# Intégration de ressources lexicales riches dans un analyseur syntaxique probabiliste

Anthony Sigogne

UNIVERSITÉ PARIS-EST

# Intégration de ressources lexicales riches dans un analyseur syntaxique probabiliste

## THÈSE DE DOCTORAT

présentée et soutenue publiquement le 3 décembre 2012

pour l'obtention du

**Doctorat de l'Université Paris-Est**

**(spécialité informatique linguistique)**

**au titre de l'Ecole Doctorale MSTIC**

par

Sigogne Anthony

**Composition du jury**

| | |
|---|---|
| *Directeur de thèse :* | Éric Laporte (Université Paris-Est) |
| *Co-directeur de thèse :* | Matthieu Constant (Université Paris-Est) |
| *Rapporteurs :* | Alexis Nasr (Université Aix-Marseille)<br>Thierry Poibeau (LaTTiCe) |
| *Examinateurs :* | Djamé Seddah (Université Paris-Sorbonne Paris 4)<br>Isabelle Tellier (Université Sorbonne Nouvelle Paris 3) |

**Laboratoire d'Informatique Gaspard Monge — UMR 8049 LIGM Labex Bezout**

# Remerciements

En premier lieu, je souhaiterais remercier les membres du jury, à savoir Alexis Nasr, Thierry Poibeau, Djamé Seddah et Isabelle Tellier, pour leur remarques et critiques pertinentes sur mon travail.

Je remercie également mon directeur de thèse, Eric Laporte, ainsi que mon co-directeur, Matthieu Constant, pour toute l'aide apportée tout au long de mon parcours de jeune chercheur. Grâce à eux, il m'a été possible de développer de nombreuses compétences utiles à ma future carrière.

De manière plus générale, j'aimerais également remercier les différents membres de l'équipe INFOLINGU :
– Rosa Cetro et Myriam Rakho, mes deux collègues de bureau qui m'ont supporté durant ces 3 années.
– Li Chi et Elsa Tolone, qui accomplissent à présent une brillante carrière dans la recherche.
– Tita Kyriacopoulou, pour sa bonne humeur et l'impulsion qu'elle donne à l'équipe.
– Sébastien Paumier, pour m'avoir permis d'intégrer une partie de mon travail dans le logiciel collaboratif Unitex.
– ainsi que tous les autres...

Je voudrais saluer les membres d'autres équipes avec qui j'ai discuté, que ce soit pour des réunions de travail ou simplement lors de conférences :
– Thomas François et Seyed Abolghasem Mirroshandel, avec qui j'ai participé à plusieurs conférences et excursions.
– Joseph Le Roux et Patrick Watrin, avec qui j'ai participé à l'écriture de plusieurs articles.
– Marie Candito et Djamé Seddah, qui m'ont donné de nombreux conseils et n'ont pas hésité à m'aider lorsque j'en avais besoin.

Pour finir, je remercie ma famille et mes amis qui m'ont soutenu pendant ces 3 années.

# Résumé

Cette thèse porte sur l'intégration de ressources lexicales et syntaxiques du français dans deux tâches fondamentales du Traitement Automatique des Langues [TAL] que sont l'étiquetage morpho-syntaxique probabiliste et l'analyse syntaxique probabiliste. Dans le cadre du français, nous disposons d'une multitude de données lexicales et syntaxiques créées par des processus automatiques ou par des linguistes. De plus, un certain nombre d'expériences ont montré l'intérêt d'utiliser de telles ressources dans les processus probabilistes comme l'étiquetage ou l'analyse, car elles sont capables d'améliorer significativement les performances des systèmes. Dans ce mémoire, nous utilisons ces ressources afin de donner une réponse à deux problématiques que nous décrivons succinctement ci-dessous : la dispersion des données et la segmentation automatique des textes.

Grâce à des algorithmes d'analyse syntaxique de plus en plus évolués, les performances actuelles des analyseurs sont de plus en plus élevées, et ce pour de nombreuses langues dont le français. Cependant, il existe plusieurs problèmes inhérents aux formalismes mathématiques permettant de modéliser statistiquement cette tâche (grammaire, modèles discriminants,...). La dispersion des données est l'un de ces problèmes, et est causée principalement par la faible taille des corpus annotés disponibles pour la langue. La dispersion représente la difficulté d'estimer la probabilité de phénomènes syntaxiques apparaissant dans les textes à analyser mais qui sont rares ou absents du corpus ayant servi à l'apprentissage des analyseurs. De plus, il est prouvé que la dispersion est en partie un problème lexical, car plus la flexion d'une langue est importante, moins les phénomènes lexicaux sont représentés dans les corpus annotés. Notre première problématique repose donc sur l'atténuation de l'effet négatif de la dispersion lexicale des données sur les performances des analyseurs.

Dans cette optique, nous nous sommes intéressé à une méthode appelée *regroupement lexical*, et qui consiste à regrouper les mots du corpus et des textes en classes. Ces classes réduisent le nombre de mots inconnus et donc le nombre de phénomènes syntaxiques rares ou inconnus, liés au lexique, des textes à analyser. Notre objectif est donc de proposer des regroupements lexicaux à partir d'informations tirées des lexiques syntaxiques du français, et d'observer leur impact sur les performances d'analyseurs syntaxiques.

Par ailleurs, la plupart des évaluations concernant l'étiquetage morpho-syntaxique probabiliste et l'analyse syntaxique probabiliste ont été réalisées avec une segmentation parfaite du texte, car identique à celle du corpus évalué. Or, dans les cas réels d'application, la segmentation d'un texte est très rarement disponible et les segmenteurs automatiques actuels sont loin de proposer une segmentation de bonne qualité, et ce, à cause de la présence de nombreuses unités multi-mots (mots composés, entités nommées,...). Dans ce mémoire, nous nous focalisons sur les unités multi-mots dites *continues* qui forment des unités lexicales auxquelles on peut associer une étiquette morpho-syntaxique, et que nous appelons *mots composés*. Par exemple, *cordon bleu* est un nom composé, et *tout à fait* un adverbe composé. Nous pouvons assimiler la tâche de repérage des mots composés à celle de la segmentation du texte. Notre deuxième

problématique portera donc sur la segmentation automatique des textes français et son impact sur les performances des processus automatiques.
Pour ce faire, nous nous sommes penché sur une approche consistant à coupler, dans un même modèle probabiliste, la reconnaissance des mots composés et une autre tâche automatique. Dans notre cas, il peut s'agir de l'analyse syntaxique ou de l'étiquetage morpho-syntaxique. La reconnaissance des mots composés est donc réalisée au sein du processus probabiliste et non plus dans une phase préalable. Notre objectif est donc de proposer des stratégies innovantes permettant d'intégrer des ressources de mots composés dans deux processus probabilistes combinant l'étiquetage ou l'analyse à la segmentation du texte.

**Mots-clés:** Analyse syntaxique, Étiquetage morpho-syntaxique, Probabilités, Lexiques, Hybridation, Dispersion des données, Segmentation automatique

# Abstract

This thesis focuses on the integration of lexical and syntactic resources of French in two fundamental tasks of Natural Language Processing [NLP], that are probabilistic part-of-speech tagging and probabilistic parsing. In the case of French, there are a lot of lexical and syntactic data created by automatic processes or by linguists. In addition, a number of experiments have shown interest to use such resources in processes such as tagging or parsing, since they can significantly improve system performances. In this paper, we use these resources to give an answer to two problems that we describe briefly below : data sparseness and automatic segmentation of texts.

Through more and more sophisticated parsing algorithms, parsing accuracy is becoming higher for many languages including French. However, there are several problems inherent in mathematical formalisms that statistically model the task (grammar, discriminant models,...). Data sparseness is one of those problems, and is mainly caused by the small size of annotated corpora available for the language. Data sparseness is the difficulty of estimating the probability of syntactic phenomena, appearing in the texts to be analyzed, that are rare or absent from the corpus used for learning parsers. Moreover, it is proved that sparsness is partly a lexical problem, because the richer the morphology of a language is, the sparser the lexicons built from a treebank will be for that language. Our first problem is therefore based on mitigating the negative impact of lexical data sparseness on parsing performance.

To this end, we were interested in a method called *word clustering* that consists in grouping words of corpus and texts into clusters. These clusters reduce the number of unknown words, and therefore the number of rare or unknown syntactic phenomena, related to the lexicon, in texts to be analyzed. Our goal is to propose word clustering methods based on syntactic information from French lexicons, and observe their impact on parsers accuracy.

Furthermore, most evaluations about probabilistic tagging and parsing were performed with a perfect segmentation of the text, as identical to the evaluated corpus. But in real cases of application, the segmentation of a text is rarely available and automatic segmentation tools fall short of proposing a high quality segmentation, because of the presence of many multi-word units (compound words, named entities, ...). In this paper, we focus on *continuous* multi-word units, called *compound words*, that form lexical units which we can associate a part-of-speech tag. We may see the task of searching compound words as text segmentation. Our second issue will therefore focus on automatic segmentation of French texts and its impact on the performance of automatic processes.

In order to do this, we focused on an approach of coupling, in a unique probabilistic model, the recognition of compound words and another task. In our case, it may be parsing or tagging. Recognition of compound words is performed within the probabilistic process rather than in a preliminary phase. Our goal is to propose innovative strategies for integrating resources of compound words in both processes combining probabilistic tagging, or parsing, and text segmentation.

**Keywords:** Parsing, Part-Of-Speech Tagging, Probabilities, Lexicons, Hybridisation, Segmentation, Data sparseness

# Table des matières

## Partie II Stratégies d'exploitation de ressources lexicales et syntaxiques pour l'étiquetage morpho-syntaxique et l'analyse syntaxique

# Table des figures

# Introduction

## 1 Contexte

Notre étude porte sur l'intégration de ressources lexicales et syntaxiques dans deux tâches fondamentales du Traitement Automatique des Langues [TAL] que sont l'étiquetage morpho-syntaxique probabiliste et l'analyse syntaxique probabiliste. Dans le cadre de l'analyse syntaxique, deux modèles de la syntaxe sont majoritairement utilisés : structures de constituants et dépendances syntaxiques. Pour ces deux modèles, il existe une multitude de formalismes permettant de modéliser statistiquement le problème, comme par exemple les grammaires. Actuellement, les grammaires dites *hors-contexte* probabilistes permettent aux analyseurs d'obtenir les meilleures performances dans le cadre de l'analyse en constituants, et ce pour diverses langues dont le français (Crabbé & Candito, 2008; Candito & Crabbé, 2009). Il existe, en parallèle de ces grammaires, d'autres modélisations possibles du problème, et notamment les modèles discriminants (ou discriminatifs). Ce type de modélisation est mathématiquement plus complexe que les grammaires et permet de modéliser un plus grand nombre d'informations lexicales et syntaxiques. Ces modèles sont principalement utilisés pour l'analyse en dépendances (McDonald, 2006; Nivre, 2008; Candito *et al.*, 2010b).
Les grammaires, et, dans une moindre mesure, les modèles discriminants, souffrent de divers problèmes liés à leur formalisme mathématique (Jurafsky & Martin, 2000). La dispersion des données est l'un de ces problèmes, et représente la difficulté d'estimer la probabilité de phénomènes syntaxiques apparaissant dans les textes à analyser mais qui sont rares ou absents du corpus ayant servi à l'apprentissage des analyseurs. Ces phénomènes qui posent un problème de dispersion des données sont en général des phénomènes courants : les locuteurs en ont déjà entendu et produit des exemples innombrables, et ils en entendent et en produisent régulièrement. Ces phénomènes sont rares dans les corpus annotés actuellement utilisés, d'une taille sans commune mesure avec la masse de formes linguistiques à laquelle est exposé un locuteur. Notre premier objet d'étude portera donc sur la résolution, ou du moins l'atténuation de l'impact négatif de la dispersion des données sur les performances des analyseurs syntaxiques du français.
Par ailleurs, la plupart des évaluations concernant l'étiquetage morpho-syntaxique probabiliste et l'analyse syntaxique probabiliste ont été réalisées avec une segmentation parfaite du texte, car identique à celle du corpus évalué. Or, dans les cas réels d'application, la segmentation d'un texte est très rarement disponible et les segmenteurs automatiques actuels sont loin de proposer une segmentation de bonne qualité, et ce, à cause de la présence de nombreuses unités multi-

mots (mots composés, entités nommées,...). Notre deuxième objet d'étude portera donc sur la segmentation automatique des textes français et son impact sur les performances des processus automatiques.

# 2 Objectifs

Pour nos deux objets d'étude, l'objectif de cette thèse est de proposer des solutions pertinentes et performantes. Pour ce faire, nous allons nous aider de lexiques morphologiques et syntaxiques. Il a été prouvé, par de nombreuses expériences, que les performances de systèmes d'étiquetage et d'analyse pouvaient être significativement améliorées en introduisant des données lexicales et syntaxiques provenant de lexiques indépendants du corpus d'apprentissage. Pour l'étiquetage, on peut citer par exemple (Denis & Sagot, 2009, 2010) qui ont utilisé le lexique du français Le*fff* (Sagot, 2010) pour augmenter, entre autres, la couverture de l'étiqueteur afin d'améliorer l'annotation des mots inconnus. En effet, un mot inconnu du corpus ayant servi à l'apprentissage du modèle pourrait en revanche être potentiellement présent dans un lexique indépendant. Quant à l'analyse syntaxique, on peut citer (Collins & Singer, 1999; Deoskar *et al.*, 2009) qui ont amélioré la qualité de l'analyse grâce à l'ajout d'informations de sous-catégorisation, comme la distinction entre les compléments essentiels des verbes et les simples modifieurs. Dans la suite de ce mémoire, nous employerons le terme *lexique externe* pour désigner un lexique indépendant du corpus d'apprentissage.
Dans le consensus actuel du TAL, les expressions multi-mots forment des unités linguistiques aux comportements lexicaux, syntaxiques et/ou sémantiques particuliers (expressions figées et semi-figées, collocations, verbes à particule,...) (Sag *et al.*, 2002). Leur identification est donc cruciale avant toute analyse de plus haut niveau (sémantique, syntaxique,...). Dans ce mémoire, nous nous focalisons sur les unités multi-mots dites *continues* qui forment des unités lexicales auxquelles on peut associer une étiquette morpho-syntaxique, et que nous appelons *mots composés*. Par exemple, *cordon bleu* est un nom composé, et *tout à fait* un adverbe composé. Nous pouvons assimiler la tâche de repérage des mots composés à celle de la segmentation du texte. Une approche récente consiste à coupler, dans un même modèle, l'annotation automatique des mots composés avec un analyseur syntaxique probabiliste (Finkel & Manning, 2009; Green *et al.*, 2011). Ces derniers ont intégré la reconnaissance des mots composés au sein de la grammaire et non plus dans une phase préalable, et ont montré l'efficacité de cette approche dans le cadre du français. Il existe également une méthode plus ancienne consistant à coupler la segmentation du texte avec le processus d'étiquetage morpho-syntaxique. L'approche BIO (Ramshaw & Marcus, 1995) permet de réaliser cette tâche et repose sur l'annotation des unités minimales du texte avec un jeu d'étiquettes particulier. Ce jeu est composé d'étiquettes morpho-syntaxiques combinées à des étiquettes de délimitation des mots composés. À partir de ces travaux de référence, notre objectif est donc de proposer des stratégies innovantes permettant d'intégrer des ressources de mots composés dans deux processus probabilistes combinant l'étiquetage ou l'analyse à la segmentation du texte (chapitre 5).
Quant à la dispersion des données, de nombreuses expériences ont montré que ce problème était en partie un problème lexical, causé notamment par le degré de complexité plus ou moins

important de la flexion d'une langue (Crabbé & Candito, 2008; Candito & Crabbé, 2009). En effet, le corpus annoté d'une langue ne dispose que d'un lexique de taille limitée, généralement proportionnelle à la taille de ce corpus. Aussi, plus la flexion est importante, moins les phénomènes lexicaux sont représentés. Or, la prise en compte du lexique est un point crucial des algorithmes d'analyse syntaxique, car, pour obtenir un analyseur ayant des performances au niveau de l'état de l'art, il est nécessaire de modéliser des affinités lexicales et syntaxiques entre les mots d'une phrase. Pour résoudre cette problématique, nous nous sommes penché sur une méthode appelée *regroupement lexical* qui consiste à regrouper les mots du corpus et des textes en classes. Grâce à cette approche, le nombre de phénomènes rares ou inconnus, liés au lexique, que l'on pourrait rencontrer dans les textes à analyser diminue. Dans la littérature, les classes ont été préalablement calculées à partir d'informations lexicales (Koo *et al.*, 2008; Versley & Rehbein, 2009; Crabbé & Candito, 2008; Candito & Crabbé, 2009), syntaxiques (Sagae & Gordon, 2009), ou encore sémantiques (Agirre *et al.*, 2008, 2011). Dans le cadre du français, cette méthode a prouvé son efficacité (Candito & Crabbé, 2009; Candito & Seddah, 2010; Candito *et al.*, 2010b). Notre objectif est donc de proposer des regroupements lexicaux à partir d'informations tirées des lexiques syntaxiques du français, et d'observer leur impact sur les performances d'analyseurs syntaxiques (chapitre 6).

# 3 Plan de la thèse

La partie I intitulée **État de l'art** dresse, comme son nom l'indique, une vue d'ensemble de deux tâches fondamentales que sont l'étiquetage morpho-syntaxique et l'analyse syntaxique, dans leur version probabiliste. Cette partie contient les quatre chapitres suivants :

– Le chapitre 1 (**Corpus annotés**) présente les divers corpus annotés du français qui sont actuellement utilisés dans la plupart des expériences. Les deux modèles de la syntaxe nécessitent chacun un type de corpus ayant un schéma d'annotations propre.

– Le chapitre 2 (**Analyse syntaxique probabiliste**) évoque l'analyse syntaxique probabiliste. Pour les deux modèles de la syntaxe, nous présentons les différents algorithmes permettant de les traiter au mieux, et ce grâce à des formalismes probabilistes particulièrs (grammaires, modèles discriminants,...). Nous indiquons et discutons des performances de ces algorithmes sur le français. Nous évoquons également les problèmes et les limites de ces algorithmes, et notamment l'un de nos deux objets d'étude, à savoir la dispersion des données.

– Le chapitre 3 (**Étiquetage morpho-syntaxique**) évoque l'étiquetage morpho-syntaxique probabiliste. Comme pour l'analyse syntaxique, nous présentons les divers algorithmes permettant de réaliser cette tâche. Nous nous intéresserons notamment à des expériences effectuées sur diverses langues, dont le français, qui ont consisté à intégrer des données provenant de ressources lexicales externes. Par ailleurs, la plupart des évaluations ont été réalisées avec une segmentation parfaite du texte, car identique à celle du corpus évalué (idem pour l'analyse syntaxique). Ce constat nous a amené à nous pencher sur notre deuxième objet d'étude,

à savoir la segmentation automatique d'un texte à travers l'identification automatique des unités multi-mots.

– Dans le chapitre 4 (**Ressources lexicales et syntaxiques**), nous présentons les lexiques syntaxiques du français que nous utilisons dans nos expériences. Pour chaque lexique, nous décrivons le principe général, l'architecture ainsi que les différentes informations codées pour chaque entrée. Il s'agit de déterminer les informations potentiellement utiles et exploitables pour traiter au mieux nos deux objets d'étude.

La partie II intitulée **Stratégies d'exploitation de ressources lexicales et syntaxiques pour l'étiquetage morpho-syntaxique et l'analyse syntaxique** se penche sur les solutions apportées aux deux objets d'étude. Cette partie contient les deux chapitres suivants :

– Dans Le chapitre 5 (**Les unités multi-mots dans l'étiquetage morpho-syntaxique et l'analyse syntaxique**), nous proposons et discutons des résultats obtenus par plusieurs stratégies innovantes permettant de combiner la phase de segmentation d'un texte avec l'étiquetage ou l'analyse syntaxique. Afin d'améliorer les performances de ces combinaisons, nous nous sommes aidé de ressources lexicales traitant des unités multi-mots.

– Dans le chapitre 6 (**Analyse syntaxique et algorithmes de regroupements lexicaux basés sur les lexiques**), nous tentons de répondre au problème de la dispersion des données en explorant l'approche basée sur les regroupements lexicaux. Nous commencerons par présenter les solutions existantes. Et, en partant de ces travaux de référence, nous proposerons nos propres classes lexicales calculées à partir des lexiques syntaxiques du français. Nous discuterons des résultats obtenus par l'intégration de ces classes dans divers systèmes d'analyse syntaxique probabiliste.

Nous finirons ce mémoire par une conclusion et des perspectives.

# Première partie

# État de l’art

# 1

# Corpus annotés

## 1.1 Introduction

Les corpus de textes sont de plus en plus utilisés dans le cadre du traitement des langues naturelles. En revanche, seule une petite portion des corpus disponibles sont actuellement annotés au niveau syntaxique. En effet, les meilleurs analyseurs syntaxiques probabilistes nécessitent un corpus annoté afin d'être capable de modéliser statistiquement le problème, mais également, de pouvoir par la suite effectuer des évaluations pertinentes[1]. Cependant, les corpus annotés ont un coût de création (humain et pécunier) non négligeable ce qui explique pourquoi il y a aujourd'hui peu de langues qui disposent de corpus annotés mais également pourquoi ces corpus sont souvent de petite taille.
Il existe plusieurs modèles de la syntaxe différents qui nécessitent chacun un type de corpus ayant un schéma d'annotations propre. Dans le cadre de l'analyse syntaxique probabiliste, deux modèles de la syntaxe sont très utilisés actuellement, l'un basé sur des corpus annotés en constituants, et l'autre sur des corpus annotés en dépendances. Les corpus dits *en constituants* contiennent des arbres de constituants syntaxiques (syntagmes nominaux, groupes verbaux,...) exploitables uniquement par des modèles probabilistes ayant un formalisme dérivé de ces constituants. Parmi les nombreux corpus de ce type, on peut citer le Penn Treebank [PTB] pour l'anglais (Marcus *et al.*, 1994), le Tiger Treebank pour l'allemand (Brants *et al.*, 2002) ou encore le French Treebank pour le français [FTB] (Abeillé *et al.*, 2003). Quant aux corpus dits *en dépendances*, ils contiennent des arbres annotés en dépendances syntaxiques (structures prédicat-arguments) et exploitables par des analyseurs en dépendances uniquement. La représentation des phrases sous forme d'arbre de dépendances est souvent utilisée dans les tâches comme l'extraction d'informations ou encore les systèmes de questions-réponses. Il existe plusieurs solutions pour représenter ces dépendances, parmi lesquelles nous en citons deux. Une première solution consiste à modéliser des relations entre syntagmes de premier

[1]Nous verrons dans la section 2.3 que certains algorithmes d'apprentissage ne font pas usage d'un corpus annoté, mais cette spécificité se répercute sur leurs performances, plus faibles.

niveau [2]. Pour le français, on peut citer les corpus des projets EAsy (Paroubek P., 2005)[3] et Passage[4] (Villemonte De La Clergerie *et al.*, 2008). Une deuxième solution est basée sur des relations gouverneur-dépendants entre les mots de la phrase.
Actuellement, la plupart des meilleurs algorithmes probabilistes d'analyse en dépendances ont adopté la deuxième solution (Nivre *et al.*, 2007). Or, de nombreuses langues dont le français et l'anglais ne disposent pas de corpus de ce type. La conversion des corpus des projets EAsy et Passage pourrait être une solution à envisager mais la tâche est rendue complexe par leur schéma d'annotation particulier (Candito *et al.*, 2010a). Pour éviter de créer un corpus de toutes pièces, de nombreux travaux se sont penchés sur la conversion automatique des corpus en constituants vers les corpus en dépendances, ce qui est rendu possible par des informations comme les fonctions syntaxiques. Par exemple, le PTB-DEP (Johansson & Nugues, 2007) et le FTB-DEP (Candito *et al.*, 2010a) sont respectivement les versions en dépendances du PTB et du FTB.
Dans la section 1.2, nous verrons en détail le schéma d'annotation du corpus du français FTB, ainsi que les corpus dérivés de celui-ci actuellement utilisés dans la plupart des expériences réalisées sur le français. Dans la section 1.3, nous décrivons la procédure de conversion du FTB vers sa version en dépendances, FTB-DEP.

## 1.2 French Treebank, un corpus arboré du français

Le corpus arboré du français appelé French Treebank (Abeillé *et al.*, 2003), a été rendu disponible sous licence à partir de 2003[5]. Ce corpus, au format XML, contient 20648 phrases (pour 580 945 mots) provenant d'articles du journal Le Monde. Dans ce corpus, chaque phrase est annotée par un arbre syntaxique en constituants et les annotations sont à la fois morphologiques et syntaxiques. La figure 1.1 montre l'arbre syntaxique d'une phrase extraite du FTB, *Aujourd'hui toutes doivent pouvoir prétendre à l'excellence.*
Un arbre syntaxique en constituants est composé de trois types de noeuds. Les noeuds terminaux sont les feuilles de l'arbre et correspondent aux tokens de la phrase[6]. Les noeuds préterminaux sont les étiquettes morpho-syntaxiques associées aux tokens et chaque noeud préterminal possède un unique noeud terminal fils. Quant aux noeuds non-terminaux, ils ont pour valeur une étiquette syntaxique. Sur l'exemple précédent, *toutes* est un token de la phrase et donc la valeur d'un terminal de l'arbre. Son noeud préterminal père a pour valeur l'étiquette morpho-syntaxique `PRO-ind-3fp`. Le père de ce noeud préterminal est un non-terminal

[2]Les syntagmes de premier niveau, aussi appelés *chunks* en anglais, regroupent les mots de la phrase en séquences complexes (groupes nominaux, groupes verbaux,...) sans conduire à la création d'un arbre. La reconnaissance de ces unités est généralement réalisée par une analyse syntaxique dite *de surface*.

