBJ, NP_OBJ et NP_CMP sont marqués comme une opération de substitution et les autres étiquettes telles que VP_MOD et NP (pour *geonseol* ('construction')) sont marquées comme une opération d'adjonction.

Le Tableau 1 ne contient pas NP_AJT. Le syntagme du nœud étiqueté par NP_AJT, en général, finit par une postposition auxiliaire ou une postposition adverbiale. Il peut être considéré comme le syntagme adverbial et il est contrôlé par l'opération d'adjonction. Cependant les syntagmes locatifs ou datifs obligatoires dans la phrase sont aussi étiquetés comme NP_AJT et il ne peut pas être traité comme une opération d'adjonction. Par exemple, *jungguk.ey* dans (2) est NP_AJT qui est un argument complémentaire pour le verbe *ga*

('aller') et il doit être traité comme une opération de substitution. Pour résoudre ce problème, nous adoptons la méthode de Nasr (2004) et nous renvoyons à la section 3.6 « Syntagme adjoint (_AJT) ».

(2) a. 중국에 갈

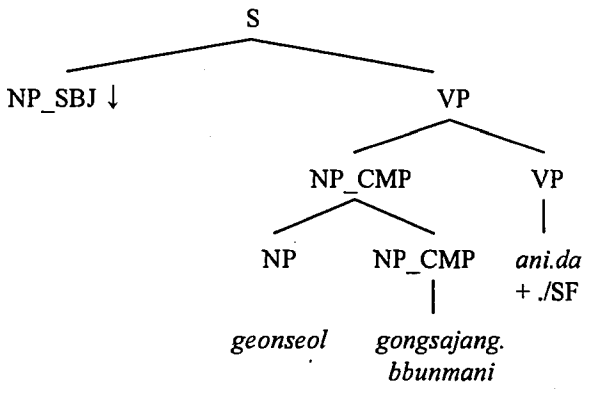
jungguk.ey *ga.l*

chine.ey aller.l

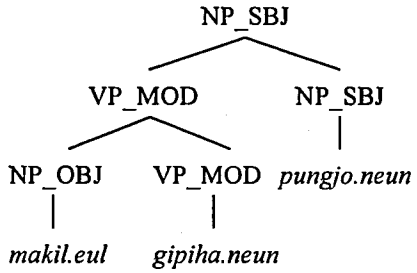
'Aller en Chine'

b. (VP (NP_AJT *jungguk* ('chine')/NNP + *ey* /JKB)
(VP *ga* ('aller')/VV + *l*/ETM))

Les nœuds fils marqués comme une opération de substitution sont remplacés par des nœuds terminaux de substitution (e.g. NP_SBJ↓) et appellent récursivement la procédure d'extraction avec le sous-arbre où le nœud racine est celui du nœud fils lui-même. Dans la Figure 4, le nœud NP_SBJ qui est marqué comme une opération de substitution est remplacé par un nœud terminal « NP_SBJ↓ » dans l'arbre principal et le sous-arbre (NP_SBJ (VP_MOD *makil.eul*... *gipiha.neun*...) (NP_SBJ *pungjo.neun*)) appelle récursivement la procédure d'extraction (Voir la Figure 5).



a. Arbre principal après l'opération de substitution

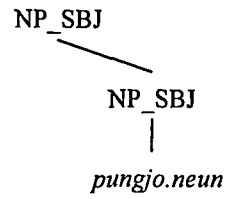


b. Sous-arbre après l'opération de substitution

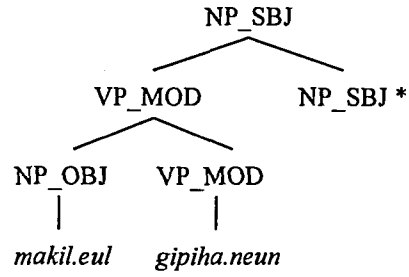
Figure 5. Opération de substitution

Les nœuds fils marqués comme une opération d'adjonction sont supprimés de l'arbre principal et appellent récursivement la procédure d'extraction avec le sous-arbre où nous ajoutons le nœud racine avec celui du nœud parent du nœud donné et son frère (le nœud pied, e.g. NP_SJB*) avec celui du nœud donné. Comme il est défini dans le formalisme, la racine et le nœud pied du sous-arbre pour l'opération d'adjonction partagent la même étiquette. Dans la Figure 4, le nœud VP_MOD qui est marqué comme une opération d'adjonction est supprimé dans l'arbre principal et le sous-arbre (NP_SBJ (VP_MOD *makil.eul... gipiha.neun...*) NP_SBJ*) appelle récursivement la procédure d'extraction[9] (Voir la Figure 6).

[9] Effectivement le sous-arbre (VP_MOD *makil.eul... gipiha.neun...*) appelle récursivement la procédure d'extraction dans l'exemple de la Figure 4. Ensuite, nous ajoutons le nœud racine celui du nœud parent du nœud donné et son frère comme le nœud pied pour former un arbre auxiliaire. Si le sous-arbre (NP_SBJ (VP_MOD *makil.eul... gipiha.neun...*) NP_SBJ*) appelle récursivement la procédure d'extraction, le système d'extraction tourne indéfiniment.



a. Arbre après l'opération d'adjonction



b. Sous-arbre après l'opération d'adjonction

Figure 6. Opération d'adjonction

2.3 Elagage

Les grammaires extraites expliquées au-dessous ne sont pas les grammaires TAG lexicalisées « correctes. » L'élagage signifie une réduction du tronc qui reste après avoir supprimé les nœuds d'adjonction. Après avoir supprimé les nœuds d'adjonction, il reste encore des nœuds dont nous n'avons pas besoin pour construire une grammaire TAG lexicalisée. La Figure 7 montre cette opération de réduction du tronc pour le verbe *malhada* ('parler').

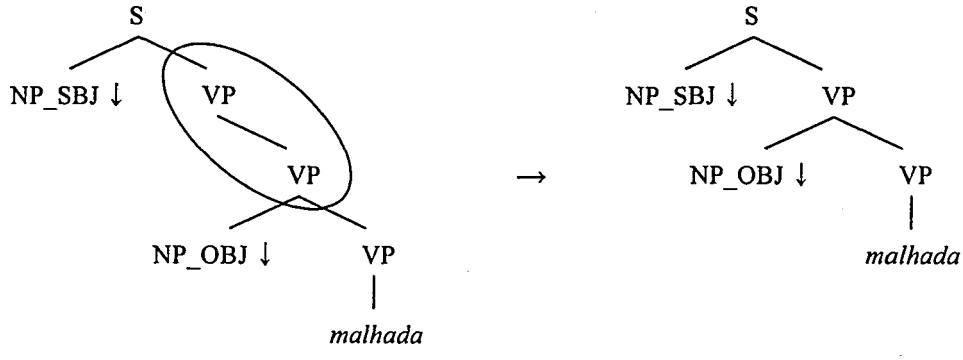
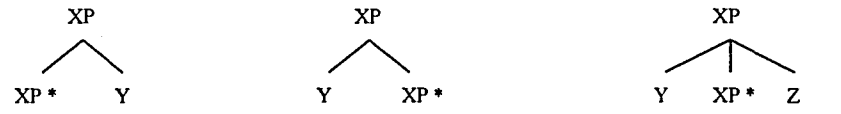


Figure 7. Procédure d'élagage

Voir l'Annexe pour l'algorithme d'extraction.

3. Les différents types d'arbre auxiliaires

L'arbre auxiliaire extrait du verbe auxiliaire est un arbre gauche. Toutefois, nous distinguons trois types d'arbres auxiliaires selon la position du nœud pied : un arbre auxiliaire dans lequel le nœud pied est la feuille la plus à gauche est un arbre auxiliaire gauche (par exemple, l'arbre pour le symbole final dans une grammaire extraite du SJTree), si le noeud pied est la feuille la plus à droite, il s'agit d'un arbre auxiliaire droite (l'arbre pour le modifieur verbal (VP_MOD) et l'adverbe dans une grammaire extraite du SJTree). Si le nœud pied ne se trouve ni à l'extrême gauche ni l'extrême droite, il est un arbre auxiliaire enveloppant (marqueur de la citation " " ou ' ') (Johansen 2004)[10]. Les trois types d'arbres auxiliaires sont décrits dans la Figure 8.



a. Arbre auxiliaire gauche | b. Arbre auxiliaire droite | c. Arbre auxiliaire enveloppant

Figure 8. Trois types d'arbres auxiliaires

4. Expériences d'extraction

Initialement, nous extrayons 18 146 arbres lexicalisés c'est-à-dire 5 419 arbres initiaux et 12 727 arbres auxiliaires, directement à partir du corpus SJTree. Dans cette section, nous extrayons non seulement les arbres lexicalisés sans modification du corpus SJTree, mais aussi les grammaires avec modification du corpus en utilisant certaines contraintes pour améliorer les grammaires extraites et leur couverture lexicale.

- G_1 : Nous extrayons la grammaire en utilisant les *eojeol*s directement sans modification du corpus SJTree

- G_2 : Nous séparons les symboles des *eojeol*s. Les symboles contiennent non seulement les signes de ponctuation mais aussi les chiffres et les caractères étrangers. Les symboles séparés sont extraits et divisés en α et β arbres sur leur type.

[10] Dams cette thèse, nous n'extrayons pas l'arbre auxiliaire enveloppant.

- G_3 : Dans le cas de d'*eojeol* [NOM + POSTPOSITION], nous séparons les postpositions des noms. Les postpositions séparées sont extraites comme des arbres initiaux. Les postpositions complexes qui contiennent deux ou plus postpositions sont aussi extraites comme un arbre initial.

- G_4 : Nous convertissons les arbres auxiliaires pour le syntagme nominal en arbres initiaux. Par exemple, dans la grammaire extraite dans la Figure 4, un arbre auxiliaire (NP (NNG *geonseol* 'construction') (NP *)) est converti comme un arbre initial (NP (NNG *geonseol* 'construction')). Nous enlevons aussi les étiquettes syntaxiques des arbres initiaux dans le syntagme nominal. Par exemple, dans la grammaire extraite dans la Figure 4, (NP_SBJ (NNG *pungjo* 'tendance')) est enlevée son etiquette syntaxique NP_SBJ et remplacée par NP, c'est-à-dire (NP (NNG *pungjo* ('tendance'))).

Dans les expériences d'extraction d'une grammaire lexicalisée ci-dessus, nous supposons que les grammaires supérieures suivent les grammaires inférieures. Par exemple, la séparation des postpositions dans la grammaire G_3 résulte de la séparation des symboles dans la grammaire G_2.

Le Tableau 2 montre la taille des grammaires extraites. Les chiffres donnés indiquent aussi le nombre d'arbres initiaux et auxiliaires.

	arbes élémentaires	arbres initiaux	arbres auxiliaires
G_1	18080	5399	12681
G_2	17976	5635	12341
G_3	17382	5586	11796
G_4	15551	9175	6376

Tableau 2. Taille des grammaires extraites

Nous additionnons 436 arbres initiaux et 32 arbres auxiliaires pour les symboles séparés dans la grammaire G_2 et 299 arbres auxiliaires pour les postpositions dans la grammaire G_3. Pour les arbres extraits des symboles et des postpositions, nous ajoutons les étiquettes syntaxiques

SYM et POSTP comme un nœud non-terminal intermédiaire que le corpus SJTree n'utilise pas. La Figure 9 montre les grammaires extraites de G_1 à G_4 à partir de la phrase en (3).

(3) 일본 외무성은즉각해명성명을발표했다.

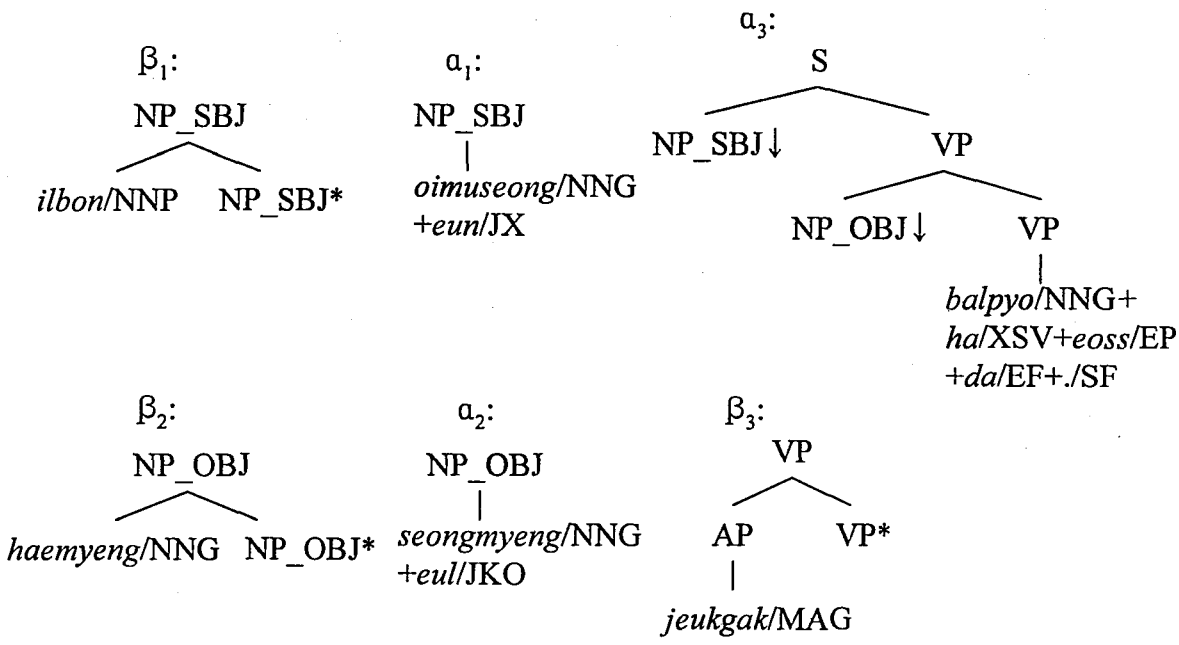
ilbon *oimuseong.eun* *jeukgak* *haemyeng*

Japon ministre_des_affaires_étrangères.Nom immédiatement élucidation

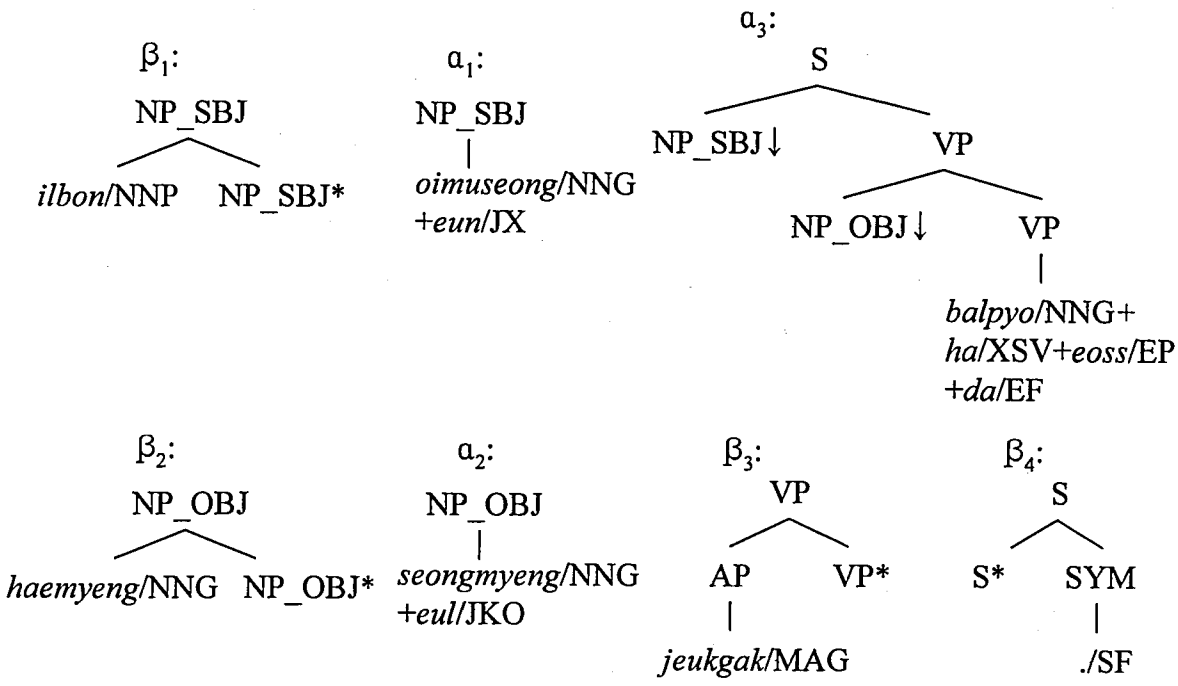
seongmyeng.eul *balpyo.ha.eoss.da*

déclaration.Acc annoncer.Pass.Ter

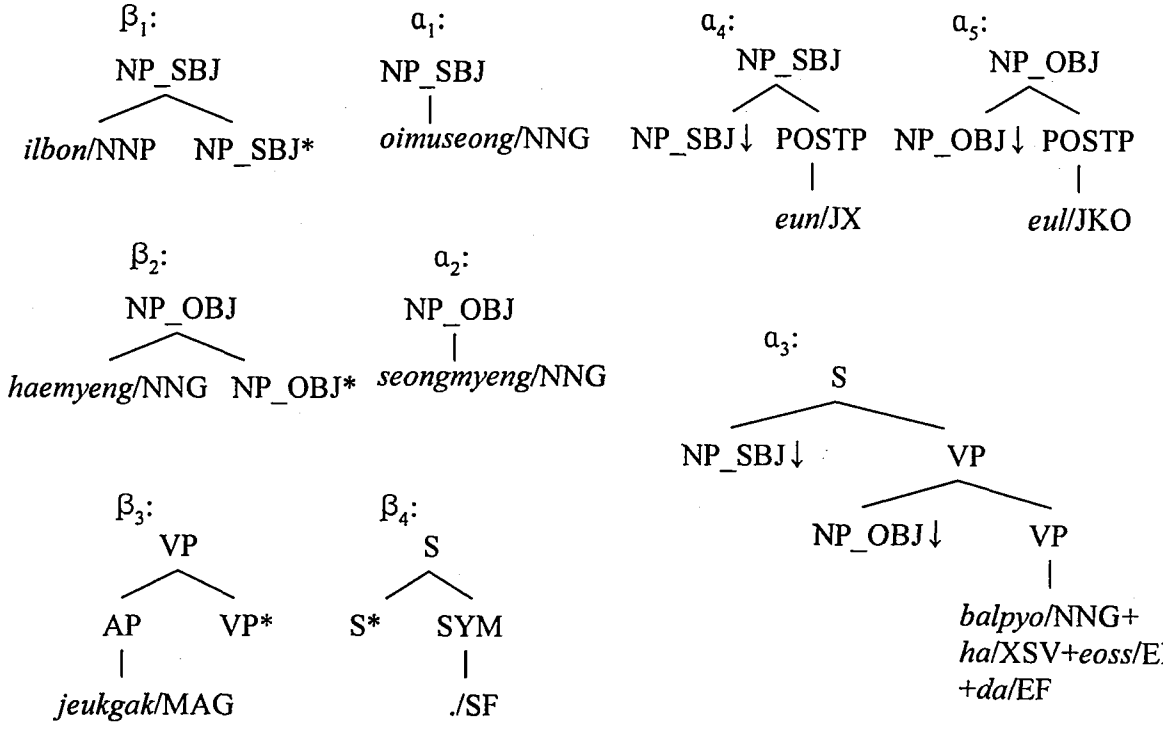
'Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation.'



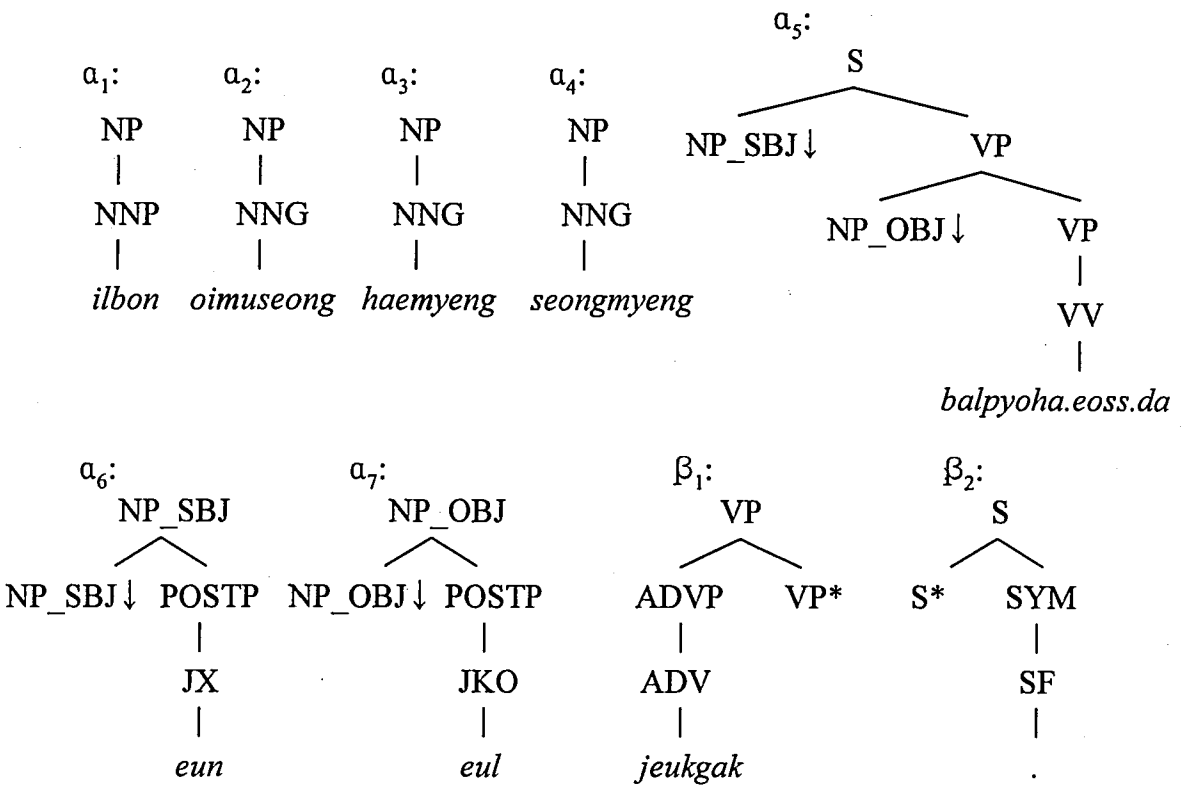
a. Grammaire extraite G_1



b. Grammaire extraite G_2



c. Grammaire extraite G_3



d. Grammaire extraite G_4

Figure 9. Grammaires extraites à partir de la phrase en (2)

La raison principale pour laquelle nous séparons les symboles et les postpositions est bien indiquée dans le Tableau 3. Toutes les fréquences moyennes (`sum(freq)/count(ltree)` où `sum(freq)` est un nombre total de fréquence de la grammaire extraite et `count(ltree)` est un nombre d'arbres lexicalisés des grammaires extraites ne dépassent pas 4. Cependant, les fréquences moyennes des symboles et des postpositions sont, respectivement, 13.80 et 31.59. Alors que les grammaires extraites ne garantissent pas toutes les possibilités de la composition d'*eojeol* tels que [syntagme + symbole] et [syntagme + postposition], nous décidons la séparation des symboles et des postpositions. Cette séparation implique la modification du corpus SJTree[11].

G_1	G_2	G_3	G_4
1.38	1.71	2.30	2.57

Tableau 3. Fréquences moyennes des grammaires extraites

Dans la Figure 10 et le Tableau 4, la couverture lexicale des grammaires extraites ci-dessus montre aussi que les grammaires supérieures sont meilleures que les grammaires inférieures. Nous faisons cette expérimentation de la couverture du lexique en appliquant les grammaires extraites à un corpus de 770 000 *eojeol*s qui est morphologiquement analysé. Cette couverture lexicale ne concerne pas la couverture syntaxique des grammaires extraites. La couverture du lexique de la grammaire G_4, où nous convertissons les arbres auxiliaires pour le syntagme nominal en arbres initiaux et enlevons les étiquettes syntaxiques des arbres initiaux dans le syntagme nominal de la grammaire G_3, est la même que celle de G_3.

[11] La complexité du parseur par rapport à la taille de grammaire est indiquée dans Nasr (2004). Dans sa thèse d'habilitation, il a montré que la complexité est diminuée quand la taille de la grammaire est augmentée. Dans cette thèse, nous ne considérons pas la complexité du parseur, nous considérons seulement la couverture lexicale des grammaires extraites.