[3]Le corpus produit pour la campagne EAsy est composé d'environ 400000 mots provenant de textes de différents genres (journalistique, littéraire,...). Une description complète du projet est disponible à l'adresse `http://archives.limsi.fr/RS2005/chm/lir/lir11/`

[4]Le corpus Passage est une extension du corpus généré par EAsy. une description complète du projet est disponible à l'adresse `http://atoll.inria.fr/passage`

[5]`http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php`

[6]On considère qu'un mot de la phrase est assimilé à un token, e.g. une séquence de caractères alphanumériques.

```
<SENT>
    <w compound="yes" lemma="aujourd'hui" ei="ADV" ee="ADV" cat="ADV">
        <w catint="P">aujourd'</w>
        <w catint="N">hui</w>
    </w>
    <NP fct="SUJ">
        <w cat="PRO" ee="PRO-ind-3fp" ei="PROfp" lemma="tout" mph="3fp" subcat="ind">toutes</w></NP>
        <VN>
            <w cat="V" ee="V--P3p" ei="VP3p" lemma="devoir" mph="P3p" subcat="">doivent</w>
        </VN>
        <VPinf fct="OBJ">
            <VN>
                <w cat="V" ee="V--W" ei="VW" lemma="pouvoir" mph="W" subcat="">pouvoir</w>
            </VN>
            <VPinf fct="OBJ">
                <VN>
                    <w cat="V" ee="V--W" ei="VW" lemma="prétendre" mph="W" subcat="">prétendre</w>
                </VN>
                <PP fct="A-OBJ"> <w cat="P" ee="P" ei="P" lemma="à">à</w>
                    <NP>
                        <w cat="D" ee="D-def-fs" ei="Dfs" lemma="le" mph="fs" subcat="def">l'</w>
                        <w cat="N" ee="N-C-fs" ei="NCfs" lemma="excellence" mph="fs" subcat="C">excellence</w>
                    </NP>
                </PP>
            </VPinf>
        </VPinf>
    <w cat="PONCT" ee="PONCT-S" ei="PONCTS" lemma="." subcat="S">.</w>
</SENT>
```

FIG. 1.1: Exemple de phrase annotée par son arbre de constituants dans le FTB.

ayant pour valeur l'étiquette syntaxique `NP`. En ce qui concerne la description des étiquettes morpho-syntaxiques, le FTB fournit les traits flexionnels (champ `mph`) ainsi que le lemme (champ `lemma`) pour tous les tokens du corpus. De plus, un champ `subcat` indique une sous-catégorisation possible pour le token. Par exemple, pour un déterminant, la sous-catégorisation indique s'il est défini ou indéfini. Pour un nom, le champ peut préciser s'il est commun ou propre. Il existe 34 valeurs différentes de sous-catégorisation dans le FTB.
Le tableau 1.1 indique les différentes valeurs d'étiquettes syntaxiques utilisées pour annoter les noeuds non-terminaux des arbres (12 étiquettes) ainsi que les possibles catégories grammaticales d'une étiquette morpho-syntaxique (13 catégories). Une partie du corpus, composée de 9357 phrases, a été annotée en fonctions syntaxiques. Ces fonctions sont liées aux étiquettes syntaxiques décrites précédemment. Le tableau 1.1 recense également l'ensemble des fonctions syntaxiques utilisées dans le FTB. Quant aux mots composés, ils sont explicitement annotés et environ 15% des tokens du corpus font partie d'un mot composé. Les différents types de mots composés présents dans le FTB sont indiqués dans le tableau 1.2. On peut noter que leur structure interne n'est pas analysée syntaxiquement. Ainsi, tous les mots internes au mot composé sont frères et leur père est la catégorie grammaticale du mot composé. La représentation des mots composés dans le FTB est illustrée par la figure 1.2 avec le mot *petit écran*. D'un point de vue critique, nous avons remarqué que le corpus ne couvre pas la reconnaissance de tous les mots composés. Tout d'abord, certains déterminants ou certaines entités nommées ne sont pas identifiés, comme les déterminants nominaux (*un certain nombre de mesures*), les dates (*31 décembre 1992*), les noms de personne (*Jacques Chirac*), ou encore les adresses postales (*rue de la Liberté*). Par ailleurs, de nombreux noms composés sont manquants (*le grand public, les chaînes hôtelières*).
Par la suite, un certain nombre de critiques ont été émises sur la difficulté de réaliser des tâches d'analyses syntaxiques en prenant le FTB comme référence (Arun & Keller, 2005; Schluter &

| Étiquette syntaxique | Signification | Catégorie grammaticale | Signification | Fonction syntaxique | Signification |
|---|---|---|---|---|---|
| AP | syntagme adjectival | N | nom | ATO | attribut objet |
| AdP | syntagme adverbial | A | adjectif | ATS | attribut sujet |
| COORD | phrase coordonnée | ADV | adverbe | A-OBJ | à-objet |
| NP | syntagme nominal | V | verbe | DE-OBJ | de-objet |
| Sint | proposition conjuguée interne | P | préposition | MOD | modifieur |
| Srel | proposition relative | D | déterminant | OBJ | objet |
| Ssub | proposition subordonnée | C | conjonction | P-OBJ | prép.-objet |
| VN | noyau verbal | CL | clitique | SUJ | sujet |
| PP | syntagme prépositionnel | PONCT | ponctuation | | |
| VPinf | proposition infinitive | PRO | pronom | | |
| VPpart | proposition participiale | I | interjection | | |
| SENT | phrase indépendante | ET | mot étranger | | |
| | | PREF | préfixe | | |

TAB. 1.1: Ensemble des jeux d'annotations utilisé dans le FTB.

| Type de mot composé | Exemple |
|---|---|
| métonymie figée | *petit écran* |
| syntaxe non régulière | *couvre-chef* |
| expression verbale | *porter préjudice* |
| entité nommée | *Le Monde* |
| semi-figé ou figé | *le jour du Seigneur* |
| mots internes non isolables | *aujourd'hui* |

TAB. 1.2: Types de mots composés présents dans le FTB.

Genabith, 2007). Ces critiques sont fondées sur deux observations : d'une part, sur le fait que la structure hiérarchique du FTB a un aspect très plat contrairement à de nombreux corpus dont le PTB, et d'autre part, que de multiples erreurs apparaissent dans le corpus. Structurellement d'abord, le FTB est plus plat que le PTB. Par exemple, il n'y a pas de noeud `VP` (syntagme verbal) mais seulement des noeuds `VN` qui regroupent uniquement le noyau verbal, les clitiques, les adverbes, les modifieurs et les négations. La figure 1.3 illustre cette différence avec la phrase *Marc joue dans le couloir*. De même, les syntagmes nominaux sont relativement plats car ils regroupent le noyau nominal ainsi que les déterminants et adjectifs. Ces critiques ont débouché sur la création de plusieurs corpus, dérivés de l'original, ayant subi des modifications structurelles et des normalisations (Arun & Keller, 2005; Schluter & Genabith, 2007). Récemment, (Seddah *et al.*, 2009b) ont prouvé que la platitude du corpus n'implique pas obligatoirement une difficulté pour la tâche d'analyse syntaxique. Les expériences qu'ils ont réalisées sur le FTB ont permis de montrer que plusieurs analyseurs syntaxiques sont plus performants sur le corpus original que sur les dérivés.

Au sujet des erreurs du corpus, on peut signaler deux problèmes récurrents. Le premier concerne les directives d'annotation qui ne sont pas respectées à divers endroits du corpus, ce qui conduit à créér de l'ambiguité structurelle pour un même phénomène. Par exemple, les directives d'an-

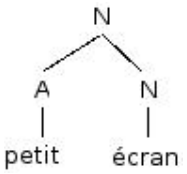
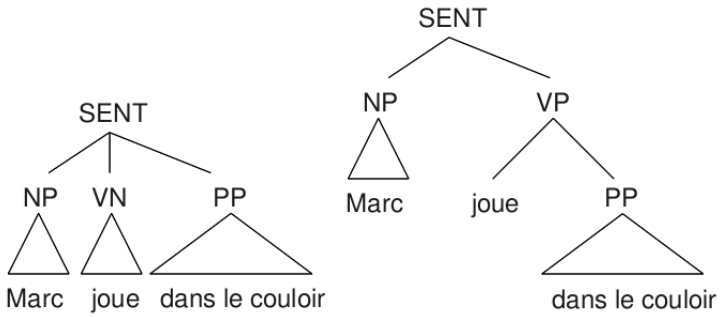



FIG. 1.2: Représentation du mot composé *petit écran* dans le FTB.




(a) FTB (b) PTB

FIG. 1.3: Exemple de différence structurelle entre le PTB et le FTB.

notation nous disent que s'il y a ellipse du sujet[7] alors il ne peut y avoir de constituant `Sint`, or, de nombreuses fois, cette règle n'est pas appliquée. Ce problème est illustré par l'exemple ci-dessous. La phrase interne *aura beaucoup varié en cours d'année* est annotée `Sint`, et ceci par erreur car il y a ellipse du sujet pour le verbe *avoir* :

*En 1992, le taux de croissance a été globalement nul mais (Sint aura beaucoup varié en cours d'année).*

Le deuxième type d'erreur concerne les oublis d'annotation des étiquettes morpho-syntaxiques, principalement pour les mots internes aux mots composés. Actuellement, la plupart des travaux menés sur le français se basent sur une version partiellement corrigée du FTB original, appelée FTB-UC, qui comporte 12351 phrases pour 350931 tokens (Candito & Crabbé, 2009). Dans ce corpus, certains mots composés, ayant un schéma syntaxique régulier, ont été défaits automatiquement en sous arbres contenant les mots simples internes au mot composé. La figure 1.4 montre un exemple de mot composé défait (*petit écran*). Les mots composés restants ont été fusionnés en une unité simple, les mots internes étant séparés par un trait de soulignement. Par exemple, l'adverbe composé *y compris* devient *y_compris*. Défaire les mots composés permet de réduire de 6125 à 3053 le nombre de mots composés distincts dans le corpus.
Selon nous, ce processus automatique fait des choix tout à fait discutables. D'une part, seuls les 200 mots composés les plus fréquents du corpus ont été défaits. Cela veut tout simplement dire qu'un mot composé du FTB ayant un schéma syntaxique régulier est considéré comme

[7]L'ellipse consiste à omettre certains mots. Dans le cas d'une ellipse du sujet, le sujet du verbe n'est pas indiqué.

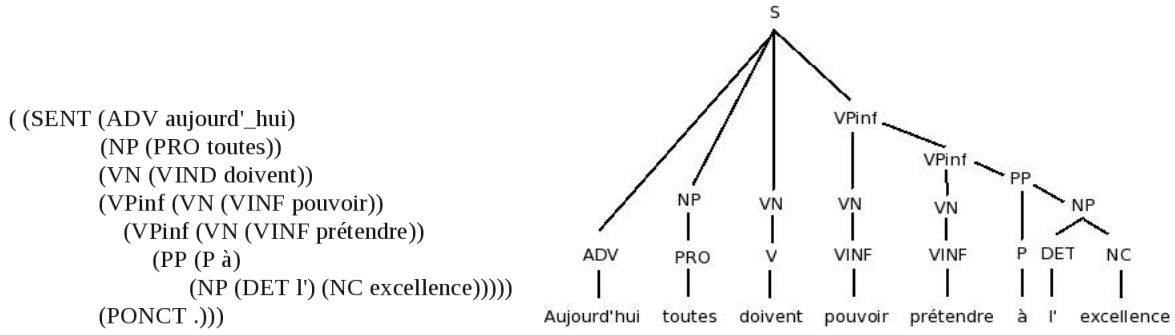
composé uniquement s'il apparaît moins de fois qu'un palier de fréquence fixé manuellement. D'autre part, le corpus étant de petite taille, ce processus a été appliqué principalement dans le but d'augmenter artificiellement le vocabulaire connu afin d'effectuer des expériences d'analyse syntaxique probabiliste. Dans le cas de nos propres expériences, nous devons nous comparer aux résultats obtenus sur ce corpus. Or, d'un point de vue linguistique, il n'est pas logique de reconnaître certains mots composés plutôt que d'autres, surtout en se basant sur un palier de fréquence.


FIG. 1.4: Mot composé *petit écran* défait. A gauche, dans le FTB et à droite, dans le FTB-UC.

### 1.2.1 Format parenthésé

Comme nous l'avons énoncé précédemment, le FTB (toutes versions confondues) est un corpus au format XML. Or, ce format n'est pas le format utilisé pour l'entraînement de la plupart des analyseurs syntaxiques. Le format standard est une structure arborée, appelé format parenthésé. Un mot et son étiquette morpho-syntaxique sont représentés sous la forme d'un tuple `(étiquette mot)`. Les noeuds syntaxiques sont représentés sous la forme `(étiquette X)`, avec $X$ une séquence de paires mot-étiquette et/ou de noeuds syntaxiques. Par convention, un noeud racine ayant une étiquette vide est ajouté à l'arbre. La figure 1.5 montre l'analyse syntaxique de la phrase *Aujourd'hui toutes doivent prétendre à l'excellence.* au format parenthésé et sa représentation graphique arborée correspondante.
La plupart des travaux menés sur l'analyse syntaxique du français se basent sur une instance du FTB-UC au format parenthésé. Cette instance a la particularité d'avoir un jeu de 28 étiquettes morpho-syntaxiques. Ces étiquettes sont la composition des catégories grammaticales avec certains traits flexionnels (`mph`) et quelques champs de sous-catégorisation (`subcat`). Par exemple, l'étiquette `CS` (conjonction de subordination) est la composition de la catégorie grammaticale `C` (conjonction) avec le trait de sous-catégorisation `S` (de subordination). Ce jeu d'étiquettes est le résultat d'expériences visant à optimiser les performances de certains analyseurs syntaxiques (Crabbé & Candito, 2008). L'ensemble du jeu est montré dans le tableau 1.3.

Afin de mener des expériences sur ce corpus et évaluer les résultats, la procédure standard utilisée dans la plupart des travaux consiste à découper ce corpus en trois parties distinctes selon la répartition suivante, 80% pour la partie apprentissage, 10% pour la partie développement et 10% pour la partie évaluation. La partie apprentissage est utilisée pour entraîner un processus stochastique. La partie développement permet d'évaluer et améliorer ce processus d'après les erreurs produites. Et enfin, la partie évaluation permet de déterminer les performances du processus sur un texte qui n'a pas été utilisé pour le développement de ce processus. Les ca-




(a) Format parenthésé (b) Arbre correspondant

FIG. 1.5: Analyse de la phrase *Aujourd'hui toutes doivent prétendre à l'excellence.* : (a) au format parenthésé dans le FTB-UC (b) sa représentation graphique arborée correspondante.

| Étiquette | Signification | Étiquette | Signification |
|---|---|---|---|
| V | verbe au mode indicatif | CLS | clitique sujet |
| VIMP | verbe au mode impératif | CLO | clitique objet |
| VINF | verbe au mode infinitif | CLR | clitique réfléchi |
| VS | verbe au mode subjonctif | P | préposition |
| VPP | verbe au participe passé | P+D | prép-déterminant |
| VPR | verbe au participe présent | P+PRO | prép-pronom |
| NPP | nom propre | I | interjection |
| NC | nom commun | PONCT | ponctuation |
| CS | conjonction de subordination | ET | mot étranger |
| CC | conjonction de coordination | ADJWH | adjectif interrogatif |
| ADJ | adjectif | ADVWH | adverbe interrogatif |
| ADV | adverbe | PROWH | pronom interrogatif |
| PROREL | pronom relatif | PRO | pronom |
| DETWH | déterminant interrogatif | DET | déterminant |

TAB. 1.3: Jeu d'étiquettes présent dans l'instance du FTB-UC.

ractéristiques de chaque partie sont indiquées dans le tableau 1.4. Les colonnes `#phrases` et `#phrases<40` indiquent respectivement le nombre total de phrases du corpus et le nombre de phrases inférieures ou égales à 40 mots. `#mots` est le nombre total de mots du corpus, et `#mots inconnus` est le nombre total de mots inconnus. Les mots inconnus sont des mots présents dans les phrases à analyser mais qui n'apparaissent pas dans le corpus d'apprentissage. Un modèle stochastique entraîné sur ce dernier ne dispose donc d'aucune information sur ces mots (étiquette morpho-syntaxiques, lemme,...). Or, l'enjeu est important puisqu'ils représentent environ 6% du total des mots des corpus de développement et d'évaluation. Nous verrons par la suite que de nombreux mécanismes permettent de traiter efficacement les problèmes liés à ce type de mot.

Dans les expériences décrites dans ce mémoire, nous avons fait le choix d'utiliser ce corpus et cette partition particulière afin de pouvoir nous comparer aisément aux travaux réalisés précédemment sur l'analyse syntaxique en constituants et l'étiquetage morpho-syntaxique du fran-

| Corpus | #phrases | #phrases<40 | #mots | #mots inconnus |
|---|---|---|---|---|
| Apprentissage | 9881 | - | 287662 | - |
| Développement | 1235 | 951 | 31729 | 1807 (5,7%) |
| Évaluation | 1235 | 969 | 31540 | 1674 (5,3%) |

TAB. 1.4: Caractéristiques de la partition du corpus FTB-UC

çais.

## 1.3 French Treebank en dépendances

Comme énoncé dans l'introduction, de nombreux corpus en dépendances sont simplement le résultat d'une conversion d'un corpus en constituants. En ce qui concerne le PTB, on peut trouver plusieurs méthodes de conversion (Collins, 1999; Yamada & Matsumoto, 2003; Johansson & Nugues, 2007) dont la plus utilisée est implémentée dans l'outil Penn2Malt[8] (Nivre, 2006). Cependant, toutes ces méthodes sont généralement dérivées d'un même algorithme basé sur des règles de propagation des têtes lexicales (Magerman, 1995). La tête lexicale est le mot noyau d'un syntagme, c'est-à-dire celui qui donne le sens principal au syntagme. Pour les syntagmes nominaux, il s'agit souvent d'un nom, comme dans les deux cas suivants :
– *un été chaud* a pour tête lexicale le nom commun *été*.
– *les ouvriers du bâtiment* a pour tête lexicale le nom commun *ouvriers*.
En se basant également sur cet algorithme, (Candito *et al.*, 2010a) proposent une procédure de conversion du corpus en constituants FTB-UC vers un format en dépendances. Cette procédure est constituée de trois étapes successives ayant comme finalité la création d'un corpus en dépendances appelé FTB-DEP :

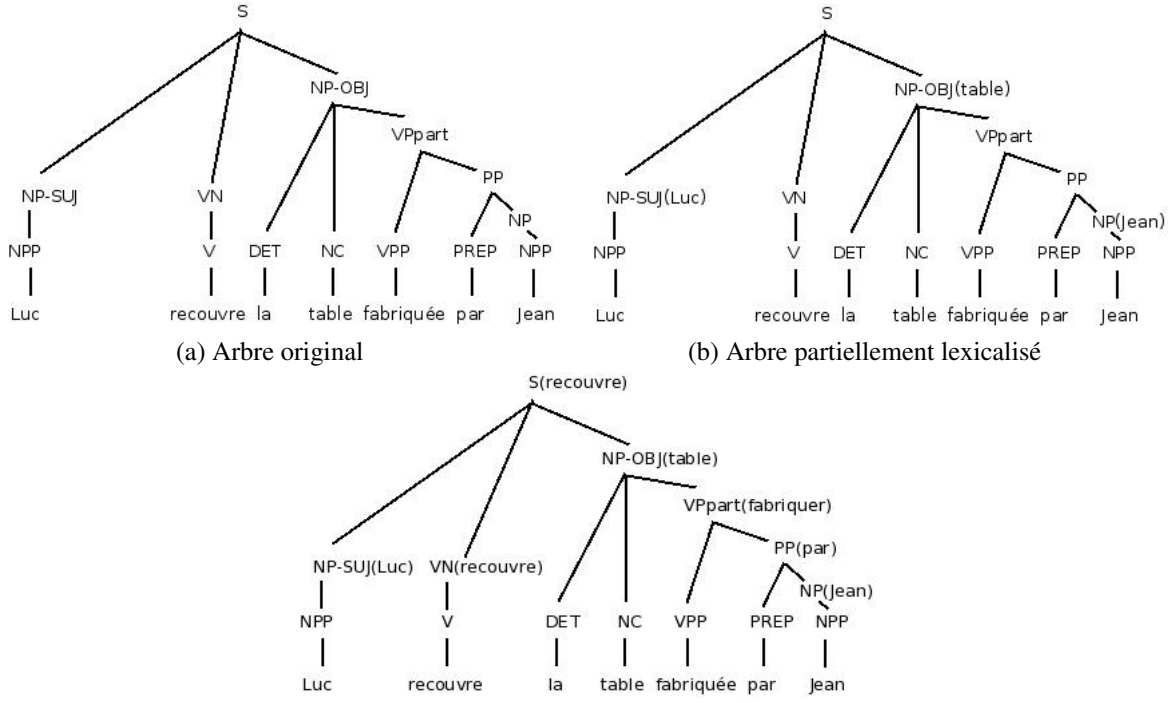
1. Les règles de propagation des têtes ont pour but premier l'identification de la tête syntaxique de chaque noeud de l'arbre. La tête syntaxique d'un constituant est l'élément (le noeud fils) qui détermine la valence passive de ce constituant, c'est-à-dire l'élément qui contrôle la distribution de ce constituant (Kahane, 2001). Puis, la tête lexicale de la tête syntaxique est propagée dans le noeud père. La règle suivante, simplifiée pour l'exemple et sous forme d'expression régulière, indique que la tête syntaxique (symbole $\star$) du noeud père de symbole `NP` est le noeud fils d'étiquette `NC` ou `NPP` pouvant être précédé d'un déterminant et possiblement suivi d'un syntagme prépositionnel :

$$NP \rightarrow DET?[NC|NPP]^{\star}PP? \tag{1.1}$$

   Ces règles sont appliquées sur chaque noeud des arbres du FTB-UC afin de les annoter par leur tête lexicale. Pour ce faire, (Candito *et al.*, 2010a) se basent sur une version améliorée de l'ensemble des règles décrit dans (Arun & Keller, 2005).
   La figure 1.6b montre l'arbre syntaxique partiellement lexicalisé de la phrase *Luc recouvre la table fabriquée par Jean* après l'application de la règle 1.1 sur l'arbre original de cette

[8] `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`

phrase, montré figure 1.6a. On peut voir que la règle a pu être appliquée avec succès sur trois syntagmes nominaux ayant pour têtes *Luc, table* et *Jean*.




(a) Arbre original

(b) Arbre partiellement lexicalisé

(c) Arbre lexicalisé

FIG. 1.6: (a) Arbre de la phrase *Luc recouvre la table fabriquée par Jean*. (b) Application de la règle de propagation des têtes 1.1. (c) Arbre lexicalisé par l'ensemble des règles.

2. Ensuite, les dépendances bilexicales (un dépendant associé à un gouverneur) sont extraites de chaque arbre en se basant sur les têtes lexicales ajoutées à l'étape précédente. Dans l'arbre en constituants, si le noeud correspondant au dépendant est annoté par une fonction syntaxique alors cette fonction est utilisée de nouveau pour caractériser la relation de dépendance extraite. En considérant que le verbe *recouvre* est la tête lexicale du noeud de symbole `S` (tête de la phrase), la relation suivante peut être extraite à partir de l'arbre de la figure 1.6c :

$$recouvre_{(gouverneur)} \overset{SUJ}{\Longleftrightarrow} Luc_{(dépendant)} \tag{1.2}$$

   En effet, *recouvre* étant le gouverneur (verbal) de la phrase, les têtes lexicales des noeuds fils de `S` sont toutes dépendantes de celui-ci.
3. Les fonctions syntaxiques présentes dans le FTB-UC ne sont utilisées que pour modéliser les relations de dépendance entre un gouverneur verbal et ses dépendants. Après l'étape précédente, il reste donc des dépendances non annotées par une fonction syntaxique. Aussi, (Candito *et al.*, 2010a) se basent sur un ensemble d'heuristiques afin d'annoter ces dépendances. Elles sont appliquées notamment sur les coordinations, les relations à gouverneur autre que verbal ou encore les dépendants qui ne projetent pas une phrase (proposition participiale, infinitive,...). Par exemple, dans l'arbre de la figure 1.6a, le noeud ayant pour symbole `PP` et représentant le syntagme prépositionnel *par Jean* n'est associé à aucune fonction

syntaxique car il est inclus dans une proposition participiale (symbole `VPpart`). Une heuristique l'annote comme `P-OBJ` (préposition-objet) dépendant du verbe au participe passé *fabriquée*.

Quelques cas particuliers requièrent des choix qui ne peuvent être complètement induits à partir des structures arborées du corpus. Pour les résoudre, des choix arbitraires ont été faits parmi lesquels[9] :

– Les auxiliaires sont considérés comme des dépendants du verbe à l'infinitif ou au participe passé qu'ils introduisent.
– Les prépositions sont considérées comme les têtes lexicales du complément qu'elles introduisent.
– Pour les structures coordonnées, la tête syntaxique est le premier conjoint.

La figure 1.7 montre l'arbre entièrement lexicalisé et annoté en fonctions syntaxiques de la phrase *Luc recouvre la table fabriquée par Jean*.