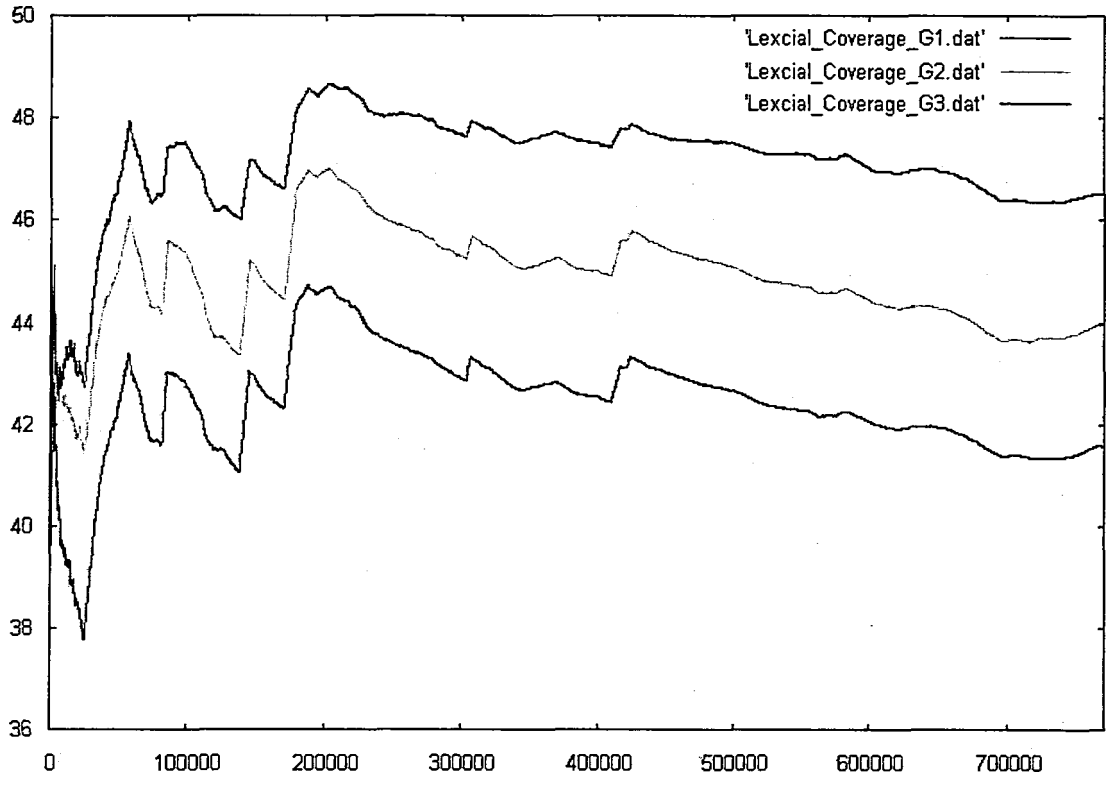


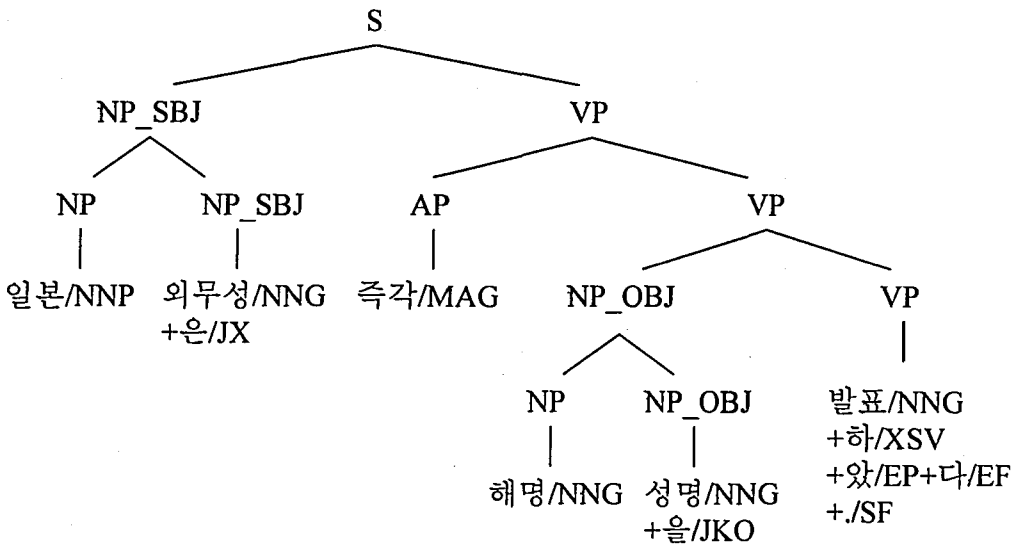
Figure 10. Couverture lexicale des grammaires extraites (x : # d'*eojeol*s et y : couverture lexcale (%))

G_1	G_2	G_3	G_4
36,86	40,21	43,41	43,41

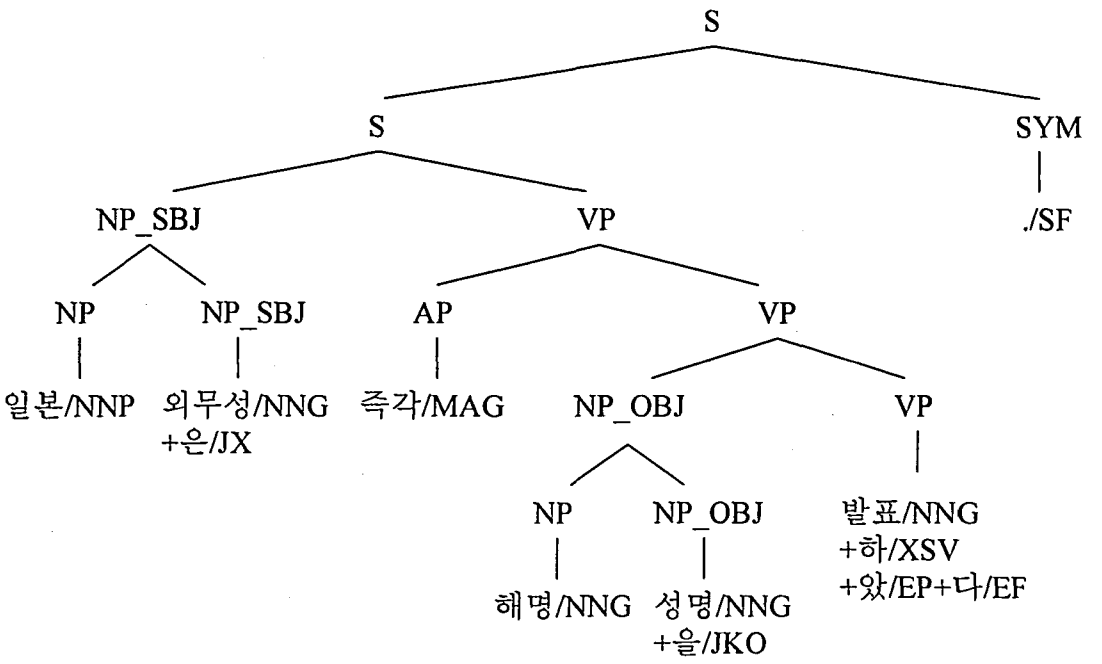
Tableau 4. Couverture lexicale moyenne des grammaires extraites

La Figure 11 montre les analyses différentes avec les grammaires extraites différentes pour la phrase en (3). L'analyse avec la grammaire G_1 est montrée dans la Figure 11a qui est tout à fait la même analyse que celle du corpus SJTree, celle avec la grammaire G_2 dans la Figure 11b montre une séparation du symbole et celle avec la grammaire G_3 dans la Figure 11c montre une séparation des postpositions. Cependant, les grammaires G_4 ne peuvent êtres analysées comme dans la Figure 11d que si nous adoptons les propositions de Park (2004b) et Park (2004c) qui proposent une structure plate intérieure pour les syntagmes nominaux et un *shallow* parseur avant l'analyse syntaxique. Nous avons aussi besoin de certaines contraintes comme « Tous les nœuds pour l'opération de substitution peuvent être substitués par

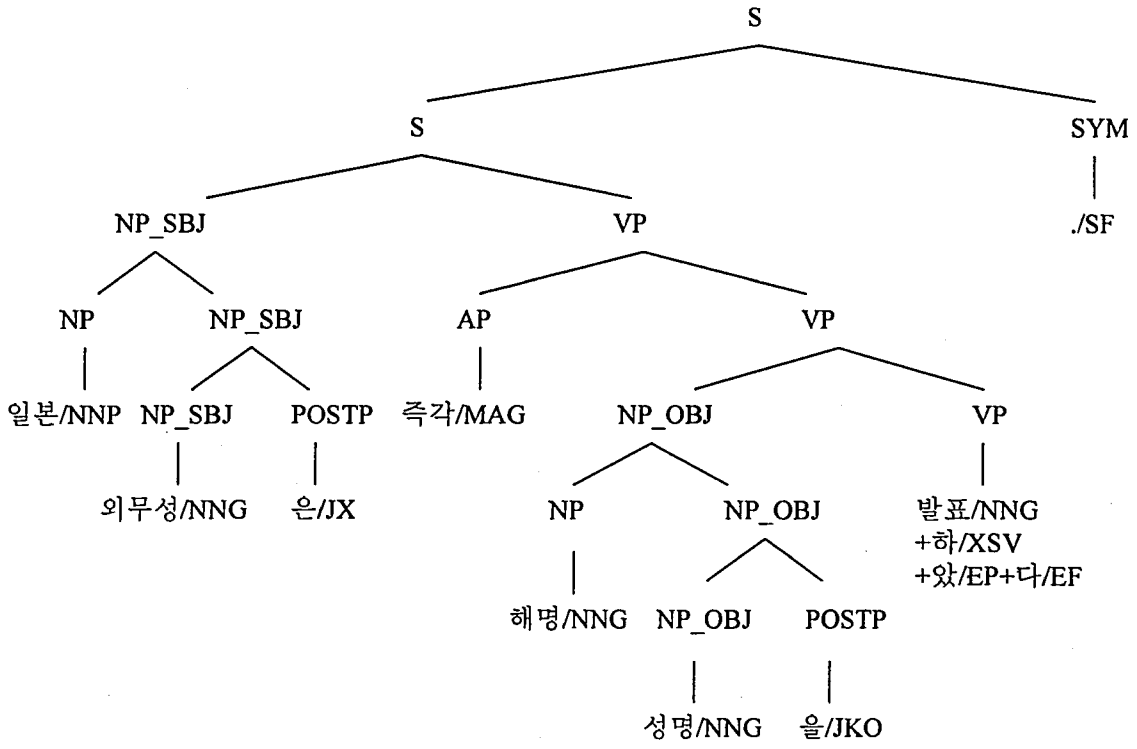
n'importe quel syntagme nominal» Nous renvoyons à l'Annexe « *chunker* » pour le détail de ces propositions.



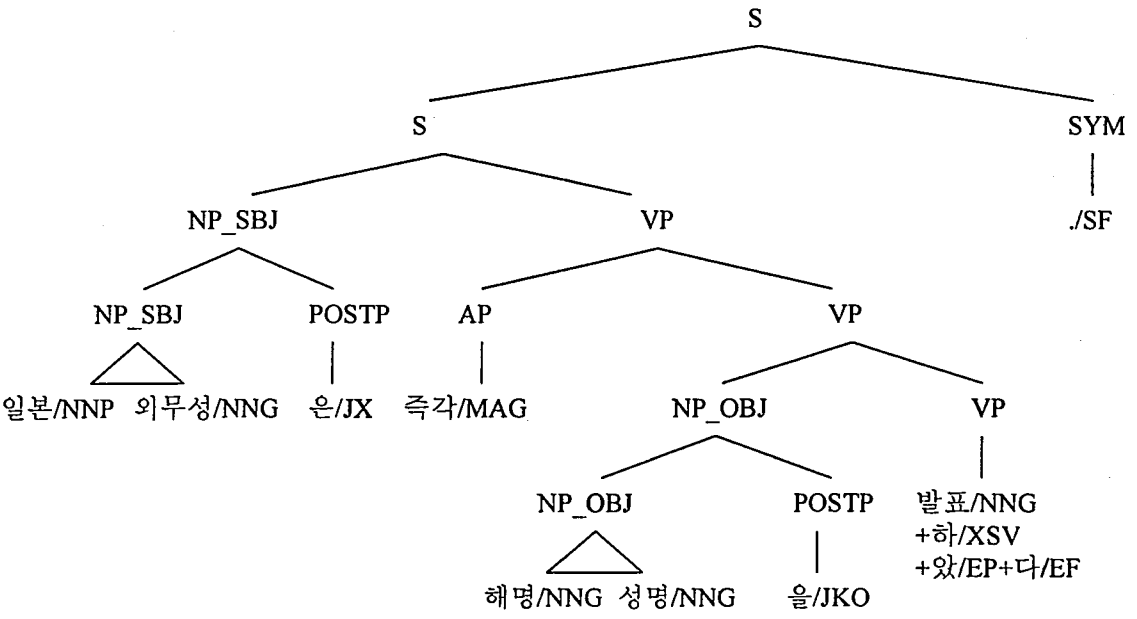
a. Analyse avec la grammaire G_1



b. Analyse avec la grammaire G_2



c. Analyse avec la grammaire G_3



d. Analyse avec la grammaire $G_{4/5}$

Figure 11. Analyses différentes avec les grammaires extraites différentes

Comme nous avons déjà mentionné dans notre « Introduction », un des problèmes de l'extraction automatique d'une grammaire est le fait d'avoir une couverture lexicale limitée. Pour résoudre ce problème, nous agrandissons les grammaires extraites en utilisant des gabarits appelés « schéma d'arbre ». L'ancre lexicale dans les grammaires extraites est enlevée et remplacée par un marqueur d'ancre pour former une grammaire de gabarits. Par exemple, un marqueur d'ancre est @NNG où l'ancre lexicale des grammaires extraites est un nom commun qui remplace l'ancre lexicale.

Le résultat d'expérimentation après conversion en schéma d'arbres est montré dans le Tableau 5. Comme l'expérimentation d'extraction des grammaires lexicalisées, nous supposons que les schémas d'arbre supérieurs résultent des schémas d'arbre inférieurs. Par exemples, la séparation des postpositions dans le schéma d'arbre T_3 suit la séparation des symboles dans le schéma d'arbre T_2. Les chiffres donnés dans le Tableau 5 indiquent aussi le nombre de schémas d'arbre initiaux et auxiliaires.

	nombre des schémas	α schémas	β schémas
$G_1 \rightarrow T_1$	1158	326	832
$G_2 \rightarrow T_2$	1153	315	838
$G_3 \rightarrow T_3$	1192	372	820
$G_4 \rightarrow T_4$	1077	379	698

Tableau 5. Chiffres des schémas d'arbres

Le Tableau 6 montre les fréquences moyennes des schémas d'arbre.

T_1	T_2	T_3	T_4
21.55	26.62	33.48	37.05

Tableau 6. Fréquences moyennes des schémas d'arbres

Dans la section suivante, nous explorons les exemples selon les types de syntagmes.

5. Les exemples selon les types de syntagmes

Les arbres extraits qui sont présentés dans cette section appartiennent à la grammaire G_4 si nous n'indiquons pas de grammaire spécifique.

5.1 Le syntagme nominal

5.1.1 Les noms simples

Le nom simple que le système extrait a une structure du type (NP *lexique*/NNG)), les noms communs ont la structure (NP *lexique*/NNG)) et les noms propres (NP *lexique*/NNP))[12] (Voir α_1 , α_3 , α_5 et α_6 de la Figure 2).

5.1.2 Les noms composés

Le corpus SJTree s'appuie sur certaines particularités du nom composé[13] en coréen qui sont problématiques du point de vue linguistique. Selon la linéarisation générale du coréen, la tête nominale est en dernière position :

(4) 건설 현장이

geonseol *hyeonjang.i*

construction chantier.Nom

'le chantier de construction'

[12] La structure (NP *lexique*/NNG)) est une représentation semblable à celle du corpus SJTree, pourtant elle représente une structure (NP (NNG (*lexique*))) comme décrit dans la Figure 1. Dans cette thèse, nous adoptons les deux représentations pour présenter la structure de la grammaire.

[13] Dans cette section, nous limitons la discussion sur les noms composés aux séquences de noms uniquement et n'adordons pas les autres séquences telles que [NOM + ADJECTIF], [DETERMINANT + NOM].

(NP_SBJ
(NP 건설/NNG)
(NP_SBJ 현장/NNG + 이/JKS))

Figure 10. Exemple de nom composé tiré du corpus SJTree

En (4), la postposition attachée à la tête marque la fonction casuelle (syntaxique), en l'occurrence le sujet, et cette fonction est héritée par le syntagme entier. Cependant il existe aussi des noms composés qui ne présentent pas explicitement leur tête :

(5) 일반 주제
ilban *juje*
général sujet
'le sujet général'

(NP
(NP 일반/NNG)
(NP 주제/NNG))

Figure 11. Exemple de nom composé tiré du corpus SJTree (suite)

Voyons un autre exemple :

(6) 산업 공동화 우려
saneob gongdonghwoa uryeo
industrie crise inquiétude
'l'inquiétude concernant la crise industrielle

(NP
(NP
(NP 산업/NNG)
(NP 공동화/NNG))
(NP 우려/NNG))

Figure 12. Exemple de nom composé tiré du corpus SJTree (suite)

Le nom composé ci-dessus n'a pas de postposition comme en (5). On pourrait supposer que 우려 *uryeo* ('inquiétude') est la tête dans (6) selon l'ordre général du coréen. Cette analyse peut être justifiée par le fait que les syntagmes nominaux 산업 공동화 *saneob gongdonghwoa* ('la crise industrielle') et 우려 *uryeo* ('l'inquiétdue') presentent une relation de sous-catégorisation. Donc, *saneob gongdonghwoa* est un argument du nom prédicatif *uryeo*, ce qui permet de former un syntagme verbal d'un verbe dérivé de *uryeo* avec son argument *saneob gongdonghwoa* comme décrit en (6').

(6') 산업의 공동화를 우려한다

saneob.ui *gongdonghwoa.reul* *uryeo.ha.n.da.*

industrie.Gen creux.Acc souci.faire.Pre.Ter

'On s'inquète de la crise industrielle'

Les arbres des noms composés des exemples (4) à (6) sont représentés respectivement dans les schéma (a) à (c) de la Figure 12.

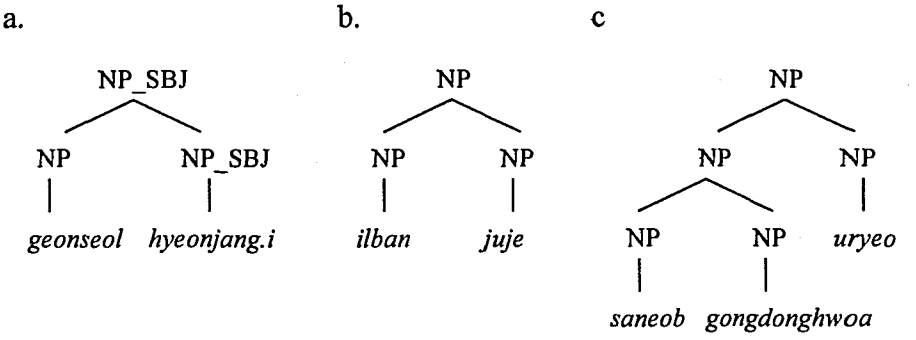


Figure 12. Trois types de syntagmes nominaux dans le corpus SJTree

Comme il a été déjà mentionné, la relation entre les noms est un sujet très difficile en linguistique coréenne qui peut faire l'objet de plusieurs thèses. Pour cette raison et pour faciliter le travail présent, nous considérerons le nom composé comme une « structure plate », donc tous les noms extraits vont avoir une structure du type (NP *lexique*/NNG)) pour les noms communs et (NP *lexique*/NNP)) pour les noms propres comme les noms simples.

Contrairement à notre travail, Xia (2001) extrait le nom en représentant une relation de tête. L'autre raison pour laquelle nous ne représentons pas cette relation dans le nom composé est

que c'est une redondance pour la grammaire extraite du nom. Si l'on accepte la méthode d'extraction de Xia (2001) pour les noms, il y aurait trop de grammaires pour les noms car un nom peut aussi bien apparaître en position de gouvernant qu'en position de gouverné (Park 2004a). Avec une structure plate pour le nom composé nous n'avons pas cette redondance, et nous pouvons analyser cette structure plate plus en détail dans les cas nécessaires comme le *deep parsing* (Park 2004b, Park 2004c).

5.1.3 Les noms dépendants

Le terme « dépendant » ou « défectif » n'est pas une notion syntaxique mais sémantique. Syntaxiquement, le nom construit son propre syntagme nominal mais sémantiquement il a un sens défectif, ce qui lui fait requérir toujours un modifieur. Il existe deux types de noms dépendants : les noms dépendants d'unité et les noms dépendants de formalité. Les noms dépendants d'unité se trouvent souvent à la fin du syntagme nominal comme : 4/SN + 일/NNB (*4 il*, '4 jours'). La phrase en (7) « *chaj.a bo.l* ***su*** *eobs.eoss.da* » présente un exemple des noms dépendants de formalité.

(7) 찾아 볼 수 없었다

chaj.a	*bo.l*	***su***	*eobs.eoss.da*
trouver.*a*	essayer.*l*	*su*	ne_pas.Pass.Ter

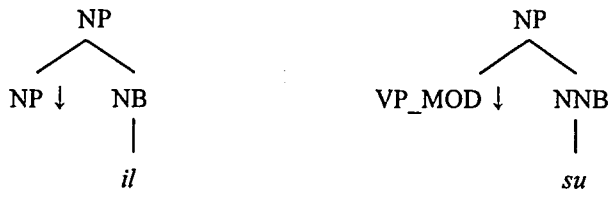
'On ne pouvait pas le trouver'

(S (NP_SBJ (VP_MOD (VP 찾/VV + 아/EC)
(VP_MOD 보/VX + ㄹ/ETM))
(NP_SBJ 수/NNB))
(VP 없/VA + 었/EP + 다/EF))))

Figure 13. L'analyse syntaxique de la phrase en (10)

Le nom dépendant est traité comme les autres noms car sa caractéristique est presque la même, sauf que le nom dépendant a besoin d'un modifieur qui le précède. Nous pouvons ainsi considérer un arbre initial qui a besoin d'un nœud de substitution pour le modifieur comme dans la Figure 14. Cependant, l'arbre initial pour le nom dépendant de formalité n'est pas approprié aux contraintes imposées par le formalisme TAG lexicalisée. De plus, dans le corpus SJTree, les noms dépendants d'unité et ceux dépendants de formalité ne sont pas bien

distingués. Donc tous les noms dépendants extraits vont avoir un type (NP *lexique*/NNB)) comme les noms simples ou composés.



a. Arbre initial pour le nom dépendant d'unité

a. Arbre initial pour le nom dépendant de formalité

Figure 14. Arbre initial « *supposé* » pour le nom dépendant d'unité et pour le nom dépendant de formalité.

Les arbres pour le nom dépendant vont associer un trait spécifique, rendant l'adjonction d'un modifieur obligatoire dans la grammaire G_5. (Voir le chapitre 4 « L'extraction de grammaires avec traits »).

5.2 Les postpositions

Un autre problème à traiter dans la procédure d'extraction est celui des postpositions. En effet, avec l'extraction des postpositions séparément, il y a une modification du corpus. Comme nous l'avons montré en α_3 et α_4 de la Figure 2, dans la grammaire TAG lexicalisée extraite, les postpositions sont traitées comme substitutions. Dans les syntagmes nominaux, les postpositions s'attachent toujours à l'*eojeol* des noms, par exemple : *pwungjo.neun* ('tendance.Nom'), mais elles sont considérées comme une unité indépendante dans la grammaire extraite. Autrement dit, la postposition est extraite séparément des noms, donc un arbre initial.

L'analyse de la séparation de la postposition des noms se retrouve dans plusieurs références (Nam K-S. et Ko Y-K. 2003 et Nam K-S. 2001). Par conséquent, l'analyse avec la grammaire résultat est [NP_SBJ [NP_SBJ *pwungjo*][JKS *neun*]] pour NP_SBJ de la Figure 1 (voir aussi la Figure 9c et 9d).

Les postpositions complexes qui sont composées de deux postpositions ou plus sont extraites comme un arbre initial unique (Voir α_7 de la Figure 2). Pourtant, la postposition qui est

attachée aux verbes avec une terminaison en forme fléchie comme 들어서는 *deul.eosoe.neun* (들/VV + 어서/EC + 는/JX, 'entrer dans') n'est pas considérée dans la procédure d'extraction, et celle qui est attachée aux adverbes, par exemple, 아직도 *ajik.do* (아직/MAG + 도/JX, 'encore') est extraite avec l'adverbe comme un arbre auxiliaire droit.

5.3 Le syntagme verbal

5.3.1 Les verbes simples

Les verbes sont extraits par la procédure d'extraction expliquée plus haut : supprimer les nœuds d'adjonction, substituer les nœuds de substitution et réduire le tronc (le chemin entre la racine et l'ancre) de l'arbre syntaxique. La grammaire extraite pour le verbe ne contient pas la terminaison verbale car la productivité de la forme fléchie du verbe en coréen est exponentielle. Dans l'extraction du verbe, nous devons aussi considérer les verbes auxiliaires et les verbes composés.

5.3.2 Les verbes auxiliaires

Les verbes auxiliaires en coréen ont les caractéristiques suivantes :

1. Plusieurs verbes auxiliaires peuvent être présentés dans une phrase.

2. Les verbes auxiliaires ne peuvent pas apparaître comme un verbe unique, mais doivent être précédé par le verbe principal

3. Tous les verbes auxiliaires suivent le verbe principal dans la séquence verbale.

Le verbe auxiliaire est extrait comme un arbre auxiliaire, car dans le syntagme verbal comme une composition [VP@VV VP@VX] où le premier VP a une ancre VV (verbe) et le deuxième VP a une ancre VX (verbe auxiliaire), le verbe auxiliaire (VX) est systématiquement marqué comme une opération d'adjonction et extrait comme un arbre auxiliaire.

5.3.3 Les verbes composés

La plupart des séquences [VP [VP *lexique*][VP *lexique*]] ne sont pas des verbes composés, comme, par exemple les séquences [VP [VP *adjectif*] [VP *verbe*]] telles que 크게 오르다 *keu.gey oreu.da* ('augmenter beaucoup') en (8). Pendant la procédure d'extraction, le deuxième VP est déterminé comme la tête, c'est-à-dire que le premier est traité comme adjoint.

(8) a. 크게 오르다
keu.gey oreu.da
grand.*gey* augmenter.Ter
'Augmenter beaucoup'

b. (VP (VP 크/VA + 게/EC)
(VP 오르/VV + 다/EF))

Dans le cas du verbe composé [VP [VP *verbe*] [VP *verbe*]] comme 전해 들었다 *jeonha.a deul.eoss.da* ('entendre par l'intermédiaire de') en (9), le système traite aussi l'extraction du premier comme adjonction et celle du deuxième comme la tête.

(9) a. 전해 들었다
jeonha.a *deud.eoss.da*
passer.a entendre.Pass.Ter
'Entendre par l'intermédiaire de'

b. (VP (VP 전하/VV + 아/EC)
(VP 듣/VV + 었/EP + 다/EF))

5.4 Le syntagme adverbial

Quand le système rencontre le nœud du syntagme adverbial (AP), il réfère au parent du nœud actuel qui permet de faire le nœud de la racine et le nœud pied d'arbre auxiliaire et génère un arbre pour l'adverbe (Voir la Figure 15 pour la phrase en (10)).

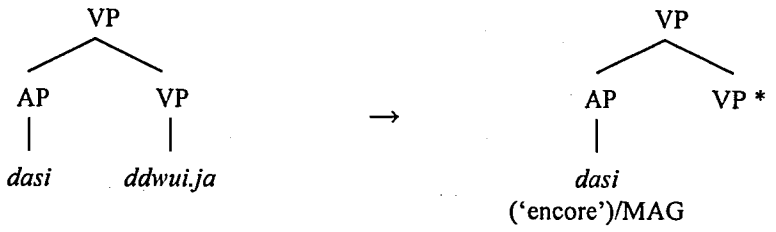


Figure 15. Exemple d'adverbe tiré du corpus SJTree et la grammaire extraite.

(10) 다시 뛰자

dasi *ddwui.ja*

encore courir.Ter

'Courrons encore'

Cependant il existe aussi le syntagme adverbial co-tête. Dans le cas d'une co-tête adverbial, nous extrayons deux arbres auxiliaires séparément (Voir la Figure 16 pour la phrase en (11)). Cependant, la procédure d'extraction donne (AP (AP *iri*('autant')/MAG) (AP*)) et (AP (AP *oai*('pourquoi')/MAG) (AP*)) qui ne sont pas appropriés à une grammaire des adverbes pour la phrase en (10). Pour cette raison, le système d'extraction génère systèmatiquement un arbre auxiliaire des adverbes comme une racine et un nœud nœud VP si le parent du nœud actuel est AP. L'analyse avec une grammaire extraite pour le syntagme adverbial co-tête ne sera pas semblable à la structure origine du corpus SJTree (Voir la Figure 17).

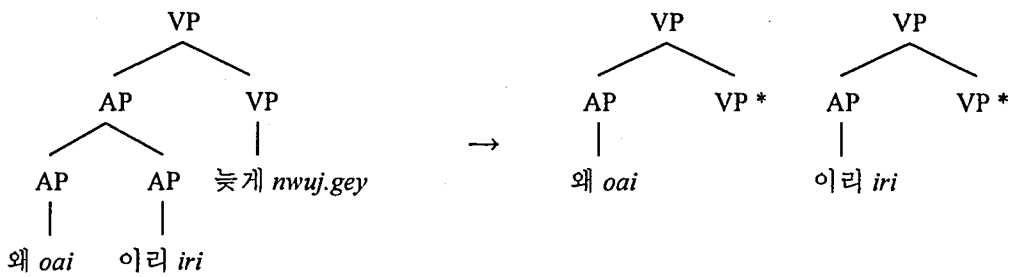


Figure 16. Exemple d'adverbe tiré du corpus SJTree et la grammaire extraite.