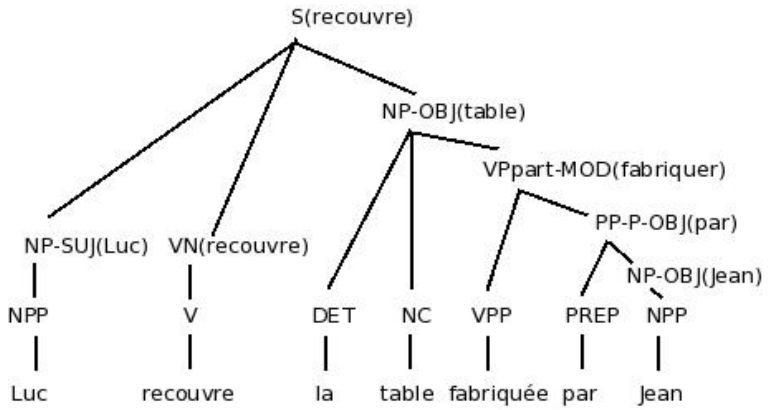



FIG. 1.7: Arbre entièrement lexicalisé de la phrase *Luc recouvre la table fabriquée par Jean*

Les dépendances bilexicales extraites de chaque arbre en constituants peuvent être représentées sous la forme d'un arbre de dépendances. Ce type d'arbre particulier modélise les relations entre les gouverneurs et leurs dépendants, chaque relation étant représentée par un arc et annotée par une fonction syntaxique. Par exemple, l'arbre de la figure 1.8 est l'arbre en dépendances de la phrase *Luc recouvre la table fabriquée par Jean*. On peut voir que les mots *Luc* et *table* sont bien reliés au gouverneur verbal *recouvre* par les relations `SUJ` et `OBJ`. Afin d'être exploités par des processus automatiques, les arbres de dépendances sont finalement convertis dans le format CoNLL[10] qui est un format standard utilisé par la plupart des analyseurs en dépendances probabilistes.

Selon ce format, chaque ligne correspond à un mot d'un arbre de dépendances et chaque arbre est séparé par une ligne vide. De plus, un ensemble de propriétés est assigné à chaque mot sous forme de colonnes (une colonne par propriété). Les propriétés communes à la plupart des corpus en dépendances sont le mot, son gouverneur (en général l'indice de ce mot dans la phrase) et la fonction syntaxique. (Candito *et al.*, 2010a) ont ajouté d'autres propriétés comme

[9]Une description complète du schéma d'annotation est disponible à l'adresse `http://www.linguist.univ-paris-diderot.fr/~mcandito/Rech/FTBDeps/GuideDepSurface.pdf`

[10]`http://nextens.uvt.nl/~conll`

le lemme du mot, des éventuelles propriétés morphologiques (genre, nombre,...) ainsi que deux étiquettes morpho-syntaxiques, l'une étant tiré du jeu d'étiquettes du FTB-UC et une autre plus grossière provenant du jeu du FTB.
La figure 1.9 montre l'arbre de dépendances de la figure 1.8 au format CoNLL avec les propriétés définies par (Candito *et al.*, 2010a). Dans cet exemple, le mot *fabriquée* est relié au gouverneur à l'indice 4 de la phrase, *table* (colonne `ID_Gouv`), par une relation `MOD` (colonne `Fct_Synt`). De plus, ce mot a pour étiquettes `VPP` et `V` (colonnes $\texttt{T}_{FTB-UC}$ et $\texttt{T}_{FTB}$) et certains traits morphologiques (colonne `Prop_Morph`). Par convention, le gouverneur de la phrase, ici *recouvre*, est dépendant d'un noeud fictif dans une relation appelée `ROOT`.

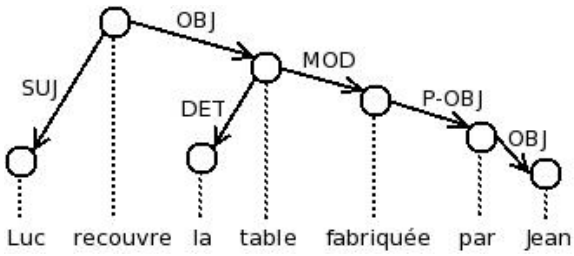



FIG. 1.8: Arbre de dépendances de la phrase *Luc recouvre la table fabriquée par Jean*

| Indice | Mot | Lemme | ID_Gouv | Fct_Synt | $\texttt{T}_{FTB-UC}$ | $\texttt{T}_{FTB}$ | Prop_Morph |
|---|---|---|---|---|---|---|---|
| 1 | Luc | Luc | 2 | SUJ | NPP | N | g=m\|n=s\|s=p |
| 2 | recouvre | recouvrir | 0 | ROOT | V | V | m=ind\|n=s\|p=3\|t=pst |
| 3 | la | le | 4 | DET | DET | D | g=f\|n=s\|s=def |
| 4 | table | table | 2 | OBJ | NC | N | g=f\|n=s\|s=c |
| 5 | fabriquée | fabriquer | 4 | MOD | VPP | V | g=f\|m=part\|n=s\|t=past |
| 6 | par | par | 5 | P-OBJ | P | P | - |
| 7 | Jean | Jean | 6 | OBJ | NPP | N | g=m\|n=s\|s=p |

FIG. 1.9: Arbre de dépendances au format CoNLL de la phrase *Luc recouvre la table fabriquée par Jean*

Pour les mêmes raisons que pour le corpus FTB-UC, nous avons fait le choix d'utiliser le corpus FTB-DEP pour toutes les expériences en rapport avec l'analyse syntaxique en dépendances du français.

# 2

# Analyse syntaxique probabiliste

## 2.1 Introduction

Dans le cadre de l'analyse syntaxique, deux modèles de la syntaxe sont majoritairement utilisés : structures de constituants et dépendances syntaxiques. Pour ces deux modèles, il existe une multitude d'algorithmes permettant de modéliser statistiquement le problème. Les grammaires sont une de ces possibilités, et sont principalement utilisées pour l'analyse en constituants. On peut distinguer plusieurs types de grammaires. Les grammaires de type *génératif* dont les représentants sont les grammaires hors-contexte CFG[11] et leur version probabiliste PCFG[12]. Quant aux grammaires *d'unification*, on peut en citer quelques-unes comme les grammaires catégorielles[13], les grammaires dirigées par les têtes[14], ou encore les grammaires lexicales fonctionnelles[15]. Parmi cette multitude d'algorithmes, les PCFG sont un des plus utilisés, et ceci pour deux raisons principales. Tout d'abord, des hypothèses d'indépendances entre les variables de la grammaire sont posées, et conduisent donc à une modélisation du problème mathématiquement et graphiquement peu complexe. En effet, dans une PCFG, la dérivation d'un symbole non-terminal (et la probabilité associée) est indépendante de son contexte dans l'arbre (ancêtres, frères,...), ce qui est plus que discutable d'un point de vue linguistique. Malgré cette apparente simplicité mathématique, de nombreux travaux ont montré que les analyseurs syntaxiques basés sur ce type de grammaire obtiennent des résultats au niveau de l'état de l'art, et ceci pour diverses langues dont l'anglais (Charniak, 2000; Petrov, 2010), et le français (Crabbé & Candito, 2008; Seddah *et al.*, 2009b). On peut néanmoins noter que ces performances élevées sont atteintes grâce à des algorithmes modifiant en profondeur le modèle mathématique de base des PCFG. Ces algorithmes ont pour but de rendre une grammaire plus efficace en corrigeant partiellement certains problèmes liés au formalisme génératif, comme la dispersion des données. La dispersion représente la difficulté d'estimer la probabilité de phénomènes syn-

[11] Context-Free Grammar (Booth, 1969).
[12] Probabilistic Context-Free Grammar (Booth, 1969).
[13] Combinatory Categorial Grammars [CCG] (Curry & Feys, 1958).
[14] Head-driven Phrase Structure Grammar [HPSG] (Pollard & Sag, 1994).
[15] Lexical-Functional Grammars [LFG] (Kaplan & Bresnan, 1995).

taxiques apparaissant dans les textes à analyser mais qui sont rares ou absents du corpus ayant servi à l'apprentissage des analyseurs. Ces phénomènes qui posent un problème de dispersion des données sont en général des phénomènes courants : les locuteurs en ont déjà entendu et produit des exemples innombrables, et ils en entendent et en produisent régulièrement. Ils sont rares dans les corpus annotés actuellement utilisés, d'une taille sans commune mesure avec la masse de formes linguistiques à laquelle est exposé un locuteur.
Il existe, en parallèle de ces grammaires, d'autres modélisations possibles du problème, et notamment les modèles discriminants (ou discriminatifs). Un algorithme d'apprentissage discriminatif est basé sur une modélisation complexe de phénomènes syntaxiques, dont la particularité est, théoriquement, de ne pas poser initialement d'hypothèses d'indépendances entre les variables[16]. L'avantage par rapport aux PCFG est donc de pouvoir incorporer une multitude d'informations, pouvant provenir de différentes sources, sans avoir à altérer le formalisme mathématique afin de les prendre en compte efficacement. Parmi les modèles les plus utilisés, on peut citer les modèles Maximum d'Entropie (Jaynes, 1957; Berger *et al.*, 1996), les Champs Conditionnels Aléatoires (Lafferty *et al.*, 2001), ou encore les algorithmes de classification linéaire comme le Perceptron (Rosenblatt, 1958) ou les Séparateurs à Vastes Marges (Boser *et al.*, 1992). Bien que ce type de modélisation soit plus adapté à l'analyse en dépendances (tâches de classification), nous verrons que quelques systèmes d'analyse en constituants ont réussi à dompter la complexité relative de ces modèles, et ceci de différentes façons et pour différents buts.
Dans la section suivante, nous indiquons l'ensemble des métriques permettant d'évaluer les performances, et donc la qualité, des analyseurs syntaxiques. Puis, dans la section 2.3, nous verrons en détail le principe des grammaires PCFG, ainsi que les problèmes associés, et les principales solutions pour les résoudre. Ensuite, dans la section 2.4, nous décrirons plusieurs modélisations statistiques discriminatives, et les systèmes d'analyse qui en découlent. Pour finir, nous faisons, dans la section 2.5, une petite parenthèse sur l'adaptation des analyseurs à de petits corpus et à des textes de genres différents, puis, nous concluons.

## 2.2 Évaluation de la qualité des analyseurs syntaxiques

L'évaluation des analyseurs syntaxiques est une étape essentielle et ce, dans le but d'établir un classement des analyseurs en terme de performances mais également de repérer les forces et les faiblesses de chacun d'entre eux. De manière générale, les expérimentations sont menées dans le but de comparer les résultats de deux expériences ayant les mêmes données d'éntrées, mais avec un paramétrage différent des algorithmes sous-jacents. Typiquement, on pourrait vouloir comparer les performances d'un analyseur entraîné sur deux corpus distincts, un corpus d'apprentissage de base et ce même corpus avec un jeu d'étiquettes morpho-syntaxiques plus complexe. Il existe de nombreuses métriques d'évaluation parmi lesquelles quatre sont utilisées majoritairement dans les travaux traitant de l'analyse syntaxique. Deux sont spécifiques à l'analyse en constituants, à savoir PARSEVAL (Black *et al.*, 1991) et Leaf-Ancestor (Sampson

[16]En pratique, il est la plupart nécessaire de poser des hypothèses de dépendances à travers les équations de calcul des probabilités.

& Babarczy, 2003). Deux autres sont en revanche spécifiques à l'analyse en dépendances, UAS et LAS[17]. Il est également courant de calculer le score de l'étiquetage morpho-syntaxique car il peut nous renseigner sur un éventuel problème de couverture lexicale de l'analyseur.
Les mesures que nous venons d'énumérer ne suffisent pas toujours à déterminer si le gain, obtenu par une expérience par rapport à une autre, est statistiquement pertinent. Ce qu'on appelle *significativité* des résultats dépend de plusieurs paramètres dont la quantité de données à évaluer. En effet, un gain de +0.5 obtenu en évaluant 1000 phrases est plus significatif que si cela porte sur seulement 50 phrases. Pour résoudre cette problématique, il existe une multitude d'algorithmes statistiques permettant de mesurer la significativité des résultats. Dans notre travail, nous avons utilisé l'un de ces algorithmes calculant une valeur appelée *t-test*. En parallèle de celui-ci, nous avons également utilisé une méthode d'estimation de la fiabilité d'un modèle statistique appelée *validation croisée*. Cette méthode d'évaluation n'est pas à proprement parler un algorithme établissant la significativité des résultats. Elle permet de calculer un score moyen sur un ensemble de données d'évaluation de taille importante.
Dans les sections qui suivent, nous décrivons en détail les différentes métriques d'évaluation ainsi que les mesures permettant d'établir la significativité des résultats. Pour illustrer concrètement ces mesures, nous les appliquerons sur des analyses de la phrase d'exemple suivante :

$$Luc\ recouvre\ la\ table\ de\ bois\ . \tag{2.1}$$

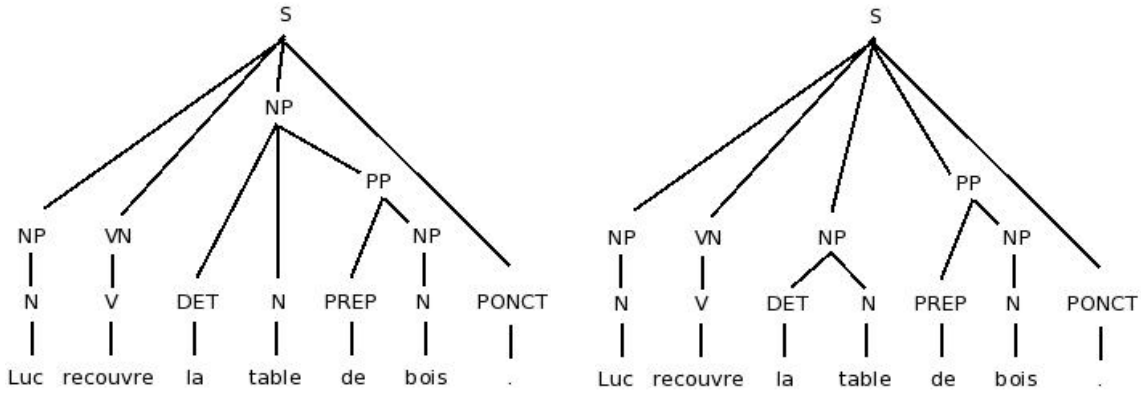
### 2.2.1 PARSEVAL

La mesure PARSEVAL (Black *et al.*, 1991) permet de déterminer trois scores, le *rappel*, la *précision* et le *f1-score* [F1]. Cette mesure considère qu'un noeud[18] de l'analyse générée est correct si l'étiquette du noeud ainsi que sa couverture en terme de tokens correspondent à l'identique à ceux d'un noeud de l'analyse correcte. D'après ce principe, le rappel est le nombre de noeuds générés corrects sur le nombre total de noeuds de l'analyse correcte. La précision est le nombre de noeuds générés corrects sur le nombre total de noeuds de l'analyse générée. Le score F1 est une valeur harmonique moyenne basée sur le rappel et la précision. Il équivaut à :

$$F_1 = \frac{2 \times precision \times rappel}{precision + rappel} \tag{2.2}$$

La figure 2.1a montre l'analyse correcte de la phrase d'exemple 2.1. Dans l'analyse générée, montrée figure 2.1b, le groupe prépositionnel *de bois* a été incorrectement rattaché au noyau verbal au lieu du groupe nominal *la table*. Aussi, l'application de la mesure PARSEVAL sur l'analyse générée indique que 5 noeuds sont considérés comme corrects, ce qui induit un rappel et une précision de 83.33% (5/6).
Dans le cadre de nos expériences, les scores PARSEVAL sont obtenus grâce à l'outil d'évaluation *evalb*[19] avec les paramètres standard de Collins[20].

[17] Unlabeled/Labeled Attachement Score
[18] Les noeuds terminaux et préterminaux ne sont pas pris en compte lors du calcul des scores.
[19] `http://nlp.cs.nyu.edu/evalb/`
[20] Les noeuds de ponctuations sont conservés lors des évaluations.




(a) Analyse correcte (b) Analyse générée

FIG. 2.1: Analyses en constituants de la phrase d'exemple 2.1.

### 2.2.2 Leaf Ancestor

Le score Leaf-Ancestor [LA] (Sampson & Babarczy, 2003) mesure la similarité entre les chemins partant de chaque noeud terminal et arrivant au noeud racine de l'analyse générée à ceux de l'analyse correcte. Un chemin est représenté par la séquence de noeuds entre un noeud terminal et le noeud racine. Le score global de l'analyse générée est égal à la moyenne des scores des noeuds terminaux. La similarité entre deux chemins est calculée par la distance de Levenshtein (Levenshtein, 1966). Cette distance est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre, une séquence de noeuds étant vue comme une chaîne de caractères (symboles associés aux noeuds). Dans le cadre de nos expériences, les scores LA sont obtenus grâce à l'outil d'évaluation *leaf-ancestor assessment software*[21]. La figure 2.2 montre les scores LA obtenus par l'analyse générée de la figure 2.1b. Chaque ligne représente l'analyse d'un noeud terminal de la phrase et contient le score LA local, le token-mot de la phrase et les chemins à comparer partant du même noeud terminal (séparés par un " :"). Le chemin de gauche correspond à celui de l'analyse générée, alors que le chemin de droite correspond à celui de l'analyse correcte. Le score global LA de l'analyse générée est ici de 92,8% (ligne démarrant par `Sentence 1 : ave.`).

### 2.2.3 Unlabeled/Labeled Attachement Score

L'analyse syntaxique en dépendances utilise des scores spécifiques, nommés sous les acronymes de UAS et LAS (pour Unlabeled/Labeled Attachement Score). Ils sont calculés grâce aux relations de dépendances entre les mots de la phrase. Le score UAS est la proportion de mots dont le gouverneur associé est correct[22]. Le score LAS est la proportion de mots dont le

[21]Les noeuds de ponctuation sont conservés lors des évaluations. L'outil est disponible à l'adresse `http://www.grsampson.net/Resources.html`

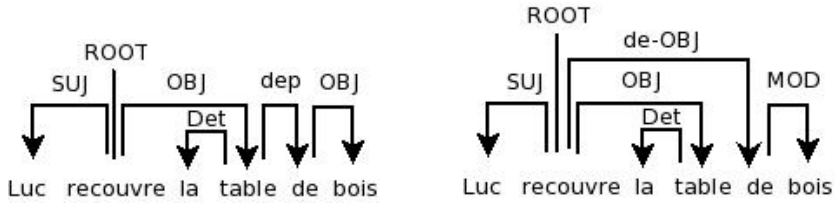
[22]Dans le cadre de relations de dépendances surfaciques, il ne peut y avoir qu'un seul gouverneur par mot.

1.000 Luc NP ] SENT [ : NP ] SENT [
1.000 recouvre [ VN ] SENT : [ VN ] SENT
1.000 la [ NP SENT : [ NP SENT
1.000 table NP ] SENT : NP SENT
0.857 de [ PP SENT : [ PP NP SENT
0.909 bois [ NP PP SENT ] : [ NP PP NP SENT ]

Sentence 1 : ave. 0.928

FIG. 2.2: Scores LA de l'analyse de la figure 2.1b.

gouverneur et le type de dépendance associés sont corrects[23]. Afin d'être capable de calculer ces scores sur les arbres retournés par les analyseurs en constituants, nous avons converti automatiquement ces derniers en arbres de dépendances typées grâce aux algorithmes décrits dans (Candito *et al.*, 2009, 2010a)[24]. Par convention, la tête de la phrase a pour gouverneur un noeud artificiel communément appelé `ROOT`.
La figure 2.3a montre l'analyse en dépendance correcte de notre phrase d'exemple 2.1. Dans l'analyse générée montrée figure 2.3b, le gouverneur *recouvre* de la préposition *de* est incorrect car il s'agit en fait de *table*. Ainsi, le score UAS est égal à 83.33% (5/6). De plus, le type de dépendance entre la préposition *de* et le nom *bois* est marquée `MOD` au lieu de `OBJ`, ce qui induit un score LAS de 66.66% (4/6).




(a) Analyse correcte (b) Analyse générée

FIG. 2.3: Analyses en dépendances de la phrase d'exemple 2.1.

Dans le cadre de nos expériences, les scores UAS et LAS sont obtenus grâce à l'outil d'évaluation *CoNLL-X evaluation script*[25] avec les paramètres standard[26].

### 2.2.4 Validation croisée et significativité des résultats

Dans le cas des corpus de petite taille, comme le FTB-UC, l'estimation de la qualité d'un analyseur ne peut se faire que sur une petite portion de ce corpus, la plus grande partie étant

[23]Le type de dépendance est l'étiquette de la relation qui peut prendre, par exemple, les valeurs `suj` (sujet), `obj` (objet),...
[24]Le programme de conversion en dépendances est inclus dans le package *Bonsaï*.
[25]`http://ilk.uvt.nl/conll/software.html`
[26]Les noeuds de ponctuations sont conservés lors des évaluations.

réservée à l'apprentissage. La méthode dite de validation croisée permet de pallier ce problème. Elle consiste à découper le corpus en *p* parties égales puis à effectuer l'apprentissage sur *p-1* parties et les évaluations sur la partie restante. On peut itérer *p* fois ce processus. Cela permet donc de calculer un score moyen ($F_1$, UAS ou LA) sur un échantillon aussi grand que le corpus initial. Dans notre cas, nous avons fixé le paramètre *p* à une valeur standard de 10. Cette valeur conduit à créér 9 parties de 1235 phrases et une partie de 1236 phrases. Cet algorithme étant très couteux en temps, nous l'utiliserons uniquement dans le but d'établir la significativité des résultats de certaines expériences après une analyse préalable des erreurs sur les corpus de développement et d'évaluation.
En parallèle de la validation croisée, nous avons fait le choix d'utiliser un algorithme calculant la significativité des résultats, qui prend en entrée uniquement deux ensembles de résultats obtenus pour deux expériences distinctes. À partir de ces résultats, il retourne un score unidirectionnel d'écarts à la moyenne pour échantillons appariés appelé t-test[27]. En dessous d'un certain seuil (en général $p < 0.5$), le gain obtenu par l'une des deux expériences est considéré comme significatif. Nous avons appliqué ce score sur les mesures PARSEVAL et UAS uniquement[28].

### 2.2.5 Évaluation de l'étiquetage morpho-syntaxique

L'évaluation de l'étiquetage morpho-syntaxique consiste à déterminer le nombre d'étiquettes correctement assignées sur la totalité des étiquettes du texte évalué. La figure 2.4 montre deux séquences d'étiquettes pour la phrase d'exemple 2.1 : la séquence correcte ainsi qu'une séquence générée par un étiqueteur lambda. Dans cette dernière, les erreurs d'étiquetage sont signalées en gras. On peut remarquer que deux mots sont mal étiquetés (*la* et *de*) sur un total de 7 mots[29]. Le score pour cette analyse est donc de $5/7$.

| | | | | | | |
|---|---|---|---|---|---|---|
| Luc/NPP | recouvre/V | la/DET | table/NC | de/P | bois/NC | ./PONCT |
| Luc/NPP | recouvre/V | **la/NC** | table/NC | **de/DET** | bois/NC | ./PONCT |

FIG. 2.4: Deux séquences d'étiquettes pour la phrase d'exemple 2.1 : la séquence correcte et une séquence générée automatiquement.

En plus du score global, il est en général utile d'indiquer le score pour les mots inconnus[30] uniquement. On peut noter que la formule que nous venons de décrire ne s'applique qu'à un texte dont la segmentation est considérée comme parfaite. En effet, il faut qu'elle soit identique à celle du corpus de référence pour que tout mot du texte corresponde à un mot du corpus. Nous verrons dans le chapitre 5 que la formule diffère lorsque ce n'est pas le cas.

[27]Ce score est calculé grâcé au programme de Dan Bikel disponible à l'adresse `http://www.cis.upenn.edu/~dbikel/software.html`

[28]De manière empirique, on peut supposer qu'un gain LA supérieur ou égal à +0.3 peut être considéré comme significatif.

[29]Le signe de ponctuation compte pour un mot.

[30]On peut rappeler qu'un mot est inconnu s'il n'est pas présent dans le corpus d'apprentissage.

### 2.2.6 Conclusion

Il n'est plus à démontrer l'intérêt d'utiliser plusieurs métriques lorsque l'on souhaite effectuer une évaluation de qualité. En effet, des métriques peuvent passer sous silence certains problèmes ou au contraire accorder trop d'importances à des erreurs très particulières. De nombreux travaux (Carroll *et al.*, 1998; Briscoe *et al.*, 2002; Sampson & Babarczy, 2003) ont par exemple critiqué le fait que la mesure PARSEVAL soit utilisée depuis les années 90 comme un standard lors des évaluations des analyseurs en constituants. Il semblerait que cette mesure ne reflète pas la qualité réelle des analyses produites. Pour illustrer cette hypothèse, (Rehbein & van Genabith, 2007) ont effectué une expérience consistant à entraîner un analyseur sur deux corpus de la langue allemande, *NEGRA*[31] (Skut *et al.*, 1997) et *TüBa-D/Z*[32] (Telljohann *et al.*, 2004), ayant un schéma d'annotation différent. Dans la littérature, la plupart des expériences (Kübler & Tübingen, 2005; Kübler *et al.*, 2006; Maier, 2006) ont montré que le schéma d'annotation du corpus NEGRA était moins approprié pour certains formalismes d'analyse syntaxique que celui du TüBa-D/Z au vu des résultats obtenus par plusieurs analyseurs. Les scores PARSEVAL indiqués par (Rehbein & van Genabith, 2007) sur les deux corpus d'évaluations pourraient valider le constat précédent s'ils étaient en accord avec les résultats des autres métriques. Cependant, ils ont signalé des scores LA et UAS significativement à l'avantage de l'analyseur entraîné sur le corpus NEGRA. D'après leurs observations, il semblerait que le score UAS soit à même de refléter au mieux la qualité d'un analyseur.