(11) 왜 이리 늦게

oai *iri* *nwuj.gey*

pourquoi autant tard.*gey*

'Pourquoi est-il autant en retard…'

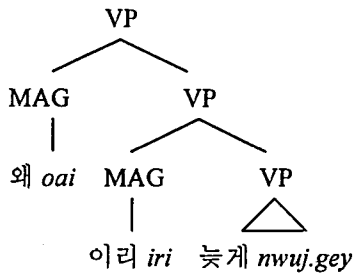


Figure 17. L'analyse proposée avec une grammaire extraite pour la phrase en (11)

Sinon il y a des adverbes comme 함께 *hamggey* ('ensemble') qui ont besoin d'un argument nominal NP_AJT. On pourrait extraire ainsi leurs arguments pour l'arbre auxiliaire (Voir la Figure 19 pour la phrase en (12)). Cette procédure d'extraction sera considérée seulement si nous extrayons une grammaire avec les arguments NP_AJT (voir la section 5.6)

(12) 김씨와 함께

gim.ssi.wa	*hamggey*
Kim.Monsieur.avec	ensemble
'avec Mr. Kim'	

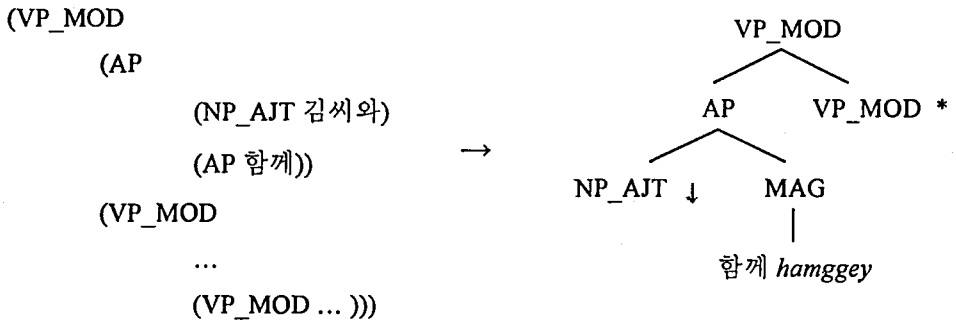


Figure 18. Exemple tiré du corpus SJTree et l'arbre extrait pour l'adverbe qui a besoin d'un argument nominal NP_AJT.

5.5 Le syntagme déterminant

Quand le système rencontre le syntagme déterminant prénominal, il génère un arbre auxiliaire gauche comme (NP (*han* ('un')/MM) (NP*)) pour le syntagme déterminant 한/MM + 달/NNG *han dal* ('un mois').

5.6 Le syntagme adjoint (_AJT)

Le syntagme adjoint (le syntagme nominal avec une postposition adverbiale) dans le corpus SJTree, qui est représenté en AP_AJT, DP_AJT, NP_AJT, S_AJT, VNP_AJT, VP_AJT et X_AJT pose le problème de l'étiquetage complément-adjoint, c'est-à-dire la distinction entre l'opération de substitution et d'adjonction. La distribution des syntagmes adjoints dans le corpus SJTree est présentée dans le Tableau ci-dessous :

AP_AJT	DP_AJT	NP_AJT	S_AJT	VNP_AJT	VP_AJT	X_AJT
6	2	5895	24	27	129	27

Tableau 7. Distribution de l'étiquette _AJT dans le corpus SJTree

Le traitement du syntagme étiqueté NP_AJT est le même que celui du syntagme prépositionnel en anglais ou en français. Comme nous avons déjà mentionné dans le Chapitre 1, Nasr (2004) a proposé une manière de distinguer entre l'opération de substitution et d'adjonction pour le français : la grammaire G_1 revient à considérer tous les groupes prépositionnels comme compléments, tandis que G_3 considère, quant à elle, tous les groupes prépositionnels comme adjoints. G_2 constitue une position intermédiaire entre G_1 et G_3. Elle repose sur une liste de 14 prépositions établie manuellement pour le français dont on considère qu'elles introduisent des compléments. Les 103 autres prépositions (principalement des locutions prépositionnelles) introduisent des ajouts.

Grammaire	Taille	Couverture	Ambiguïté moyenne	Temps d'analyse moyen
G_1	3 808	0,993	11 003	2,22 ms
G_2	5 910	0,988	750	1,39 ms
G_3	7 123	0,985	346	0,13 ms

Tableau 8. Taille des trois grammaires G_1, G_2 et G_3 ainsi que leur couverture, leur ambiguïté moyenne et le temps moyen d'analyse par phrase sur le corpus de test (Nasr 2004).

« Les résultats de Tableau 8 confirment et quantifient les prédictions théoriques. G_3 définit approximativement deux fois plus de catégories que G_1, et G_2 se situe entre les deux. De plus, la taille de la grammaire influence directement l'ambiguïté moyenne et le temps d'analyse. La comparaison de l'ambiguïté moyenne pour G_1 et pour G_3 permet en outre de quantifier la part d'ambiguïté provenant des rattachements prépositionnels. En effet, ces deux grammaires ne se distinguent que par leur traitement des rattachements prépositionnels, qui ne sont jamais ambigus pour G_3. La différence de l'ambiguïté moyenne pour G_1 et G_3 est très largement dûe aux rattachements prépositionnels. Parmi les 11 003 analyses construites en moyenne par phrase pour G_1, à peu près 10 600 proviennent de rattachements prépositionnels, soit une proportion de l'ordre de 95 %. Le Tableau 3 permet aussi d'observer que les trois heuristiques exercent une faible influence sur les couvertures des grammaires qu'elles définissent : le rapport des tailles des grammaires G_1 et G_3 vaut 0,53 alors que le rapport de leur couverture n'est que de 1,008. Un accroissement considérable du nombre de catégories ne diminue donc que très faiblement la couverture des grammaires » (Nasr 2004, pp.141-142).

Dans cette thèse, nous distinguons tous les nœuds d'adjonction pour le syntagme adjoint « _AJT » comme adjonction.

5.7 Le syntagme _MOD

Le résultat de l'extraction pour les nœuds étiquetés par _MOD est un arbre auxiliaire et il a plusieurs types de famille. Dans le corpus SJTree le syntagme représenté par une étiquette avec le suffixe _MOD a 5 types d'étiquettes _MOD, telles que NP_MOD, S_MOD, VNP_MOD, VP_MOD et X_MOD. Le Tableau 9 montre la distribution de l'étiquette _MOD.

NP_MOD	S_MOD	VNP_MOD	VP_MOD	X_MOD
1982	804	471	7127	15

Tableau 9. La distribution de l'étiquette _MOD.

5.7.1 NP_MOD

Dans le corpus SJTree, NP_MOD est composé par le nom et la postposition du génitif 의*ui*. Pour tous les syntagmes NP_MOD, la grammaire est séparément extraite comme les autres syntagmes nominaux, tel qu'un arbre initial pour le syntagme nominal et un arbre auxiliaire pour la postposition du génitif comme décrit dans la Figure 19 pour la phrase en (13).

(13) 대부분의 건설 현장이

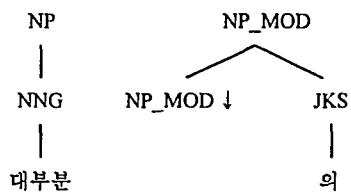
daebubun.ui *geonseol* *hyenjang.i*

plupart.Gen construction chantier.Nom

'la plupart des chantiers'

(NP_SBJ
(NP_MOD 대부분/NNG + 의/JKG)
(NP_SBJ
(NP 건설/NNG)
(NP_SBJ 현장/NNG + 이/JKS)))

a. Exemple tiré du corpus SJTree



b. Grammaires extraites

Figure 19. Exemple du syntagme NP_MOD tiré du corpus SJTree et les grammaires extraites.

5.7.2 S_MOD

S_MOD est une phrase qui modifie un syntagme nominal. Il forme un arbre auxiliaire extrait qui comporte S_MOD et un nœud NP. S_MOD. S_MOD est composé d'un verbe et de ses arguments en tant que phrase.

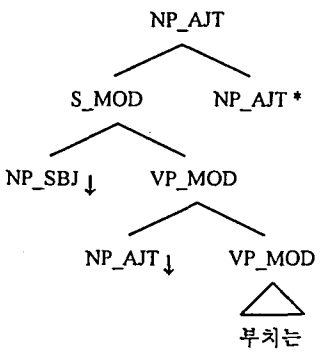
(14) 하고 있는 일이 힘에부친 듯

ha.go	*iss.neun*	*il.i*	*him.ey*	*buchin dues*
faire.go	être.neun	travail.Nom	être_au-delà.	semblant

'il semble que le travail qu'il fait est au-delà de ses capacités'

```
(NP_AJT
    (S_MOD
        (NP_SBJ
            (VP_MOD
                (VP 하/VV + 고/EC)
                (VP_MOD 있/VX + 는/ETM))
            (NP_SBJ 일/NNG + 이/JKS))
        (VP_MOD
            (NP_AJT 힘/NNG + 에/JKB)
            (VP_MOD 부치/VV + ㄴ/ETM)))
    (NP_AJT 듯/NNB)
```

a. Exemple tiré du corpus SJTree



b. Grammaire extraite

Figure 20. Exemple du syntagme S_MOD tiré du corpus SJTree et la grammaire extraite.

5.7.3 VNP_MOD

VNP_MOD représente la construction de la copule. Nous pouvons assimiler VNP à un syntagme verbal. Donc, VNP_MOD va être traité comme une sorte de VP_MOD qui est expliqué dans la section 3.7.4 VP_MOD.

(15) 궂은 일이라는 생각때문에

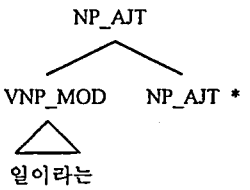
guj.eun *il.i.raneun* *saenggak.ddaemun.ey*

dur.*eun* travail.Cop.*raneun* pensée.car.*ey*

'à cause de la pensée que le travail est dur'

(NP_AJT
(VNP_MOD
(VP_MOD 궂/VA + 은/ETM)
(VNP_MOD 일/NNG + 이/VCP + 라는/ETM))
(NP_AJT 생각/NNG + 때문/NNB + 에/JKB))

a. Exemple tiré du corpus SJTree



b. Grammaire extraite

Figure 21. Exemple du syntagme VNP_MOD tiré du corpus SJTree et la grammaire extraite.

5.7.4 VP_MOD

Comme S_MOD, VP_MOD est un arbre auxiliaire qui contient une tête verbale, les arguments qui sont nécessaires, et le modifié. Certains VP_MOD correspondent aux participes en anglais ou français.

(16) 일할 사람이

il.ha.l *saram.i*

travail.faire.l homme.Nom

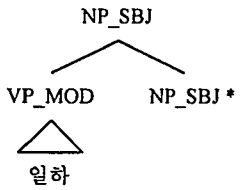
'l'homme qui va travailler'

(NP_SBJ

(VP_MOD 일/NNG + 하/XSV + ㄹ/ETM)

(NP_SBJ 사람/NNG + 이/JKS))

a. Exemple tiré du corpus SJTree



b. Grammaire extraite

Figure 22. Exemple du corpus pour le syntagme VP_MOD et la grammaire extraite.

Comme VP_MOD fonctionne comme un participe (un modifieur de NP), nous analysons le syntagme VP_MOD différemment du corpus SJTree. Park (2004b) propose une analyse syntaxique en découpant la phrase comme « *il.ha.l saram* » + « *i* » ('l'homme qui va travailler') en construisant un shallow parseur pour une analyse syntaxique. Cette analyse peut permettre d'expliquer comment VP_MOD modifie VNP_MOD en (15). La phrase en (15) dans la section de VNP_MOD, par exemple, VP_MOD « *guj.eun* ('dur') » modifie VNP_MOD « *il.i.raneun* ('être un travail'). » Du point vue linguistique, VNP_MOD ne peut pas être VP_MOD. Cependant d'après la proposition de Park (2004b), l'exemple en (15) a une structure « *guj.eun il* » + « *i.raneun.* » C'est-à-dire, ce n'est pas le nom qui s'attache à la copule, c'est le syntagme nominal de forme « VNP_MOD + nom » qui est ajouté à la copule, et forme VNP. Par conséquent, nous pouvons avoir une structure qui reprend l'idée de la grammaire scolaire selon laquelle « VP_MOD modifie un nom. » Selon cette thèse, cependant, nous extrayons les VP_MOD comme les autres syntagmes de _MOD donc comme des arbres auxiliaires pour faciliter le travail.

5.7.5 X_MOD

Dans le corpus SJTree, il y a deux cas pour X_MOD : (X_MOD 의/JKG), la postposition du génitif *ui* et (X_MOD 는/ETM), la terminaison verbale *neun*. Cette étiquette est utilisée pour marquer le syntagme cité dans la phrase, donc (X_MOD 의/JKG) est utilisée pour un syntagme nominal et (X_MOD 는/ETM) est utilisée pour une phrase citée. En général dans cette thèse, nous considérons que la postposition du génitif *ui* forme un arbre initial et que la

terminaison verbale n'est pas considérée pour l'extraction de la grammaire. Nous extrayons un arbre pour la terminaison verbale *neun* dans le syntagme X_MOD alors que en général la terminaison verbale n'est pas extraite comme une structure indépendante comme nous l'avons mentionné dans la section sur les verbes.

Le syntagme avec la postposition du génitif (X_MOD 의/JKG) produit un arbre initial comme les autres postpositions. Comme le corpus SJTree adopte la théorie du branchement binaire, l'annotation pour les marqueurs de citation « L » et « R » est représentée à des niveaux différents. Pour l'extraction des marqueurs de citation, nous modifions l'arbre extrait pour avoir une structure de pluri-furcation, c'est-à-dire que nous mettons ces marqueurs au même niveau, ce qui nous semble plus raisonnable. La Figure 23 et la Figure 24 montrent un exemple tiré du corpus SJTree pour X_MOD avec l'étiquette JKG et la grammaire supposée.

(17) '쥬라기 공원'의 캐릭터...

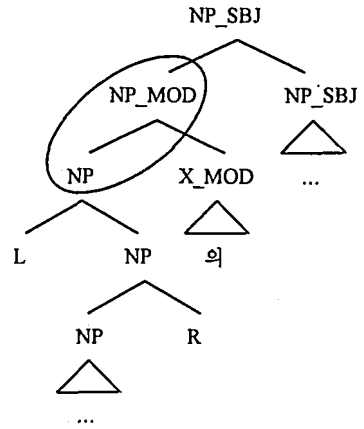
'jyuragi *gongwon'.ui* *kaerikteo...*

'Jurassic Park'.Gen personnage...

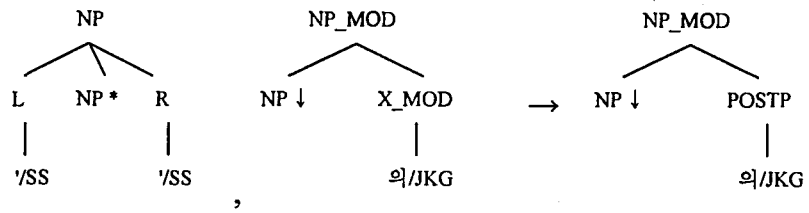
'... le personnage de « Jurassic Park »'

```
(NP_SBJ
    (NP_MOD
        (NP
            (L '/SS)
            (NP
                (NP
                    (NP 쥬라기/NNG)
                    (NP 공원/NNG))
                (R '/SS)))
        (X_MOD 의/JKG))
    (NP_SBJ
        (NP 캐릭터/NNG)
        (NP_SBJ 상품/NNG + 은/JX)))
```

Figure 23. Exemple tiré du corpus SJTree pour X_MOD avec l'étiquette JKG



a. Arbre syntaxique partiel pour la phrase en (17)



b. Arbres extraits supposés pour le syntagme *ui*/JKG de X_MOD

Figure 24. Grammaire extraite pour X_MOD à partir de la phrase en (17)

(X_MOD 는/ETM) est représenté par un arbre initial comme les postpositions. Cependant, puisque le syntagme a l'étiquette ETM, une terminaison verbale, nous changeons la structure du corpus SJTree car il ne fournit pas ce qu'il faut. Dans le corpus SJTree, X_MOD avec l'étiquette ETM modifie directement VNP dans la Figure 26. Toutefois, il s'agit d'une terminaison de la phrase modifieur, donc elle sera déplacée sous S_MOD (celui qui est modifié de S de la Figure 26a) (Voir la Figure 26b). Pour l'extraction, la structure extraite a une structure plate.

(18) '... 기다리었다'는 것이다.

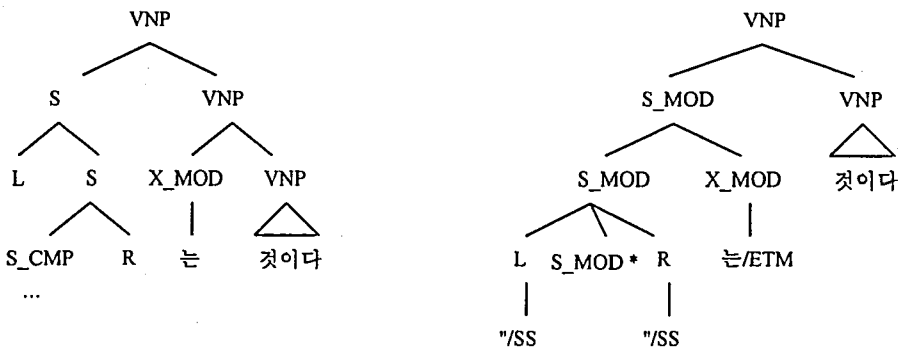
'... *gidari.eoss.da*'.*neun* *geos.i.da*

'... attendre.Pass.Ter'.Toc chose.Cop.Ter

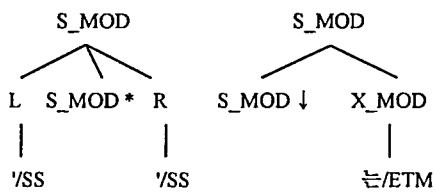
'C'est qu' « il attend »'

(VNP
(S
(L "/SS)
(S
(S_CMP)
...
(S_OBJ
(NP_SBJ 정책/NNG + 이/JKS)
(VP_OBJ 바뀌/VV + 기/ETN + 를/JKO))
(VP_CMP 기다리/VV + 었/EP + 다/EC))
(R "/SS)))
(VNP
(X_MOD + 는/ETM)
(VNP 것/NNB + 이/VCP + 다/EF + ./SF)))

Figure 25. Exemple tiré du corpus SJTree pour X_MOD avec l'étiquette ETM



a. Structure syntaxique donnée b. Structure syntaxique modifiée



c. Arbres extraits supposés pour le syntagme *neun*/ETM de X_MOD

Figure 26. Les arbres syntaxiques pour la Figure 29 et la grammaire supposée.

6. Conclusion

Enfin, nous fournissons un tableau avec chaque catégorie.

Catégorie	Nombre
Noms (NNG + NNP + NNB)	7867 = 6644 + 916 + 307
Prénoms (NP)	181
Numéral (NR)	97
Postpositions	299
Prédicats (VV + VA + VX + COP)	5423 = 3400 + 914 + 349 + 760
Déterminant (MM)	177
Adverbes (MAG + MAJ)	819 = 725 + 94
Exclamatifs (IC)	44
Symboles	632
Terminaisons verbales (syntagme X)	12
	15 551

Tableau 10. Nombre de chaque catégorie (G_4)

Chapitre 4. L'extraction de grammaires avec traits

Les grammaires extraites dans les autres travaux n'ont pas de traits. Nous proposons ici une méthode d'extraction de grammaires TAG avec traits.

1. Présentation générale

Dans Abeillé (2002, p.34), « on définit des traits, c'est-à-dire des couples attribut-valeur, les valeurs pouvant être des symboles atomiques ou de traits. Des ensembles de traits (conjoints) peuvent être associés à un mot ou à un syntagme. ... Ils peuvent se représenter comme des matrices ou comme des graphes orientés sans cycles (l'arc portant l'attribut et pointant vers sa valeur) ».

« L'unification entre deux structures de traits produit une structure résultante sauf si les deux structures portent des informations incompatibles (par exemple, si un attribut présent dans les deux structures y a une valeur différente) ; on dit alors que l'unification « échoue ». La structure résultante est la plus petite structure qui contient toute l'information contenue dans la première structure et toute l'information continue dans la deuxième structure » (Abeillé 2002, p.34).

Dans l'unification dans une grammaire TAG lexicalisée avec traits, « tous les nœuds d'un arbre élémentaire peuvent avoir une structure de traits (Vijay-Shanker 1987). Deux types d'information peuvent ainsi figurer dans les arbres élémentaires : soit la valeur (absolue) de tel trait à tel nœud (par exemple <mode> = indicatif au nœud P racine d'une arbre phrastique),

soit l'identité de valeur entre deux traits pour un même nœud ou pour nœuds différents (par exemple, l'égalité entre les traits d'accord du nœud correspondant au sujet et ceux du nœud V dans un arbre élémentaire de racine P) » (Abeillé 2002).

« Les équations ainsi exprimées sur les traits peuvent contraindre la flexion des éléments lexicaux présents dans l'arbre élémentaires (si elles sont associées à des préterminaux). Elles peuvent également contraindre la combinaison de cet arbre avec d'autres arbres élémentaires, si elles sont associées à des nœuds feuilles (à substitution) ou à des nœuds intérieurs ou racine susceptibles de recevoir des adjonctions. Elles jouent donc un rôle essential dans la bonne formation des phrases. La mise à jour des structures de traits a été définie pour tenir compte de cette opération particulière qu'est l'adjonction (Vijay-Shanker 1987). À chaque nœud d'un arbre élémentaire est associée au départ une structure de traits bipartite, divisée en une partie « amont » (en anglais « top ») et une partie « aval » (en anglais « bottom »). En « amont » sont les traits indiquant les relations du nœud avec les nœuds qui le dominent, en « aval » ses relations avec ceux qu'il domine. Les nœuds auxquels ne peut se produire aucune adjonction, par exemple les nœuds à substitution et les nœuds pieds des arbres auxiliaires, peuvent avoir une structure de traits unique. À la fin d'une dérivation, parties amont (top) et aval (bottom) doivent s'unifier (selon la définition ci-dessus) à chaque nœud » (Abeillé 2002).

À cause de l'opération d'adjonction, l'unification a donc lieu en deux temps dans une dérivation : tout d'abord, lors de la combinaison, entre les structures de traits des arbres élémentaires combinés, et puis après la combinaison, à l'intérieur de chacun des nœuds de l'arbre dérivé.

Lorsque deux arbres élémentaires sont combinés, leurs structures de traits sont modifiées de la façon suivante en cas d'adjonction :

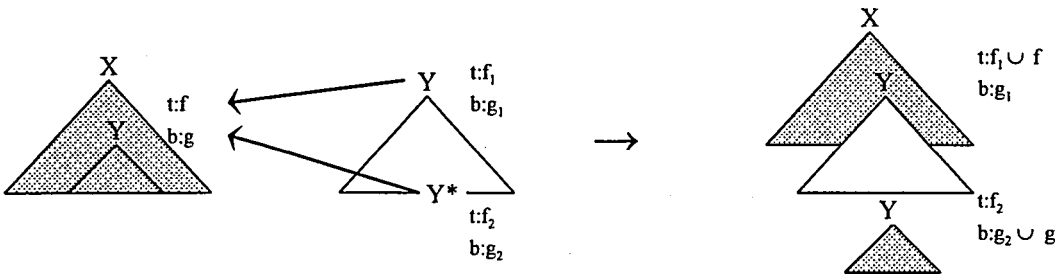
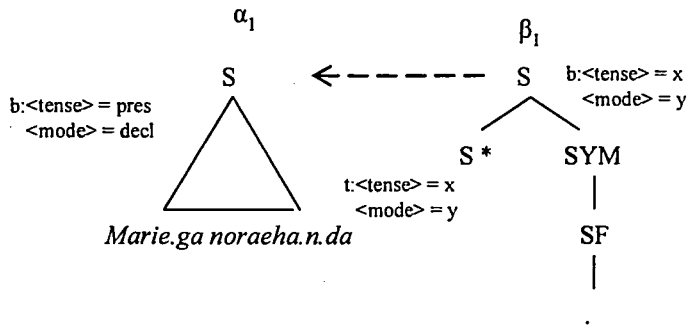


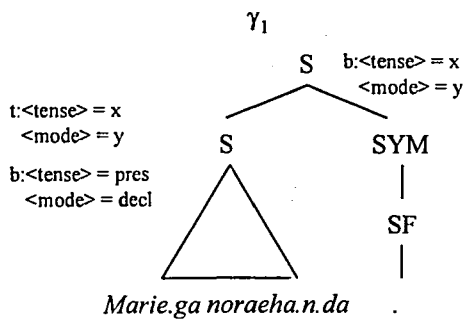
Figure 1. Adjonction dans FB-LTAG

Le nœud qui reçoit l'adjonction sera divisé et ses traits amont (f) s'unifient avec traits amont de la racine du nœud de l'arbre auxiliaire (f_1), tandis que les traits aval (g) s'unifient avec traits aval du nœud pied de l'arbre auxiliaire (g_2).

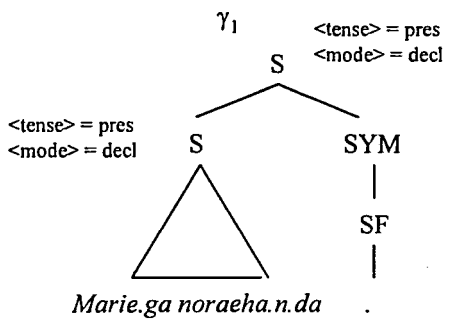
Prenons un exemple très simple de la mise à jour des traits <tense> et <mode> en cas d'adjonction d'un symbole dans des arbres élémentaires[14] :



a. Arbres élémentaires



b. Après adjunction



b. Après unification

Figure 2. Exemple d'adjonction avec unification

[14] Les lettres x, et éventuellement y et z dans les Figures indiquent des valeurs variables.

La structure des traits du nouveau nœud créée par la substitution est l'unification des traits des nœuds originaux. Les traits de la partie amont du nouveau nœud équivalent à l'unification des traits « amont » des deux nœuds originaux, tandis que les traits aval du nouveau nœud sont simplement les traits « aval » du nœud amont de l'arbre de substitution car le nœud de substitution n'a pas de trait aval.