## 2.3 Modèles génératifs pour l'analyse syntaxique : Grammaires hors-contexte probabilistes

Les grammaires hors-contexte probabilistes [PCFG] (Booth, 1969), connues aussi sous le nom de grammaires stochastiques hors-contexte, sont, à l'heure actuelle, un des formalismes les plus utilisés dans le cadre de l'analyse syntaxique probabiliste. Il s'agit d'une extension des grammaires hors-contexte [CFG]. Elles sont définies par quatre paramètres *(N,*$\Sigma$*,P,S)*, avec :
– *N*, un ensemble de symboles non-terminaux.
– $\Sigma$, un ensemble de symboles terminaux (disjoint de *N*).
– P, un ensemble de règles, appelées aussi *productions*.
– S, un symbole de départ (racine de l'arbre syntaxique).
L'ensemble *P* contient des règles de la forme suivante :

$$\Lambda \rightarrow \beta[p] \tag{2.3}$$

où $\Lambda$ est un symbole non-terminal appartenant à l'ensemble *N*, $\beta$ est une séquence de symboles appartenant à l'ensemble $(N \cup \Sigma)*$, et *p* est la probabilité, comprise entre 0 et 1, associée à la règle. Cette probabilité *p*, étrangère aux grammaires CFG, correspond à la probabilité qu'un symbole non-terminal $\Lambda$ soit dérivé en une production $\beta$. Cette probabilité conditionnelle

[31] Version 2 composée de 50000 phrases provenant en partie du corpus NEGRA.
[32] Version 2 composée d'environ 22000 phrases.

peut être représentée par $P(\Lambda \rightarrow \beta|\Lambda)$. La façon la plus courante d'estimer une probabilité conditionnelle est d'utiliser un corpus annoté de la langue traitée. Cela consiste à compter le nombre d'occurrences de la règle $\Lambda \rightarrow \beta$ dans le corpus. Puis, il est normalisé en le divisant par le nombre de fois que $\Lambda$ est membre gauche d'une règle :

$$P(\Lambda \rightarrow \beta) = \frac{occ(\Lambda \rightarrow \beta)}{\sum_{\gamma} occ(\Lambda \rightarrow \gamma)} \tag{2.4}$$

De manière générale, l'usage veut qu'une grammaire PCFG soit consistante. Cela signifie que la somme des probabilités des dérivations possibles d'un symbole non-terminal doit être égale à 1 :

$$\sum_{\beta} P(\Lambda \rightarrow \beta) = 1 \tag{2.5}$$

La figure 2.5 montre un exemple de grammaire PCFG que l'on pourrait extraire d'un corpus du français. Les règles sont classées en deux catégories, les règles contextuelles (symbole non-terminal → symboles non-terminaux) et les règles lexicales (symbole non-terminal → mot). De plus, elle vérifie la propriété de consistance (par exemple, $P(NC \rightarrow ouvriers) + P(NC \rightarrow sacs) + P(NC \rightarrow gravats) = 1$)

| Règles contextuelles | | Règles lexicales |
|---|---|---|
| S → NP VN NP | [0.5] | DET → les [1.0] |
| S → NP VN NP PP | [0.5] | NC → ouvriers [0.4] |
| NP → DET NC | [0.4] | NC → sacs [0.3] |
| NP → DET NC PP | [0.6] | NC → gravats [0.3] |
| PP → P NP | [1.0] | V → jettent [1.0] |
| VN → V | [1.0] | P → de [1.0] |

FIG. 2.5: Exemple de grammaire PCFG du français.

Un algorithme non-supervisé, appelé *Inside-Outside* [IO] (Baker, 1979; Lari & Young, 1990), permet de produire une grammaire PCFG sans l'aide d'un corpus annoté. Cet algorithme itératif prend en entrée un corpus de phrases brutes ainsi qu'une grammaire non probabiliste CFG, à laquelle on ajoute une distribution de probabilité uniforme aux règles. À chaque itération, la grammaire génère les analyses possibles de chaque phrase brute du corpus. Puis, la probabilité de chaque analyse est calculée et est ensuite utilisée pour pondérer le nombre d'occurrences des règles apparaissant dans ces analyses. La dernière étape consiste à réestimer les probabilités associées aux règles du modèle. L'algorithme se termine lorsque les probabilités ont convergé. Cette méthode, dérivée de l'algorithme Expectation-Maximisation [EM], a prouvé son efficacité dans le cadre de l'analyse syntaxique (Briscoe & Waegner, 1992; Deoskar, 2008; Deoskar *et al.*, 2009), mais les grammaires créées ont généralement des performances plus faibles que des grammaires apprises sur des corpus annotés[33].
Dans la section suivante, nous décrivons formellement la procédure de calcul de l'analyse ayant la meilleure probabilité à partir d'une grammaire PCFG. Cette procédure est également appelée

[33](Deoskar *et al.*, 2009) signalent un score $F_1$ de 5 points inférieur à l'état de l'art actuel des grammaires de corpus.

*désambiguisation syntaxique* d'une phrase. Puis, dans les section 2.3.2 et 2.3.3, nous verrons en détail les deux problèmes liés au formalisme PCFG, à savoir les hypothèses d'indépendances trop fortes entre les règles, et le faible conditionnement lexical. Pour chacun d'eux, nous évoquons les solutions apportées par les différents travaux sur le sujet. Nous finirons, section 2.3.4, par indiquer les performances de divers analyseurs syntaxiques sur le corpus du français FTB-UC.

### 2.3.1 Désambiguisation des analyses syntaxiques d'une phrase

Les hypothèses fortes d'indépendances induites par les grammaires CFG font que la dérivation d'un symbole non-terminal est indépendante de son contexte dans l'arbre. De fait, dans une grammaire PCFG, la probabilité d'une règle est également indépendante de son contexte dans l'arbre. Par définition, la probabilité d'un ensemble d'événements indépendants est le produit des probabilités associées aux événements. Ainsi, la probabilité d'une analyse *T* d'une phrase *S* est définie par le produit des probabilités des *n* règles qui dérivent des *n* noeuds non-terminaux composant l'arbre *T* :

$$P(T,S) = P(T) = \prod_{i=1}^{n} P(\Lambda_i \rightarrow \beta_i) \tag{2.6}$$

avec $\Lambda_i \rightarrow \beta_i$ la *i*ème règle de $T$. L'avantage d'une grammaire PCFG, par rapport à une grammaire CFG, est de pouvoir désambiguiser la forêt d'analyses possibles d'une phrase en sélectionnant la meilleure analyse en terme de score. La désambiguisation est définie formellement par :

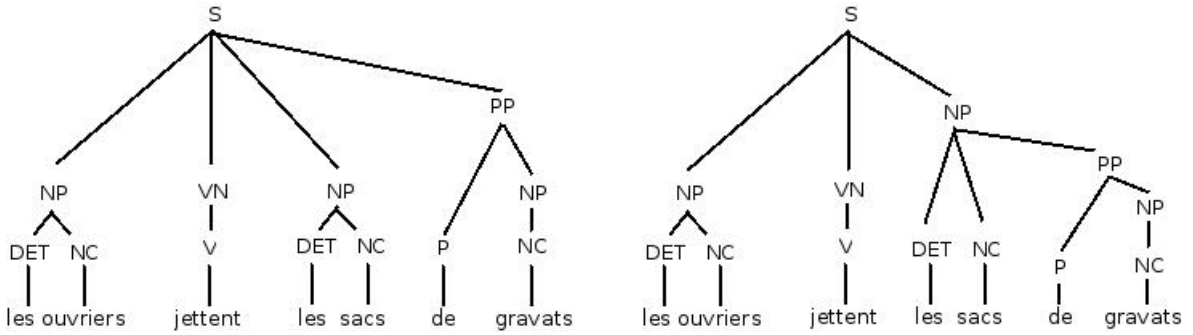
$$T_\star(S) = \underset{T \in \tau(S)}{argmax} P(T) \tag{2.7}$$

avec $\tau(S)$ l'ensemble des analyses produites par la grammaire à partir de *S*. Par exemple, la figure 2.6 montre deux analyses possibles pour la phrase *Les ouvriers jettent les sacs de gravats*. L'unique différence entre les deux analyses est le rattachement du groupe prépositionnel *de gravats*, soit au noeud racine et donc au verbe *jettent* comme dans la première analyse, soit au groupe nominal *les sacs*. Le sens dégagé par la deuxième analyse semble correct (*les sacs de gravats sont jetés*), contrairement à celui du premier arbre (∗*les sacs sont jetés de gravats*). À l'aide de la grammaire PCFG montrée figure 2.5, nous sommes capable de calculer la probabilité de chacun de ces deux arbres :

$$P(2.6a) = P(S \rightarrow NP\ \ VN\ \ NP\ \ PP) \times P(NP \rightarrow DET\ \ NC) \times ... = 2.9\times^{10^{-3}} \tag{2.8}$$

$$P(2.6b) = P(S \rightarrow NP\ \ VN\ \ NP) \times P(NP \rightarrow DET\ \ NC\ \ PP) \times ... = 4.3\times^{10^{-3}} \tag{2.9}$$

L'intuition que nous avions à propos de la meilleure solution pour la phrase en entrée, à savoir la première analyse, est validée par la grammaire PCFG qui lui a attribuée une probabilité légèrement supérieure. De nombreux algorithmes standards à base d'analyse tabulaire, notamment *Cocke-Kasami-Younger* (Kasami, 1965; Ney, 1991) ou encore *Earley* (Earley, 1970), permettent de déterminer avec une complexité réduite l'ensemble $\tau(S)$ des analyses d'une phrase *S* et d'en effectuer la désambiguisation.

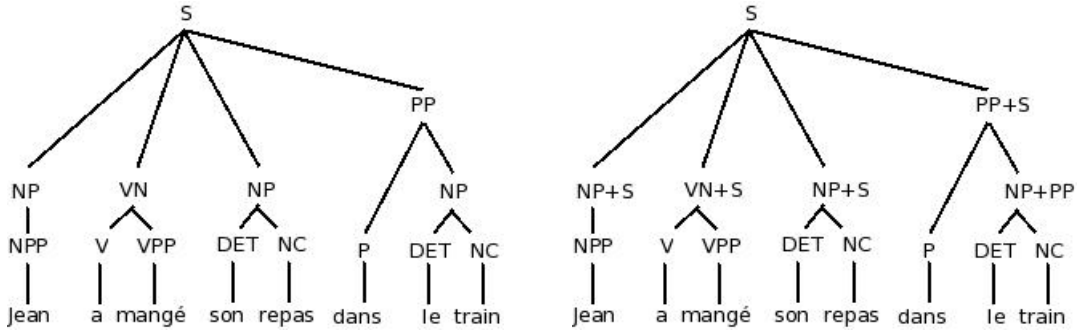



(a) solution 1 (b) solution 2

FIG. 2.6: Deux analyses possibles pour la phrase *Les ouvriers jettent les sacs de gravats*.

### 2.3.2 Hypothèses d'indépendance des grammaires PCFG

Les hypothèses d'indépendances très fortes entre les règles d'une grammaire PCFG conduisent à une faible contextualisation des probabilités des règles. On peut rappeler que la probabilité d'une règle $\Lambda \rightarrow \beta$ est calculée grâce à des statistiques sur les occurrences des membres gauche et droit de la règle uniquement. Or, on aimerait conditionner ces probabilités avec d'autres informations tirées des arbres, notamment l'environnement direct dans lequel apparaît une règle. (Johnson, 1998) propose une manière de résoudre ce problème grâce à une méthode appelée *parent annotation* qui effectue des manipulations sur les arbres du corpus d'apprentissage de la grammaire. Chaque noeud non-terminal d'un arbre (excepté les préterminaux) voit son étiquette augmentée par le symbole de son noeud père, ce qui conduit à obtenir des probabilités conditionnées par les symboles pères. La figure 2.7 montre un exemple d'application de cette méthode sur l'arbre original de la phrase *Jean a mangé son repas dans le train*.




(a) arbre original (b) arbre modifié

FIG. 2.7: Application de la méthode *parent annotation* sur l'arbre original de la phrase *Jean a mangé son repas dans le train*.

Bien que cette méthode ait prouvé son efficacité sur le PTB[34], elle n'est parfois pas assez puissante pour discriminer certaines ambiguités syntaxiques. Par exemple, dans le cadre du français

[34](Johnson, 1998) obtient un gain $F_1$ de +4.3 par rapport à une grammaire PCFG de base.

et plus particulièrement du FTB-UC, le syntagme nominal NP a le même père, qu'il soit en position de sujet ou en position d'objet du verbe. (Klein & Manning, 2003) ont donné une réponse à ce problème en adoptant un point de vue linguistique. Ils ont écrit manuellement des règles de modification de l'étiquetage des noeuds non-terminaux des arbres. Chaque règle divise un symbole en sous-catégories d'après des observations linguistiques faites sur le corpus annoté. Par exemple, une règle applicable sur le PTB, appelée *SPLIT-AUX*, impose que les auxiliaires soient marqués au niveau de leur étiquette morpho-syntaxique par `+BE` pour toutes les formes du verbe *be*, et `+HAVE` pour toutes les formes du verbe *have*. La figure 2.8 montre l'application de cette règle sur l'arbre de la phrase *A passenger plane has crashed* (*L'avion de transport de passagers s'est écrasé*).

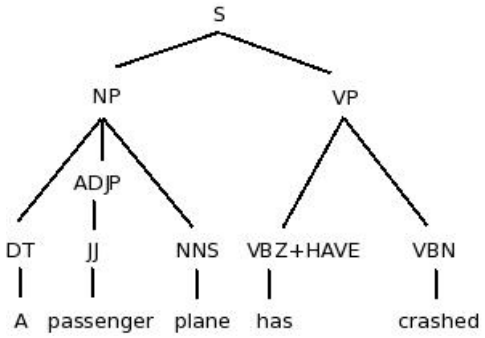
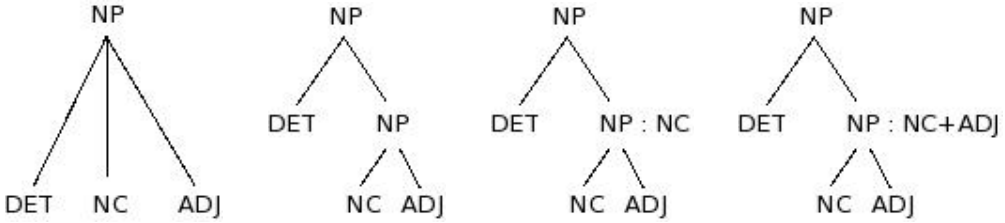



FIG. 2.8: Application de la règle SPLIT-AUX de (Klein & Manning, 2003) sur l'arbre de la phrase *A passenger plane has crashed*.

Les deux méthodes que nous venons de voir causent une augmentation significative de la taille de la grammaire extraite du corpus, ayant pour conséquence un effet indésirable appelé *dispersion des données*. La multiplicité accrue des règles réduit la quantité de données disponibles pour estimer la probabilité de chaque règle de la grammaire. Cela conduit à surestimer les probabilités des règles apparaissant rarement dans le corpus et à sous-estimer celles des règles présentes dans les phrases à analyser mais inconnues du corpus d'apprentissage. Pour pallier la dispersion des données, (Klein & Manning, 2003) effectuent des modifications de structure des arbres, à savoir une binarisation des arbres appelée *markovisation horizontale*. Chaque membre droit des règles est maintenant vu comme une séquence de symboles qui sont générés de la gauche vers la droite. Pour déterminer la probabilité d'un symbole généré, l'historique utilisé (le contexte) est composé des $h$ symboles précédents. Cette markovisation a donc pour effet de découper le calcul de la probabilité d'une règle en sous produits de probabilités plus simples à estimer :

$$P(\Lambda \rightarrow \beta) = P(\Lambda \rightarrow \beta_0 ... \beta_n) = \prod_{i=1}^{n-1} P(\beta_i | \beta_0, ..., \beta_{i-1}, \Lambda) \approx \prod_{i=1}^{n-1} P(\beta_i | \beta_0, ..., \beta_{i+h}, \Lambda) \quad (2.10)$$

La figure 2.9 illustre une markovisation horizontale d'une règle de base selon divers ordres $h$. Les meilleurs résultats de (Klein & Manning, 2003) sur le corpus d'évaluation du PTB sont obtenus avec seulement quinze règles et une markovisation horizontale de 2. Le gain $F_1$ de +10.8, indiqué dans le tableau 2.1, est significatif par rapport à une grammaire PCFG de base




(a) règle originale (b) ordre 0 (c) ordre 1 (d) ordre 2

FIG. 2.9: Exemples de markovisations horizontales d'ordre $h$ d'une règle

(expérience notée `Base`). Les règles de division des symboles de (Klein & Manning, 2003), malgré leur efficacité sur le PTB, ont plusieurs inconvénients. Il y a tout d'abord un problème de maintenance de ces règles car si le corpus évolue dans une future version, les règles doivent également évoluer pour prendre en compte les nouvelles modifications structurelles et syntaxiques. De plus, les règles étant interdépendantes, la maintenance peut se révéler fastidieuse. Et enfin, il faut adapter ou créer des règles afin de les appliquer sur tout nouveau corpus ayant un schéma d'annotation spécifique.
Aussi, plus récemment, (Matsuzaki *et al.*, 2005; Prescher, 2005) ont proposé une approche automatique du problème à travers l'utilisation de grammaires PCFG augmentées de symboles latents [PCFG-LA] qui sont une extension des grammaires PCFG classiques. Dans le formalisme PCFG-LA, tout arbre observé dans un corpus est un arbre incomplet et l'arbre complet correspondant est un arbre augmenté de symboles latents. Chaque noeud non-terminal d'un arbre complet est étiqueté par un symbole de la forme $A[x]$ où $A$ est le symbole de ce noeud dans l'arbre observé et $x$ est un symbole latent pris parmi un ensemble prédéfini $H$. En supprimant les annotations latentes d'une grammaire PCFG-LA, nous sommes capable d'extraire automatiquement une grammaire PCFG. La taille $n$ de l'ensemble $H$ est fixé manuellement lors de l'initialisation de l'algorithme EM, qui permet d'estimer les paramètres de la grammaire PCFG-LA. Par exemple, si $n = 8$ alors le symbole `NP` sera divisé en 8 symboles, $\texttt{NP}_1$ à $\texttt{NP}_8$, chacun ayant sa propre distribution de probabilités d'après les observations faites sur le corpus d'apprentissage. Le tableau 2.1 montre les meilleurs résultats obtenus par la grammaire PCFG-LA de (Matsuzaki *et al.*, 2005) apprise sur le corpus d'apprentissage du PTB et évaluée sur le corpus d'évaluation du PTB. Le gain $F_1$ de +11.2 est obtenu en fixant la taille $n$ de l'ensemble $H$ à 16. Les performances de leur méthode automatique sont donc légèrement supérieures à la méthode à base de règle de (Klein & Manning, 2003).

| Analyseur | Rappel | Précision | $F_1$ | $\Delta_{F_1}$ |
|---|---|---|---|---|
| Base | 72.8 | 77.2 | 74.9 | - |
| (Klein & Manning, 2003) | 85.1 | 86.3 | 85.7 | +10.8 |
| (Matsuzaki *et al.*, 2005) | 86.0 | 86.1 | 86.1 | **+11.2** |

TAB. 2.1: Performances sur le corpus d'évaluation du PTB des analyseurs proposés par (Klein & Manning, 2003) et (Matsuzaki *et al.*, 2005). $\Delta_{F_1}$ est le gain par rapport à une grammaire PCFG de base (expérience notée `Base`).

Le fait de devoir fixer la taille $n$ de l'ensemble $H$ impose de diviser tout symbole en $n$ sous-

symboles, or certaines divisions ne sont pas ou peu utiles en terme de gain sur les performances. De plus, elles causent une augmentation superflue de la taille de la grammaire PCFG-LA. L'analyseur Berkeley[35] [BKY] (Petrov *et al.*, 2006; Petrov & Klein, 2007; Petrov, 2010) permet d'optimiser la phase d'ajout des symboles latents en déterminant automatiquement si la division d'un symbole est nécessaire ou non. Cet algorithme créé itérativement $n$ grammaires PCFG-LA $G_1 ... G_n$ possédant un jeu de symboles non-terminaux de plus en plus complexes grâce à l'ajout d'annotations latentes. La grammaire de départ $G_1$ est une grammaire PCFG simple extraite du corpus annoté. À chaque itération, plusieurs étapes successives sont effectuées :

1. La première étape consiste à ajouter les annotations latentes aux symboles de la grammaire. Pour cela, une nouvelle grammaire $G_{i+1}$ est créée à partir de $G_i$ en divisant chaque symbole non-terminal de $G_i$ en deux nouveaux symboles non-terminaux. Les paramètres de la grammaire $G_{i+1}$ sont ensuite estimés par l'algorithme EM sur le corpus annoté. Un biais aléatoire est introduit pour briser la symétrie lors des divisions.
2. La création d'annotations latentes permet d'améliorer les performances de la grammaire, mais augmente sensiblement la taille de celle-ci. Pour éviter un éventuel problème de dispersion des données et de surentraînement de la grammaire, la division d'un symbole n'est effectuée que si elle est utile, i.e. qu'elle permette de discriminer des ambiguités syntaxiques. Ainsi, chaque symbole divisé à l'étape précédente est fusionné à nouveau. Puis, la vraisemblance de ce symbole est calculée sur le corpus annoté (section 2.4.3). La vraisemblance consiste à quantifier la probabilité que les observations faites sur le corpus (après division) proviennent effectivement d'un échantillon théorique du modèle probabiliste. Si la perte de vraisemblance est faible, la division n'est pas conservée car elle n'apporterait pas assez d'informations supplémentaires.
3. La dernière étape consiste à lisser par interpolation les probabilités des règles ayant le même membre gauche (sans symbole latent). Par exemple, deux symboles $\texttt{DET}_0$ et $\texttt{DET}_1$ d'une grammaire PCFG-LA ont en commun le symbole de base non-latent DET. Cependant, ils ont une distribution de probabilités qui leur est propre. Or, il pourrait être utile de faire bénéficier $\texttt{DET}_0$ des données statistiques de $\texttt{DET}_1$ et vice versa. L'interpolation permet de créer une interdépendance entre les distributions lors de la phase d'estimation des probabilités des règles. Ainsi, la probabilité $p_x$ d'une règle $A_x \rightarrow B_y C_z$ est lissée par interpolation avec la moyenne des probabilités des règles ayant $A$ comme membre gauche, pour tous les symboles latents possibles associés à $A$ (dont $x$ fait partie) :

$$p_x^{'} = (1 - \alpha) p_x + \alpha \overline{p} \tag{2.11}$$

où $\overline{p} = \frac{1}{n} \sum_x p_x$ et $\alpha$ est une constante de faible valeur.

La figure 2.10 montre un exemple de division du symbole DT (symbole du déterminant dans le PTB) après chaque itération de l'algorithme[36]. On peut voir qu'après la première division, les deux groupes de déterminants semblent faire respectivement référence aux déterminants démonstratifs *that, this, some* et non démonstratifs *the, a, The*.

Afin de pouvoir donner une étiquette morpho-syntaxique aux mots rares et inconnus, BKY se base sur un algorithme robuste de classification. Pour un mot de ce type rencontré dans une

[35] http://code.google.com/p/berkeleyparser/

[36] Cet exemple est directement tiré de (Petrov *et al.*, 2006).

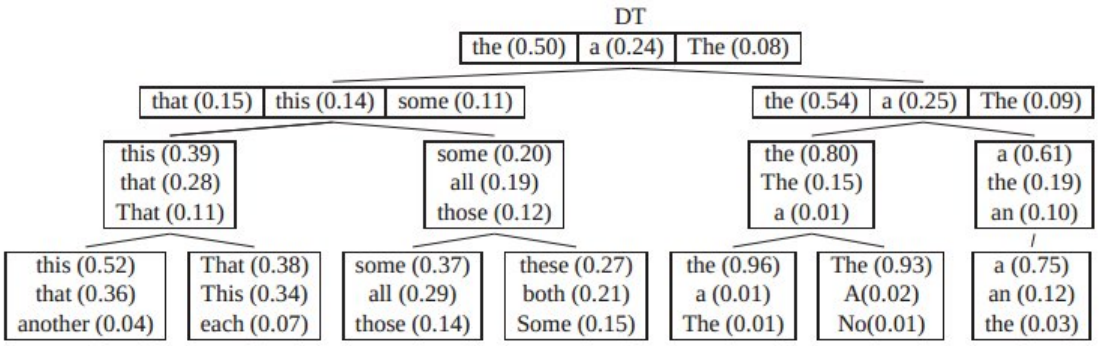



FIG. 2.10: Divisions successives du symbole `DT` (symbole du déterminant dans le PTB) après chaque itération de l'algorithme de BKY. Pour chaque symbole, les trois mots les plus fréquents sont indiqués avec leur probabilité associée.

phrase à analyser, BKY applique des patrons sur ce mot afin d'extraire des informations morphologiques (capitalisation, suffixes,...). Ces informations sont ensuite fusionnées pour former un nouveau mot appelé *signature du mot* (ou classe du mot). Cette signature permet de donner une probabilité d'étiquetage au mot original d'après la formule suivante :

$$p(mot|étiquette) = p(signature|étiquette) \tag{2.12}$$

Les statistiques permettant d'estimer ces probabilités sont extraites au cours de la phase d'apprentissage de BKY. La plupart des expériences effectuées avec BKY sur le français se base sur une version améliorée[37] pour le traitement des mots rares et inconnus grâce à des traits spécifiques à la langue française (Crabbé & Candito, 2008).
Dans l'algorithme de BKY, la valeur du biais aléatoire fixée à l'initialisation de l'apprentissage EM introduit potentiellement de grandes différences dans les grammaires produites. En effet, une grammaire générée avec une valeur spécifique ne sera pas identique à une grammaire générée avec une autre valeur. De même, leurs performances respectives ne seront probablement pas similaires. Nous avons constaté que, dans le cadre du FTB-UC, les performances pouvaient varier jusqu'à 1% en terme de score $F_1$. Cette observation pose des problèmes évidents pour l'évaluation de l'analyseur. En général, le protocole consiste à faire la moyenne des résultats obtenus par $n$ grammaires, ayant chacune une valeur de biais propre. Plutôt que d'exploiter indépendamment chaque grammaire créée, (Petrov, 2010) ont proposé de les combiner lors de l'analyse d'un texte. Dans leur modélisation du problème, la probabilité d'une règle est simplement égale au produit des probabilités postérieures données par chaque grammaire.
Les performances de BKY sur le corpus d'évaluation du PTB, indiquées dans le tableau 2.2, sont significativement supérieures aux méthodes décrites précédemment (gains $F_1$ de +14.8). Cela prouve la pertinence et l'efficacité de leur algorithme de division automatique des noeuds non-terminaux. L'expériences réalisée dans (Petrov, 2010), notée $BKY_p$[38], a consisté à générer 8 grammaires avec une valeur de biais comprise entre 1 et 8. On peut voir que le gain $F_1$ est de +2.2 par rapport à l'algorithme BKY classique.

[37] Version 1.0 de BKY disponible dans l'outil Bonsaï.
[38] `http://code.google.com/p/berkeleyparser/`

| Analyseur | Rappel | Précision | $F_1$ | $\Delta_{F_1}$ |
|---|---|---|---|---|
| Base | 72.8 | 77.2 | 74.9 | - |
| (Klein & Manning, 2003) | 85.1 | 86.3 | 85.7 | +10.8 |
| (Matsuzaki *et al.*, 2005) | 86.0 | 86.1 | 86.1 | +11.2 |
| BKY | **89.6** | **89.8** | **89.7** | **+14.8** |
| $BKY_p$ | **92.0** | **91.7** | **91.9** | **+17** |

TAB. 2.2: Performances sur le corpus d'évaluation du PTB des analyseurs basés sur les divisions de symboles non-terminaux. $\Delta_{F_1}$ est le gain par rapport à une grammaire PCFG de base (expérience notée *Base*).

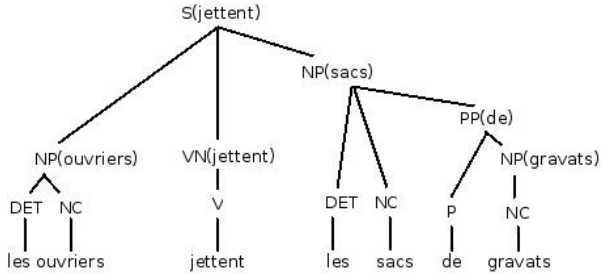
### 2.3.3 Faible conditionnement lexical

Les grammaires PCFG utilisent le lexique du corpus uniquement au niveau des règles lexicales. Ce faible conditionnement lexical ne permet pas de modéliser des structures syntaxiques propres à certains mots, ce qui entraîne principalement des problèmes de rattachement prépositionnel et de coordination. Pour illustrer cet état de fait, reprenons la figure 2.6 nous montrant deux analyses pour la phrase *Les ouvriers jettent les sacs de gravats*, où l'unique différence entre les deux analyses concerne le rattachement prépositionnel de *de gravats*. Selon la distribution de probabilité du modèle, une grammaire PCFG de base va toujours choisir soit le rattachement au nom soit le rattachement au verbe. Or, si dans le corpus d'apprentissage, le rattachement d'un groupe prépositionnel au nom est plus fréquent alors la solution choisie sera fausse quelles que soient les informations apportées par les mots de la phrase. Les grammaires PCFG dites lexicalisées (Charniak, 1997; Collins & Singer, 1999; Charniak, 2000; Collins, 2003) tentent de résoudre ce problème en annotant les noeuds non-terminaux par leur tête lexicale, ce qui conduit à modéliser des dépendances lexicalisées entre les mots de la phrase. On peut rappeler que la tête lexicale est le mot noyau d'un syntagme, i.e. celui qui donne le sens principal au syntagme. Pour les syntagmes nominaux, il s'agit souvent d'un nom[39]. La figure 2.11a montre un exemple de lexicalisation de l'arbre de la phrase *Les ouvriers jettent les sacs de gravats* (figure 2.6a). On considère que la tête lexicale de la phrase (noeud racine de l'arbre) est le verbe *jettent*. La lexicalisation des arbres permet d'obtenir une grammaire PCFG lexicalisée contenant des règles conditionnées par les mots de la phrase, comme par exemple :

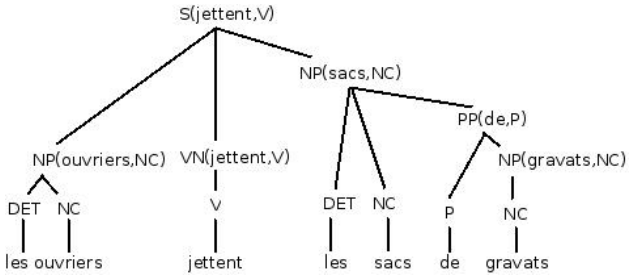
$$S(jettent) \rightarrow NP(ouvriers)VN(jettent)NP(sacs) \tag{2.13}$$

De fait, ces règles lexicalisées permettent de modéliser des affinités entre les mots de la phrase. Une variante de la lexicalisation consiste à faire remonter également les étiquettes morpho-syntaxiques des mots têtes. La figure 2.11b montre cette variante sur l'arbre lexicalisé par les mots têtes. Une grammaire lexicalisée souffre cependant d'un problème de dispersion des données. Les règles deviennent trop spécifiques pour espérer obtenir une bonne estimation des probabilités à partir du corpus. En effet, la probabilité de la règle 2.13 se calcule par la formule

[39]Par exemple, le syntagme nominal *Les ouvriers du bâtiment* a pour tête lexicale le nom *ouvriers*.




(a) mots têtes




(b) ajout des étiquettes des mots têtes

FIG. 2.11: Exemples de lexicalisation des arbres par les têtes.

suivante :

$$\begin{aligned} &P(S(jettent,V) \rightarrow NP(ouvriers,NC)VN(jettent,V)NP(sacs,NC)) \\ &= \frac{occ(S(jettent,V) \rightarrow NP(ouvriers,NC)VN(jettent,V)NP(sacs,NC))}{occ(S(jettent))} \end{aligned} \tag{2.14}$$

Selon (Charniak, 1997), on peut résoudre ce problème efficacement en décomposant le membre droit d'une règle PCFG comme :

$$LHS \rightarrow L_n L_{n-1} ... L_1 H R_1 ... R_{n-1} R_n \tag{2.15}$$

où $LHS$ est le membre gauche de la règle. La tête lexicale de la règle *H* possède à sa gauche une séquence de non-terminaux $L_{1...n}$ ainsi qu'une autre séquence de non-terminaux à droite $R_{1...n}$. Cette formulation particulière d'une règle permet de découper le calcul de la probabilité en sous produit de probabilités. Ce produit est obtenu en utilisant un algorithme génératif à base d'historique qui ajoute des hypothèses d'indépendance entre les règles. Il génère d'abord la tête lexicale de la règle puis génère un par un les dépendants de la tête. Chaque génération de symbole conduit à calculer une probabilité indépendante. Un symbole spécial *STOP* est ajouté à gauche et à droite du membre droit de chaque règle. Ce symbole permet d'indiquer à l'algorithme génératif que la génération se termine. La règle 2.15 devient :

$$LHS \rightarrow STOP\ L_n L_{n-1} ... L_1 H R_1 ... R_{n-1} R_n\ STOP \tag{2.16}$$

Ainsi, les symboles non-terminaux à gauche de la tête sont générés jusqu'à rencontrer le symbole STOP. Puis, la génération continue à droite de la tête jusqu'au symbole STOP. Formellement, le calcul de la probabilité de la règle 2.16 est effectué de la façon suivante, avec $h$ la tête lexicale de la règle, $l$ et $r$ les têtes lexicales des symboles du membre droit, et enfin, $w$ fait référence au mot tête et $t$ à l'étiquette morpho-syntaxique de cette tête :

1. Génération de la tête lexicale $H(hw, ht)$ ayant une probabilité $P_H(H(hw, ht)|hw, ht)$.
2. Génération des dépendants gauches de la tête $\prod_{i=1}^{n+1} P_L(L_i(lw_i, lt_i)|H, hw, ht)$ avec $L_{n+1}(lw_{n+1}, lt_{n+1}) = STOP$.
3. Génération des dépendants droits de la tête $\prod_{i=1}^{n+1} P_R(R_i(rw_i, rt_i)|H, hw, ht)$ avec $R_{n+1}(rw_{n+1}, rt_{n+1}) = STOP$.

Nous montrons dans la figure 2.12 les différentes étapes de la génération de la règle 2.14 ainsi que la probabilité associée à chaque nouveau symbole généré.

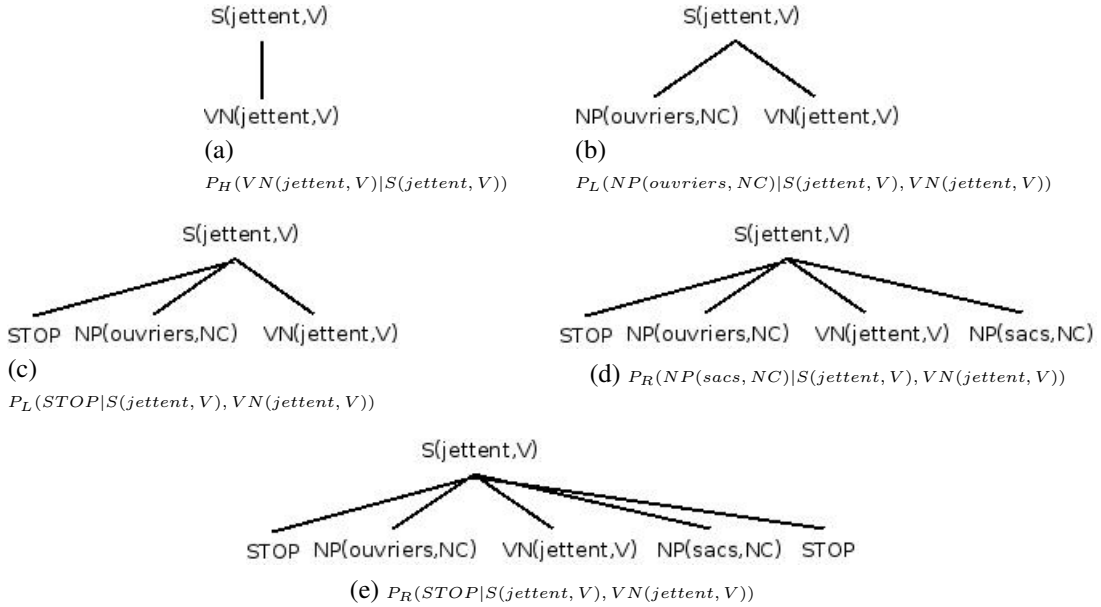



FIG. 2.12: Génération de la règle 2.14.

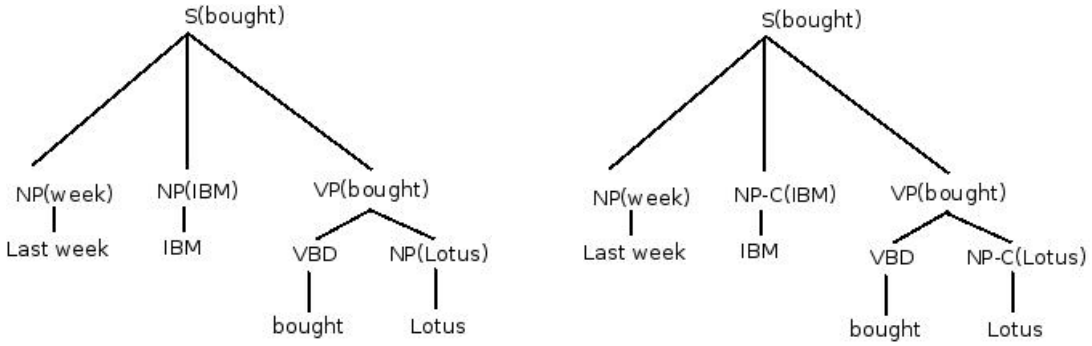
La probabilité de la règle 2.14 est donc maintenant égale au produit de sous-probabilités suivant :

$$\begin{aligned} P_H(VN(jettent, V)|S(jettent, V)) \quad & \times P_L(NP(ouvriers, NC)|S(jettent, V), VN(jettent, V)) \\ & \times P_L(NP(ouvriers, NC)|S(jettent, V), VN(jettent, V)) \\ & \times P_L(STOP|S(jettent, V), VN(jettent, V)) \\ & \times P_R(NP(sacs, NC)|S(jettent, V), VN(jettent, V)) \\ & \times P_R(STOP|S(jettent, V), VN(jettent, V)) \end{aligned} \tag{2.17}$$

Dans cette approche, la probabilité de chaque règle est conditionnée par quelques informations comme la tête lexicale ainsi que le membre gauche de la règle. (Collins & Singer, 1999) reprennent cette approche en ajoutant d'autres informations de conditionnement qu'ils considèrent comme pertinentes afin d'améliorer les performances générales. (Collins, 1996) a, par exemple, montré que la distance entre la tête et les dépendants est une information importante car elle permet, en particulier, de capturer des préférences de rattachement pour les relations de longue distance. Ainsi, les équations de calcul des sous-probabilités sont réécrites de la façon suivante :

$$\begin{aligned} & P_L(L_i(lw_i, lt_i)|H, hw, ht, distance(i-1)) \\ & P_R(R_i(rw_i, rt_i)|H, hw, ht, distance(i-1)) \end{aligned} \tag{2.18}$$

où $distance$ est une fonction renvoyant la distance entre le symbole contenant la tête de la règle et le symbole courant. Collins pose également l'hypothèse que l'ajout d'informations de sous-catégorisation, à travers la distinction entre les compléments essentiels des verbes et les simples modifieurs, pourrait améliorer la qualité de l'analyse. Par exemple, l'arbre de la phrase *Last week IBM bought Lotus* (*La semaine dernière IBM a acheté Lotus*), montré dans la figure 2.13a, ne modélise aucune information qui pourraient guider l'apprentissage du modèle sur le rôle respectif de chacun des syntagmes nominaux présents autour du verbe *bought*. Pour ce faire, Collins a appliqué des règles créées manuellement sur les arbres du corpus. Ces règles identifient tout d'abord les compléments des verbes, puis, ajoutent un suffixe *-C* au symbole du noeud non-terminal correspondant. La figure 2.13b montre l'arbre de la figure 2.13a après l'application de ces règles. Par cette méthode, un suffixe *-C* a été ajouté aux compléments, sujet (syntagme nominal de tête *IBM*) et objet direct (syntagme nominal de tête *Lotus*), du verbe *bought*. Afin de maximiser l'impact de ces informations de sous-catégorisation, celles-ci sont également présentes dans le conditionnement des probabilités des règles.




(a) arbre original (b) arbre modifié par les règles

FIG. 2.13: Application des règles de distinction entre compléments et modifieurs sur l'arbre de la phrase *Last week IBM bought Lotus*.

La complexification des règles de la grammaire (lexicalisation, contextes divers,...) peut entraîner un éventuel problème de dispersion des données. En effet, si un symbole, situé à droite de la tête lexicale de la règle, n'est jamais associé à cette tête dans le corpus d'apprentissage alors la probabilité de cette règle et donc celle de la règle globale sont égales à 0. Pour remédier au problème, Collins propose un algorithme de lissage qui conditionne les probabilités selon un contexte variable. Le contexte de base est composé du mot et de l'étiquette de la tête ainsi que du symbole du membre gauche. Si la probabilité conditionnée par ce contexte vaut 0 alors, le mot tête est ôté du contexte et la probabilité est recalculée. Enfin, si le résultat obtenu est identique, le calcul est effectué avec un contexte final composé uniquement du membre gauche. Les statistiques permettant de lisser les probabilités sont extraites au moment de l'apprentissage de la grammaire.
Nous avons indiqué dans le tableau 2.3 les performances (score $F_1$) de l'analyseur de Collins sur le corpus d'évaluation du PTB selon les diverses informations ajoutées dans le conditionnement des probabilités des règles. Nous appelons `m1`, le modèle intégrant la notion de distance et `m2` celui faisant la distinction entre compléments et modifieurs. À titre de comparaison, nous indiquons les résultats obtenus par (Charniak, 1997) qui reposent sur un modèle moins com-

plexe que celui de Collins. On peut voir que les gains obtenus par les deux modèles de Collins sont significatifs par rapport à l'analyseur de Charniak, ce qui valide ses hypothèses. L'ajout d'informations de sous-catégorisation des verbes semble malgré tout avoir plus d'impact sur les performances que la notion de distance.

| Modèle | $F_1$ |
|---|---|
| (Charniak, 1997) | 86.6 |
| m1 | 87.6 |
| m2 | **88.2** |

TAB. 2.3: Performances de l'analyseur de Collins sur le corpus d'évaluation du PTB selon le modèle utilisé.

(Collins & Singer, 1999) ont prouvé que le conditionnement des probabilités par un contexte riche permet d'améliorer significativement les performances générales. Cependant, plus le contexte est conséquent plus l'estimation des probabilités devient difficile. (Charniak, 2000) a proposé une solution efficace à ce problème en s'inspirant du fonctionnement des modèles discriminants (section 2.4). Ces modèles complexes ont l'avantage de pouvoir intégrer des millions d'informations, appelées *traits*, tirées des observations faites sur le corpus sans poser d'hypothèses d'indépendance entre ces traits. Afin de simuler ce principe dans le formalisme PCFG, Charniak a modifié les équations de calcul des probabilités conditionnelles. Cela lui permet d'intégrer de nombreuses informations supplémentaires comme les symboles des frères du membre gauche de la règle ou encore le symbole du père de celui-ci. Cette approche a été implémentée dans l'analyseur Brown[40] (Charniak, 2000; Charniak & Johnson, 2005).
Dans le tableau 2.4, nous avons indiqué les résultats (score $F_1$) sur le corpus d'évaluation du PTB des trois analyseurs lexicalisés décrits dans cette section (Charniak, 1997; Collins & Singer, 1999; Charniak, 2000). On peut voir que les analyseurs intégrant de multiples informations au conditionnement des probabilités sont les plus performants, avec un avantage significatif pour celui de Brown grâce à son formalisme PCFG particulier.

| Analyseur | $F_1$ |
|---|---|
| (Charniak, 1997) | 86.6 |
| (Collins & Singer, 1999) | 88.2 |
| Brown | 89.5 |

TAB. 2.4: Performances des analyseurs lexicalisés sur le corpus d'évaluation du PTB

### 2.3.4 Évaluation comparative des performances de plusieurs analyseurs PCFG sur le FTB-UC

Le formalisme PCFG permet de modéliser efficacement un processus syntaxique tout en étant simple dans sa description. Cette simplicité est malgré tout établie par deux contraintes sur

[40] `http://www.cog.brown.edu/~mj/Software.htm`

les règles de la grammaire que sont le faible conditionnement lexical et les hypothèses d'indépendances fortes entre ces règles. Ces deux contraintes nuisent aux performances générales de la grammaire. Nous avons donc vu émerger, au cours de la dernière décennie, deux courants majeurs qui tentent d'atténuer les effets de ces contraintes. Le premier, basé sur le paradigme non lexicalisé (Klein & Manning, 2003; Matsuzaki *et al.*, 2005; Petrov & Klein, 2007; Petrov, 2010), réduit l'effet produit par les hypothèses d'indépendances fortes entre les règles. Des divisions de symboles sont effectuées afin d'intégrer des informations contextuelles plus nombreuses aux règles. Quant au paradigme lexicalisé (Charniak, 1997; Collins & Singer, 1999; Charniak, 2000; Collins, 2003), il tente de répondre au problème du faible conditionnement lexical par l'ajout d'informations lexicales comme les têtes des constituants. Nous avons regroupé dans le tableau 2.5 les résultats en terme de score $F_1$ des principaux analyseurs lexicalisés, (Collins & Singer, 1999) et Brown (Charniak, 2000) ainsi que non lexicalisés, BKY (Petrov & Klein, 2007) et BKY$_p$ (Petrov, 2010). On peut voir que les performances sont presque identiques sur l'anglais quel que soit le paradigme utilisé, excepté pour l'analyseur basé sur le produit de grammaires introduit par (Petrov, 2010).

| Analyseur | Paradigme | $F_1$ |
|---|---|---|
| (Collins & Singer, 1999) | lexicalisé | 88.2 |
| Brown | lexicalisé | 89.5 |
| BKY | non lexicalisé | 90.1 |
| BKY$_p$ | non lexicalisé | **91.8** |

TAB. 2.5: Performances, sur le corpus d'évaluation du PTB, d'analyseurs lexicalisés (Collins & Singer, 1999) et Brown, et non lexicalisés BKY et BKY$_p$.

Afin d'observer le comportement de ces analyseurs sur notre langue d'étude à savoir le français, nous avons indiqué, dans les tableaux 2.6 et 2.7, leurs performances ($F_1$, UAS et LA) sur les corpus de développement et d'évaluation du FTB-UC. L'entraînement des analyseurs a été effectué sur le corpus d'apprentissage du FTB-UC. Pour les analyseurs lexicalisés Brown et StanfordParser[41], la table de propagation des têtes décrite par (Dybro-Johansenn, 2004) nous a permis de lexicaliser les arbres de constituants. En ce qui concerne les analyseurs non lexicalisés, nous avons généré 8 grammaires pour BKY, chacune ayant une valeur de biais comprise entre 1 et 8. L'évaluation a été effectuée en calculant la moyenne des résultats obtenus par chaque grammaire. Nous avons réutilisé ensuite ces 8 grammaires pour l'approche par produit de grammaires BKY$_p$. D'après les résultats obtenus sur les deux corpus, on peut observer que les analyseurs non lexicalisés ont des performances significativement meilleures que celles des analyseurs lexicalisés, et ceci tout score confondu. Bien que la lexicalisation des grammaires semble inadaptée pour le français, (Seddah *et al.*, 2009a) ont toutefois nuancé ce constat. En effet, ils ont constaté que la courbe d'apprentissage[42] de Brown est parallèle à celle de BKY. Cela voudrait dire que certains analyseurs lexicalisés pourraient obtenir de meilleures perfor-

[41]Le StanfordParser est une version lexicalisée de l'algorithme à base de règles de division de (Klein & Manning, 2003). Nous utilisons la version 2.0.1 adaptée pour le français par (Green *et al.*, 2011), disponible à l'adresse `http://nlp.stanford.edu/software/lex-parser.shtml`

[42]Une courbe d'apprentissage indique les performances d'un analyseur en fonction de la quantité de données d'entraînement utilisée.

mances si le corpus d'apprentissage était de plus grande taille, comme c'est le cas du PTB pour l'anglais. Les analyseurs lexicalisés souffriraient donc plus du problème de la dispersion des données.

| Analyseur | Paradigme | $F_1$ | UAS | LA |
|---|---|---|---|---|
| Brown | lexicalisé | 79.59 | 85.41 | 83.8 |
| StanfordParser | lexicalisé | 77.43 | 84.01 | 82.8 |
| BKY | non lexicalisé | 82.63 | 87.67 | 86.45 |
| $BKY_p$ | non lexicalisé | **84.05** | **88.62** | **87.6** |

TAB. 2.6: Performances, sur le corpus de développement du FTB-UC, d'analyseurs lexicalisés (Brown et StanfordParser) et non lexicalisés (BKY et $BKY_p$).

| Analyseur | Paradigme | $F_1$ | UAS | LA |
|---|---|---|---|---|
| Brown | lexicalisé | 80.86 | 85.96 | 84.5 |
| StanfordParser | lexicalisé | 77.43 | 84.01 | 82.8 |
| BKY | non lexicalisé | 84.26 | 88.22 | 86.77 |
| $BKY_p$ | non lexicalisé | **85.51** | **89.04** | **87.7** |

TAB. 2.7: Performances, sur le corpus d'évaluation du FTB-UC, d'analyseurs lexicalisés (Brown et StanfordParser) et non lexicalisés (BKY et $BKY_p$).

## 2.4 Modèles discriminants pour l'analyse syntaxique

Les performances des ordinateurs se sont considérablement accrues au cours des dernières décénnies. De nombreux analyseurs syntaxiques basés sur des modèles probabilistes complexes ont vu le jour. Ces modèles, qualifiés de *discriminants*, demandent beaucoup de ressources mais possèdent de nombreux avantages. Quatre modèles discriminants majeurs sont utilisés dans le cadre de l'analyse syntaxique probabiliste. L'algorithme *Maximum d'Entropie*[43] [ME] (Jaynes, 1957; Berger *et al.*, 1996) consiste à utiliser un ensemble de données d'apprentissage afin de construire un modèle probabiliste qui va s'adapter à ces données. Par exemple, les traders de la finance construisent des modèles statistiques qui prédisent le cours de la bourse des prochains jours à partir des prix du marché des jours précédents. De la même façon, un modèle ME va tenir compte uniquement des informations disponibles afin de déterminer une loi de probabilité. On va donc chercher un modèle qui maximise l'entropie, le maximum d'informations issues du corpus d'apprentissage, tout en tenant compte de contraintes sur ces informations. Dans le cadre de l'analyse syntaxique, le modèle ME a été privilégié de nombreuses années du fait de sa simplicité de modélisation et d'utilisation (Ratnaparkhi, 1999; Collins & Koo, 2005; Charniak & Johnson, 2005). D'autres analyseurs plus récents (Huang, 2008; Finkel *et al.*, 2008; Tsuruoka *et al.*, 2009) font état de l'utilisation de l'approche appelée *Champs conditionnels*

[43] Aussi appelée Entropie Maximale.