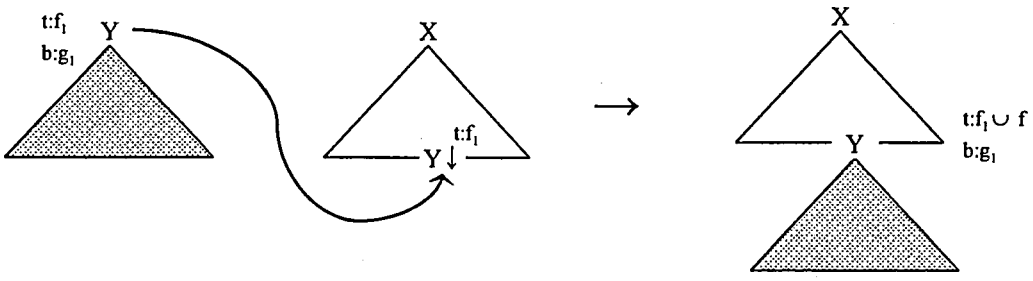
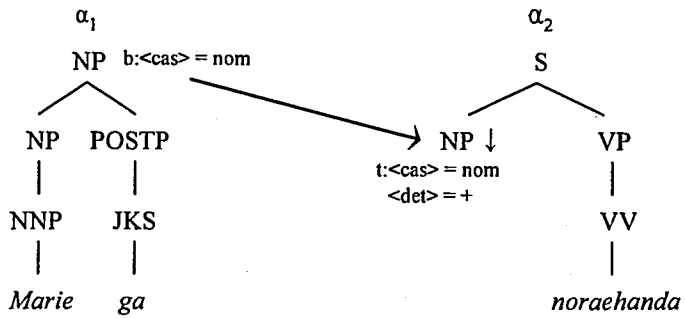
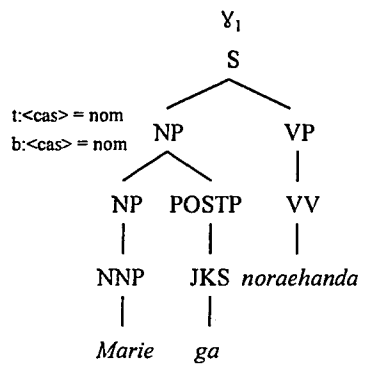


Figure 3. Substitution dans FB-LTAG

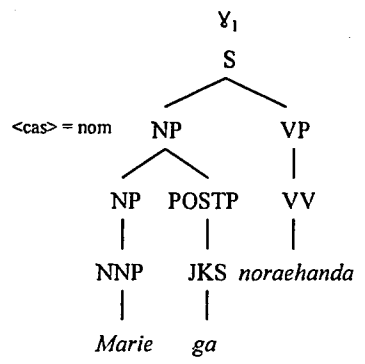
Prenons un exemple de la mise à jour des traits <cas> et <fonc> en cas de substitution d'un syntagme nominal dans un arbre phrastique :



a. Arbres élémentaires



b. Après substitution



b. Après unification

Figure 4. Exemple de substitution avec unification

(1) 마리가 노래한다.

Marie.ga *noraeha.n.da.*

Marie.Nom chanter.Pres.Ter

'Marie chante.'

L'unification contraint donc les combinaisons entre arbres élémentaires et permet d'associer des contraintes d'adjonction à chaque nœud (adjonction obligatoire, adjonction nulle, adjonction sélective). Les combinaisons entre arbres élémentaires (ou arbre dérivé) réglées par unification peuvent être (Abeillé 2002) :

- interdites : il n'y a pas d'unification possible des structures de traits des nœuds où la substitution ou l'adjonction a lieu,

- obligatoires : les traits amont et aval d'un nœud donné ne peuvent s'unifier : seule l'adjonction d'un arbre auxiliaire compatible permettra de résoudre ce conflit (en séparant amont et aval) et d'achever la dérivation,

- facultatives : les structures de traits des nœuds racine et pied d'un arbre auxiliaire peuvent s'unifier respectivement avec les parties amont et aval d'un nœud de même catégorie.

« Pour l'adjonction nulle, qui interdit toute adjonction à un nœud donné, il faudrait définir un trait tel qu'il ne puisse s'unifier avec aucune autre structure de la grammaire. Nous avons préféré, par souci de clarté, garder une contrainte explicite (notée NA ou ø). L'adjonction obligatoire peut, grâce aux équations sur les traits, être représentée comme une contrainte globale sur un arbre élémentaire, dont les nœuds partagent certains traits ; elle peut aussi apparaître dynamiquement à un ou plusieurs nœuds au cours d'une dérivation. Si l'on interdit les structures de traits cycliques (où un attribut peut avoir pour valeur une structure de traits où il figure lui-même), les TAG avec unification ont même capacité générative que les TAG avec contraintes d'adjonction (Vijay-Shanker 1987) » (Abeillé 2002).

2. Extraction des traits pour le coréen

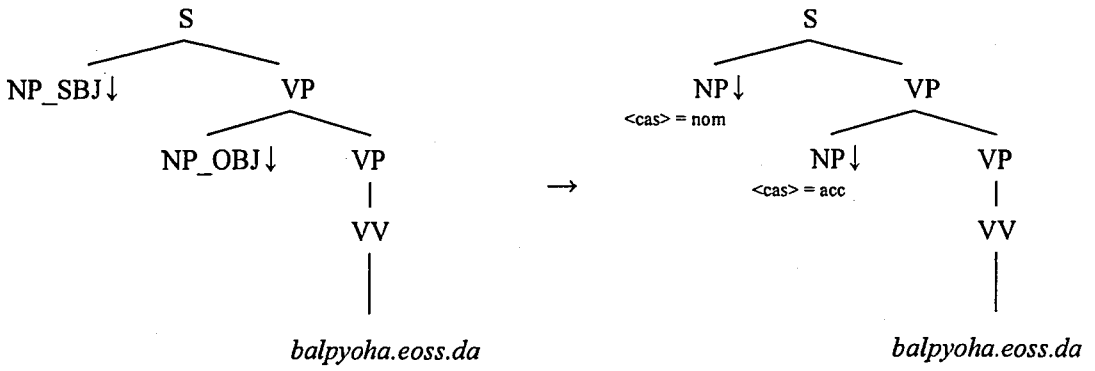
Les 55 étiquettes syntaxiques et les analyses morphologiques dans le corpus SJTree nous permettent d'extraire les traits syntaxique automatiquement et de développer une grammaire d'arbres adjoints lexicalisée avec des traits (FB-LTAG, en anglais Feature-based Lexicalized Tree Adjoining Grammar).

En fait, la procédure d'extraction de la grammaire avec traits se déroule en 2 phases : la première est la conversion d'étiquettes syntaxiques du corpus en catégories plus traits, et la deuxième est l'ajout d'équations plus générales sur ces mêmes traits en se basant sur la notion d'épine dorsale.

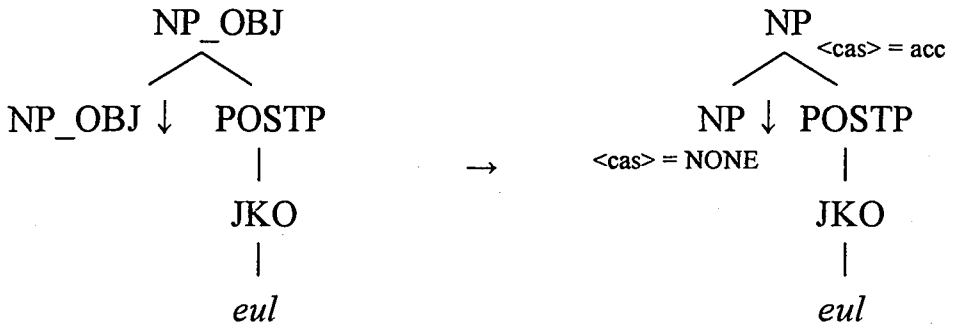
2.1 La conversion d'étiquettes syntaxiques

La grammaire FB-LTAG extraite emploie un ensemble d'étiquettes réduit parce qu'elle contient leur information syntaxique dans la structure des traits. Par exemple, une étiquette syntaxique NP_SBJ dans la grammaire LTAG est changée à NP et le trait syntaxique [<cas> = sujet] est ajouté. Par conséquent, nous utilisons 13 étiquettes syntaxiques pour la grammaire FB-LTAG. A partir des étiquettes syntaxiques qui finissent par _SBJ (le sujet), _OBJ (l'objet), et _CMP (l'attribut), nous pouvons extraire le trait <cas> qui décrit une structure des arguments dans la phrase. C'est-à-dire, le système extrait les traits <cas> à partir des arbres élémentaires d'une grammaire lexicalisée extraite pour les verbes et les adjectifs qui contiennent les étiquettes syntaxiques des arguments.

Tandis que les arbres auxiliaires pour les postpositions sont aussi extraits des traits <cas>, nous n'extrayons pas le trait <cas> avec l'argument qui correspond à l'étiquette syntaxique pour le nœud pied mais, le trait [<cas> = none] est extrait pour empêcher la multiple opération d'adjonction. La Figure 1 montre l'extraction des traits <cas> à partir d'arbre élémentaire du verbe et de la postposition. Les traits dans la Figure 5 ne sont pas complets, nous montrons les arbres avec tous les traits extractibles dans la Figure ultérieure.



(a) Extraction des traits <cas> pour le verbe *balpyoha.eoss.da* ('avoir annoncé')



(a) Extraction des traits <cas> pour la postposition accusative *eul*

Figure 5. Extraction des traits <cas>

A côté du trait <cas>, nous pouvons aussi extraire les traits <mode> et <temps> à partir du corpus SJTree. Puisque les analyses morphologiques pour les terminaisons verbales et adjectivales dans le corpus SJTree sont simplement divisées en étiquettes morphologiques EP, EF et EC qui signifient, respectivement, une terminaison non-finale, une terminaison finale et une terminaison conjonctive, les traits <mode> et <temps> ne sont pas extraits directement à partir du corpus SJTree. Dans cet article, nous préanalysons 7 terminaisons non-finales (EP) et 77 terminaisons finales (EF) utilisées dans le corpus SJTree pour extraire les traits <mode> et <temps> automatiquement. En général, l'étiquette morphologique EF porte la flexion du mode pour le trait <mode> et EP porte la flexion du temps pour le trait <temps>. Lorsque le système rencontre ces terminaisons non-finales ou finales pendant la procédure d'extraction, il réfère aux listes de EP et de EF pour extraire les traits <mode> et <temps>.

En général, les terminaisons conjonctives (EC) ne concernent pas l'extraction des traits <mode> et <temps>, nous extrayons seulement le trait <ec> qui contient la valeur de chaîne[15]. Quand les prédicats portent EC précédé par EP comme « *olreu* ('monter')/VV + *ass* ('PASS')/EP + *go*/EC, » le système aussi extrait le trait <temps >, mais on se trouve cet exemple très rarement dans le corpus SJTree. Les traits <ef> et <ep> sont aussi extraits et contiennent la valeur de chaîne. Quelques terminaisons non-finales comme *si* sont extraites par le trait [<hor> = +] qui a un sens honorifique. Dans la grammaire FB-LTAG extraite, nous représentons la tête du lexique sous la forme infinitive et les traits contiennent la forme fléchie.

Un arbre initial pour le verbe *balpyoha.eoss.da* ('avoir annoncé') de la Figure 5a est changé comme la Figure 6.

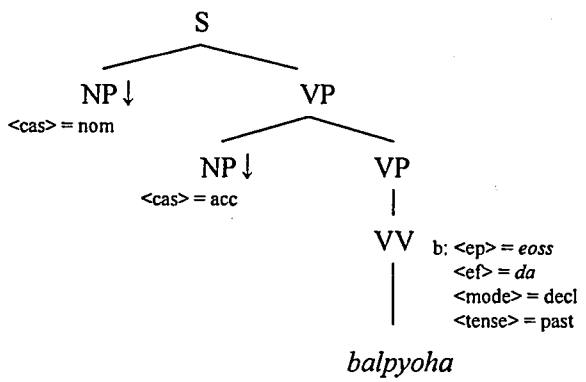


Figure 6. Un exemple d'une grammaire FB-LTAG pour *balpyoha.eoss.da* ('avoir annoncé')

Le trait <det> est aussi automatiquement extractible dans le corpus SJTree et il est extrait à partir de l'étiquette syntaxique et de l'analyse morphologique, à la différence des autres traits

[15] Le coréen ne porte pas la terminaison non-finale pour le temps dans la phrase conjointe comme *meok.go* ('manger et …') dans la phrase (1').

(1') 마리는 … 먹고, … 마시었다.
Maire.neun … meok.go, … masi.eoss.da
Marie.Nom … manger.go, … boire.Past.Ter
'Marie a mangé … et bu …'

extraits[16]. Par exemple, tandis que le trait [<det> = -] est extrait à partir des noms dépendants qui ont toujours besoin d'un modifieur (extrait par l'analyse morphologique), le trait [<det> = +] est extrait à partir des modifieurs comme les syntagmes _MOD (extrait par l'étiquette syntaxique). A partir de l'étiquette syntaxique DP qui contient les MMs (déterminants ou démonstratifs), le trait [<det> = +] est aussi extrait. Le Figure 7 montre l'extraction des traits <det>. Nous remarquons que le nœud pied NP dans la Figure 7b n'a pas de trait [<det> = -] comme les articles définis en français parce que un NP ne doit pas être nécessairement précédé par un article en coréen.

(2) 이철희 씨는 ...

icheolheui/NNP	*ssi*/NNB + *neun*/JX
Cheolheui.Lee	monsieur.Top

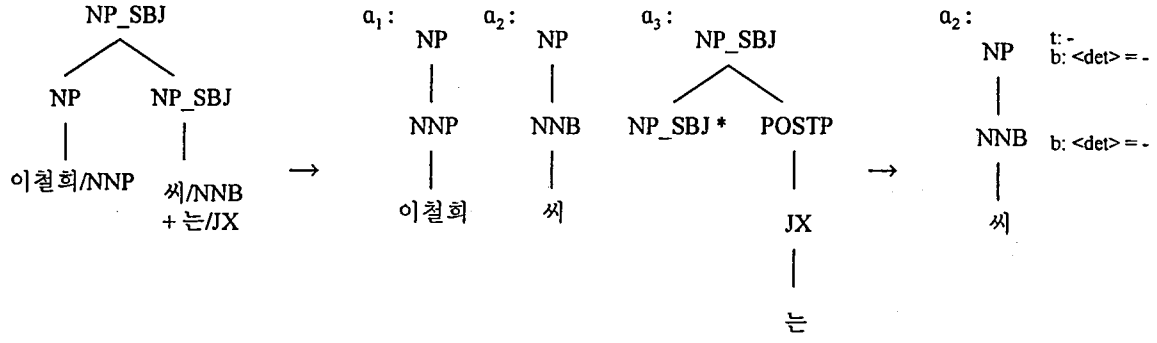
'Monsieur Cheolheui Lee ...'

(3) 일하는 사람이 ...

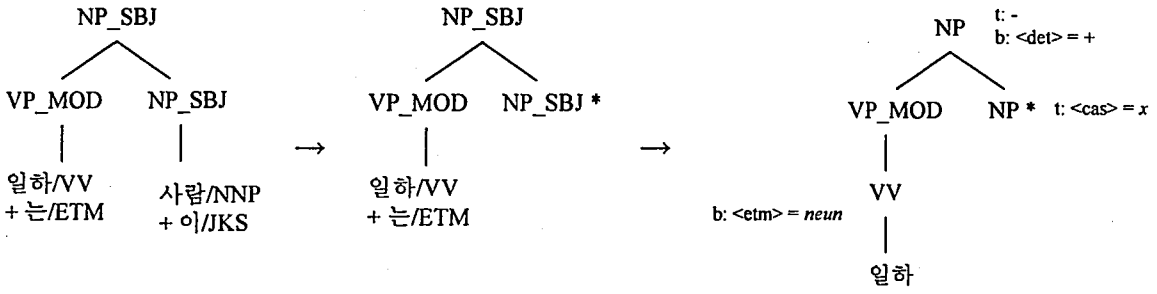
ilha/VV + neun/ETM	*saram*/NNG + *i*/JKS
travailler.neun	homme

'l'homme qui travaille ...'

[16] Le trait <det> ne signifie pas nécessairement un déterminant. Il a un sens plus général comprnant déterminant, modifieur, etc.



(a) Extraction du trait [<det> = -] à partir des noms dépendants



(b) Extraction du trait [<det> = -] à partir des syntagmes _MOD

Figure 7. Extraction du trait <det>

Le verbe *balpyoha.eoss.da* ('avoir annoncé') prend un syntagme nominal pour sujet et pour objet, et impose donc le trait (amont) [<det> = +] aux nœuds de sujet et d'objet. On peut substituer un syntagme nominal comme un nom propre comme *icheolheui* au nœud [NP <cas> = nom], puisque le syntagme nominal avec un nom propre est marqué [b : <det> = +]. En revanche, si on substitue seulement un nom dépendant sans modifieur comme *ssi.neun* au nœud [NP <cas> = nom], il en résulte un conflit au nœud [NP <cas> = nom] dans l'arbre dérivé. Par conséquent, dans l'arbre initial pour le verbe *balpyoha.eoss.da* ('avoir annoncé'), on ajoute le trait [<det> = +] aux nœuds de substitution comme le montre la Figure 8.

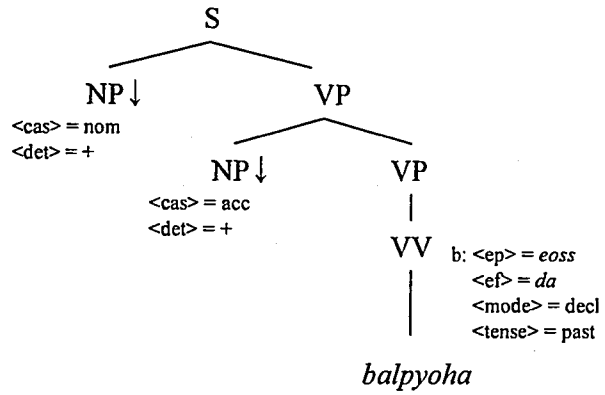


Figure 8. Arbre initial pour *balpyoha.eoss.da* ('avoir annoncé') dans FB-LTAG avec trait <det>

Le Tableau 1 montre les exemples de conversion d'étiquettes syntaxiques du corpus en catégories plus traits. Un tableau complet de conversion d'étiquettes syntaxiques est fourni dans un Annexe.

Ancre	Type d'arbre	Etiquettes syntaxiques	Type de nœud	Exemple de conversion
verbe	α	NP_SBJ	subst	NP [<cas> = nom <det> = +]
verbe	α	NP_OBJ	subst	NP [<cas> = acc <det> = +]
verbe	α	NP_CMP	subst	NP [<cas> = attr <det> = +]
verbe	α|β	VP, VP_MOD	-	VP [<ep> <ef> <mode> <tense>]
postposition	α	NP_SBJ	racine	NP [<cas> = nom]
postposition	α	NP_SBJ	subst	NP [<cas> = NONE]
postposition	α	NP_OBJ	racine	NP [<cas> = acc]
postposition	α	NP_OBJ	subst	NP [<cas> = NONE]
postposition	α	NP_CMP	racine	NP [<cas> = attr]
postposition	α	NP_CMP	subst	NP [<cas> = NONE]

Tableau 1. La conversion d'étiquettes syntaxiques

Le coréen n'a pas besoin d'un trait <personne> comme en anglais et des traits <genre> ou <nombre> comme en français. Han *et al.* (2000) propose plusieurs traits de la grammaire FB-LTAG pour le coréen qui nous n'utilisons pas dans cet article comme <adv-pp>, <top> et <aux-pp> pour les noms et <clause-type> pour les prédicats. Tandis que les postpositions sont séparées à partir de l'*eojeol* pendant la procédure d'extraction d'une grammaire lexicalisée, Han *et al.* considère la combinaison des noms et postpositions comme un *eojeol* (voir la section 3). La séparation des postpositions à partir de l'*eojeol* est plus efficace pour un développement de la grammaire. Dans Han *et al.* (2000), le trait <adv-pp> contient simplement la valeur de chaîne des postpositions adverbiales. Le trait <aux-pp> ajoute un sens sémantique des postpositions auxiliaires comme *seulement*, *aussi*, etc. que nous ne pouvons pas extraire automatiquement à partir du corpus SJTree ou des autres corpus arborée pour le coréen car le corpus annoté syntaxiquement ne contient pas telle information sémantique. Le trait <top> marque la présence ou l'absence de marqueur topique en coréen comme *neun*, cependant le marqueur topique dans le corpus SJTree est annoté comme le sujet, c'est-à-dire que le seul trait [<cas> = sujet] est extrait à partir du marqueur topique. Le trait <clause-type> indique le type de phrases qui a des valeurs comme main ('phrase principale'), coord ('phrase coordinative'), subordi ('phrase subordonnée'), adnom ('phrase adnominale'), nominal ('phrase nominale', et aux-connect ('phrase auxiliaire'). Puisque la distinction du type de phrases est très vague, sauf la phrase principale, nous n'adoptons pas ce trait. Au lieu de cela, le trait <ef> est extrait si la phrase est principale et le trait <ec> est extrait pour les autres types de la phrase.

Le Tableau 2 suivant montre tous les traits que nous avons extraits à partir du corpus SJTree.

Trait	Description	Valeurs
<cas>	un trait du cas est assigné par le prédicat	nom(inatif), acc(usatif), et attr(ibut)
<det>	un trait avec un sens très général qui contient déterminant, modifieur, etc.	+, -
<mode>	un trait pour la mode ; il est extrait quand le trait <ef> existe (c'est-à-dire, la phrase est principale).	ind(icatif), imp(ératif), int(érrogatif), excl(amatif).
<temps>	un trait temporel	pres(ent), pass(é), et fut(ur)
<ep>, <ef>, <ec>	des traits pour la terminaison non-finale, finale, et conjonctive	valeur de chaîne comme *eoss*, *da*, *go* etc.
<hor>	un trait honorifique	+, -

Tableau 2. La liste des traits extractibles à partir du corpus SJTree.

2.2 L'ajout d'équations

La deuxième phase d'extraction des traits est l'ajout d'équations plus générales sur ces mêmes traits en se basant sur la notion d'épine dorsale. Par exemple, on ajoute les mêmes traits de VV aval sur les nœuds S aval, VP amont/aval et VV amont dans la Figure 6 parce que les nœuds d'épine dorsale partagent un certain nombre de traits.

Un arbre initial pour le verbe *balpyoha.eoss.da* ('avoir annoncé') de la Figure 13 est enfin, changé comme la Figure 9.

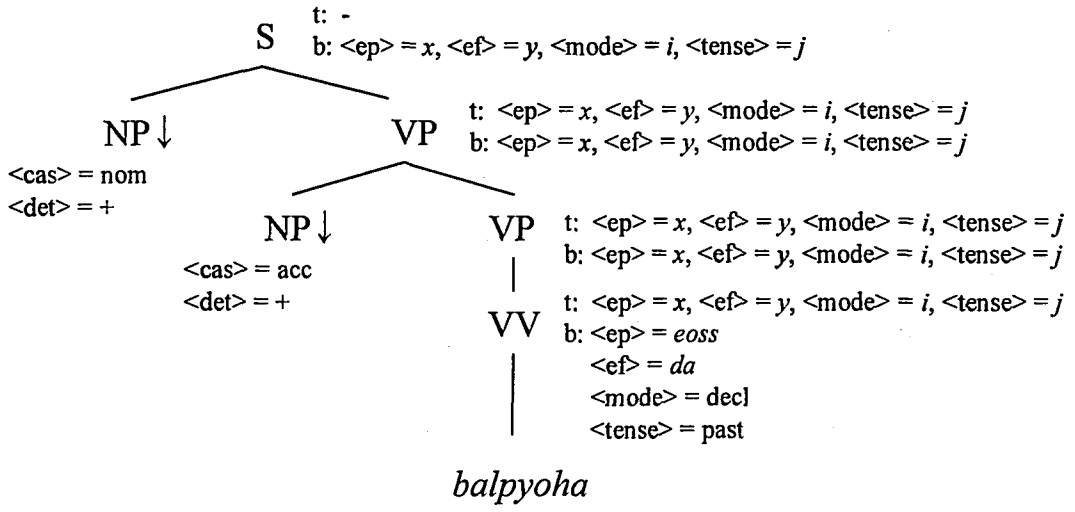


Figure 9. Arbre initial pour *balpyoha.eoss.da* ('avoir annoncé') dans FB-LTAG après l'ajout d'équations sur les mêmes traits en se basant sur la notion d'épine dorsale

Il y a deux types d'épines dorsales pour les arbres auxiliaires dans Abeillé (2002) : les arbres auxiliaires prédicatifs et les arbres auxiliaires modifieurs. Cependant, dans cette thèse nous n'extrayons que les arbres auxiliaires modifieurs pour les adverbes, les syntagmes _MOD, et les signes de ponctuation. Dans les arbres auxiliaires pour les adverbes et les signes de ponctuation, on ajoute systématiquement les traits [<ep> = *x*, <ep> = *y*, <mode> = *i*, <tense> = *j*] sur le nœud racine aval et sur le nœud pied amont si les nœuds sont VP ou S (Voir β_1 et β_2 dans la Figure 10). Dans les arbres auxiliaires pour les syntagmes _MOD, on ajoute le trait [<det> = +] sur le nœuds racine aval et le trait [<det> = *x*] sur le nœud pied amont (Voir la Figure 7b).

3. Expérience d'extraction

Nous extrayons une grammaire FB-LTAG qui emploie un ensemble d'étiquettes réduites parce qu'elle contient leur information syntaxique dans la structure des traits. La grammaire extraite G_5 enlève les étiquettes syntaxiques, utilise finalement un ensemble d'étiquettes réduit, ajoute la structure des traits, et introduit une forme infinitive pour son ancre lexicale.

- G_5: Nous employons un ensemble d'étiquettes réduit et son ancre lexique sous la forme infinitive avec une structure des traits.

Le Tableau 2 montre le résultat de procédure d'extraction au-dessous. La Figure 10 montre la grammaire extraite G_5 à partir de la phrase en (4).

(4) 일본 외무성은즉각해명성명을발표했다.

ilbon *oibwuseon.eun* *jeukgak* *haemyeng*

Japon ministre_des_affaires_étrangères.Nom immédiatement élucidation

seongmyeng.eul *balpyo.ha.eoss.da*

déclaration.Acc annoncer.Pass.Ter

'Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation.'

	nombre de ltrees (α + β)	nombre de schémas d'arbres (α + β)	fréquences moyennes de ltrees	fréquences moyennes de schémas
G_5	12 429 (6 981 + 5 448)	385 (166 + 219)	3,21	103,65

Tableau 2. Résultat d'expérimentation pour FB-LTAG

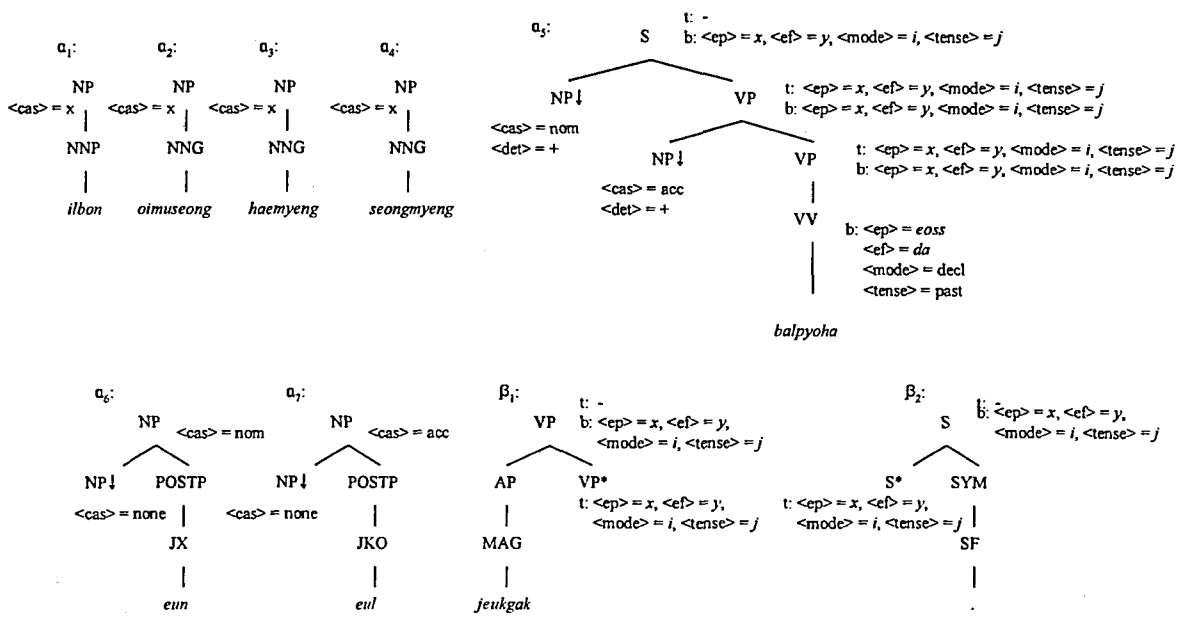


Figure 10. FB-LTAG extraite G_5 à partir de la phrase (1).

La Figure 11 montre l'analyse avec la grammaire extraite G_5 pour la phrase en (4).

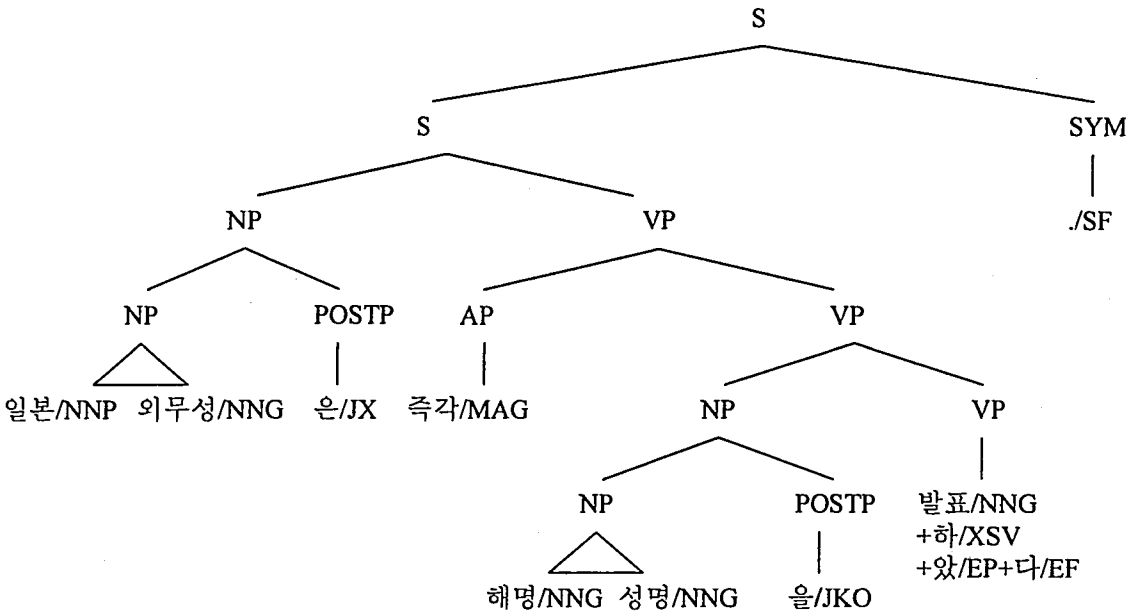


Figure 11. Analyse avec la grammaire G_5

Le nombre de schémas d'arbres de G_4 est beaucoup plus inférieur par rapport à celui des grammaires $G_1 \sim G_3$ car la grammaire extraite G_4 emploie un ensemble d'étiquettes réduit, donc 13 étiquettes syntaxiques qui contiennent leur information syntaxique dans la structure des traits.

Enfin, nous fournissons un schéma qui résume la procédure d'extraction des grammaires $G_1 \sim G_5$. Dans la figure, il y deux extracteurs de grammaire (celui de la grammaire lexicalisée et celui de la grammaire lexicalisée avec traits), trois convertisseurs de grammaire qui convertissent les grammaires inférieures en grammaires supérieures, un délexicaliseur qui délexicalise pour construire les schémas d'arbres. Les deux extracteurs, le convertisseur 1, et le délexicaliseur sont indépendants du corpus et de la langue, mais les convertisseurs 2 et 3 sont dépendants du coréen.

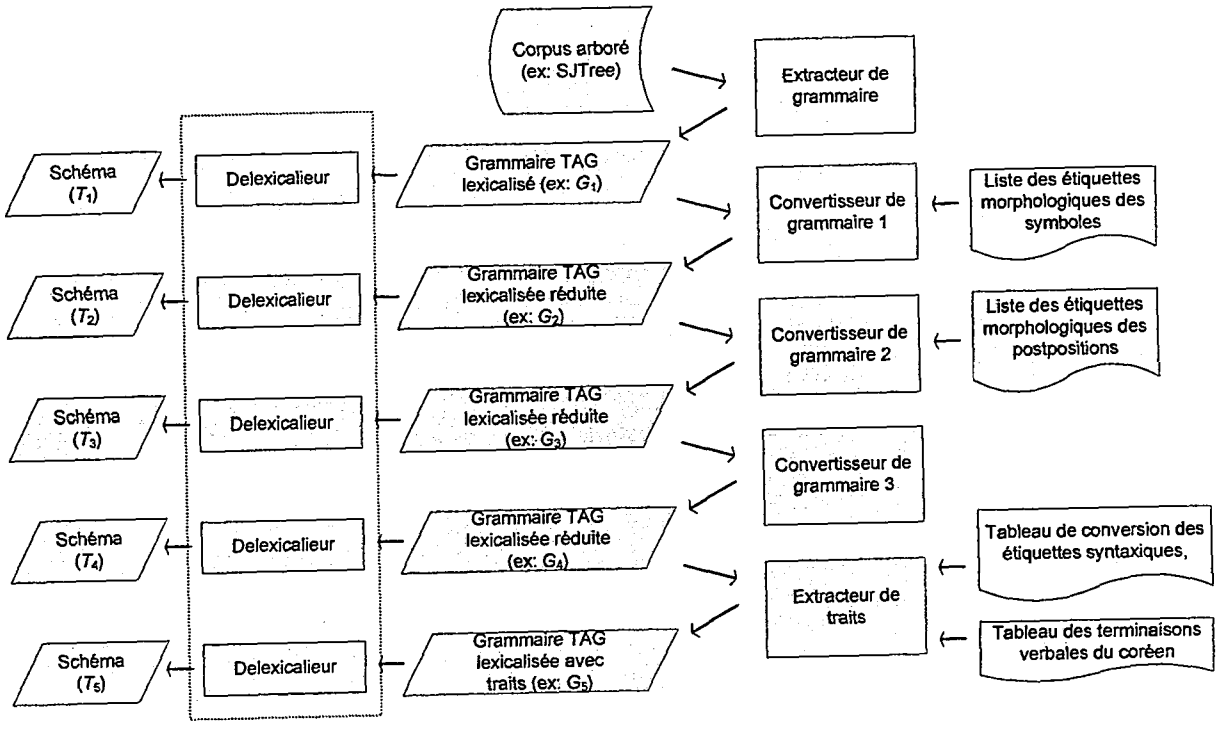


Figure 12. Schéma d'extraction des grammaires

Chapitre 5. Evaluations

Les méthodes d'évaluation dans ce chapitre sont empruntées à la thèse d'habilitation de Nasr (2004). Les grammaires extraites sont évaluées par leur taille, leur couverture et leur ambiguïté moyenne. La taille des grammaires est le nombre d'arbres lexicalisés et le nombre de catégories (les schémas d'arbre). La couverture des grammaires, étant donné un corpus, est le rapport du nombre d'occurrences de catégories inconnues dans le corpus à la taille du corpus (le nombre total d'occurrences de mots dans le corpus). L'ambiguïté moyenne des grammaires est la moyenne du nombre d'analyses différentes que la grammaire associe aux phrases du corpus.

1. Evaluation par la taille de la grammaire extraite

Les courbes de la Figure 1 montrent la croissance du nombre d'arbres lexicalisés. Les courbes de la Figure 2 montrent que la croissance du nombre de schémas d'arbre n'est pas stabilisée à l'issue de l'extraction, ce qui semble indiquer que la taille du corpus n'est pas suffisante pour atteindre la convergence des grammaires.

« Cette conclusion doit néanmoins être modérée, en observant que le nombre de schémas apparaissant au moins deux fois dans le corpus est quasiment stabilisé à l'issue de l'extraction. Seules les courbes retraçant l'évolution des *hapax* continuent de croître à l'issue du processus d'extraction. Malgré son influence limitée sur la couverture de la grammaire, l'évolution de la courbe des *hapax* est un phénomène troublant. Il semble s'expliquer, en partie, par des erreurs dans le corpus d'apprentissage. [...] Les autres *hapax* modélisent des phénomènes rares, susceptibles de se répéter dans un corpus plus important. » (Nasr 2004, p142)

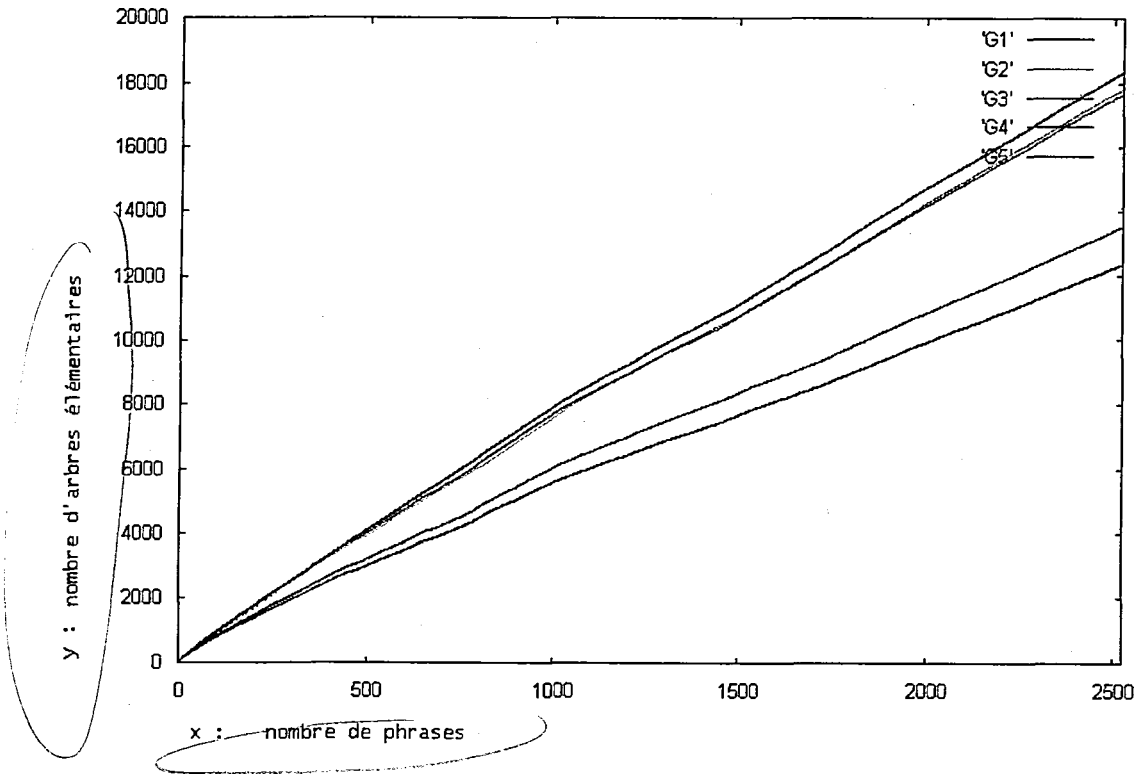
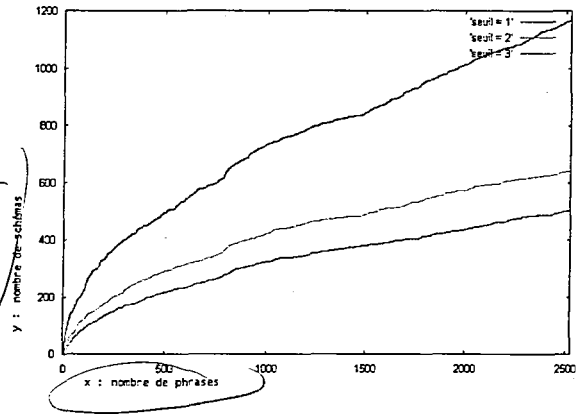
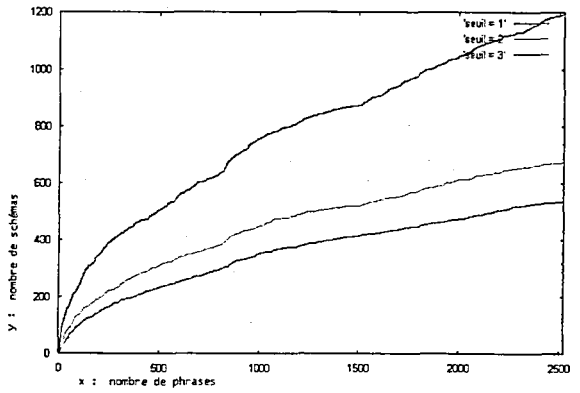


Figure 1. Croissance de la taille des grammaires lexicalisées (axe x : nombre d'arbres élémentaires, axe y : nombre de phrases).

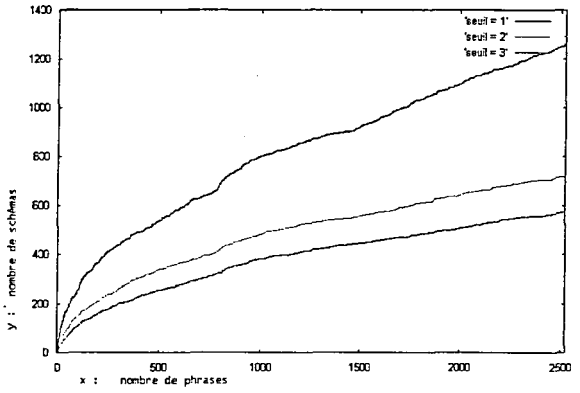
Dans la Figure 2, le nombre de schémas apparaissant au moins deux fois dans le corpus est quasiment stabilisé à l'issue de l'extraction.



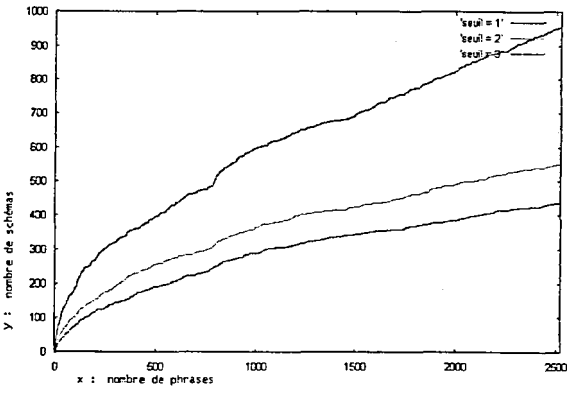
a. G_1



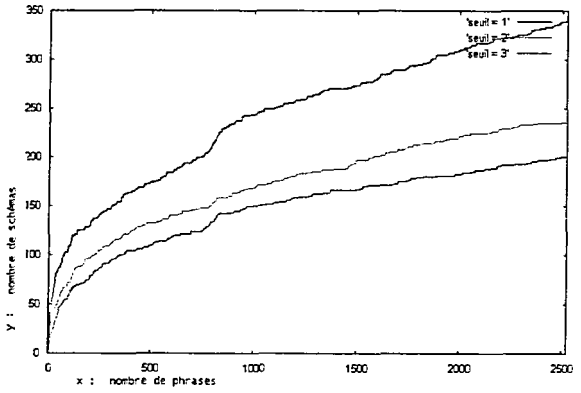
b. G_2



c. G_3



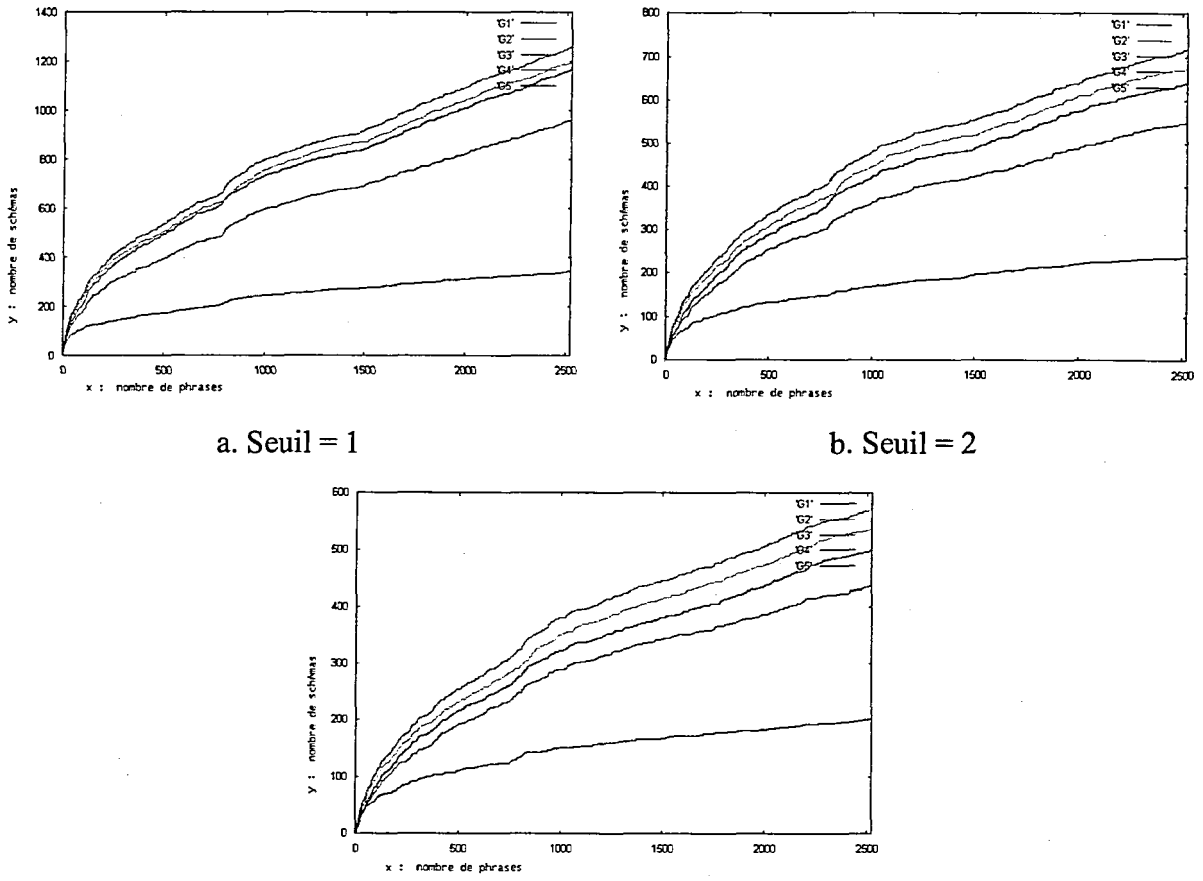
d. G_4



e. G_5

Figure 2. Croissance de la taille des schémas d'arbres des grammaires extraites (axe x : nombre de schémas, axe y : nombre de phrases).

Pour la comparaison de la croissance de la taille des schémas d'arbres entre les grammaires, nous fournissons aussi la Figure 3 qui représente la taille des schémas d'arbres observés au moins une fois (seuil = 1), au moins deux fois (seuil = 2) et au moins trois fois (seuil = 3). La Figure 3 indique une possibilité de la stabilisation des grammaires supérieures.



a. Seuil = 1

b. Seuil = 2

c. Seuil = 3

Figure 3. Comparaison de la croissance de la taille des schémas d'arbres observés au moins une fois (seuil = 1), au moins deux fois (seuil = 2) et au moins trois fois (seuil = 3) entre les grammaires extraites
(axe x : nombre de schémas, axe y : nombre de phrases).

2. Evaluation par la couverture de la grammaire extraite

Pour l'évaluation par la couverture, le corpus est divisé en deux parties : un corpus d'apprentissage et un corpus de test. Le corpus d'apprentissage sert à l'extraction des grammaires, et le corpus de test est utilisé pour évaluer la couverture des grammaires. Nous effectuons cette évaluation sur 60 % (1 511 phrases, voir le Tableau 1) et 90 % (2 265 phrases,

voir le Tableau 2) du corpus complet, comme le corpus d'apprentissage, pour les schémas d'arbres observés au moins une fois (seuil = 1), au moins deux fois (seuil = 2) et au moins trois fois (seuil = 3).

	seuil = 1	seuil = 2	seuil = 3
G_1	0,6073	0,6055	0,6036
G_2	0,5925	0,5911	0,5896
G_3	0,6075	0,6063	0,6051
G_4	0,6075	0,6066	0,6060
G_5	0,6075	0,6070	0,6066

Tableau 1. Couverture des grammaires extraites avec un corpus d'apprentissage par la fréquence : 60 % du corpus complet, donc 1 511 phrases.

	seuil = 1	seuil = 2	seuil = 3
G_1	0,9113	0,9112	0,9105
G_2	0,8889	0,8888	0,8882
G_3	0,9114	0,9113	0,9108
G_4	0,9114	0,9114	0,9111
G_5	0,9114	0,9114	0,9111

Tableau 2. Couverture lexicale des grammaires extraites avec un corpus d'apprentissage par la fréquence : 90 % du corpus complet, donc 2 265 phrases.

Les Tableaux au-dessus sont calculés par la fréquence des schémas d'arbres. Supposons qu'il y a un corpus complet qui contient les catégories *a* avec la fréquence 7, *b* avec 2 et *c* avec 1 et un corpus d'apprentissage qui contient les catégories *a* avec la fréquence 7 et *b* avec 2. La couverture par la fréquence de ce corpus d'apprentissage est 90 % alors qu'elle est de 66.67 % pour celle par le nombre de schémas d'arbres. Pour éviter ce genre de problème, nous calculons la couverture par la fréquence. Nous fournissons aussi les tableaux qui sont calculés

par le nombre de schémas d'arbres dont résulte une couverture meilleure par rapport à la couverture par la fréquence pour nos grammaires extraites.

	seuil = 1	seuil = 2	seuil = 3
G_1	0,7245	0,7622	0,7657
G_2	0,7119	0,7532	0,7517
G_3	0,7332	0,7739	0,7715
G_4	0,7255	0,7698	0,7816
G_5	0,8	0,8069	0,8103

Tableau 3. Couverture des grammaires extraites avec un corpus d'apprentissage par le nombre de schémas: 60 % du corpus complet, donc 1 511 phrases.

	seuil = 1	seuil = 2	seuil = 3
G_1	0,9326	0,9591	0,9556
G_2	0,9329	0,9551	0,9539
G_3	0,9388	0,9561	0,9483
G_4	0,9326	0,9525	0,9488
G_5	0,9579	0,9638	0,9430

Tableau 4. Couverture lexicale des grammaires extraites avec un corpus par le nombre de schémas : 90 % du corpus complet, donc 2 265 phrases.

3. Evaluation par l'ambiguïté moyenne de la grammaire extraite

Pour cette évaluation, nous utilisons le parseur LLP2 de Loria[17] est basé sur l'algorithme de Lopez (2000). Le parseur LLP2 emploie TAGML qui est un standard de description de ressources nécessaires à un analyseur LTAG en XML[18]. Par exemple, la grammaire extraite

[17] Pour le détail de LLP2, voir http://www.loria.fr/~azim/LLP2

[18] Pour le détail de TAGML, voir http://www.loria.fr/~azim/LLP2/help/fr/tagml2/index.html

G_1 pour la phrase (1) est transformée comme la Figure 4 en TAGML. L'entrée et la sortie de LLP2 pour la phrase (1) sont fournies dans l'Annexe.

(1) 일본 외부성은 즉각 해명 성명을 발표했다.

ilbon *oimuseong.eun* *jeukgak* *haemyeng*

Japon ministre_des_affaires_étrangères.Nom immédiatement élucidation

seongmyeng.eul *balpyoha.eoss.da*

déclaration.Acc annoncer.Pass.Ter

'Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation.'

```
<tagml>

<!-- LES ARBRES TAG        -->
<!-- ===================== -->

<treeLib>
  <tree id="np_sbj">
      <node cat="NP_SBJ" name="_head" type="anchor"/>
  </tree>
  <tree id="np_obj">
      <node cat="NP_OBJ" name="_head" type="anchor"/>
  </tree>
  <tree id="np_sbj_np_obj_v">
    <node cat="S">
      <node cat="NP_SBJ" name="_n0" type="subst"/>
      <node cat="VP">
        <node cat="NP_OBJ" name="_n1" type="subst"/>
        <node cat="VP" name="_head" type="anchor"/></node>
    </node>
  </tree>
  <tree id="np_sbj_b">
    <node cat="NP_SBJ">
      <node cat="NP" name="_head" type="anchor"/>
      <node cat="NP_SBJ" type="foot"/>
    </node>
  </tree>
  <tree id="np_obj_b">
    <node cat="NP_OBJ">
      <node cat="NP" name="_head" type="anchor"/>
      <node cat="NP_OBJ" type="foot"/></node>
```

```
  </tree>
  <tree id="ap">
    <node cat="VP">
      <node cat="AP" name="_head" type="anchor"/>
      <node cat="VP" type="foot"/>
    </node>
  </tree>
  </treeLib>

<!-- LES FORMES FLECHIES -->
<!-- ================== -->

<morphLib>
   <morph lex="ilbon">
       <lemmaref cat="NP" name="ilbon/NNP"/>
    </morph><!-- ____________________________________ -->
    <morph lex="oimuseong_eun">
       <lemmaref cat="NP_SBJ" name="oimuseong/NNG+eun/JX"/>
    </morph><!-- ____________________________________ -->
    <morph lex="haemyeng">
       <lemmaref cat="NP" name="haemyeng/NNG"/>
    </morph><!-- ____________________________________ -->
    <morph lex="seongmyeng_eul">
       <lemmaref cat="NP_OBJ" name="seongmyeng/NNG+eul/JKO"/>
    </morph><!-- ____________________________________ -->
    <morph lex="jeukgak">
       <lemmaref cat="AP" name="jeukgak/MAG"/>
    </morph><!-- ____________________________________ -->
    <morph lex="balpyo_ha_eoss_da_.">
       <lemmaref cat="VP" name="balpyo/NNG+ha/XSV+eoss/EP+da/EF+./SF"/>
    </morph><!-- ____________________________________ -->
</morphLib>

<!-- LES LEXICALISATIONS -->
<!-- ================== -->
  <lexicalization>
    <tree copyof="np_sbj"/>
    <anchor noderef="_head">
      <lemmaref cat="NP_SBJ" name="oimuseong/NNG+eun/JX"/></anchor>
  </lexicalization><!-- ____________________________________ -->
  <lexicalization>
    <tree copyof="np_obj"/>
    <anchor noderef="_head">
      <lemmaref cat="NP_OBJ" name="seongmyeng/NNG+eul/JKO"/></anchor>
  </lexicalization><!-- ____________________________________ -->
```

```
  <lexicalization> <!-- balpy_ha -->
    <tree copyof="np_sbj_np_obj_v"/>
    <anchor noderef="_head">
      <lemmaref cat="VP" name="balpyo/NNG+ha/XSV+eoss/EP+da/EF+./SF"/></anchor>
  </lexicalization><!-- ____________________________________ -->
  <lexicalization>
    <tree copyof="ap"/>
    <anchor noderef="_head">
      <lemmaref cat="AP" name="jeukgak/MAG"/></anchor>
  </lexicalization><!-- ____________________________________ -->
  <lexicalization>
    <tree copyof="np_sbj_b"/>
    <anchor noderef="_head">
      <lemmaref cat="NP" name="ilbon/NNP"/></anchor>
  </lexicalization><!-- ____________________________________ -->
  <lexicalization>
    <tree copyof="np_obj_b"/>
    <anchor noderef="_head">
      <lemmaref cat="NP" name="haemyeng/NNG"/></anchor>
  </lexicalization><!-- ____________________________________ -->

</tagml>
```

Figure 4. Exemple de TAGML pour la phrase (1).