*aléatoires* (Lafferty *et al.*, 2001), qui est une généralisation des algorithmes ME. Elle permet de modéliser des structures complexes, et notamment des séquences et des arbres. Empiriquement, ce type de modèle donne de meilleures performances que le modèle ME. Et enfin, il existe également une famille d'algorithmes d'apprentissage sous le nom de *classifieur linéaire* dont font partie les modèles Perceptron (Rosenblatt, 1958) et Séparateurs à Vastes Marges (Boser *et al.*, 1992) [SVM]. Le rôle d'un classifieur est de regrouper les échantillons qui ont des propriétés similaires qui ont été mesurées sur des observations. Un classifieur linéaire est un type particulier de classifieur, qui calcule la décision par combinaison linéaire des échantillons. Quelques analyseurs se basent sur de tels algorithmes (Yamada & Matsumoto, 2003; Collins & Roark, 2004; Sagae & Lavie, 2005).
Dans les sections 2.4.1 et 2.4.2, nous décrivons formellement et en détail deux modèles discriminants, ME et CRF. Puis, sections 2.4.3 et 2.4.4, nous détaillons les méthodes permettant d'estimer les paramètres de ces modèles. Nous citons également quelques travaux qui se sont penchés sur l'optimisation du temps de calcul à l'apprentissage du modèle qui peut être très long. En effet, les modèles discriminants sont souvent amenés à traiter avec des millions de paramètres rendant la tâche d'apprentissage très complexe. De nos jours, nombreux sont les analyseurs syntaxiques qui profitent de la puissance des modèles discriminants. Bien que certains analyseurs en constituants discriminatifs (Finkel *et al.*, 2008; Tsuruoka *et al.*, 2009) atteignent des performances au niveau de l'état de l'art, ils restent en deça des analyseurs PCFG (Charniak, 2000; Petrov & Klein, 2007). Aussi, dans la section 2.4.5, nous verrons uniquement des analyseurs en dépendances discriminatifs. Pour finir, nous nous intéresserons à l'approche appelée *réordonnancement discriminatif* qui peut être vue comme un post-traitement d'un analyseur syntaxique. Les analyses les plus probables renvoyées par ce dernier sont ordonnées différement par le réordonnanceur grâce à des informations plus riches. Cette approche peut aussi bien être utilisée pour l'analyse en constituants que pour celle en dépendances (LeRoux *et al.*, 2011).

### 2.4.1 Maximum d'Entropie

Le problème principal de la modélisation stochastique d'un processus aléatoire est de savoir quelles informations d'un ensemble de données d'apprentissage sont pertinentes. Par exemple, supposons que l'on veuille construire le modèle d'un processus d'étiquetage morpho-syntaxique. Le modèle assigne pour chaque étiquette *t* une probabilité $p(t|h)$ que le processus prédise *t* étant donné un contexte *h*. Quelles sont les informations à utiliser dans $h$ afin de déterminer une distribution de probabilités cohérente ?
L'algorithme de modélisation ME peut être vu comme deux étapes distinctes. La première étape consiste à tirer une série d'observations du corpus, que l'on appelle *traits*. Ces traits sont ensuite intégrés au modèle stochastique. Le but est d'obtenir un modèle uniforme sous contraintes de ces traits afin qu'il puisse refléter la distribution initiale du corpus. Dans les sections qui suivent, nous répondons à deux questions essentielles liées au modèle ME :
– Comment trouver un modèle à partir des contraintes sur les traits ?
– Qu'est ce qu'un modèle uniforme et comment mesurer cette uniformisation ?

On peut noter que le modèle ME a beaucoup d'adeptes et est utilisé dans de nombreux domaines du TAL comme l'étiquetage morpho-syntaxique (Ratnaparkhi, 1996; Toutanova & Manning, 2000; Denis & Sagot, 2009), la classification de documents (Nigam, 1999; Zhu *et al.*, 2005) ou encore l'extraction d'informations (McCallum *et al.*, 2000; Chieu & Ng, 2002).

### Définitions

On suppose qu'un processus aléatoire génère une variable *y*, élément d'un ensemble fini $Y$, conditionné par un éventuel contexte *x*, élément d'un ensemble fini $X$. Dans le cadre de l'étiquetage morpho-syntaxique, $y$ est une étiquette morpho-syntaxique et $x$ un historique (ou contexte) restant à définir. Le but est de construire un modèle stochastique qui reflète le comportement de ce processus aléatoire. Il s'agit donc d'une méthode d'estimation de la probabilité conditionnelle $p(y|x)$ que le processus génère $y$ étant donné un contexte $x$ (classification). Afin de déterminer le comportement de ce processus, la première étape consiste à tirer des observations d'un corpus $C$ constitué de paires d'exemples $(x, y)$ tel que $C = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$. Le corpus d'apprentissage peut être représenté par la distribution de probabilités $\hat{p}$, définie par :

$$\hat{p}(x, y) = \frac{1}{N} * nb_{occ}(x, y) \tag{2.19}$$

où $nb_{occ}(x, y)$ est le nombre d'occurrence de la paire $(x, y)$ dans le corpus, normalisée par la taille $N$ du corpus. Cette probabilité est tout simplement la distribution des paires $(x, y)$ distinctes dans le corpus. Le but est de calculer une loi de probabilité permettant de modéliser le processus générant le corpus d'apprentissage $\hat{p}$. Comme énoncé précédemment, on souhaite extraire du corpus un ensemble d'événements statistiquement pertinents à inclure dans le modèle. Pour prédire une étiquette $t$ d'un mot $w$ en utilisant un modèle de Markov caché (section 3.2.1), on utilise deux types de probabilités. La première dite d'*émission*, formellement $p(t|w_i)$, se sert d'un contexte lexical $w_i$ (le mot à étiqueter). La deuxième probabilité, dite de *transition*, formellement $p(t|t_1...t_{i-1})$, se sert des $i^{eme}$ précédentes étiquettes. Deux types d'informations contextuelles sont donc utilisées pour déterminer la probabilité de prédire une étiquette $t$. Dans un modèle Maximum-Entropy, cette notion de contextualisation est généralisée à de multiples possibilités. On pourra utiliser par exemple le mot précédent dans la phrase $w_{i-1}$, la troisième étiquette précédente $t_{i-3}$ ou encore les suffixes du mot $suffixe(w_i)$.
Ces contextes, aussi appelés *traits* dans le jargon des modèles discriminants, sont extraits du corpus d'apprentissage pour chaque symbole $y$ à prédire. Afin d'obtenir ces traits, des patrons sont appliqués sur les paires d'exemples du corpus[44]. Un patron est défini par au moins un paramètre, à savoir $y$. Des paramètres contextuels $x$ peuvent également être capturés. Par exemple, on peut définir un patron $f(y = t, x = w_i)$ qui va capturer le fait que l'étiquette $t$ ait comme contexte lexical $w_i$. Formellement, un trait $f_i$ obtenu à partir d'un patron $f$ est représenté par une fonction booléenne qui renvoie 1 si la paire courante du corpus contient

[44]Il est néanmoins possible de modéliser des traits qui ont été créés de façon manuelle.

l'événement (x',y'), 0 sinon :

$$f_i(x,y) = \{ \begin{array}{l} \text{1 si les conditions sont réunies pour la paire (x,y), i.e x=x' et y=y'} \\ \text{0 sinon} \end{array} \quad (2.20)$$

Par exemple, supposons qu'un événement est défini par le fait que l'étiquette *NC* soit assignée au mot *chien* si celui-ci est précédé du mot *le*. La fonction booléenne correspondante à ce trait est :

$$f_i(x,y) = \{ \begin{array}{l} \text{1 si x=le chien et y=NC} \\ \text{0 sinon} \end{array} \quad (2.21)$$

La valeur attendue de chaque fonction $f_i$ est estimée à partir de la distribution empirique $\hat{p}(x,y)$ du corpus. Elle est obtenue en comptant simplement le nombre d'occurrences du trait dans les données d'apprentissage et en prenant en compte les différentes valeurs des variables (pour toutes les valeurs de $x$ et $y$) :

$$\hat{p}(f_i) = \sum_{x,y} \hat{p}(x,y) f_i(x,y) \quad (2.22)$$

Plus concrètement, cette équation signifie que la valeur $\hat{p}(f_i)$ est égale au nombre d'occurrences du trait $f_i$ ayant la valeur 1 dans le corpus divisé par la taille $N$ de $C$[45]. Quand un événement $f_i$ semble pertinent, la procédure consiste à l'intégrer au modèle. Cela signifie qu'il faut calculer également sa valeur attendue d'après la distribution du modèle $p(y|x)$ :

$$p(f_i) = \sum_{x,y} \hat{p}(x) p(y|x) f_i(x,y) \quad (2.23)$$

où $\hat{p}(x)$ est la distribution de $x$ dans le corpus.
Le modèle doit s'adapter aux données d'apprentissage. Cela signifie que, pour chaque trait $f_i$, sa valeur attendue dans la distribution empirique doit être égale à celle présente dans la distribution du modèle. On contraint donc notre modèle pour que $p(f_i) = \hat{p}(f_i)$. Cette égalité est appelée une fonction de contrainte. En établissant celle-ci, on élimine les modèles qui ne seraient pas en adéquation avec les observations faites sur le corpus.

### Modélisation

On suppose que l'on a en entrée *n* traits $f_i$ qui représentent des événements utiles du processus à modéliser. En sortie de l'algorithme d'estimation, le modèle doit valider la fonction de contrainte pour chaque trait de l'ensemble. Formellement, on souhaite obtenir un modèle $p$ d'un sous-ensemble $S$ de $P$ des modèles respectant les fonctions de contraintes, tel que :

$$S \equiv \{p \in P | p(f_i) = \hat{p}(f_i) \text{for i } \in \{1,2,...,n\}\} \quad (2.24)$$

D'après cette équation, l'ensemble $S$ peut contenir plusieurs modèles mais la modélisation ME impose que la distribution sélectionnée soit la plus uniforme possible. L'uniformité d'une distribution sous contraintes $p(y|x)$ est définie par :

$$H(p) = -\sum_{x,y} \hat{p}(x) p(y|x) log p(y|x) \quad (2.25)$$

[45] $\hat{p}(x,y)$ peut être remplacé par l'équation 2.19.

Cette équation est bornée par le bas par 0 et par le haut par $log|Y|$ avec $Y$ les valeurs de $y$. Le principe de la modélisation ME consiste à sélectionner le modèle $p_\star \in S$ ayant la meilleure entropie $H(p)$ :

$$p_\star = argmax_{p \in S}\ H(p) \tag{2.26}$$

On peut également démontrer qu'il existe un unique modèle $p_\star$ qui satisfait cette équation dans l'ensemble $S$. Il faut simplement résoudre le système d'équations suivant :

$$\begin{array}{l} p_\star = argmax\ H(p) \\ C = \{p \in P | p(f_i) = \hat{p}(f_i) \text{for i } \in \{1, 2, ..., n\}\} \\ p(f) = \sum_{x,y} \hat{p}(x) p(y|x) f(x, y) \\ H(p) = -\sum_{x,y} \hat{p}(x) p(y|x) log p(y|x) \end{array} \tag{2.27}$$

La solution de ce système est donné par l'équation :

$$p(y|x) = \frac{1}{Z(x)} \prod_{i=1}^{n} \alpha_i f_i(x, y) \tag{2.28}$$

Avec Z(x) un facteur de normalisation et $\alpha_1 ... \alpha_n$ les paramètres du modèle. Chacun des paramètres $\alpha_i$ est une fonction exponentielle qui représente le poids d'un trait $f_i$.

### 2.4.2 Champs conditionnels aléatoires

(Lafferty *et al.*, 2001) ont introduit un nouveau modèle appelé *Champs conditionnels aléatoires*[46], ou CRF, qui est à la fois une version séquentielle des modèles ME et une généralisation des modèles markoviens cachés [MMC] utilisés pour l'étiquetage de séquences (section 3.2.1). Nous rappelons que le modèle discriminant ME permet de calculer une probabilité $p(y|x)$ d'avoir un symbole $y$ étant donné un contexte $x$ composé d'une multitude de traits. Les modèles ME sont très utilisés dans les tâches de classification. Quant aux modèles génératifs MMC, ils permettent de modéliser la distribution de probabilités $p(\overrightarrow{y}|\overrightarrow{x})$ d'avoir une séquence de symboles $\overrightarrow{y}$ étant donnée une séquence d'observations $\overrightarrow{x}$. Ce dernier souffre de deux défauts majeurs. Tout d'abord les hypothèses d'indépendances entre les observations sont très fortes. Chaque observation est liée à une variable de sortie $y$ spécifique. Dans un modèle CRF, de telles hypothèses n'existent pas car les observations sont interdépendantes. Une variable à prédire est ainsi conditionnée sur la globalité du contexte (normalisation globale). De plus, les MMC ne permettent de traiter qu'avec des structures linéaires (étiquetage morpho-syntaxique, analyse syntaxique de surface,...) qui sont incompatibles avec les structures arborées. Le modèle CRF permet potentiellement de modéliser tout processus traitant avec des structures complexes.
Les CRF associent à une observation $x$ une annotation $y$ en se basant sur un ensemble d'exemples annotés, c'est-à-dire un ensemble de couples $(x, y)$. La plupart du temps $x$ est une séquence d'unités et $y$ la séquence des étiquettes correspondante. Les CRF sont définis par $X$ et $Y$, deux champs aléatoires décrivant respectivement chaque unité de l'observation $x$ et son annotation

[46] Aussi appelé Conditional Random Fields en anglais.

$y$, et par un graphe $G = (V, E)$ dont $V = X \cup Y$ est l'ensemble des noeuds et $E \subseteq V \times V$ l'ensemble des arcs. Deux variables sont reliées dans le graphe si elles dépendent l'une de l'autre. Le graphe sur le champ $Y$ des CRF dits linéaires[47], dessiné en figure 2.14, traduit le fait que chaque symbole $y$ est supposé dépendre du symbole précédent et du suivant et, implicitement, de la donnée $x$ complète. En effet, dans le graphe, chaque variable $Y_i$ est reliée à chaque variable du champ $X$.

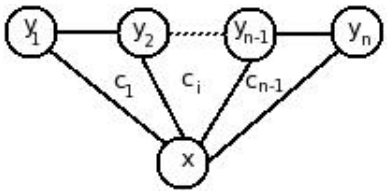



FIG. 2.14: Graphe associé à un CRF linéaire.

L'équation générale d'un CRF pour tout type de structure modélisée est la suivante :

$$p(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \Psi_c(x, y_c, c) = \frac{1}{Z(x)} \prod_{c \in C} exp(\sum_k \alpha_k f_k(y_c, x, c)) \tag{2.29}$$

Avec,

- $C$, l'ensemble des cliques (sous-graphes complètement connectés) de $G$ sur $Y$ : dans le cas du graphe de la figure 2.14, ces cliques ($c_1$, $c_i$, $c_n$) sont constituées soit d'un noeud isolé, soit d'un couple de noeuds successifs.
- $y_c$, l'ensemble des valeurs prises par les variables de $Y$ sur la clique $c$ pour un $y$ donné.
- $Z(x)$ est un coefficient de normalisation, défini de telle sorte que la somme sur $y$ de toutes les probabilités $p(y|x)$ pour une donnée $x$ fixée soit égale à 1.
- Les fonctions $f_k$ sont appelées fonctions caractéristiques (traits). Elles sont définies à l'intérieur de chaque clique $c$ et sont à valeurs réelles, mais souvent choisies pour donner un résultat binaire (0 ou 1). Elles doivent être fournies au système par l'utilisateur grâce aux patrons de traits. Par définition, la valeur de ces fonctions peut dépendre des symboles $y$ présents dans une certaine clique $c$ ainsi que de la valeur de $x$ n'importe où dans la donnée. En effet, les traits peuvent être non locaux, c'est à dire que les informations provenant de $x$ ne s'arrêtent pas uniquement aux indices correspondants à la clique $c$.
- Les poids $\alpha_k$, à valeurs réelles, permettent d'accorder plus ou moins d'importance à chaque fonction $f_k$ dont ils caractérisent le pouvoir discriminant. Ce sont les paramètres du modèle.

L'intérêt et l'efficacité des CRF proviennent du fait qu'ils prennent en compte des dépendances entre symboles reliés les uns aux autres dans le graphe. En cherchant le meilleur $y$ avec l'aide d'une donnée complète $x$, ils se comportent en général mieux qu'une série de classifications d'unités isolées comme le ferait un modèle ME. On peut d'ailleurs noter que la modélisation ME est en réalité un cas particulier de CRF. En effet, dans le graphe correspondant à un modèle ME, chaque clique est composée d'un unique symbole $y$ relié au contexte $x$ (classification), ce

[47] Aussi appelé *Linear-Chain CRF* en anglais.

qui nous donne l'équivalence suivante :

$$p(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \Psi_c(x, y_c, c) = \frac{1}{Z(x)} \prod_k \alpha_k f_k(x, y) \tag{2.30}$$

De nombreux travaux se sont penchés sur la modélisation CRF de structures autres que linéaires (Lafferty *et al.*, 2001; Sutton & McCallum, 2007; Finkel *et al.*, 2008). Dans le cadre de l'analyse syntaxique, nous sommes amené à traiter avec des arbres. Bien que ce type de structure soit non linéaire, un arbre peut néanmoins être vu comme une séquence de sous-arbres, un sous-arbre étant généré par une règle de dérivation d'une grammaire. Chaque sous-arbre peut donc être relié au contexte $x$ et former une clique. Ce type particulier de modélisation, illustré graphiquement dans la figure 2.15, est appelé CRF-CFG (Finkel *et al.*, 2008)

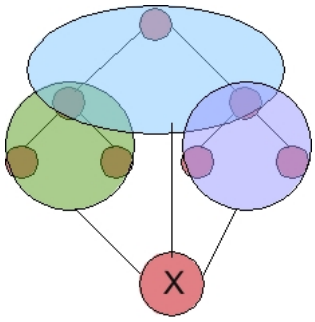



FIG. 2.15: Graphe d'une structure arborée modélisée par un CRF-CFG.

L'équation correspondante à un CRF-CFG est la suivante :

$$p(t|s, \overrightarrow{\alpha}, \overrightarrow{f}) = \frac{1}{Z(\overrightarrow{x})} \prod_{r \in t} \Psi(r, s, \overrightarrow{\alpha}, \overrightarrow{f}) \tag{2.31}$$

Le contexte $x$ est formé des mots de la phrase $s$. Les fonctions potentielles associées aux cliques prennent en paramètre une règle $r$ de la grammaire, le contexte et les fonctions caractéristiques $\overrightarrow{f}$ associées aux poids $\overrightarrow{\alpha}$.

### 2.4.3 Estimation des paramètres des modèles discriminants

L'estimation des paramètres des modèles discriminants est en général effectuée par des algorithmes itératifs d'optimisation pouvant prendre en compte efficacement des millions de paramètres. Il existe quatres méthodes majeures qui permettent de faire converger les valeurs des paramètres du modèle, *Generalized Iterative Scaling* [GIS] (Darroch & Ratcliff, 1972), *Improved Iterative Scaling* [IIS] (Della Pietra *et al.*, 1997) ainsi que des méthodes de premier et second ordres (méthode de Cauchy, quasi-Newton,...). Pour la plupart, ces algorithmes estiment les paramètres du modèle grâce à la méthode dite de Maximum de Vraisemblance (Maximum Likelihood en anglais). Prenons un échantillon de données d'observations $(x_1, ..., x_n)$ et un

modèle probabiliste $P_\alpha$ représentant une loi de probabilité. La vraisemblance consiste à quantifier la probabilité que les observations proviennent effectivement d'un échantillon théorique du modèle probabiliste. Elle est calculée par la formule suivante :

$$L_\alpha(x_1, ..., x_n) = \prod_{i=1}^{N} p_\alpha(x_i) \tag{2.32}$$

L'interprétation peut être vue de cette façon : on considère notre échantillon de variables aléatoires $(X_1, ..., X_n)$ du modèle $P$. Par définition, les variables sont indépendantes et dépendent toutes du même modèle $P$. Donc, la probabilité que l'échantillon théorique ait pour réalisation l'échantillon $(x_1, ..., x_n)$ est égal au produit des probabilités que $X_i$ prenne la valeur $x_i$.
Nous pouvons spécialiser cette équation pour notre problématique d'analyse syntaxique et plus particulièrement sur le modèle CRF. Les observations correspondent à l'ensemble des arbres $y_i$ conditionnés par les phrases $x_i$, nous définissant l'équation de vraisemblance suivante :

$$L_\alpha = \prod_{i=1}^{N} p_\alpha(\overrightarrow{y}_i | \overrightarrow{x}_i) \tag{2.33}$$

La vraisemblance logarithmique correspondante est définie par l'équation :

$$L_\alpha = \sum_{i=1}^{N} log\ p_\alpha(\overrightarrow{y}_i | \overrightarrow{x}_i) \tag{2.34}$$

La vraisemblance logarithmique est plus aisée à maximiser que la fonction classique car il s'agit d'additions de logarithmes et non plus des produits de facteurs. L'estimation d'un paramètre par la méthode de maximum de vraisemblance consiste à proposer comme valeur de ce paramètre celle qui rend maximale la vraisemblance, à savoir la probabilité d'observer les données comme réalisation d'un échantillon de la loi $P_\alpha$ :

$$\hat{\alpha} = \underset{\alpha}{argmax} L_\alpha \tag{2.35}$$

Afin de déterminer les valeurs des paramètres, il est nécessaire de calculer la fonction de dérivation partielle pour chacun des paramètres puis de déterminer le point où la dérivée est nulle. La valeur correspondant à ce point de la courbe est le paramètre optimal pour $L$. L'équation correspondante est :

$$\frac{\partial L}{\partial \alpha_j} = 0 \tag{2.36}$$

pour tous les paramètres $\alpha_j$ que l'on doit estimer. Tant que les dérivées sont non nulles, l'algorithme itératif d'optimisation doit assigner de nouvelles valeurs aux paramètres. (Malouf, 2002) a effectué une évaluation comparative des différentes méthodes d'optimisation sur plusieurs tâches automatiques comme l'analyse syntaxique de surface (Osborne, 2002a; Tan *et al.*, 2005) ou encore le résumé automatique de textes (Osborne, 2002b). Il semblerait que les méthodes de second ordre soient les plus performantes, tant en nombre d'itérations nécessaires qu'en terme d'optimalité des scores, et qu'au contraire les méthodes GIS et IIS, pourtant les plus utilisées, soient beaucoup moins efficaces.

### 2.4.4 Sélection pertinente des traits

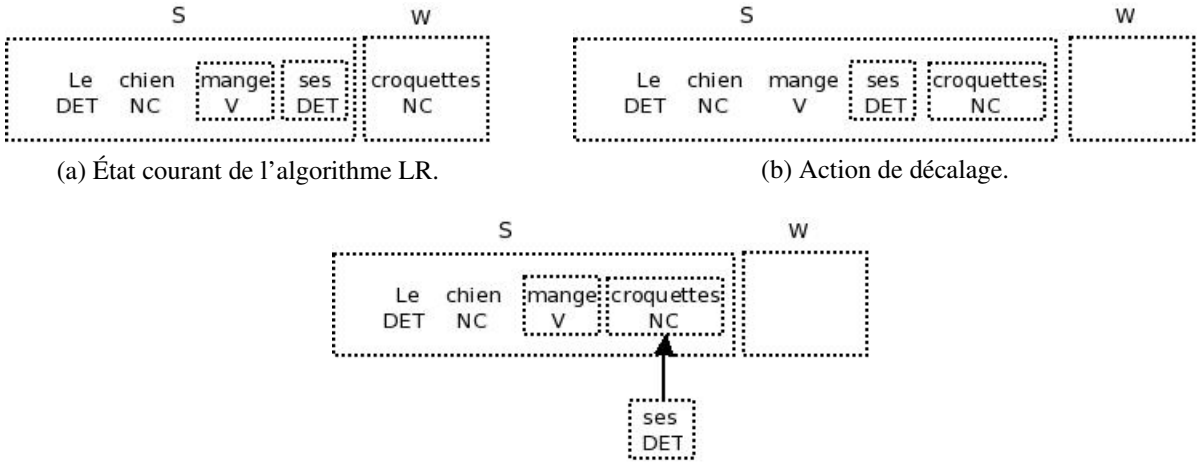
On suppose que l'on possède en entrée un ensemble de $n$ traits $f_i...f_n$. On souhaite inclure dans le modèle uniquement les traits les plus pertinents en terme d'impact sur ce modèle. En effet, un trait rare qui a peu d'occurrences dans le corpus aura un impact très limité sur les performances du modèle tout en ayant une estimation peu fiable de son poids discriminant. Cette problématique est à mettre en parallèle de la dispersion des données liée aux grammaires PCFG (section 2.3). De nombreux travaux ont proposés des méthodes de sélection automatique des traits (Ratnaparkhi, 1996; Berger *et al.*, 1996; Della Pietra *et al.*, 1997; McCallum, 2003; Zhou *et al.*, 2003; Riezler & Vasserman, 2004; Pal *et al.*, 2006; Lavergne *et al.*, 2010).
Un algorithme incrémental simple, décrit dans (Berger *et al.*, 1996; Della Pietra *et al.*, 1997), consiste, à chaque itération, à ajouter au modèle le trait qui maximise le gain obtenu par le modèle lorsqu'il est évalué sur les données d'apprentissage. Cette procédure peut être très lente et coûteuse en mémoire à cause du nombre non-négligeable de traits pouvant être extraits des données d'apprentissage. En effet, il n'est pas rare de devoir traiter avec des millions de traits. Une autre méthode, décrite par (Ratnaparkhi, 1996), considère qu'un trait est pertinent s'il apparaît au moins cinq fois dans le corpus. Dans ce cas, il doit donc être inclus dans le modèle, sinon il est supprimé de l'ensemble des traits. Cette heuristique permet de réduire la complexité en temps et en espace lors de l'apprentissage du modèle. Dans le cadre de l'étiquetage morpho-syntaxique, (Toutanova & Manning, 2000) fixent par exemple un nombre minimal d'occurrences de 5 pour les traits classiques et un nombre de 45 pour les traits utilisés pour le traitement des mots rares (section 3.3.1). D'autres travaux plus récents (Pal *et al.*, 2006; Lavergne *et al.*, 2010) [48] ont obtenu des résultats très concluants quant à l'optimisation du temps de calcul tout en gardant initialement la totalité des traits extraits.