Nous utilisons seulement les grammaires extraites G_1 pour l'évaluation par l'ambiguïté moyenne car les grammaire G_1, G_2, G_4 et G_5 requièrent aussi un pre-processus 'shallow parsing' avant l'emploi de ces grammaires dans le parseur LLP2 et une interface entre shallow parseur et LLP2 est demandée. Pour cette raison, nous laissons l'emploi des grammaires G_2, G_3, G_4 et G_5 dans le parseur LLP2 comme un travail ultérieur.

Cependant LLP2 a un problème d'allocation de mémoire et il ne peut pas gérer les phrases qui contiennent plus de 8 *eojeol*s pendant l'évaluation[19]. Donc, nous n'évaluons que 648 phrases qui contiennent moins de 8 *eojeol*s dans le corpus SJTree. Le Tableau 5 montre l'ambiguïté moyenne et le temps d'analyse moyen. La Figure 5 montre aussi l'ambiguïté selon la longueur des phrases fournis par le parseur LLP2.

[19] LLP2 ne peut pas non plus gérer certaines phrases qui contiennent moins de 8 *eojeol*s.

	Ambiguïté moyenne	Temps d'analyse moyen
G_1	122,67	17,45 ms

Tableau 5. L'ambiguïté moyenne et le temps d'analyse moyen par le parseur LLP2

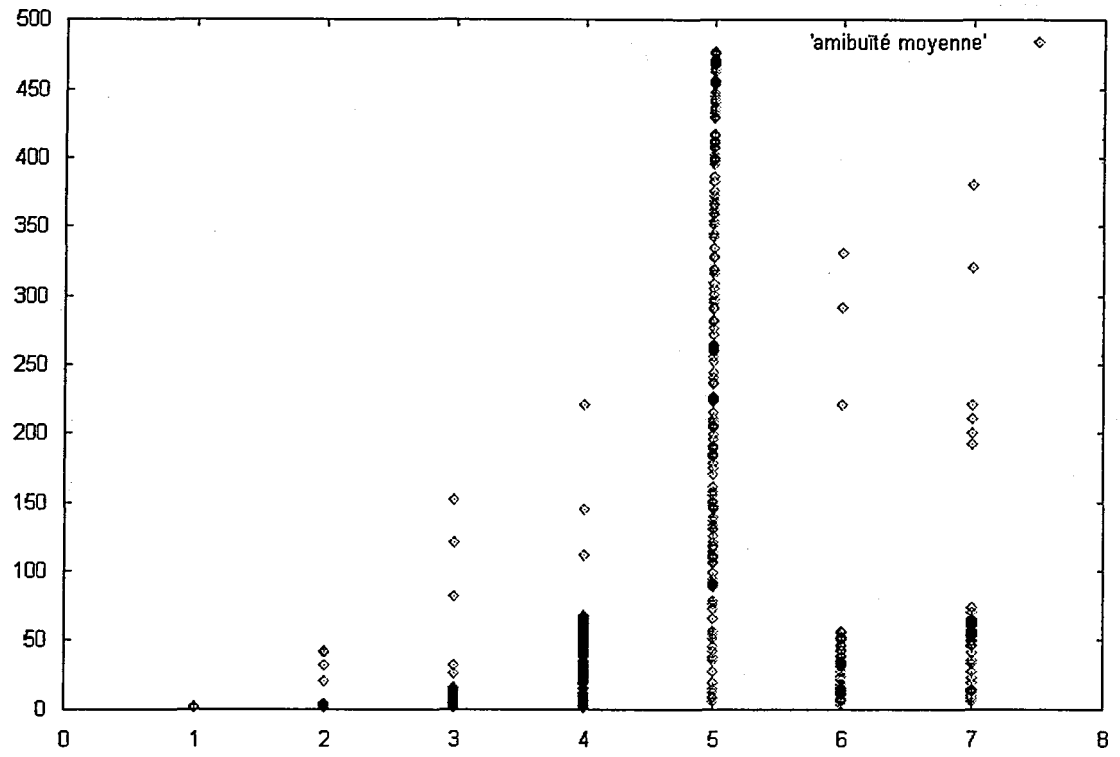


Figure 5. L'ambiguïté selon la longueur des phrases
(axe x : longueur de phrases (*eojeol*), axe y : nombre d'arbres analysés).

4. Extraction sur le Penn Korean Treebank

Le système d'extraction dans cette thèse est aussi susceptible d'être appliqué sur l'autre corpus arboré, par exemple le Penn Korean Treebank (Han et al. 2001). Nous extrayons les grammaires lexicalisées à partir de PKT (Penn Korean Treebank) avec quelques changements du corpus car la structure de SJTree et celle de PKT ne sont pas les mêmes. Dans le chapitre 2, nous avons déjà montré la différence entre SJTree et PKT au niveau de l'annotation syntaxique. Rappelons la structure de SJTree pour la phrase (1) (voir la Figure 6).

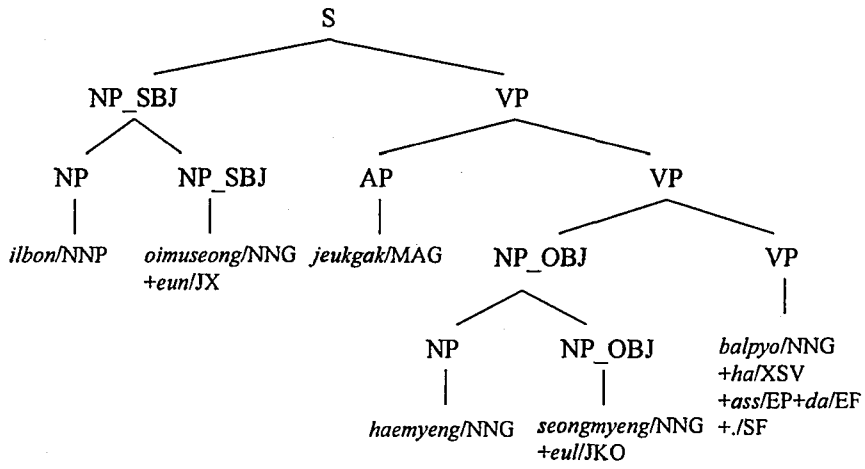


Figure 6. Structure de SJTree

Dans le corpus SJTree, chaque nœud contient un *eojeol*. Cependant, la structure de PKT est différente du traitement du syntagme nominal. Le corpus PKT considère le syntagme nominal comme un nœud même dans le cas où le syntagme contient plusieurs *eojeol*s. En plus, le symbole est séparé de l'*eojeol* dans PKT. La Figure 7 montre une structure de PKT pour la phrase (1).

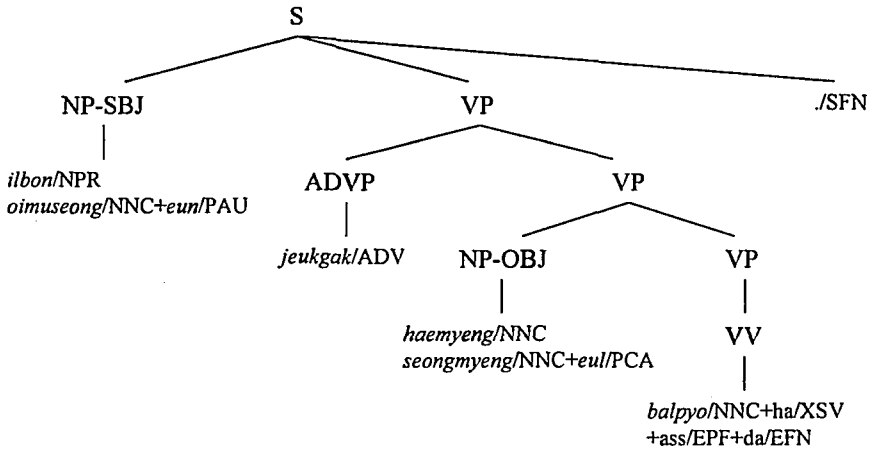


Figure 7. Structure de PKT

Puisque le symbole est déjà séparé de l'*eojeol*, nous extrayons seulement la grammaire G_2 à partir de la structure de PKT dans la Figure 7 : 5 arbres lexicalisés, donc 3 arbres initiaux et 2 arbres auxiliaires par rapport à 7 arbres lexicalisés à partir de SJTree dans la Figure 6, donc 3 arbres initiaux et 4 arbres auxiliaires pour la phrase (1).

Nous effectuons des expériences d'extraction sur une partie de PKT, 2 042 phrases, donc 19 234 *eojeol*s et extrayons 14 552 arbres lexicalisés, donc 5 093 arbres initiaux et 9 459 arbres auxiliaires avec 844 schémas d'arbres, donc 275 arbres initiaux et 569 arbres auxiliaires. Le Tableau 6 montre le résultat de l'extraction sur Penn Korean Treebank. La couverture des grammaires extraites par le nombre des schémas d'arbres est évaluée avec 90 % du corpus complet comme un corpus d'apprentissage.

Nombre d'arbres ($\alpha + \beta$)	Nombre de schéma ($\alpha + \beta$)	Couverture (seuil=1)	Couverture (seuil=2)	Couverture (seuil=3)
14 552 (5093 + 9459)	844 (275+569)	0,9609	0,9643	0,9519

Tableau 6. Résultat des expériences d'extraction sur le Penn Korean Treebank

Le résultat d'extraction sur le Penn Korean Treebank est augmenté par rapport au résultat de Xia et al. (2000) car la taille du corpus a été augmentée. Enfin, la couverture de la grammaire du PKT est légèrement plus grande que celle du corpus SJTree.

Nous pouvons aussi considérer que laquelle grammaire extraite est la plus grosse entre les grammaires extraites du SJTree et du PKT, si la grammaire extraite du PKT est différente des grammaires extraites du SJTree, et si l'on peut espérer les fusionner.

La grammaire extraite du SJTree est plus grande par la taille absolue (17 979 versus 14 552) et relative (1,78 versus 1,32 pour le nombre d'arbres élémentaires par l'*eojeol*). La grammaire extraite du PKT est différente des grammaires extraites du SJTree car, par exemple, un syntagme peut contenir plusieurs *eojeol*s dans le PKT, c'est-à-dire qu'un arbre élémentaire peut contenir plusieurs *eojeol*s. Dans les grammaires extraites du SJTree, un arbre élémentaire contient toujours un *eojeol*. La fusion entre les grammaires extraites du SJTree et du PKT est assez difficile car beaucoup de modifications de la grammaire du PKT sont requises. Par exemple, une adaptation d'étiquettes syntaxiques et morphologiques du SJTree est déjà pénible parce que les étiquettes du SJTree sont plus spécifiées que celles du PKT.

Chapitre 6. Sous-catégorisation des prédicats en coréen

1. Présentation générale

Dans ce chapitre nous construisons une sous-catégorisation des prédicats en coréen. Particulièrement, la sous-catégorisation que nous montrons ici, inclut le nombre de la grammaire lexicalisée de chaque sous-catégorisation, la fréquence et les variations syntaxiques car cette sous-catégorisation est basée sur les grammaires extraites dans les chapitres précédents. Dans le chapitre 3 concernant « la procédure d'extraction, » nous élargissons les grammaires extraites en utilisant les gabarits appelés « les schémas d'arbre. » L'ancre lexique dans les grammaires extraites est enlevée et remplacée par un marqueur d'ancre pour former une grammaire de gabarit. Par exemple, un marqueur d'ancre @NNG où l'ancre lexique des grammaires extraites est un nom commun, qui remplace l'ancre lexique. La construction de la sous-catégorisation pour les prédicats est basée sur les schémas d'arbre T_5.

En général, la sous-catégorisation est aussi considérée comme une entrée du développement semi-automatique d'une grammaire, par exemple, la métagrammaire (Candito 1999 et Crabbé 2005)[20] ou la description d'arbres (Xia 2001)[21].

[20] La métagrammaire est conçue pour générer les schèmes d'une grammaire TAG, en tant qu'unités respectant les principes de minimalité sémantique et de cooccurrence prédicat-arguments (Candito 1999). Le compilateur de la métagrammaire fonctionne en prenant en entrée une métagrammaire pour une langue donnée et fournit en sortie les schèmes de la grammaire TAG générée. Pour générer

Han *et al.* (2000) propose une sous-catégorisation pour les prédicats en coréen. C'est une sous-catégorisation initiale construite manuellement sans ancre lexical, ni fréquence ni les variations syntaxiques. Dans ce chapitre, nous comparaisons aussi les sous-catégorisations initiales de Han *et el.* (2000) avec les nôtres.

2. Sous-catégorisations initiales

2.1 Construction des sous-catégorisations à partir des schémas d'arbres

Parmi 391 schèmes d'arbre extraits automatiquement à partir du corpus SJTree, nous avons 311 schèmes d'arbre qui ont les ancres lexicales des verbes, adjectifs et copules. La liste suivante montre une sous-catégorisation avec le nombre de la grammaire lexicalisée et la fréquence. Cette liste est faite manuellement d'après les schémas d'arbre extraits automatiquement.

---

l'ensemble des classes croisées (un schème final pouvant être ancré), Candito (1999) dispose de la sous-catégorisation initiale (dimension 1), les redistributions des fonctions (dimension 2) et les réalisations grammaticales des fonctions (dimension 3). L'ensemble des classes croisées est généré en épuisant toutes les possibilités de classe terminale de la dimension 1, puis 2, puis pour chaque fonction grammaticale apparaissant dans la sous-catégorisation finale, toutes possibilités de réalisation sont utilisées. Parmi les classes croisées ainsi définies ne sont retenues que celles qui vérifient les contraintes de compatibilité.

[21] Après Vijay-Shanker et Schabes (1992), la description d'arbre est utilisée comme une représentation de la structure partagée. Les entrées du système (qui s'appelle LexOrg) sont un ensemble de la sous-catégorisation, un ensemble fini des règles lexicales, et un ensemble fini de description. LexOrg a trois composants : Frame Generator, Description Selector, et Tree Generator. Pour chaque sous-catégorisation initiale, Frame Generator y applique l'ensemble de règles lexicales et construit un ensemble de sous-catégorisations qui sont liées à la sous-catégorisation initiale. Pour chaque sous-catégorisation produite par Frame Generator, Description Selector choisit un sous-ensemble de description. Pour chaque sous-ensemble de description, Tree Generator combine la description dans le sous-ensemble pour produire un ensemble de gabarits. Par conséquent, les sorties du LexOrg sont un ensemble de gabarits pour la sous-catégorisation initiale et toutes les sous-catégorisations qui sont dérivées de la sous-catégorisation initiale par les règles lexicales appliquées (Xia 2001).

sous-catégorisation	nombre d'arbres lexicalisés
(S(NP_SBJ↓)(VP @VA))	70
(S(NP_SBJ↓)(VP @VV))	154
(S(NP_SBJ↓)(VP(NP_OBJ↓)(VP @VV)))	103
(S(NP_SBJ↓)(VP(NP_CMP↓)(VP @VV)))	17
(S(NP_SBJ↓)(VNP @COP))	190
(S(NP_SBJ↓)(S(NP_SBJ↓)(VP @VA)))	16
(S(NP_SBJ↓)(S(NP_SBJ↓)(VP @VV)))	14
(S(NP_SBJ↓)(S(NP_SBJ↓)(VP(NP_OBJ↓)(VP @VV))))	6
(S(NP_SBJ↓)(VP(VP_OBJ↓)(VP(NP_OBJ↓)(VP @VV))))	1
(S(S_SBJ↓)(VNP @COP))	5
(S(S_SBJ↓)(VP @VV))	1
(S(NP_SBJ↓)(VP(S_OBJ↓)(VP @VV)))	1
(S(NP_SBJ↓)(VP(S_CMP*)(VP @VV)))	56
(S(NP_SBJ↓)(VP(S_CMP*)(VP(NP_OBJ↓)(VP @VV))))	2

Tableau 1. Liste de sous-catégorisation

Enfin, nous explorons les schémas d'arbres élémentaires qui partagent la même sous-catégorisation. Elles consistent en les arbres élémentaires qui sont ancrés par les verbes et les adjectifs. Dans les sous-sections suivantes nous fournissons une description brève de chaque famille d'arbre.

2.1.1 (S(NP_SBJ↓)(VP @VA)) : nx0A

Ce schéma d'arbres est sélectionné par les adjectifs qui ont besoin de NP sujet comme *keu.da* ('être grand'), ou *geom.da* ('être noir'), etc. Elle est montrée dans la Figure 1.

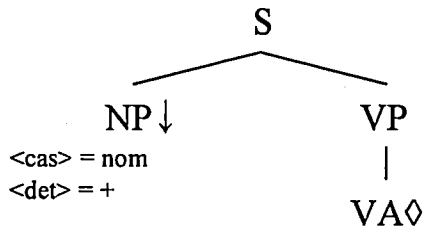


Figure 1. Schéma d'arbre nx0A

2.1.2 (S(NP_SBJ↓)(VP @VV)) : nx0V

Ce schéma d'arbres est sélectionné par les verbes qui n'ont pas besoin d'un complément objet comme *geod.da* ('marcher'), *balsaengha.da* ('se produire'), etc. Elle est montrée dans la Figure 2.

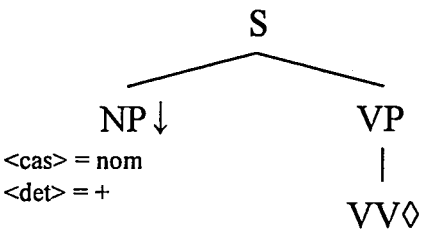


Figure 2. Schéma d'arbre nx0V

2.1.3 (S(NP_SBJ↓)(VP(NP_OBJ↓)(VP @VV))) : nx0nx1V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin d'un NP sujet et d'un NP objet comme *yujiha.da* ('maintenir') ou *jisiha.da* ('ordre'). Elle est montrée dans la Figure 3.

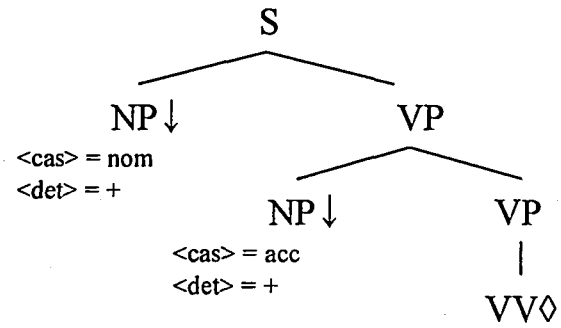


Figure 3. Schéma d'arbre nx0V

2.1.4 (S(NP_SBJ↓)(VP(NP_CMP↓)(VP @VV))) : nx0nx2V

Ce schéma d'arbres est sélectionné seulement par les verbes *doi.da* ('devenir') et *ani.da* ('n'être pas') qui ont besoin d'un attribut. Elle est montrée dans la Figure 4. L'attribut dans la syntaxe coréenne est très particulier car en coréen seul le syntagme nominal qui se trouve devant ces deux verbes est considéré comme attribut. Puisque le corpus SJTree étiquette explicitement le syntagme de l'attribut comme NP_CMP, nous avons cette sous-catégorisation[22].

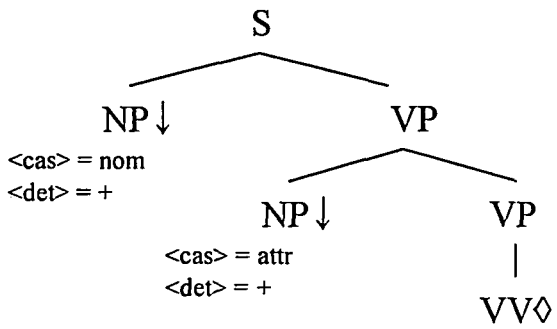


Figure 4. Schéma d'arbre nx0nx2V

[22] Han *et al.* (2000) considère cette phrase comme une phrase avec deux sujets.

2.1.5 (S(NP_SBJ↓)(VNP @COP)) : nx0COP

Ce schéma d'arbres est sélectionné par la copule *i.da* avec un NP sujet. Elle est montrée dans la Figure 5.

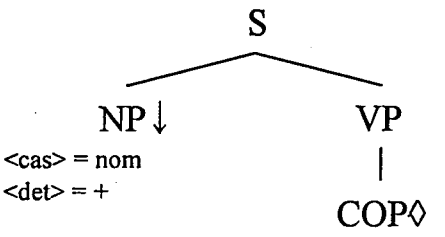


Figure 5. Schéma d'arbre nx0COP

2.1.6 (S(NP_SBJ↓)(S(NP_SBJ↓)(VP @VA))) : nx0nx0A

Ce schéma d'arbres est sélectionné par les adjectifs qui ont besoin de deux NP sujets comme *pilyoha.da* ('être nécessaire'), *joh.da* ('être bon'), etc. Elle est montrée dans la Figure 6.

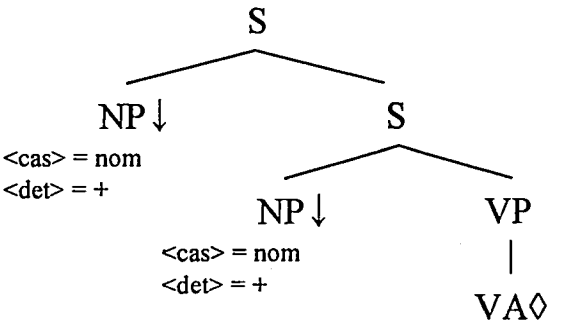


Figure 6. Schéma d'arbre nx0nx0A

2.1.7 (S(NP_SBJ↓)(S(NP_SBJ↓)(VP @VV))) : nx0nx0V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin de deux NP sujets comme *eob.da* ('n'exsiter pas'), *iss.da* ('exister'), etc. Elle est montrée dans la Figure 7.

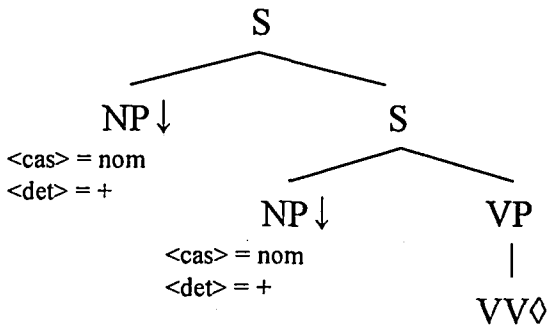


Figure 7. Schéma d'arbre nx0nx0V

2.1.8 (S(NP_SBJ↓)(S(NP_SBJ↓)(VP(NP_OBJ↓)(VP @VV)))) : nx0nx0nx1V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin de deux NP sujets et d'un NP objet comme *mandeul.da* ('faire'), *irueoji.da* ('accomplir'), etc. Elle est montrée dans la Figure 8.

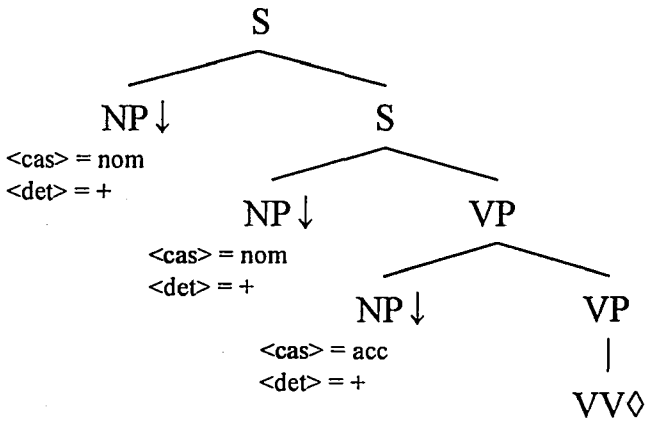


Figure 8. Schéma d'arbre nx0nx0nx1V

2.1.9 (S(NP_SBJ↓)(VP(NP_OBJ↓)(VP(NP_OBJ↓)(VP @VV)))) : nx0nx1nx1V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin d'un NP sujets et de deux NP objets comme *ha.da* ('faire'), *danha.da* ('être trompé'), etc. Elle est montrée dans la Figure 9.

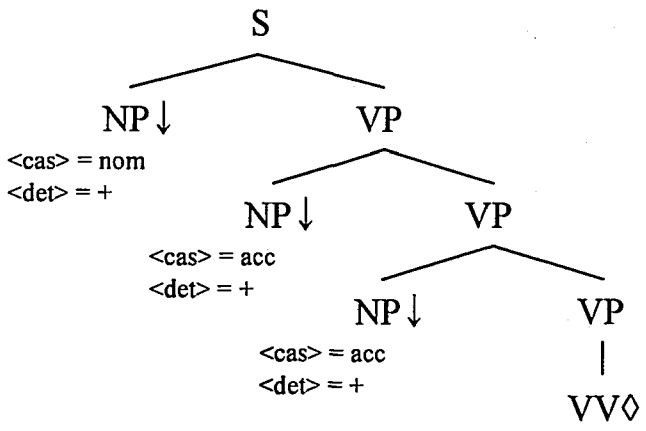


Figure 9. Schéma d'arbre nx0nx1nx1V

2.1.10 (S(S_SBJ↓)(VP @VV)) : s0V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin d'un sujet phrastique. Elle est montrée dans la Figure 10.

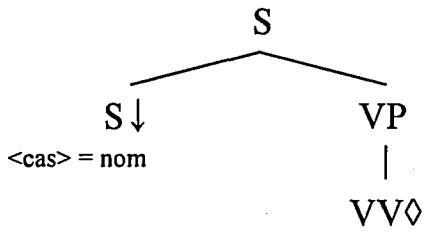


Figure 10. Schéma d'arbre s0V

2.1.11 (S(S_SBJ↓)(VP @COP)) : s0COP

Ce schéma d'arbres est sélectionné par la copule *i.da* avec un sujet phrastique. Elle est montrée dans la Figure 11.

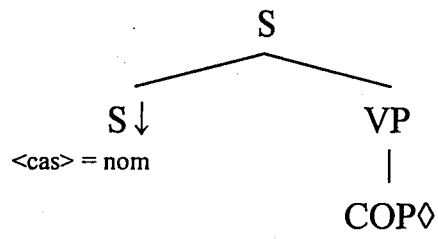


Figure 11. Schéma d'arbre s0COP

2.1.12 (S(NP_SBJ↓)(VP(S_OBJ↓)(VP @VV))) : nx0s1V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin d'un NP sujet et d'un objet phrastique. Elle est montrée dans la Figure 10.