### 2.4.5 Analyseurs en dépendances discriminatifs

De nos jours, les analyseurs en dépendances atteignent des scores élevés pour de nombreuses langues (Buchholz & Marsi, 2006; Nivre *et al.*, 2007), notamment grâce à des algorithmes exploitant la puissance des modèles discriminants. On peut distinguer deux types d'algorithmes majoritairement utilisés par les analyseurs en dépendances discriminatifs : l'analyse LR et l'analyse tabulaire. On peut noter que ces deux types d'algorithmes et donc la plupart des analyseurs considèrent l'étiquetage comme un processus externe préalablement effectué. Aussi, l'entrée des analyseurs en dépendances est un ensemble de phrases étiquetées, contrairement à une majorité d'analyseurs en constituants.
L'analyse LR (*left to right* en anglais), aussi appelée analyse *shift/reduce*, lit l'entrée de gauche à droite et produit en sortie une dérivation droite. Dans le cas présent, l'entrée est la séquence de mots de la phrase et la sortie est l'arbre de dépendances ayant la meilleure probabilité. Cet arbre est en général calculé par une analyse LR ascendante (*bottom-up* en anglais) qui tente de déduire les productions de haut niveau en les construisant à partir des feuilles. Un algorithme d'analyse LR fait intervenir deux structures de données, une pile $S$ et une queue $W$. Les éléments de $W$ sont des terminaux correspondant aux mots étiquetés de la phrase. Quant aux

[48] L'outil Wapiti (Lavergne *et al.*, 2010) est disponible à l'adresse `http://wapiti.limsi.fr/`

éléments de $S$, ils peuvent être des terminaux ou des sous-arbres de dépendances. À l'initialisation, $S$ est vide et $W$ est composée de l'ensemble des mots de la phrase ordonnés de façon à ce que le premier mot de cette phrase soit le premier élément de la queue. Dans un algorithme LR, seulement deux actions sont généralement autorisées : *décalage* (shift) et *reduction* (reduce). Une action de décalage consiste à extraire le premier élément de $W$ (un mot étiqueté) et à l'insérer ensuite en tête de la pile $S$. L'action de réduction consiste tout d'abord à extraire les $m$ premiers éléments de $S$. Puis, un nouvel élément, formé à partir des $m$ éléments précédemment extraits, est inséré en tête de $S$. Dans le cadre de l'analyse syntaxique en dépendances, le paramètre $m$ est fixé à 2 et représente une relation de dépendance "gouverneur/dépendant" entre deux mots de la phrase. Le nouvel élément inséré en tête de $S$ est le gouverneur de la relation de dépendance. Quant au dépendant, il est ajouté au sous-arbre lié au gouverneur et n'intervient plus dans l'algorithme. D'après cette définition de la réduction, l'action de décalage signifie que l'on ne génère pas de dépendances pour les deux mots courants. Si $S$ est vide alors seule une action de décalage est permise et au contraire, si $W$ est vide alors seule une action de réduction est autorisée. L'algorithme se termine lorsque $W$ est vide et $S$ contient un seul élément qui est le gouverneur de la phrase.
La figure 2.16 montre le résultat des deux types d'action sur la phrase étiquetée *Le chien mange ses croquettes*. L'état courant de l'algorithme est illustré par la figure 2.16a. On peut voir que la pile $S$ a pour mots têtes *mange* et *ses*. L'état de la pile après l'action de décalage vers la droite est illustrée par la figure 2.16b. On peut voir que les mots têtes de $S$ sont maintenant *ses* et *croquettes*. La figure 2.16c montre l'état de la pile $S$ après une action de réduction opérée sur les mots têtes *ses* et *croquettes*. Le gouverneur de la relation de dépendance étant *croquettes*, son dépendant *ses* est ajouté à son sous-arbre, puis ce gouverneur est réinséré en tête de $S$.




(a) État courant de l'algorithme LR.

(b) Action de décalage.

(c) Action de réduction.

FIG. 2.16: Schémas illustrant les deux actions autorisées par les analyseur LR : décalage et réduction.

Traditionnellement, les algorithmes LR prennent des décisions quant aux actions à effectuer grâce à une grammaire probabiliste. Cependant, la plupart des algorithmes LR pour l'analyse en dépendances sont basés sur un classifieur discriminatif qui donne une classe en sortie (une action) en fonction du contexte local présent dans l'analyse. L'entraînement d'un analyseur LR

| |
|---|
| pos($w_{i+j}$) = $X$, $j \in \{-2,-1,0,1,2,3\}$ |
| lex($w_{i+j}$) = $X$, $j \in \{-2,-1,0,1,2,3\}$ |
| ch-pos($w_{i+j}$) = $X$, $j \in \{-2,-1,0,1,2,3\}$ |
| ch-lex($w_{i+j}$) = $X$, $j \in \{-2,-1,0,1,2,3\}$ |

(a) Patrons de traits.

| |
|---|
| pos($w_{-1}$) = NC |
| lex($w_{-1}$) = chien |
| pos($w_1$) = NC |
| lex($w_1$) = croquettes |
| ch-pos($w_1$) = DET |
| ch-lex($w_1$) = ses |

(b) Quelques traits extraits pour les noeuds têtes *mange* et *croquettes*.

FIG. 2.17: (a) Patrons de traits utilisés par l'analyseur syntaxique en dépendances de (Yamada & Matsumoto, 2003). La taille des contextes gauche et droit a été fixée à 2. (b) Liste non exhaustive des traits extraits par l'application de ces patrons sur l'état de l'algorithme montré dans la figure 2.16c.

repose donc principalement sur l'apprentissage du classifieur. Pour cela, des instances d'entraînement sont nécessaires. Une instance est composée d'un ensemble de traits associé aux actions à effectuer. Afin de déterminer ces instances, une procédure standard consiste à appliquer l'algorithme LR sur les phrases du corpus d'apprentissage. Cependant, au lieu d'utiliser le classifieur pour déterminer les actions de l'analyseur, dans le cas présent, il s'agit uniquement de déterminer la liste des actions à effectuer pour obtenir l'arbre correct de la phrase, disponible dans le corpus d'apprentissage.

Les systèmes d'analyse basés sur un algorithme LR ne sont pas nouveaux. En effet, (Ratnaparkhi, 1997) l'utilisait déjà dans le cadre de l'analyse syntaxique en constituants. La particularité de son analyseur vient du fait qu'il est non déterministe. Il est capable de renvoyer plusieurs actions possibles à chaque étape, ce qui induit la création de plusieurs piles parallèles et donc l'exploration de plusieurs chemins. En sortie, cet algorithme non déterministe permet également de retourner les $n$ analyses les plus probables. Par la suite, (Yamada & Matsumoto, 2003) ont adapté avec succès l'analyseur de (Ratnaparkhi, 1997) pour l'appliquer à l'analyse en dépendances. Les patrons de traits qu'ils utilisent sont indiqués dans le tableau 2.17a. Pour ces patrons, les indices $i$ et $i+1$ représentent les deux noeuds têtes de $S$. Les contextes gauche (éléments précédents dans $S$) et droit (éléments de $W$) de ces noeuds ont donc des indices respectivement inférieurs à $i$ et supérieurs à $i+1$, sachant que la taille du contexte utilisé est un paramètre à définir manuellement. Les patrons dénotés `pos` et `lex` retournent des informations sur les mots et les étiquettes morpho-syntaxiques des noeuds têtes et des contextes. Quant aux patrons appelés `ch-pos` et `ch-lex`, ils extraient les mêmes informations mais cette fois-ci pour les noeuds fils des noeuds têtes et des contextes. La figure 2.17b montre l'ensemble des traits extraits à partir de l'état du système de la figure 2.16c avec une taille de contexte fixée à 2. Dans ce cas, les deux mots têtes sont *mange* et *croquettes*. Le contexte gauche est composé des noeuds *Le* et *chien*. Le contexte droit est vide car il n'y a plus d'éléments dans $W$. De plus, on peut remarquer que le noeud *croquettes* a un noeud fils *ses*.

Plus récemment, (Nivre, 2006; Nivre *et al.*, 2006; Nivre, 2008) ont ajouté de nouvelles fonc-

tionnalités qu'ils ont intégrées dans l'analyseur appelé *Malt*[49]. Dans leur algorithme, l'arbre de dépendances est déterminé non pas par une analyse LR ascendante classique mais par une analyse à la fois ascendante et descendante permettant ainsi de réduire drastiquement le temps de calcul nécessaire pour l'analyse d'une phrase. De plus, ils ont intègré de nouvelles actions permettant notamment de générer des dépendances typées contrairement aux autres analyseurs. Quant à l'analyseur proposé par (Sagae & Lavie, 2005), il possède une base similaire à celui de (Ratnaparkhi, 1997) en ce qui concerne l'analyse syntaxique en constituants. Cependant, en parallèle de la construction de l'arbre de constituants d'une phrase, l'arbre de dépendances syntaxiques correspondant est généré automatiquement. Cette opération est permise grâce à la lexicalisation de l'arbre de constituants au fur et à mesure de sa construction. Concrètement, les têtes lexicales sont ajoutées aux noeuds de l'arbre, ce qui permet d'extraire les dépendances bilexicales (section 1.3). La création en parallèle de deux structures arborées pour une même phrase permet au classifieur de prendre des décisions sur les actions en fonction de traits provenant de ces deux sources d'informations[50]. Une version non déterministe de cet algorithme est décrite dans (Sagae & Lavie, 2006).

Parallèlement aux expériences sur l'analyse LR, (McDonald *et al.*, 2005; McDonald, 2006) ont proposé un système d'analyse, appelé MST[51], basé sur la maximisation de la probabilité globale de l'arbre et non plus sur une suite de décisions locales. Pour ce faire, ils se sont inspirés des travaux réalisés sur les grammaires PCFG. La création d'un arbre de dépendances est effectuée à l'aide d'un algorithme tabulaire similaire à CKY (Kasami, 1965; Ney, 1991), très utilisé dans le cadre des PCFG (section 2.3.1). Les probabilités sont déterminées à partir d'un modèle appelé *Margin Infused Relaxed Algorithm* [MIRA] (Crammer & Singer, 2003; Crammer *et al.*, 2006) qui est de la même famille de modélisation que le perceptron[52]. Les patrons de traits exploités par MST sont relativement standard car, pour un noeud donné, ils extraient des informations sur ses fils (dépendants) ou encore sur son père (gouverneur). On peut noter que les algorithmes naïfs d'analyse tabulaire ont une complexité en $O(n^5)$. En effet, la forêt complète des arbres possibles pour la phrase en entrée a une taille en $O(n^5)$ et la complexité en temps pour la générer est également en $O(n^5)$. En se basant sur certaines observations empiriques, (Eisner, 1996) a néanmoins proposé un algorithme permettant réduire la complexité d'analyse à $O(n^3)$. Comme l'analyseur Malt, MST permet de générer des dépendances typées.

Dans le tableau 2.8, nous indiquons les scores UAS[53] obtenus par les différents analyseurs syntaxiques en dépendances que nous venons de décrire, sur le corpus d'évaluation du PTB-DEP. Les analyseurs (Yamada & Matsumoto, 2003; Sagae & Lavie, 2005) utilisent un modèle discriminant SVM. L'analyseur Malt[54] (Nivre, 2008) se base sur une modélisation appelée *Memory-Based*[55].À titre de comparaison, nous avons également indiqué les performances de

[49] `http://maltparser.org/`

[50] On peut noter que (Sagae & Lavie, 2005) se basent sur un ensemble de 13 patrons de traits dont 7 sont consacrés aux arbres de constituants et les 6 autres aux arbres de dépendances.

[51] `http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`

[52] Online large-margin multi-class training.

[53] Comme dit précédemment, la plupart des analyseurs ne permettent pas de générer des dépendances syntaxiques typées, ce qui nous empêche de déterminer les scores LAS.

[54] On peut noter que le score LAS obtenu par Malt est 86.3% sur le PTB-DEP.

[55] Ils utilisent l'implémentation de cet algorithme contenue dans l'outil TiMBL, disponible à l'adresse `http://ilk.uvt.nl/timbl/`

deux analyseurs PCFG (Collins & Singer, 1999; Charniak, 2000) dont les arbres de sortie ont été automatiquement transformés en arbres de dépendances.

| Analyseur | UAS |
|---|---|
| (Yamada & Matsumoto, 2003) | 90.3 |
| (McDonald *et al.*, 2005) | 90.9 |
| (Sagae & Lavie, 2005) | 90.3 |
| (Nivre, 2008) | 88.1 |
| (Collins & Singer, 1999) | 91.5 |
| (Charniak, 2000) | **92.1** |

TAB. 2.8: Performances des systèmes d'analyse syntaxique en dépendances sur le corpus d'évaluation du PTB-DEP.

On peut voir que les analyseurs PCFG obtiennent les performances les plus élevées avec un écart de 1,2% entre (Charniak, 2000) et le meilleur analyseur en dépendances discriminatif (McDonald *et al.*, 2005). Pour l'anglais, les analyses en dépendances générées à partir de celles en constituants sont donc de meilleure qualité. Ces observations ont été reprises récemment par (Cer *et al.*, 2010) dans le cadre d'expériences réalisées sur la reconnaissance de dépendances typées au format Stanford (De Marneffe *et al.*, 2006) par différents analyseurs en dépendances et en constituants.
Notre langue d'étude étant le français, nous avons également indiqué les résultats obtenus par quelques analyseurs en dépendances sur le corpus d'évaluation FTB-DEP. Les résultats, indiqués dans le tableau 2.9, sont directement tirés de (Candito *et al.*, 2010b). Leurs expériences les ont conduit à utiliser l'analyseur Malt sous la version 1.3.1 (Nivre, 2008), avec un classifieur linéaire discriminatif provenant de l'outil LIBLINEAR[56] (Fan *et al.*, 2008). En ce qui concerne l'analyseur MST, il s'agit de la version 0.4.3b. Pour ces deux analyseurs, l'étiquetage morpho-syntaxique a été fourni par un système d'étiquetage appelé MElt (Denis & Sagot, 2009) basé sur un modèle maximum d'entropie et dont les traits proviennent en partie de ressources linguistiques externes (section 3.3.2). Le troisième analyseur évalué est BKY[57] (Petrov *et al.*, 2006), et dont les analyses en constituants ont été automatiquement converties en arbres de dépendances (Candito *et al.*, 2010a) (section 1.3).

| Analyseur | UAS | LAS |
|---|---|---|
| Malt | 89.3 | 86.7 |
| MST | **90.3** | **87.6** |
| BKY | 89.6 | 85.6 |

TAB. 2.9: Performances des analyseurs sur le corpus d'évaluation du PTB.

L'analyseur MST obtient les meilleurs résultats tout score confondu. En ce qui concerne le

[56] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

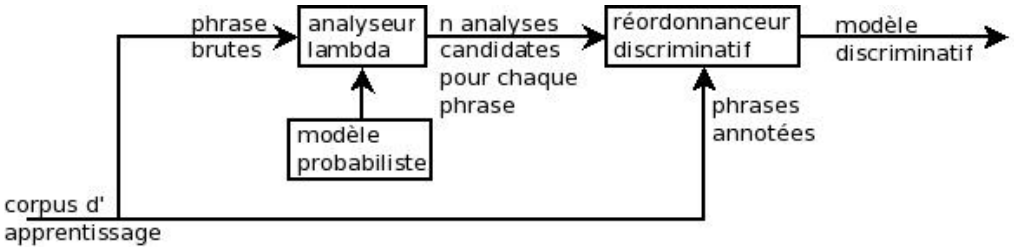
[57] La version utilisée est la 1.0 modifiée pour améliorer le traitement des mots rares et inconnus du français (Crabbé & Candito, 2008) (section 2.3).

score UAS, bien que la différence de performances entre MST et les autres analyseurs soit significative, l'écart est finalement faible (inférieur à 1%). En revanche, BKY est beaucoup plus faible dans le cadre des dépendances typées car on constate une différence de 2% avec MST. Les erreurs proviennent principalement du classifieur discriminatif attribuant une étiquette aux dépendances bilexicales. Quant à Malt, il reste à un niveau stable avec un écart de 1%. D'après ces observations, les auteurs concluent sur le fait que Malt est un bon compromis entre qualité et rapidité d'analyse. En effet, il est 10 fois plus rapide que MST et 9 fois plus rapide que BKY, ceci grâce à son algorithme en temps linéaire alors que les deux autres ont une complexité en $O(n^3)$. On peut noter que, contrairement aux résultats observés précédemment sur l'anglais, les analyseurs en constituants ne sont pas significativement meilleurs que ceux en dépendances. Cette observation rejoint celle de (Kubler, 2008) qui a effectué des expériences similaires pour la langue allemande.
Ces expériences sur le français ont été réalisées avec des traits de base. Or, comme de nombreux autres travaux, (Candito *et al.*, 2010b) ont également montré que les analyseurs en dépendances peuvent tirer profit de l'intégration de diverses informations lexicales comme des traits morphologiques (Tsarfaty, 2006), les lemmes (Seddah *et al.*, 2010) ou encore des classes lexicales (Koo *et al.*, 2008; Candito & Crabbé, 2009; Candito & Seddah, 2010). Dans la partie 6, nous proposerons nos propres algorithmes permettant d'obtenir des classes lexicales que nous pourrons intégrer directement dans les modèles. Nous verrons également que, grâce à ces informations lexicales, le problème de dispersion des données des grammaires PCFG sera partiellement résolu permettant aux analyseurs en constituant, notamment BKY, d'obtenir des performances proches de MST.

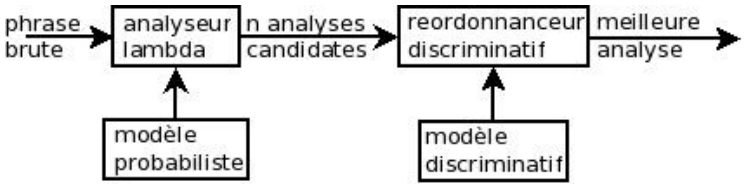
### 2.4.6 Réordonnancement discriminatif

Dans la section 2.3, nous avons pu observer que les analyseurs syntaxiques en constituants, basés sur un modèle PCFG, ont des performances au niveau de l'état de l'art sur de nombreuses langues dont l'anglais et le français. Or, la modélisation PCFG, de par son formalisme génératif, limite le type et la quantité d'informations contextuelles que l'on peut exploiter afin de calculer les probabilités des prédictions. De nombreuses techniques ont tenté de pallier cette limitation comme l'ajout de symboles latents aux règles (Petrov & Klein, 2007) ou encore la lexicalisation des règles (Charniak, 1997; Collins & Singer, 1999; Charniak, 2000; Collins, 2003). Ces méthodes complexifient fortement l'estimation des probabilités des règles, ce qui provoque des effets négatifs comme la dispersion des données. L'approche appelée *réordonnancement discriminatif* (discriminative parsing en anglais) tente de donner une solution à ce problème.
Le réordonnancement discriminatif (Collins, 2000) est une méthode séquentielle faisant intervenir, en premier lieu, un analyseur lambda qui va générer une liste restreinte d'analyses pour une phrase donnée. Puis, un algorithme basé sur un modèle discriminant va reclasser les analyses produites en tenant compte de nouvelles informations qui n'étaient pas disponibles pour le premier analyseur. Un réordonnanceur fonctionne en deux étapes : une étape d'apprentissage et une étape de prédiction. L'étape d'apprentissage, illustrée par la figure 2.18a, consiste à calculer un modèle probabiliste discriminant à partir d'exemples d'entraînement. Un exemple

d'entraînement est une phrase du corpus d'apprentissage dont on possède l'arbre syntaxique correct ainsi que l'ensemble des $n$ meilleurs candidats produits par un analyseur lambda. À partir des exemples d'entraînement, le réordonnanceur extrait des traits qui sont les variables du modèle discriminant. Les paramètres de ce modèle (les poids associés aux traits) sont ensuite estimés par un algorithme d'apprentissage automatique. Quant à l'étape de prédiction, illustrée par la figure 2.18b, elle consiste tout d'abord à produire les $n$ meilleurs candidats pour chaque phrase brute d'un texte par un analyseur[58]. Ensuite, le réordonnanceur extrait les traits de chaque candidat et calcule son score grâce aux poids associés aux traits. En sortie, pour chaque phrase, le réordonnanceur sélectionne l'analyse candidate qui a le score le plus élevé.




(a) Étape d'apprentissage du réordonnanceur.




(b) Étape de prédiction du réordonnanceur.

FIG. 2.18: Schémas illustrant les deux étapes d'un réordonnanceur discriminatif : apprentissage et prédiction.

Formellement, pour une phrase *s*, le réordonnanceur sélectionne la meilleure analyse $p^\star$ parmi l'ensemble des candidats $P(s)$ selon une fonction de score $V_\theta$ :

$$p^\star = \underset{p \in P(s)}{argmax}\ V_\theta(p) \tag{2.37}$$

L'ensemble des candidats $P(s)$ correspond à la liste des *n* meilleurs candidats en sortie d'un analyseur syntaxique avec $P(s) = \{p_1, p_2, ..., p_n\}$. La fonction de score $V_\theta$ est définie par le produit d'un vecteur de poids $\theta$ et d'un vecteur de traits $f$ :

$$V_\theta(p) = \theta.f(p) = \sum_{j=1}^{m} \theta_j.f_j(p) \tag{2.38}$$

où $f_j$ est le $j$ème trait du vecteur et $f_j(p)$ correspond au nombre de fois que le trait $f_j$ apparaît dans l'analyse $p$. D'après (Charniak & Johnson, 2005), le premier trait $f_1$ est la probabilité

[58]En règle générale, l'analyseur est le même que celui utilisé à l'apprentissage du réordonnanceur. Il y a quelques exceptions comme les travaux de (Johnson & Ural, 2010) où les analyses proviennent de plusieurs analyseurs génératifs.

pour l'analyse $p$ renvoyée par l'analyseur[59]. Le vecteur de poids $\theta$ est estimé par un algorithme d'apprentissage automatique à partir d'un corpus annoté.
On peut voir plusieurs avantages liés au réordonnancement discriminatif par rapport à l'analyse générative classique. En ayant en entrée uniquement les arbres produits par l'analyseur syntaxique, l'espace de recherche du réordonnanceur est restreint car il ne peut créer de nouvelles analyses pour la phrase. Cela permet donc de réduire drastiquement la complexité de la tâche. De plus, dans ce formalisme, une analyse est représentée sous la forme d'un vecteur de traits sans aucune hypothèse sur leur interdépendance contrairement aux règles des PCFG qui tiennent compte de la structure de l'arbre. Ces traits ont la particularité de pouvoir être non locaux, indépendants de la langue, et d'intégrer éventuellement des informations provenant de ressources externes autres que le corpus et les analyses candidates.

### Approches de réordonnancement discriminatif

Il existe actuellement de nombreux algorithmes de réordonnancement dérivés de celui de (Collins, 2000). Le réordonnanceur décrit dans (Charniak & Johnson, 2005) reprend dans les grandes lignes l'approche de Collins. Parmi les quelques différences notables avec ce dernier, on peut noter qu'ils ont exploité un plus grand ensemble de patrons de traits (13 au total contre 11 pour Collins) et les traits extraits sont plus complexes[60]. De plus, ils ont utilisé une modélisation par maximum d'entropie et non plus l'algorithme d'optimisation *Boosting Loss* (Schapire & Singer, 1999; Freund *et al.*, 2003). Ce réordonnanceur a été utilisé avec succès pour différents buts dont l'adaptation des analyseurs à de petits corpus et à des textes de genres différents (section 2.5). L'utilisation d'un analyseur LR sous forme de réordonnanceur est proposée par (Sagae & Lavie, 2006). Au lieu de générer un arbre ayant la meilleure probabilité, l'analyseur donne simplement un nouveau score aux analyses présentes dans la liste d'entrée. Pour ce faire, l'algorithme LR est appliqué sur chaque analyse comme s'il s'agissait de l'entraîner. Et, à partir des actions qui ont permi de générer l'analyse, des probabilités sont calculées en fonction des traits associés aux actions.
Plus récemment, (Huang, 2008) a posé l'hypothèse que le réordonnancement traditionnel souffrirait du champs d'action restreint offert à cet algorithme à travers la liste des $n$ analyses en entrée. En effet, de potentielles bonnes analyses pourraient ne pas avoir été sélectionnées pour apparaître dans cette liste. L'algorithme proposé par Huang consiste à effectuer le réordonnancement sur la forêt des analyses possibles de la phrase. Pour illustrer graphiquement le principe de la forêt d'arbres, prenons une phrase d'exemple :

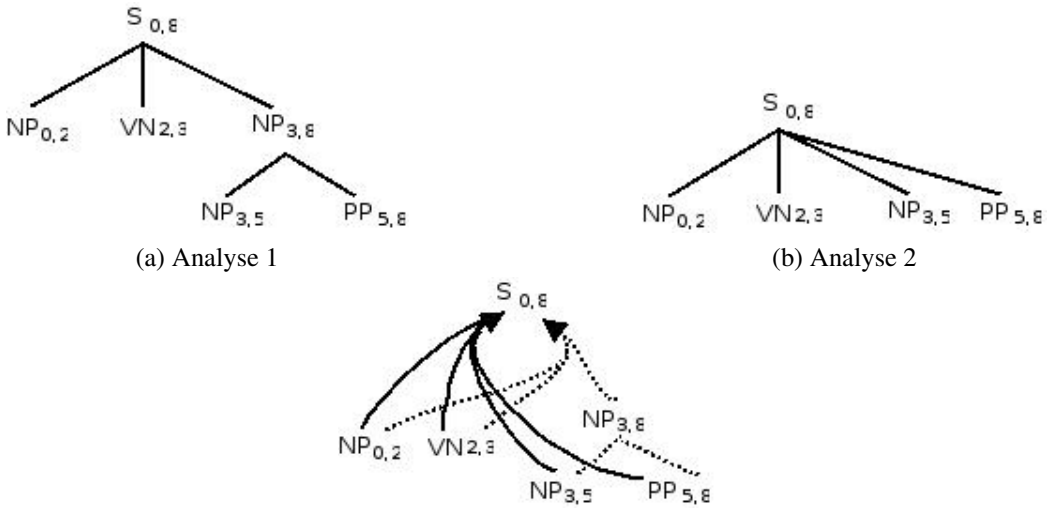
$_0$ le $_1$ chien $_2$ mange $_3$ ses $_4$ croquettes $_5$ à $_6$ la $_7$ volaille $_8$

où les chiffres entre les mots représentent les positions des mots. Dans cette phrase, le groupe prépositionnel *à la volaille* peut être rattaché soit au nom *croquettes* soit au verbe *mange*.