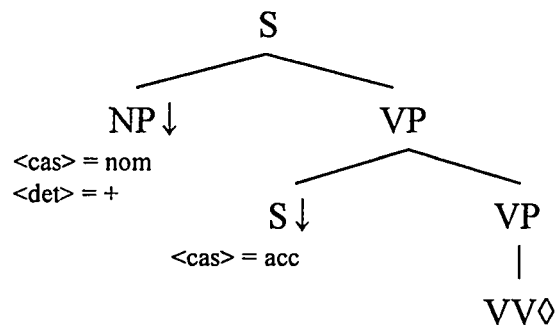


Figure 12. Schéma d'arbre nx0s1V

2.1.13 (S(NP_SBJ↓)(VP(S_CMP*)(VP @VV))) : nx0s2V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin d'un NP sujet et d'un complément phrastique. Elle est montrée dans la Figure 13.

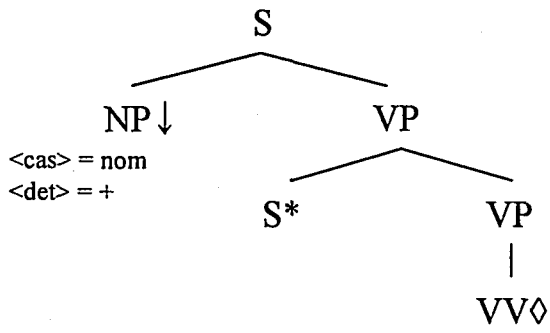


Figure 13. Schéma d'arbre nx0s2V

2.1.14 (S(NP_SBJ↓)(VP(S_CMP*)(VP(NP_OBJ↓)(VP @VV)))) : nx0S2nx1V

Ce schéma d'arbres est sélectionné par les verbes qui ont besoin d'un NP sujet, d'un complément phrastique et d'un NP objet. Elle est montrée dans la Figure 14.

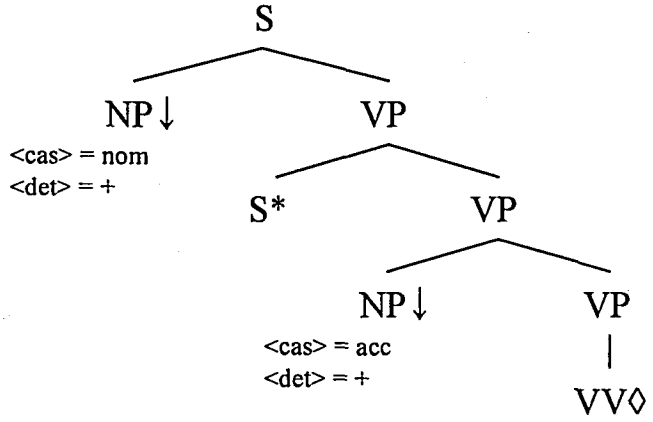


Figure 14. Schéma d'arbre nx0S2nx1V

2.2 Comparaison avec les sous-catégorisations de Han *et al.* (2000)

Nous comparaisons 11 sous-catégorisations des prédicats en coréen que Han *et al.* (2000) propose (voir le Tableau 2) avec 14 sous-catégorisations que nous avons proposé à partir des schèmes d'arbres extraits automatiquement.

Tnx0V	Intransitive verb	*balsaingha* ('avoir lieu'), *gamboha* ('diminuer')
Tnx0nx1V	Transitive verb	*yujiha* ('maintenir'), *jisiha* ('ordonner')
Tnx0nxp1V	Intransitive verb with an NP marked with a postposition	*idongha* ('déplacer'), *yulaiha* ('provenir')
Tnx0nxp1nx2V	Ditransitive verb	*ju* ('donner'), *bonai* ('envoyer')
Tnx0s1V	Verb with sentential complement	*sainggakha* ('penser'), *mid* ('croire')
Tnx0nxp1s2V	Verb with sentential complement and an NP marked with a postposition	*myengryengha* ('ordonner'), *bogoha* ('rapporter')
Tnx0nxNOM1V	Double nominative verb	*doi* ('devenir'), *iss* ('exister')
Tnx0nx1CO	Copula	*uisa.i* ('être docteur')
Tnx0A	Intransitive adjective	*keu* ('être grand'), *norah* ('être jaune')
Tnx0nxp1A	Intransitive adjective with an NP marked with a postposition	*daleu* ('être différent'), *gat* ('être même')
Tnx0nxNOM1A	Double nominative adjective	*pilyoha* ('être nécessaire'), *ani* ('ne pas être')

Tableau 2. Familles de Han et al. (2000)

La première différence est une reconnaissance du syntagme prépositionnel. Alors que Hans *et al.* (2000) a 3 sous-catégorisations qui contiennent un syntagme prépositionnel, nos sous-catégorisations n'en contient pas. Le corpus SJTree où nous avons extrait les grammaires lexicalisées et les schèmes d'arbre, ne distingue pas le syntagme prépositionnel du syntagme adverbial (en fait, le syntagme nominal avec une postposition adverbiale). Par exemple, alors que le syntagme prépositionnel *hakgyo.e* ('à l'école') dans la phrase *nai.ga hakgyo.e ga.ss.da* ('Je suis allé à l'école') est étiqueté comme NP_AJT, le syntagme adverbial *ojeon.e* ('matin') dans la phrase *nai.ga ojeon.e hakgyo.e ga.ss.da* ('Je suis allé à l'école ce matin') est aussi étiqueté comme NP_AJT. La postposition *e* de *ojeon.e* ('matin') et celle de *hakgyo.e* ('à

l'école') sont mêmes postpositions adverbiales dans la syntaxe coréenne. Pendant l'extraction de la grammaire lexicalisée, nous considérons tous les syntagmes NP_AJT comme une opération d'adjonction, c'est-à-dire que nous supprimons le syntagme NP_AJT dans l'arbre principal et le sous-arbre (dans ce cas, le syntagme NP_AJT) est extraite comme un arbre auxiliaire. Par conséquent, Han *et al.* (2000) contient une sous-catégorisation (la famille d'arbre Tnx0nxp1V) pour les phrases comme *nai.ga hakgyo.e ga.ss.da* ('Je suis allé à l'école'), la nôtre ne le contient pas. Cette distinction concerne aussi le verbe ditransif comme *ju.da* ('donner') où l'objet indirect est étiqueté comme NP_AJT (la famille d'arbre Tnx0nxp1nx2V dans celle de Han *et al.* (2000)).

La deuxième différence est la reconnaissance de l'attribut. L'attribut dans la syntaxe coréenne est très particulier car en coréen seul le syntagme nominal qui se trouve devant deux verbes *doi.da* ('devenir') et *ani.da* ('n'être pas') est considéré comme attribut. Puisque le corpus SJTree étiquette explicitement le syntagme de l'attribut comme NP_CMP, par exemple (NP_CMP *non.i*) ('champ') dans la phrase *bat.i non.i doi.eoss.da* ('Le champ est devenu une rizière'), nous avons une sous-catégorisation qui est (S(NP_SBJ↓)(VP(NP_CMP↓)(VP @VV))), alors que Han *et al.* (2000) considère cette phrase comme une phrase à deux sujets.

Nous avons les 4 sous-catégorisations qui contiennent les arguments phrastiques. Ces sous-catégorisations contiennent les 2 sous-catégorisations telles que le sujet phrastique et l'objet phrastique et les 2 sous-catégorisations des arguments phrastique de Han *et al.* (2000).

En récapitulant, nous n'avons pas les 3 sous-catégorisations de Han *et al.* (2000). D'ailleurs, Han *et al.* (2000) n'a pas les 6 sous-catégorisations chez nous.

3. Variations syntaxiques[23]

Dans la sous-catégorisation, le trait <mode> a une valeur *decl* (déclaratif). Les autres valeurs du trait <mode> sont possibles comme *imp* ('impératif'), *int* ('interrogatif'), etc. et le type de phrase n'est pas changé. Par exemple, la phrase déclarative *neo.ga hakgyo.e ga.ss.da* ('Tu es

[23] Le travail sur les variations syntaxiques nécessite une grande connaissance linguistique. Dans cette section, nous montrons seulement une possibilité de ce genre de travail basé sur les grammaires extraites automatiquement. Pour cette raison, nous le travaillerons ultérieurement.

allé à l'école') a le même type de phrase que la phrase interrogative *neo.ga hakgyo.e ga.ss.na* ('Es-tu allé à l'école')[24]. En outre, nous avons l'antéposition de l'objet, l'omission des arguments, etc. comme les variations syntaxiques.

3.1 Antéposition de l'objet

Puisque l'ordre des composants de la phrase est relativement libre en coréen, on y trouve souvent l'antéposition de l'objet au début de la phrase. Théoriquement l'attribut peut aussi être disloqué au début de la phrase, ce cas ne se trouve pas dans le corpus SJTree. Elle est montrée dans la Figure 15.

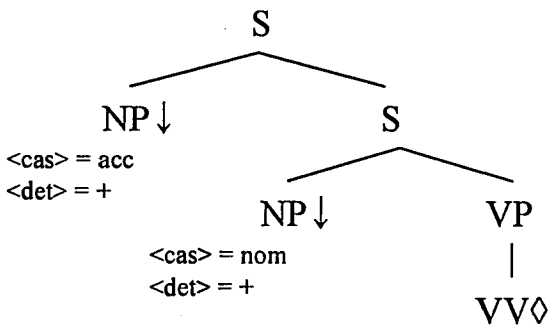


Figure 15. Antéposition de l'objet

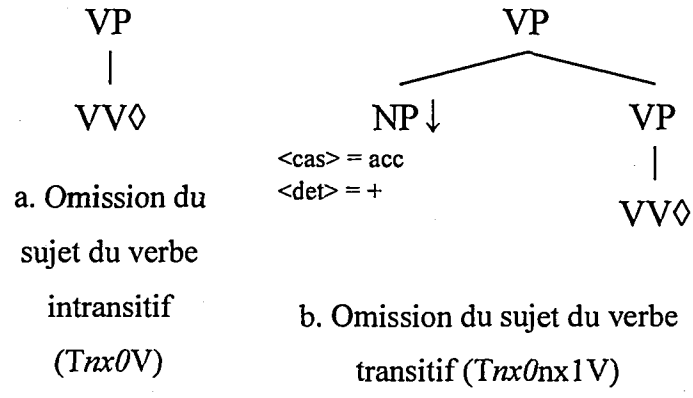
La liste suivante montre le nombre d'arbres lexicalisés que nous avons extraits à partir du corpus SJTree pour les phrases avec l'antéposition de l'objet.

- `(S(NP_OBJ↓)(S(NP_SBJ↓)(VP @VV)))  5`
- `(S(NP_OBJ↓)(S(VP_OBJ↓)(S(NP_SBJ↓)(VP @VV))))1`

[24] La phrase impérative avec un sujet comme *neo.ga hakgyo.e ga* ('(tu) Va à l'école') est aussi grammaticale dans la syntaxe coréenne. La phrase impérative a seulement un changement de la terminaison verbale comme la phrase interrogative.

3.2 Omission des arguments

Le corpus SJTree ne contient pas de catégorie vide qui correspond à un argument omis ou à une trace laissé par le déplacement. Si certains arguments sont omis dans la phrase, nous n'avons pas de moyen de le savoir. Cependant, des phrases sans sujet peuvent être explicitement reconnues comme une variation syntaxique qui omet le sujet.



a. Omission du sujet du verbe intransitif (T*nx0*V)

b. Omission du sujet du verbe transitif (T*nx0*nx1V)

Figure 16. Omission du sujet

La liste suivante montre le nombre d'arbres lexicalisés que nous avons extraits à partir du corpus SJTree pour les phrases où il y a une omission du sujet

- (VNP @COP) 209
- (VP @VA) 57
- (VP @VV) 166

3.3 Multiples sujets

Les doubles sujets pour le verbe et pour l'adjectif sont classés dans la sous-catégorisation initiale. Pourtant, les multiples sujets sont très difficiles à juger. En plus, puisque tous les marqueurs de topicalisation sont aussi annotés comme le sujet, nous rencontrons aussi le double sujet et le triple sujet pour la construction à copule dans le corpus SJTree. Ici, nous montrons seulement la liste des multiples sujets que nous rencontrons dans le corpus SJTree.

La liste suivante montre le nombre d'arbres lexicalisés que nous avons extraits à partir du corpus SJTree pour les phrases à multiples sujets.

- `(S(NP_SBJ↓)(S(NP_SBJ↓)(S(NP_SBJ↓)(VNP @COP))))  2`

- `(S(NP_SBJ↓)(S(NP_SBJ↓)(VNP @COP)))  21`

Conclusion

Les grammaires électroniques sont des ressources très importantes pour le traitement automatique des langues naturelles. Dans cette thèse, nous avons fourni un système qui peut extraire automatiquement une grammaire d'arbres adjoints lexicalisée et une grammaire d'arbres adjoints lexicalisée avec traits à partir de *Sejong* 21 Korean Treebank (le corpus SJTree). Dans le chapitre 1, nous avons commencé par une présentation brève du formalisme TAG et puis nous avons présenté les travaux d'extraction. Le chapitre 2 a présenté la structure du corpus SJTree à partir duquel une grammaire TAG lexicalisée est extraite. C'est un corpus arboré, annoté syntaxiquement pour le coréen. Nous en avons décrivé en particulier le schéma d'annotation, qui est basé sur l'analyse linguistique. Nous avons présenté également le schéma d'annotation du Penn Treebank pour le coréen (Penn Korean Treebank) et comparons les schémas des deux corpus. Nous avons proposé un algorithme d'extraction d'une grammaire TAG lexicalisée à partir d'un corpus arboré dans le chapitre 3. L'algorithme est implémenté et testé sur le corpus SJTree. Dans le chapitre 4, nous avons extrait les traits. L'extraction automatique des traits n'a pas encore été réalisée dans les autres travaux d'extraction. Ici, nous avons proposé une procédure d'extraction de traits qui permet d'établir une grammaire TAG lexicalisée avec traits pour le coréen. Dans le chapitre 5, nous avons évalué les grammaires extraites par leur taille, leur couverture et leur ambiguïté moyenne. Enfin, Dans le chapitre 6 nous avons présenté la sous-catégorisation des verbes et des adjectifs en coréen basée sur la grammaire extraite.

Parmi les perspectives de nos travaux, nous pouvons citer une extraction de grammaires utilisant d'autres formalismes, par exemple LFG et HPSG, l'emploi de la grammaire extraite, et l'étude des variations syntaxiques de la sous-catégorisation.

Nous pouvons utiliser les grammaires lexicalisées extraites automatiquement dans un parseur syntaxique pour un emploi de la grammaire extraite. Dans Park (2004b, 2004c), nous avons utilisé les grammaires extraites de cette thèse pour réaliser des analyses syntaxiques. Cependant, par manque d'un parseur statistique pour le moment, nous n'utilisons pas les statistiques et les probabilités que nous avons obtenues pendant la procédure d'extraction. Pour un travail futur, nous pourrions créer (ou utiliser) un parseur statistique pour résoudre les problèmes d'ambiguïtés apparaissant parmi les candidats d'analyse de la phrase.

Dans le chapitre 6, nous avons présenté une sous-catégorisation pour les prédicats du coréen, basée sur les schémas d'arbre extraits. Comme le travail sur les variations syntaxiques nécessite une grande connaissance linguistique, nous avons seulement montré que ce genre de travail était possible en se basant sur les grammaires extraites automatiquement. Nous allons continuer ce travail afin d'obtenir une sous-catégorisation complète pour le coréen et de l'évaluer.

Annexe A. La distribution des étiquettes dans le corpus SJTree

Le corpus SJTree emploie 42 étiquettes de partie de discours et 55 étiquettes syntaxiques.

1. Distribution des étiquettes POS

EC	4984
EF	2070
EP	1648
ETM	3698
ETN	319
IC	64
JC	333
JKB	2927
JKC	144
JKG	1247
JKO	2263
JKQ	84
JKS	1769
JKV	16
JX	2544
MAG	1552
MAJ	304
MM	988
NNB	2168
NNG	15302

NNP	1936
NP	577
NR	377
SE	52
SF	2285
SH	307
SL	167
SN	1409
SO	84
SP	900
SS	1337
SW	119
VA	1177
VCN	94
VCP	1136
VV	5294
VX	1614
XPN	64
XR	378
XSA	438
XSN	983
XSV	1284

2. Distribution des étiquettes syntaxiques

AP	1893
AP_AJT	6
DP	865
DP_AJT	2
IP	107
L	496
L_CMP	1
L_PRN	61
LP	39
LP_CMP	2
NP	9457
NP_AJT	5895
NP_CMP	335
NP_CNJ	1037
NP_INT	50
NP_IP	7
NP_MOD	1982
NP_OBJ	4145
NP_PRN	123
NP_SBJ	5406
QP	18
R	419
R_CMP	8
R_PRN	61
RP	5
S	3053
S_AJT	24
S_CMP	319
S_CMP]	2 ← Erreur d'annotation du corpus
S_CNJ	2
S_MOD	804
S_OBJ	19
S_SBJ	12
VNP	2276
VNP_AJT	27

```
VNP_CMP         145
VNP_CNJ         6
VNP_MOD         471
VNP_OBJ         27
VNP_PRN         4
VNP_SBJ         18
VP              13716
VP_AJT          129
VP_CMP          468
VP_CMP]         3       ← Erreur d'annotation du corpus
VP_CNJ          9
VP_MOD          7127
VP_OBJ          150
VP_PRN          4
VP_SBJ          98
X               159
X_AJT           27
X_CMP           81
X_CNJ           9
X_MOD           15
X_OBJ           29
X_SBJ           31

;; nombre de phrases = 2526
;; nombre d'étiquettes syntaxiques = 55
;; nombre d'étiquettes erronées = 2
```

Annexe B. L'algorithme d'extraction

Avant l'extraction automatique d'une grammaire TAG lexicalisée G_l, nous transformons la structure des phrases annotées par les parenthèses dans le corpus SJTree en arbres (`Tree tree`). Pour la construction des arbres élémentaires par méthode descendante, nous prenons un algorithme du parcours en profondeur d'abord (`depthFirstSearch(Tree tree)`). Ensuite, nous déterminons la tête (`setHead()`) et le type de l'opération (substitution ou adjonction) (`setSubNode()`) pour les nœuds des fils si le nœud donné est une noeud non-terminal. Les nœuds supprimés ou substitués pendant le parcours, sont aussi récursivement concernés par l'algorithme, afin de réduire à un arbre élémentaire la structure de l'arbre syntaxique qui reste (`reduireTronc()`). Voir la Figure 1.

L'algorithme consiste en deux parties majeurs : `depthFirstSearch(Tree tree)` et `depthFirstSearchAdjNode(Tree tree, Node parent)`. La fonction `depthFirstSearch(Tree tree)` est appelé dans l'algorithme principal et la fonction `depthFirstSearchAdjNode(Tree tree, Node parent)` est appelé pour construire des arbres auxiliaires. Nous fournissons aussi les fonctions `setHead()` et `setSubNode()` qui assignent la tête et les nœuds de substitution respectivement. La fonction `reduireTronc()` réduit à un arbre élémentaire la structure de l'arbre syntaxique qui reste et la fonction `extract()` imprime l'arbre extrait.

```
Entrée : une structure d'arbre pour les phrases du corpus SJTree.
Sortie : une grammaire TAG lexicalisée G1.
Begin Main ()

      depthFirstSearch(tree); // le parcours en profondeur d'abord

End Main ();
Entrée : une structure d'arbre du corpus SJTree
Sortie : un ensemble d'une grammaire TAG lexicalisée
```

```
Begin depthFirstSearch(Tree tree)

      if (!tree.isLeaf()) {

            // Déterminer la tête ;
            tree.setHead() ;

            // Distinguer entre l'opération de substitution
            // et d'adjonction ;
            tree.setSubNode() ;

            for (i, pour chaque nœud des fils daughter(i)) {
                  if (le fils est une tête) {
                      // Rien à faire ;
                  }
                  else if (le fils est un nœud de substitution) {
                      // Voir la Figure 5a du chapitre 3.
                      daughter(i).replaceSubNode() ;

                      // Appeler le parcours pour le nœud de substitution
                      depthFirstSearch(daughter(i));
                  }
                  eles {
                      // Le fils qui est ni tête ni nœud de substitution
                      // est un nœud d'adjonction ;
                      // voir la Figure 6a du chapitre 3.
                      daughter(i).removeAdjNode() ;

                      // Appeler le parcours pour le nœud d'adjonction ;
                      depthFirstSearchAdjNode(daughter(i), parentNode);
                  }
            }
      }

      // Appeler le parcours récursivement ;
      for (i, pour chaque nœud des fils daughter(i)) {
            depthFirstSearch(daughter(i));
      }

      // Réduire le tronc qui reste après avoir supprimé les nœuds
      // d'adjonction ;
      tree.reduireTronc() ;

      // Extraire l'arbre ;
      tree.extract() ;

End depthFristSearch();

Entrée : une partie de la structure d'arbre
Sortie : un ensemble d'une grammaire TAG lexicalisée
Begin depthFirstSearchAdjNode(Tree tree, Node parent)

      if (!tree.isLeaf()) {

            tree.setHead() ;
            tree.setSubNode() ;

            for (i, pour chaque nœud des fils daughter(i)) {
                  if (le fils est une tête) { }
                  else if (le fils est un nœud de substitution) {
                        daughter(i).replaceSubNode() ;
```

```
                        depthFirstSearch(daughter(i));
                  }
                  eles {
                        daughter(i).removeAdjNode() ;
                        depthFirstSearchAdjNode(daughter(i));
      }     }     }

      for (i, pour chaque nœud des fils daughter(i)) {
            depthFirstSearch(daughter(i));
      }

      tree.reduireTronc() ;

      // Ajouter le nœud racine celui du nœud parent du nœud donné
      // et son frère comme le nœud pied pour former un arbre auxiliaire.
      tree.addRoot(parent); tree.addFoot(parent);
      tree.extract() ;

End depthFirstSearchAdjNode() ;

Entrée : les noeuds des fils
Sortie : assigner le nœud de tête
Begine setHead()
      if (s'il y a le verbe auxiliaire) {
            tête = le nœud le plus à gauche ;
      }
      else {
            tête = le nœud le plus à droite ;
      }
End setHead() ;

Entrée : les noeuds des fils
Sortie : assigner le nœud de substitution
Begin setSubNode()
      if (l'étiquette des nœuds des fils est finie
            par _CMP || _OBJ || _SBJ) {
            nœud de substitution = thisNode() ;
      }
End setSubNode() ;

Entrée : une structure d'arbre
Sortie : une structure d'arbre réduite si c'est le cas
Begin reduireTronc()
      // Voir la Figure 7 du chapitre 3 ;
      for (parcours vers l'ancre) {
            if (l'étiquette du fils == l'étiquette du parent &&
                  il est un seul fils)
            removeParentNode() ;
      }
End reduireTronc() ;

Entrée : une structure d'arbre
Sortie : un arbre lexicalisé
Begin extract()
      //Imprimer l'arbre qui reste ;
      tree.print() ;
End extract() ;
```

Figure 1. L'algorithme d'extraction

Nous séparons les symboles à partir de la grammaire extraite G_1 pour la grammaire G_2 (Voir la Figure 2) et les postpositions à partir de la grammaire extraite G_2 pour la grammaire G_3 (Voir la Figure 3). Enfin, nous convertissons les arbres auxiliaires pour le syntagme nominal en arbres initiaux à partir de la grammaire G_3 pour la grammaire G_4 (Voir la Figure 4).

```
Entrée : une grammaire TAG lexicalisée G1
Sortie : une grammaire TAG lexicalisée G2
Begin Main ()
      // ltree est un arbre lexicalisé
      SELECT ltree FROM G1 ;
      for (pour les résultats de SELECT) {
            if (ltree contient les symboles) {
                  symbole = seperateSymbol(ltree) ;
                  ltree = removeSymobl(ltree) ;
                  if (les symboles sont les signes de ponctuation) {
                        tree = makeAuxTree(symbole) ;
                  }
                  else {
                        tree = makeInitTree(symbole) ;
                  }
                  INSERT INTO G2 VALUES (ltree) ;
                  INSERT INTO G2 VALUES (tree) ;
            }
            else {
                  INSERT INTO G2 VALUES (ltree) ;
            }
      }
END Main() ;
```

Figure 2. Extraction de la grammaire G_2

```
Entrée : une grammaire TAG lexicalisée G2
Sortie : une grammaire TAG lexicalisée G3
Begin Main ()
      // ltree est un arbre lexicalisé
      SELECT ltree FROM G2 ;
      for (pour les résultats de SELECT) {
            if (ltree contient la postposition) {
                  postp = seperatePostp(ltree) ;
                  ltree = removePostp(ltree) ;
                  tree = makeInitTree(postp) ;

                  INSERT INTO G3 VALUES (ltree) ;
                  INSERT INTO G3 VALUES (tree) ;
            }
            else {
                  INSERT INTO G3 VALUES (ltree) ;
            }
      }
END Main() ;
```

Figure 3. Extraction de la grammaire G_3

```
Entrée : une grammaire TAG lexicalisée G3
Sortie : une grammaire TAG lexicalisée G4
Begin Main ()
      // ltree est un arbre lexicalisé
      SELECT ltree FROM G3 ;
      for (pour les résultats de SELECT) {
            if (ltree est un arbre β pour le syntagme nominal) {
                  tree = makeInitTree(ltree) ;
                  INSERT INTO G4 VALUES (tree) ;
            }
            else {
                  INSERT INTO G4 VALUES (ltree) ;
            }
      }
END Main() ;
```

Figure 4. Extraction de la grammaire G_4

Pour la grammaire G_5 nous enlevons les étiquettes syntaxiques, utilisons un ensemble d'étiquettes réduit, ajoutons la structure des traits, et introduisons une forme infinitive pour son ancre lexicale à partir de la grammaire G_4 (Voir la Figure 5). Actuellement, les traits de la grammaire G_5 sont extraits pendant la transformation en TAGML. La fonction `addSynFeature(conversionTable)` remplit donc le trait *cat* de `<node>` dans la partie `<treeLib>` et la fonction `addMorphFeature(EPEFTable)` remplit les traits de `<fs>` dans la partie `<morphLib>` de TAGML.

```
Entrée : une grammaire TAG lexicalisée G4,
         Tableau de conversion d'étiquettes syntaxiques,
         Tableau de EP et EF (les terminaisons verbales)
Sortie : une grammaire TAG lexicalisée G5
Begin Main ()
      // ltree est un arbre lexicalisé
      SELECT ltree FROM G4 ;
      for (pour les résultats de SELECT) {
            if (s'il y a les étiquettes syntaxiques) {
                  tree = makeTreeWithoutSynTag(ltree) ;
                  addSynFeature(conversionTable) ;
                  addMorphFeature(EPEFTable) ;
                  // Ajouter d'équations
                  addEquation() ;
                  INSERT INTO G5 VALUES (tree) ;
            }
            else {
                  INSERT INTO G5 VALUES (ltree) ;
            }
      }
END Main() ;
```

Figure 5. Extraction de la grammaire G_5

Annexe C. Le tableau de conversion d'étiquettes syntaxiques

Ancre	Type d'arbre	Etiquettes syntaxiques	Type de nœud	Exemple de conversion
verbe	α	NP_SBJ	subst	NP [<cas> = nom <det> = +]
verbe	α	NP_OBJ	subst	NP [<cas> = acc <det> = +]
verbe	α	NP_CMP	subst	NP [<cas> = attr <det> = +]
verbe	α	S_SBJ	subst	S [<cas> = nom <det> = +]
verbe	α	S_OBJ	subst	S [<cas> = acc <det> = +]
verbe	α	S_CMP	subst	S [<cas> = attr <det> = +]
verbe	α	VP_SBJ	subst	VP [<cas> = nom <det> = +]
verbe	α	VP_OBJ	subst	VP [<cas> = acc <det> = +]
verbe	α	VP_CMP	subst	VP [<cas> = attr <det> = +]
verbe	α	VNP_SBJ	subst	VNP [<cas> = nom <det> = +]

verbe	α	VNP_OBJ	subst	VNP [<cas> = acc <det> = +]
verbe	α	VNP_CMP	subst	VNP [<cas> = attr <det> = +]
verbe	α|β	VP, VP_MOD	-	VP [<ep> <ef> <mode> <tense>]
ancré par le syntagme _MOD	β	NP|NP_CMP|NP_MOD |NP_OBJ| |NP_SBJ	racine	NP [<det> = +]
ancré par le syntagme _MOD	β	NP|NP_CMP|NP_MOD |NP_OBJ| |NP_SBJ	foot	NP [<cae = *x*>]
ancré par le syntagme _MOD	β	S|S_CMP|S_MOD |S_OBJ|S_SBJ	racine	S [<det> = +]
ancré par le syntagme _MOD	β	S|S_CMP|S_MOD |S_OBJ|S_SBJ	foot	NP [<cae = *x*>]
ancré par le syntagme _MOD	β	VP|VP_CMP |VP_MOD|VP_OBJ |VP_SBJ	racine	VP [<det> = +]
ancré par le syntagme _MOD	β	VNP|VNP_CMP |VNP_MOD|VNP_OBJ |VNP_SBJ	racine	VNP [<det> = +]
postposition	α	NP_SBJ	racine	NP [<cas> = nom]
postposition	α	NP_SBJ	subst	NP [<cas> = NONE]
postposition	α	NP_OBJ	racine	NP [<cas> = acc]
postposition	α	NP_OBJ	subst	NP [<cas> = NONE]
postposition	α	NP_CMP	racine	NP [<cas> = attr]
postposition	α	NP_CMP	subst	NP [<cas> = NONE]
postposition	α	S_SBJ	racine	S [<cas> = nom]
postposition	α	S_SBJ	subst	S [<cas> = NONE]
postposition	α	S_OBJ	racine	S [<cas> = acc]

postposition	α	S_OBJ	subst	S [<cas> = NONE]
postposition	α	S_CMP	racine	S [<cas> = attr]
postposition	α	S_CMP	subst	S [<cas> = NONE]
postposition	α	VP_SBJ	racine	VP [<cas> = nom]
postposition	α	VP_SBJ	subst	VP [<cas> = NONE]
postposition	α	VP_OBJ	racine	VP [<cas> = acc]
postposition	α	VP_OBJ	subst	VP [<cas> = NONE]
postposition	α	VP_CMP	racine	VP [<cas> = attr]
postposition	α	VP_CMP	subst	VP [<cas> = NONE]
postposition	α	VNP_SBJ	racine	VNP [<cas> = nom]
postposition	α	VNP_SBJ	subst	VNP [<cas> = NONE]
postposition	α	VNP_OBJ	racine	VNP [<cas> = acc]
postposition	α	VNP_OBJ	subst	VNP [<cas> = NONE]
postposition	α	VNP_CMP	racine	VNP [<cas> = attr]
postposition	α	VNP_CMP	subst	VNP [<cas> = NONE]

Annexe D. L'exemple d'une entrée et d'une sortie de LLP2

Dans cette annexe nous montrons un exemple d'une entrée et d'une sortie de LLP2 pour la phrase (1). LLP2 produit 20 analyses différentes pour cette phrase de 6 *eojeol*s.

(1) 일본 외부성은 즉각 해명 성명을 발표했다.

ilbon *oinuseong.eun* *jeuggag* *haimyeng*
Japon ministre_des_affaires_étrangères.Nom immédiatement élucidation
seongmyeng.eul *balpyoha.eoss.da*
déclaration.Acc annoncer.Pass.Ter

'Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation.'

1. Entrée

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE setOfSentences SYSTEM "segment.dtd">

<!--
       An example to test
-->

<setOfSentences>
  <sentence text="ilbon oimuseong_eun jeuggag haemyeng seongmyeng_eul balpyo_ha_ass_da_.">
    <segment loc = "0"><word text="ilbon"/></segment>
    <segment loc = "1"><word text="oimuseong_eun"/></segment>
    <segment loc = "2"><word text="jeuggag"/></segment>
    <segment loc = "3"><word text="haimyeng"/></segment>
    <segment loc = "4"><word text="seongmyeng_eul"/></segment>
    <segment loc = "5"><word text="balpyo_ha_ass_da_."/></segment>
  </sentence>
</setOfSentences>
```

2. Sortie

```
#=======================================
#LLP2 PARSER RESULT
#=======================================
#Parsed sentence : ilbon oimuseong_eun jeuggag haemyeng seongmyeng_eul balpyo_ha_ass_da_.
#Date :Mon Apr 24 17:43:14 CEST 2006

// 1:
derivation tree [id=dt_51] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_0] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 2:
derivation tree [id=dt_53] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_12] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 3:
derivation tree [id=dt_55] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_24] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 4:
derivation tree [id=dt_57] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_36] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 5:
derivation tree [id=dt_59] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_48] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 6:
derivation tree [id=dt_61] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_60] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 7:
derivation tree [id=dt_63] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_72] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 8:
derivation tree [id=dt_65] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_84] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 9:
```

```
derivation tree [id=dt_67] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_96] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 10:
derivation tree [id=dt_69] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_108] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 11:
derivation tree [id=dt_71] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_120] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 12:
derivation tree [id=dt_73] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_132] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 13:
derivation tree [id=dt_75] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_144] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 14:
derivation tree [id=dt_77] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_156] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 15:
derivation tree [id=dt_79] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_168] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 16:
derivation tree [id=dt_81] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_180] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 17:
derivation tree [id=dt_83] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_192] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 18:
derivation tree [id=dt_85] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_204] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
```

```
#---------------------------------------

// 19:
derivation tree [id=dt_87] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_216] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------

// 20:
derivation tree [id=dt_89] :=  balpyo_ha_ass_da_.( seongmyeng_eul( haimyeng* ) jeuggag*
oimuseong_eun( ilbon* ) )
derived tree [id=d_228] := <S>(<NP_SBJ>(<NP>(<ilbon/NNP>)<NP_SBJ>(<oimuseong/NNG +
eun/JX>))<VP>(<AP>(<jeuggag/MAG>)<VP>(<NP_OBJ>(<NP>(<haimyeng/NNG>)<NP_OBJ>(<seongmyeng/NNG +
eul/JKO>))<VP>(<balpyo/NNG + ha/XSV + ass/EP + da/EF + ./SF>))))
#---------------------------------------
Processed sentence : ilbon oimuseong_eun jeukgak haemyeng seongmyeng_eul balpyo_ha_eoss_da_.
Morpho loading time : 0ms
Resource loading time : 15703ms
Lemmatizing time : 266ms
Chart initializing time : 16ms
Parsing time : 16ms
Derivation Trees building time : 62ms
Nb of words :6
Nb of segment :16
Nb of elementary trees : 16
Nb of items at the beginning: 50
Nb of items at the end: 76
Nb of parsed sentences : 20
```

Annexe E. Le *chunker*

Dans cette annexe nous présentons notre système de *chunker*. La grammaire extraite peut être utilisée dans un système d'analyse syntaxique, donc un parseur. Ce parseur, qui est déjà présenté dans Park (2004b, 2004c), utilise deux niveaux : shallow parsing et deep parsing. Le but du shallow parsing (ou chunking) est de diminuer le temps de l'analyse au niveau du deep parsing.

Au niveau du chunking, le système analyse la phrase entrée comme un ensemble de structure plate. Le deep parseur prend le résultat du chunker pour continuer l'analyse syntaxique. A travers ce processus, nous pouvons diminuer le temps de l'analyse sans modification de l'algorithme de TAG car le nombre d'entrées pour le deep parsing est diminué en moyenne de 70% après shallow parsing. Plus on diminue le nombre d'entrées, moins l'analyse syntaxique prend de temps.

Le chunker prend une phrase étiquetée morphologiquement en entrée et produit une structure plate en sortie. La structure plate est groupée par la structure syntagmatique, mais elle ne contient pas d'informations sur le gouvernement et le liage. La fonction de cette structure plate est de diminuer la longueur de l'entrée en groupant les syntagmes et, ainsi, d'améliorer la vitesse au niveau du deep parsing.

Pour segmenter la phrase d'entrée et pour attribuer l'étiquette syntagmatique, le chunker est basé sur un ensemble de règles.

D'abord, nous avons quatre règles pour la segmentation de la phrase, ces règles coupent comme un syntagme si la fonction `ends_with(POSTPOSITION)` ou `ends_with(ENDING)` est satisfaite, ou divisent en deux segments - l'un pour le dernier mot et les autres pour ceux qui restent - si la fonction `ends_with(ADVERB)` ou `ends_with(SYMBOL)` est rempli. De plus, la fonction `ends_with(SYMBOL)` considère les symboles qui se trouvent à la fin du syntagme, tel que le signe de ponctuation comme le point et le virgule. Les symboles tels que les marqueurs de citation peuvent être au milieu du syntagme, ils sont donc considérés comme une partie du syntagme, c'est-à-dire qu'on ne les découpe pas.

En ce qui concerne l'attribution des étiquettes, JX (postposition auxiliaire) et ENDING (terminaison verbale) sont concernés par plusieurs règles. Une fonction `look_back(var)` est introduite et elle *look back* les variables (`var`) qui sont précédées par JX ou par ENDING pour vérifier l'étiquette du syntagme. S'il y a des cas ou la fonction `look_back(var)` ne trouve pas une condition donnée, par exemple, SS + JX ou XSN + JX, alors la fonction retourne NP en tant que valeur par défaut. De même, si la fonction `look_back(var)` rencontre JX + JX dans le corpus, nous considérons aussi le syntagme comme NP par défaut.

La Figure 1 montre l'entrée et la sortie du shallow parsing. Comme nous l'avons déjà décrit, le shallow parseur produit une structure plate telles celle de la Figure (1b) pour la phrase d'entrée et ne montre aucune relation de gouvernement. Ensuite nous décrivons toutes les règles de la segmentation et de l'attribution qui sont utilisées dans le chunker.

(1) 일본 외무성은 즉각 해명 성명을 발표했다.

ilbon *oimuseong.eun* *jeukgak* *haemyeng*
Japon ministre_des_affaires_étrangères.Nom immédiatement élucidation
seongmyeng.eul *balpyoha.eoss.da*
déclaration.Acc annoncer.Pass.Ter

'Le ministre des affaires étrangères du Japon a apporté immédiatement son élucidation.'

일본 *ilbon*/NNP

외무성 *oibwuseon*/NNG + 은 *eun*/JX

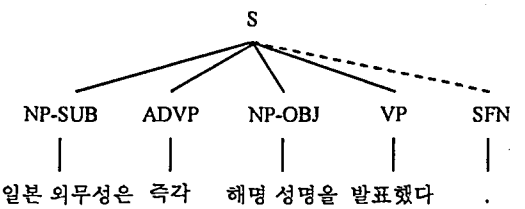
즉각 *jeukgak*/ADV

해명 *haemyeng*/NNG

성명 *seongmyeng*/NNG + 을 *eul*/JKO

발표하 *balpyoha*/VV + 었 *eoss*/EP + 다 *da*/EF + ./SF

a. Entrée : phrase étiquetée morphologiquement



b. Sortie : structure plat

Figure 1. Entrée et sortie de shallow parsing

Les règles de la segmentation

- `IF ends_with (POSTPOSTION), THEN end_of_phrase (addr);`
- `IF ends_with (ENDING), THEN end_of_phrase (addr);`
- `IF ends_with (ADVERB), THEN end_of_phrase (addr-1) AND end_of_phrase (addr);`
- `IF ends_with (SYMBOL), THEN end_of_phrase (addr-1) AND end_of_phrase (addr);`

Les règles de l'attribution du syntagme

- `IF ends_with (NNx) THEN return NP;` [25]

[25] Où NN*x* est NNG, NNP ou NNB pour le nom commun, le nom propre ou le nom dépendant respectivement. Voir le chapitre 2 pour le détail des étiquettes utilisées dans le corpus SJTree.

- IF ends_with (JKx) THEN return NP; 26
- IF ends_with (JX) AND look_aback(NOUN) THEN return NP;
- IF ends_with (JX) AND look_aback(POSTPOSITION) THEN return NP;
- IF ends_with (JX) AND look_aback(ENDING) THEN return VP;
- IF ends_with (JX) AND look_aback(ADVERB) THEN return ADVP;
- IF ends_with (JX) AND look_aback(NOT NOUN AND NOT POSTPOSITION AND NOT ENDING) THEN return NP ;;; SS + JX, XSN + JX
- IF ends_with (ENDING) AND look_aback(VERB) THEN return VP;
- IF ends_with (ENDING) AND look_aback(ADJECTIVE) THEN return AP;
- IF ends_with (ENDING) AND look_aback(VX) THEN return VXP;
- IF ends_with (ADVERB) THEN return ADVP;
- IF ends_with (EXCLAMATION) THEN return EXP;
- IF ends_with (SYMBOL) THEN return do_nothing();

Pour l'évaluation du chunker, nous l'avons testé sur une partie du corpus *Sejong* annoté morphologiquement, qui contient 270.629 phrases, avec en moyenne 12.87 *eojeols*, c'est-à-

[26] Où JK*x* est JKS, JKO, ou JKC pour la postposition nominative, la postposition accusative ou la postposition complémentaire respectivement. Voir aussi Voir le chapitre 2 pour le détail des étiquettes utilisées dans le corpus SJTree.

dire en moyenne 28.60 mots par phrase. Le nombre des syntagmes produits dans la structure plate pour une phrase est diminué en moyenne de 9.45.

Bibliographie

ABEILLE Anne, 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français*. Thèse de doctorat, Université Paris 7.

ABEILLE Anne, 2002. *Une grammaire électronique du français*, CNRS Editions, Paris.

ABEILLE Anne, 2003. *Treebanks : Buidling and Using Parsed Corpora*, Kluwer, Dordrecht.

ABEILLE Anne, BLACHE Philippe, 2000. Grammaires et analyseurs syntaxiques, in J.-M. Pierrel (éd.) *Ingénierie des Langues*, Paris, Hermès.

ABEILLE Anne, CANDITO Marie-Hélène, 2000. FTAG : a lexicalized Tree adjoining grammar for French. In ABEILLE Anne, RAMBOW Owen, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*. CSLI Publications, Stanford, CA.

ABEILLE Anne, CLEMENT Lionel, TOUSSENEL Francois, 2003. Building a Treebank for Frenche. In A. ABEILLÉ (ed) *Treebanks : Buidling and Using Parsed Corpora*, Kluwer, Dordrecht.

ABEILLE Anne, RAMBOW Owen, 2000, *Tree Adjoining Grammars: formalism, linguistic analysis and processing*, Stanford, CSLI.

BALDWIN Breckenridge, DORAN Christine, REYNAR Jerey, NIV Michael, SRINIVAS B., WASSON Mark, 1997. EAGLE: An Extensible Architecture for General Linguistic Engineering. In *Proc. of RIAO97*, Montreal.

BRESNAN Joan, 2001. *Lexical-Functional Syntax*. Oxford: Blackwell Publishers.

CAHILL Aoife, MCCARTHY Mairead, O'DONOVAN Ruth, GENABITH Josef van, WAY Andy, 2003. A Suite of Linguistic Tools for Use with the Penn-II Treebank. In *Proceedings of the LFG03 Conference*. CSLI Publications On-line,

CANDITO Marie-Hélène, 1999. *Organisation modulaire et paramétrable de grammaire électronique lexicalisées*. Thèse de doctorat, Université Paris 7.

CARPENTER R., 1992. *The Logic of typed feature structures*, Cambridge University Press.

CHANDRASEKAR R., SRINIVAS B., 1997. Gleaning information from theWeb: Using Syntax to Filter out Irrelevant Information. In *Proc. of AAAI 1997 Spring Symposium on NLP on the World Wide Web.*

CHEN John, 2001. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information.* PhD Thesis, University of Delaware.

CHEN John, VIJAY-SHANKER K., 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*, p. 65-76, 2000.

CHIANG David, 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Data Oriented Parsing*, CSLI Publication, pp. 299-316.

COLLINS Mike, 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proc. of ACL-1997.*

COPESTAKE Ann, 2002. *Implementing Typed Feature Structure Grammars*, CSLI., Chicago Press

CRABBE Benoit, 2005. *Computational representation of strongly lexicalised syntactic formalisms: application to Tree Adjoining Grammars*. PhD, University of Nancy 2.

DALRYMPLE Mary, 2001. *Lexical Functional Grammar*. New York: Academic Press.

DORAN Christine, HOCKEY Beth Ann, SARKAR Anoop, SRINIVAS B., XIA Fei, 2000. Evolution of the XTAG System. In Anne ABEILLÉ, Owen RAMBOW, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing.* CSLI Publications, Stanford, CA.

DORAN Christine, SRINIVAS B., HOCKEY Beth Ann, ZAIDEL Martin, EGEDI Dania, 1994. XTAG System - A Wide Coverage Grammar for English, *Coling 1994.*

DUCHIER D., LEROUX J., PARMENTIER Y., 2005. XMG un compilateur de métagrammaires extensible, in *Proceedings TALN 2005*, Dorudan.

ETRI, 1999. *Critère du Tag Set*, Electronics and Telecommunications Research Institute. (in Korean).

FRANK Anette, 2001. Treebank Conversion. Converting the NEGRA treebank to an LTAG grammar. in *Proceedings of the Workshop on Multi-layer Corpus-based Analysis*, July 30, 2001, Iasi, Romania.

FRANK Anette, 2002. A (Discourse) Functional Analysis of Asymmetric Coordination. in *Proceedings of the LFG 2002 Conference*, Athens, CSLI Online Publications.

GROSS Maurice, 1975. *Méthode en syntaxe*, Paris: Hermann.

HABASH Nizar, RAMBOW Owen, 2004. Extracting a Tree Adjoining Grammar from the Penn Arabic Treebank. In *Proceedings of Traitement Automatique du Langage Naturel (TALN-04)*. Fez, Morocco, 2004.

HAN Chunghye, HAN Narae, KO Eonsuk, 2001. *Bracketing Guidelines for Penn Korean TreeBank.* Technical Report IRCS-01-10, University of Pennsylvania

HAN Chunghye, YOON Juntae, KIM Nari, PALMER Martha, 2000. *A Feature-Based Lexicalized Tree Adjoining Grammar for Korean.* IRCS Technical Report 00-04. University of Pennsylvania.

JOHANSEN Ane Dybro, 2004. *Extraction des grammaires LTAG à partir d'un corpus étiquette syntaxiquement.* DEA mémoire, Université Paris 7.

JOSHI Aravind K., LEVY L., TAKAHASHI M., 1975. Tree Adjunct Grammars. *Journal of Computer and System Sciences.*

JOSHI Aravind, VIJAY-SHANKER K., 1999. Compositional Semantics with LTAG: How Much Underspecication Is Necessary? in *Proc. of 3nd International Workshop on Computational Semantics.*

KALLMEYER Laura, JOSHI Aravind, 1999. *Underspecied Semantics with LTAG.*

KAY M., GAWRON J.-M., NORVIG P., 1994. *Verbmobil : a Translation System for Face-to-Face Dialog*, CSLI, Chicago Press.

KIPPER Karin, DANG Hoa Trang, SCHULER William, PALMER Martha, 2000. Building a class-based verb lexicon using TAGs. In *Proceedings of TAG+5*, Université Paris 7, Jussieu May 25-27, 2000.

LOPEZ Patrice, 2000. Extended Partial Parsing for Lexicalized Tree Grammars. In *International Workshop on Parsing Technology*, IWPT 2000, Trento, Italy.

MAGERMAN David M., 1995. Statistical Decision-Tree Models for Parsing. In *Proc. of ACL-1995.*

MARCUS Mitchell, KIM Grace, MARCINKIEWICZ Mary Ann, 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proc of ARPA speech and Natural language workshop.*

MCCOY K. F., VIJAY-SHANKER K., YANG G., 1992. A Functional Approach to Generation with TAG. In *Proc. of ACL-1992.*

MÜLLER Stefan, 1999. Deutsche Syntax deckarative. Head-Driven Phrase Structure Grammar für das Deutsche. *Linguistische Arbeiten*, No. 394, Tübingen : Max Niemeyer Verlag.

NASR Alexis, 2004. *Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement.* Habilitation à diriger des recherches, Université Paris 7.

NEUMANN Gunter, 1998. Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks. in *Proc. of the 4th International Workshop on TAG and Related Frameworks* (TAG+4).

NEUMANN Günter, 2003. A Uniform Method for Automatically Extracting Stochastic Lexicalized Tree Grammar from Treebank and HPSG, In A. ABEILLÉ (ed) *Treebanks : Buidling and Using Parsed Corpora*, Kluwer, Dordrecht.

PALMER Martha, RAMBOW Owen, NASR Alexis, 1998. Rapid Prototyping of Domain Specific Machine Translation Systems, 8 pg., *AMTA-98*, Langhorne, PA, Oct 28-31, 1998.

PALMER Martha, ROSENZWEIG Joseph, SCHULER William, 1998. Capturing Motion Verb General-izations with Synchronous TAGs, *Predicative Forms in NLP*, pp 250-277, ed by Patrick St. Dizier, Kluwer Press, December, 1998.

PARK Jungyeul, 2004a. LTAG Extraction from a Treebank for Korean. In *Proceeding of Korea Information Science Society Annual Meeting*, Seoul. (in Korean)

PARK Jungyeul, 2004b. Automatic Bracketing using 2 Level Analysis for Korean. In *Proceeding of Korea Information Science Society Annual Meeting*, Seoul. (in Korean)

PARK Jungyeul, 2004c. Partially Lexicalized TAG Parser, In *The 15th Meeting of Computational Linguistics in the Netherlands*, University of Leiden, Leiden.

PAROUBEK P., SCHABES Y., JOSHI A., 1992. XTAG: a graphical Workbench for developing TAGs, *4th Conference on Applied NLP*, Trento, p.223-227.

PRASAD Rashmi, SARKAR Anoop, 2000. Comparing Test-Suite Based Evaluation and Corpus-Based Evaluation of a Wide-Coverage Grammar for English. *Using Evaluation within HLT Programs: Results and Trends*. LREC 2000 Satellite Workshop, p.7-12, 2000.

SAG Ivan A., POLLARD Carl J., 1987. *Head-Driven Phrase Structure Grammar: An Informal Synopsis*. CSLI Report 87-79, Stanford University.

SAG Ivan A., POLLARD Carl J., 1989. Subcategorization and Head-Driven Phrase Structure. In BALTIN Mark and KROCH Anthony, editor(s), *Alternate Conceptions of Phrase Structure*. 139–181. Chicago: University of Chicago Press.

SAG Ivan A., WASOW Thomas, BENDER Emily M., 2003. *Syntactic Theory: A Formal Introduction*. Stanford: CSLI Publications. Second edition.

SAGER N., 1981. *Natural language information processing: a computer grammar for English and its applications*, Reading, Addison Wesley.

SALKOFF M., 1973. *Une grammaire en chaîne du français*, Dunod.

SCHABES Yves, 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania.

SCHABES Yves, SHIEBER Stuart, 1992. An Alternative Conception of Tree-Adjoining Derivation. in *Proceeding of ACL-1992*.

SCHABES Yves, WATERS Richard, 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*.

SEJONG PROJECT, 2003. *Final Report for Sejong Korean Treebank*. Minstre d'éducation en Corée.

SIMOV Kiril, 2001. *Grammar Extraction from an HPSG Corpus*. BulTreeBank Project. Linguistic Modelling Laboratory, Bulgarian Academy of Sciences Acad. G. Bonchev St. 25A, 1113 So a, Bulgaria.

SKUT W., KRENN B., BRANTS T., USZKOREIT H., 1997. An Annotation Scheme for Free Word Order Languages. In *Proc. of 5th International Conference of Applied Natural Language*.

SRINIVAS B., 1997. *Complexity of Lexical Descriptions and Its Relevance to Partial Parsing.* Ph.D. thesis, University of Pennsylvania.

SRINIVAS B., SARKAR A., DORAN C., HOCKEY B. A., 1998. Grammar and Parser Evaluation in the XTAG Project. In *Workshop on Evaluation of Parsing Systems*, Granada, Spain, 26 May. 1998.

STEEDMAN Mark, 1996. *Surface Structure and Interpretation.* Linguistic Inquiry Monograph No.30, MIT Press.

STEEDMAN Mark, 2000. *The Syntactic Process*, MIT Press.

STEEDMAN Mark, BALDRIDGE Jason, 2003. *Combinatory Categorial Grammar.* Unpublished Tutorial Paper.

STONE Matthew, DORAN Christine, 1997. Sentence Planning as Description Using Tree Adjoining Grammar. In *Proc. of ACL-1997.*

TAYLOR Ann, MARCUS Mitchell, SANTORINI Beatrice, 2003. The Penn Treebank : An Overview. In ABEILLÉ A. (ed) *Treebanks : Buidling and Using Parsed Corpora*, Kluwer, Dordrecht.

THE XTAG GROUP, 1999. *A Lexicalized Tree Adjoining Grammar for English*, Technical Report, University of Pennsylvania. http://www.cis.upenn.edu/~xtag

THOMASSET F., DE LA CLERGERIE E., 2005. Comment obtenir plus des méta-grammaires, in *Proceedings TALN 2005*, Dorudan.

Tseng J., 2003. A French HPSG Grammar, *ESSLLI Workshop on Grammar Development*, Vienna.

VIJAY-SHANKER K., 1987. *A Study of Tree Adjoining Grammar.* PhD thesis, University of Pennsylvania, Philadelphia.

WEBBER Bonnie, JOSHI Aravind. 1998. Anchoring a Lexicalized Tree Adjoining Grammar for Discourse. In *Proc. of ACL-COLING Workshop on Discourse Relations and Discourse Markers.*

WEBBER Bonnie, KNOTT Alistair, STONE Matthew, JOSHI Aravind. 1999. What Are Little Trees Made of: A Structural and Presuppositional Account Using Lexicalized TAG. In *Proc. of International Workshop on Levels of Representation in Discourse* (LORID-1999).

XIA Fei, 2001. *Automatic Grammar Generation from Two Different Perspectives*. PhD Thesis, University of Pennsylvania, PA.

XIA Fei, HAN Chunghye, PALMER Martha, JOSHI Aravind, 2001. Automatically Extracting and Comparing Lexicalized Grammars for Dierent Languages. In *Proc. of the Seventeenth International Joint Conference on Articial Intelligence* (IJCAI-2001), Seattle, Washington.

XIA Fei, PALMER Martha, JOSHI Aravind K., 2000. A Uniform Method of Grammar Extraction and Its Application. In *The Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong, Oct 7-8, 2000.

Sommaire