[59]Le poids associé à $f_1$ est un paramètre fixé manuellement par l'utilisateur contrairement aux autres paramètres.

[60]Par "complexe", nous entendons qu'ils contiennent plus d'informations.

Cette ambiguité conduit à obtenir deux analyses possibles pour cette phrase, illustrées par les figures 2.19a et 2.19b. Ces différentes analyses peuvent être représentées sous une forme compacte à travers une forêt d'arbres, montrée dans la figure 2.19c. On peut voir que les noeuds communs aux deux analyses ne sont présents qu'une seule fois dans la forêt. Cependant, la




(a) Analyse 1

(b) Analyse 2

(c) Forêt correspondante

FIG. 2.19: Représentation compacte de deux analyses possibles d'une phrase sous forme d'une forêt d'arbres. Chaque noeud est associé à son étiquette syntaxique et à sa couverture en terme de mots de la phrase (indices).

forêt peut potentiellement contenir une infinité d'analyses rendant la tâche très complexe. Afin de résoudre ce problème efficacement, Huang utilise un algorithme d'approximation appelé *Cube Pruning* (Hopkins & Langmead, 2009) fonctionnant en deux passes. La première étape consiste à parcourir les noeuds internes de la forêt du bas vers le haut (*bottom-up* en anglais) et à chaque noeud, les $n$ meilleurs sous arbres en terme de probabilité sont conservés et ordonnés. Puis, en effectuant un parcours descendant, les noeuds de la forêt faisant partie de la meilleure analyse sont extraits. Les patrons exploités par Huang sont au nombre de 15 et proviennent de différents travaux (Collins, 2000; Charniak & Johnson, 2005).
Dans le tableau 2.10, nous avons indiqué les meilleurs résultats obtenus (score $F_1$) par les divers réordonnanceurs sur le corpus d'évaluation du PTB lorsqu'ils sont intégrés dans un système d'analyse syntaxique en constituants. Le réordonnanceur de (Collins, 2000) est combiné avec l'analyseur génératif (Collins & Singer, 1999)[61]. Le réordonnanceur de (Charniak & Johnson, 2005) est combiné avec l'analyseur (Charniak, 2000) dans un système appelé Brown Parser[62]. Environ 1,5 millions de traits ont été extraits avec $n$ fixé à 50. (Sagae & Lavie, 2006) a également utilisé l'analyseur (Charniak, 2000) avec $n$ fixé à 10. Et enfin, (Huang, 2008) a modifié cet analyseur PCFG afin qu'il renvoie une forêt d'arbres plutôt qu'une liste d'analyses. Le paramètre $n$ a été fixé à 50 pour extraire environ 800000 traits.
On peut voir que le meilleur système est celui de (Huang, 2008) ce qui validerait son hypothèse

[61] Collins ne précise pas le nombre de traits extraits ni la valeur du paramètre $n$.
[62] `http://web.science.mq.edu.au/~mjohnson/Software.htm`

de départ, à savoir que le réordonnancement traditionnel souffre de la vision restreinte du problème. Puis, viennent ensuite les réordonnanceurs de (Charniak & Johnson, 2005) et (Sagae & Lavie, 2006) qui ont des performances relativement similaires.

| Système | $F_1$ |
|---|---|
| (Collins, 2000) | 89.7 |
| (Charniak & Johnson, 2005) | 91.0 |
| (Sagae & Lavie, 2006) | 90.8 |
| (Huang, 2008) | **91.7** |

TAB. 2.10: Performances sur le corpus d'évaluation du PTB de divers systèmes d'analyse en constituants intégrant un réordonnanceur.

Plus récemment, (LeRoux *et al.*, 2011) ont proposé d'appliquer un réordonnanceur, appelé MIRA[63], sur des arbres de dépendances syntaxiques plutôt que sur des arbres de constituants. Bien que la majorité des meilleurs analyseurs en dépendances utilisent une modélisation discriminative, il est tout de même intéressant d'observer le comportement d'un réordonnanceur sur ce type d'analyse. Á l'heure actuelle, il existe quelques analyseurs en dépendances capables de générer les $n$ analyses les plus probables (Hall, 2007). Cependant, les auteurs ont préféré adopter une approche consistant à convertir des arbres de constituants vers un format en dépendances typées. Cette approche, déjà observée pour la conversion de corpus (section 1.3), est composée de trois étapes successives. La première étape consiste à produire les $n$ analyses par un analyseur lambda. Ensuite, des annotations fonctionnelles (sujet, objet direct,...) sont automatiquement ajoutées aux noeuds des arbres grâce à un étiqueteur fonctionnel (classifieur). Et enfin, les arbres sont convertis en structures de dépendances par un ensemble de règles de propagations de têtes lexicales et d'heuristiques. Les patrons de traits utilisés par MIRA lors des phases d'apprentissage et de prédiction sont proches de ceux définis par (McDonald *et al.*, 2005) pour leur analyseur MST (section 2.4.5).
Dans le cadre d'expériences réalisées sur le français, (LeRoux *et al.*, 2011) ont utilisé un analyseur génératif basé sur une grammaire PCFG-LA similaire à BKY, ainsi qu'un étiqueteur fonctionnel et un convertisseur disponibles dans l'outil Bonsaï. Dans le tableau 2.11, nous indiquons les résultats $F_1$, UAS et LAS obtenus sur le corpus d'évaluation du FTB-UC par l'analyseur PCFG-LA utilisé seul, puis combiné avec le réordonnanceur[64]. À titre de comparaison, nous indiquons également les performances d'autres systèmes d'analyse en dépendances. L'expérience notée BKY correspond à la version de BKY utilisée dans (Candito *et al.*, 2010b) et dont la sortie a été transformée en arbres de dépendances (section 2.4.5). Par rapport à l'analyseur utilisé seul, les gains obtenus par MIRA sur les trois métriques sont significatifs ($p < 0.001$). De plus, la combinaison PCFG-LA+MIRA permet d'obtenir les meilleurs résultats sur les scores de dépendances UAS et LAS mais également sur le score $F_1$.
Bien que ce réordonnanceur soit très performant sur le français, nous avons préféré utilisé celui de (Charniak & Johnson, 2005), et ceci pour plusieurs raisons. À l'heure où nous écrivons ce mémoire, MIRA est toujours en développement. Il n'est donc pas encore tout à fait optimisé en

[63] `https://github.com/jihelhere/adMIRAble`

[64] Les meilleurs résultats sont obtenus avec *n* fixé à 50, ce qui conduit à extraire environ 75 millions de traits à l'apprentissage.

| Analyseur | $F_1 < 40$ | LAS | UAS |
|---|---|---|---|
| PCFG-LA | 88.4 | 87.3 | 91.0 |
| PCFG-LA + MIRA | **88.7** | **88.4** | **91.9** |
| BKY | 88.2 | 86.8 | 91.0 |
| MST | - | 88.2 | 90.9 |

TAB. 2.11: Performances sur le corpus d'évaluation du FTB-UC de divers systèmes d'analyse en dépendances

temps et en mémoire. En effet, il est nécessaire de posséder une machine avec environ 50Go de RAM et la phase d'apprentissage prend environ 20 heures avec $n$=50.

### Réimplémentation du réordonnanceur

L'implémentation disponible du réordonnanceur du Brown Parser a toutefois, selon nous, plusieurs inconvénients. D'une part, les fichiers de configuration inclus dans la distribution ne permettent pas de configurer aisément le réordonnanceur pour une langue autre que l'anglais. En effet, ce fichier nous impose, par exemple, d'avoir un corpus d'apprentissage prédécoupé comme l'est le PTB, c'est-à-dire sous forme de *sections* (des dossiers contenant les fichiers d'arbres). Cette spécificité n'est pas présente dans beaucoup de corpus dont le FTB-UC. D'autre part, dans cet outil, la création et la manipulation des patrons de traits est une tâche assez complexe. Le langage C++ impose une relative lourdeur d'écriture empêchant le développement rapide de patrons. De plus, à moins de modifier directement le code, il n'est pas possible d'intégrer des informations provenant de ressources externes.
Pour toutes ces raisons, nous avons donc décidé de réimplémenter complètement le réordonnanceur en langage Python[65], tout en essayant d'améliorer l'interface homme-machine pour le rendre plus intuitif pour un utilisateur lambda. Il est également intéressant pour la communauté de disposer d'un outil dans plusieurs langages, ceci afin qu'un utilisateur non familier d'un langage particulier puisse tout de même manipuler l'outil. La représentation des arbres de constituants est effectuée grâce au module Python appelé *NetworkX*[66]. Notre outil accepte en entrée tout corpus d'apprentissage au format parenthésé car il s'agit d'un format standard utilisé notamment par le FTB-UC et le PTB (section 1.2.1). Les patrons de traits sont représentés sous la forme de fonctions Python qui partagent toutes le même prototype :

```
def nom_patron(arbre,noeud_courant,donnees)
        traits = []
        #code python
        return traits
```

où *nom_patron* est le nom du patron de traits. Les paramètres d'entrée *arbre* et *noeud_courant*

[65]Librement disponible à l'adresse `http://igm.univ-mlv.fr/~sigogne/`
[66]`http://networkx.lanl.gov/`

sont des pointeurs sur l'arbre et le noeud courant. Quant au paramètre *donnees*, il permet l'utilisation de données externes (dictionnaires, lexiques syntaxiques,...). Chaque patron doit déclarer une variable de type *liste* qui va contenir les traits extraits à partir du noeud courant. Cette variable, appelée *traits* dans le prototype, doit être renvoyée en sortie de la fonction. De plus, nous mettons à disposition des fonctions utiles déjà écrites dans le but d'accélérer la création de nouveaux patrons. Par exemple, une fonction appelée *ancestors* retourne l'ensemble des ancêtres d'un noeud de l'arbre.
Un fichier de configuration permet de specifier la valeur de certains paramètres du réordonnanceur, comme le chemin vers le module contenant les patrons de traits, les patrons devant être utilisés durant les étapes d'apprentissage et de prédiction, ou encore le nom de fichier du modèle discriminant. De plus, deux paramètres numériques sont à définir manuellement. Le premier, appelé *freq_feature*, est un palier de filtrage fixant le nombre minimal d'occurrences d'un trait dans le corpus d'apprentissage (Ratnaparkhi, 1996). Si un trait apparaît 1 fois dans le corpus et que le palier est fixé à 5, alors ce trait n'est pas pris en compte dans la modélisation du problème. Le deuxième paramètre, appelé *weight_parser_score*, correspond au poids associé au trait composé de la probabilité renvoyée par l'analyse lambda pour une analyse. On peut rappeler que ce trait est utilisé uniquement lors de la phase de prédiction. Il est nécessaire de donner une valeur pertinente à ce poids car un poids trop fort pourrait accorder trop d'importance à l'analyseur. Une méthode simple consiste à fixer initialement le paramètre à 0, puis à l'incrémenter, et ceci afin de maximiser les scores obtenus sur le corpus de développement. Par exemple, dans le cadre du FTB-UC et de l'analyseur BKY, le poids idéal est de 2.
À l'origine, le réordonnanceur de Brown se base sur un modèle maximum d'entropie dont les paramètres sont déterminés par un algorithme d'optimisation de second ordre appelé *Limited Memory Variable Metric*. Concrètement, ils utilisent une implémentation de cet algorithme disponible dans les bibliotèques mathématiques PETSc et TAO[67]. Nous avons décidé de conserver uniquement cette partie de leur outil pour des raisons pratiques.

### Patrons de traits

L'extraction des traits par le réordonnanceur est effectuée par l'application systématique de patrons de traits sur chaque noeud des arbres de constituants. Chaque patron permet d'extraire des traits bien spécifiques représentés informatiquement sous la forme de chaînes de caractères. On peut noter que certains patrons nécessitent que les arbres soient lexicalisés. Pour ce faire, nous avons fait le choix d'utiliser la table de propagation des têtes lexicales proposée par (Dybro-Johansenn, 2004). Dans le tableau 2.12, nous avons indiqué l'ensemble des patrons inclus dans la distribution de notre outil. Ces patrons sont issus pour la plupart des travaux décrits dans (Charniak & Johnson, 2005). Ceux repérés par un $^{\triangle}$ proviennent de (Collins, 2000). La possibilité de les utiliser pour diverses langues, comme le français ou l'anglais, provient du fait qu'ils ne font pas d'hypothèses linguistiques sur la langue traitée[68].
Dans la suite de cette section, nous donnons une description succincte de ces patrons. Pour cha-

[67]Ces bibliotèques, écrites en C++, sont téléchargeables aux adresses suivantes `http://www.mcs.anl.gov/petsc/` et `http://www.mcs.anl.gov/research/projects/tao/`
[68]Ces patrons sont dits *indépendants de la langue*.

| | |
|---|---|
| Rule | Edges |
| Word | WordEdges |
| Heavy | Heads |
| HeadTree | WProj |
| Bigrams$^{\triangle}$ | NgramTree |
| Trigrams$^{\triangle}$ | Score |

TAB. 2.12: Patrons de traits utilisés par notre implémentation du réordonnanceur, et pour la plupart issus de (Charniak & Johnson, 2005). Ceux repérés par un $^{\triangle}$ proviennent de (Collins, 2000).

cun d'entre eux, nous avons indiqué un ou plusieurs traits issus de l'application du patron sur un arbre de constituants de la phrase *Luc recouvre la table de bois*, montré figure 2.20. Pour faciliter la compréhension des exemples, nous avons également numéroté chaque noeud de l'arbre.

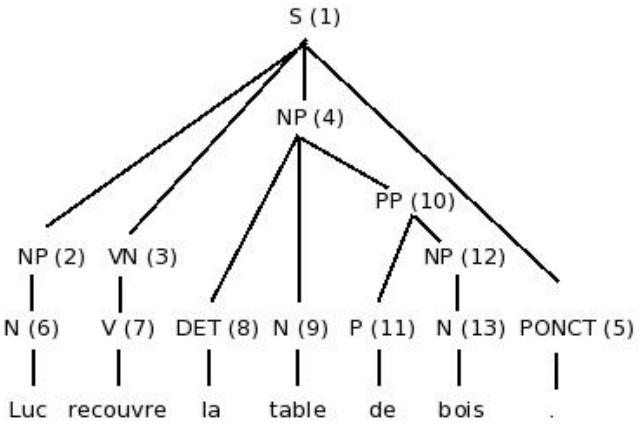



FIG. 2.20: Une analyse en constituants de la phrase *Luc recouvre la table de bois*.

**Rule** Le patron appelé *Rule* extrait les règles CFG locales, à savoir le noeud non-terminal courant et ses noeuds fils. Si le noeud courant possède un noeud père, il est également extrait. La figure 2.21 montre le résultat de l'application du patron sur le noeud 10 (`PP`). Les fils de ce noeud sont les noeuds 11 et 12 (respectivement `P` et `NP`) et son père est le noeud 4 (`NP`).

**Words** Le patron appelé *Words* extrait la liste des $n$ ancêtres d'un noeud terminal, $n$ étant un paramètre défini par l'utilisateur. La figure 2.22 montre le résultat de l'application du patron sur le terminal *bois* avec $n$ fixé à 3. Le patron extrait les noeuds ancêtres 12 et 10 (respectivement `NP` et `PP`).

**NgramTree** Le patron appelé *NgramTree* extrait le sous-arbre ayant pour racine le premier ancêtre commun à plusieurs noeuds terminaux voisins (mots de la phrase). Ce sous-arbre est

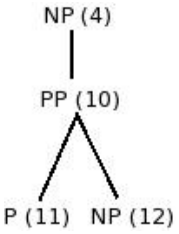



FIG. 2.21: Application du patron *Rule* sur le noeud 10.

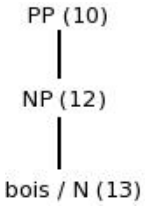



FIG. 2.22: Application du patron *Word* sur le terminal *bois*.

partiellement extrait car il ne contient que les noeuds qui ont pour feuilles au moins un des mots terminaux spécifiés. La figure 2.23a montre le résultat de l'application du patron sur les noeuds terminaux *la* et *table*. Le premier ancêtre commun de ces noeuds est le noeud 4 (NP). La figure 2.23b montre l'application sur les noeuds terminaux *recouvre*, *la* et *table*. Le premier ancêtre commun de ces noeuds est le noeud 1 (S).

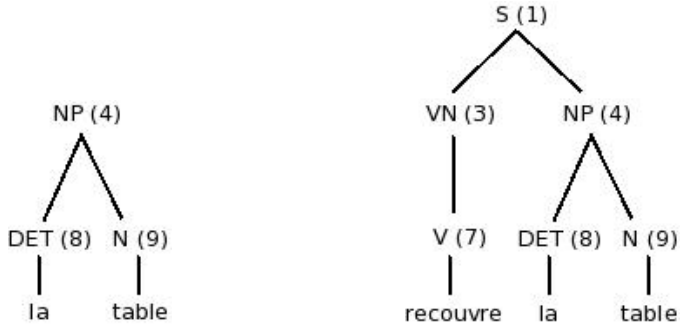



FIG. 2.23: Exemples d'application du patron *NgramTree*.

**Heavy** Le patron appelé *Heavy* extrait plusieurs informations sur le noeud courant de l'arbre :
– la taille de la couverture en terme de nombre de terminaux
– la distance avant la fin en terme de nombre de terminaux
– indique si le dernier mot de la couverture du noeud est une ponctuation
– indique si une ponctuation est située à droite de la couverture du noeud
Le tableau 2.13 montre le résultat de l'application du patron sur le noeud 4 (NP). Le nombre de noeuds terminaux couverts par ce noeud est de 4. La distance avant la fin est de 1 car il ne reste

qu'un noeud terminal à droite de la couverture, à savoir le noeud 5 (PONCT). Le dernier mot de la couverture n'est pas une ponctuation et il y a bien une ponctuation à droite de la couverture.

| Information | Valeur |
|---|---|
| taille(couverture) | 4 |
| distance(fin) | 1 |
| ponct(dernier_mot_couverture) | non |
| ponct(mot_après_couverture) | oui |

TAB. 2.13: Application du patron *Heavy* sur le noeud 4.

**Wproj** Le patron appelé *Wproj* extrait la liste des ancêtres faisant partie de la projection maximale d'un noeud terminal de l'arbre. La projection maximale est le premier ancêtre du noeud terminal dont la tête lexicale n'est pas ce noeud terminal. La figure 2.24 montre le résultat de l'application du patron sur le noeud terminal *bois*. Les ancêtres 12 et 10 (respectivement NP et PP) ont pour tête lexicale *bois*. En revanche, le noeud 4 (NP) a pour tête lexicale *table*, donc la projection maximale de *bois* est le noeud 4.

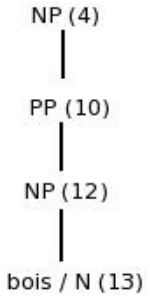
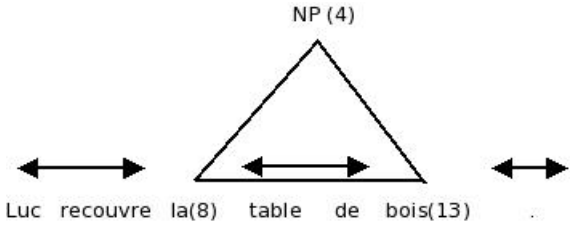



FIG. 2.24: Application du patron *Wproj* sur le noeud terminal *bois*.

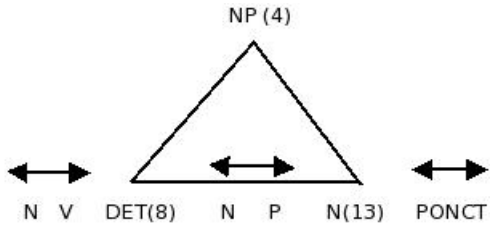
**WordEdges** Le patron appelé *WordEdges* extrait les $n$ symboles terminaux situés à gauche et à droite du premier et du dernier noeud de la couverture d'un noeud de l'arbre. $n$ est un paramètre à définir par l'utilisateur. La figure 2.25 montre le résultat de l'application du patron sur le noeud 4 (NP) avec $n$ fixé à 2. Dans cet exemple, les deux mots situés à gauche de l'extremité gauche de la couverture du noeud 4 sont *Luc* et *recouvre*. À droite de ce noeud, ce sont *table* et *de*. Ces mêmes mots sont extraits à gauche de l'extremité droite du noeud 4, à savoir le noeud 13. Et enfin, à droite de ce noeud, seul le signe de ponctuation "." est extrait.

**Edges** Le patron appelé *Edges* est dérivé du patron *WordEdges* et extrait les $n$ symboles préterminaux situés à gauche et à droite du premier et du dernier noeud de la couverture d'un noeud de l'arbre. $n$ est un paramètre à définir par l'utilisateur. La figure 2.26 montre le résultat de l'application du patron sur le noeud 4 (NP) avec $n$ fixé à 2.




| Noeud | Direction | Terminaux |
|---|---|---|
| 8 (DET) | gauche | Luc recouvre |
| 8 (DET) | droite | table de |
| 13 (N) | gauche | table de |
| 13 (N) | droite | . |

FIG. 2.25: Application du patron *WordEdges* sur le noeud 4 : (a) informations extraites à partir de l'arbre. (b) trait généré.




| Noeud | Direction | Préterminaux |
|---|---|---|
| 8 (DET) | gauche | N V |
| 8 (DET) | droite | N P |
| 13 (N) | gauche | N P |
| 13 (N) | droite | PONCT |

FIG. 2.26: Application du patron *Edges* sur le noeud 4 : (a) informations extraites à partir de l'arbre. (b) trait généré.

**HeadTree** Le patron appelé *HeadTree* extrait un sous-arbre à partir du noeud courant. Ce sous-arbre contient les noeuds ayant la même tête lexicale que le noeud courant, jusqu'aux noeuds terminaux. De plus, tous les fils d'un noeud présent dans le sous-arbre font également partie de cet arbre. La figure 2.27 montre le résultat de l'application du patron sur le noeud 1 (`S`). La tête lexicale de ce noeud est *recouvre*. Tout d'abord, le noeud 1 fait partie du sous-arbre, ce qui veut dire que ses fils également (noeuds 2, 3, 4 et 5). Le noeud fils 3 (`VN`) contient la tête lexicale *recouvre*. On inclut donc son unique fils 7 (`V`) dans le sous-arbre. Ce noeud a pour fils la tête lexicale *recouvre*. Le sous-arbre est terminé.

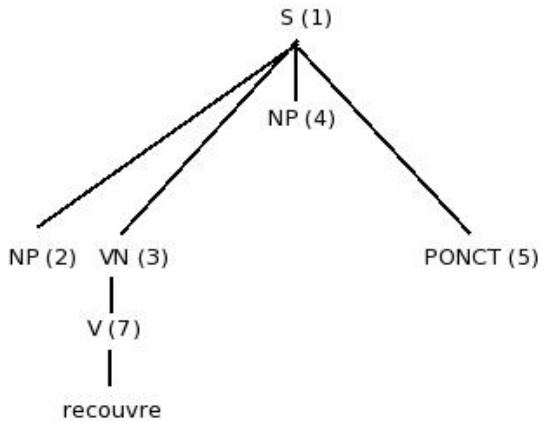



FIG. 2.27: Application du patron *HeadTree* sur le noeud 1.

**Heads** Le patron appelé *Heads* extrait les $n$-grammes de noeuds fils issus d'un même père. Pour chacun de ces noeuds fils, on précise quelle est la tête lexicale du noeud. Le tableau 2.14 montre le résultat de l'application du patron sur le noeud 1 (S) avec $n$ fixé à 2.

| Noeuds | Têtes lexicales associées |
|---|---|
| 2 (NP), 3 (VN) | Luc, recouvre |
| 3 (VN), 4 (NP) | recouvre, table |
| 4 (NP), 5 (PONCT) | table, . |

TAB. 2.14: Application du patron *Heads* sur le noeud 1.

**Bigrams** Le patron appelé *Bigrams* extrait les bigrammes de noeuds fils issus d'un même père. De plus, pour chaque bigramme extrait, on précise dans quelle direction (gauche ou droite), par rapport à ce bigramme, se situe le noeud contenant la tête lexicale de la règle. Un noeud artificiel appelé STOP permet de créer des bigrammes pour le premier et le dernier noeud de la couverture. Le tableau 2.15 montre le résultat de l'application du patron sur le noeud 4 (NP). La tête lexicale de ce noeud est *table* propagée à travers le noeud fils 9 (N).

| Bigramme | Direction |
|---|---|
| STOP DET | droite |
| DET N | droite |
| N PP | gauche |
| PP STOP | gauche |

TAB. 2.15: Application du patron *Bigrams* sur le noeud 4.

**Trigrams** Le patron appelé *Trigrams* extrait les trigrammes de noeuds fils issus d'un même père. Si le noeud contenant la tête lexicale de la règle est présent dans le trigramme, une marque est apposée sur celui-ci. Le tableau 2.16 montre le résultat de l'application du patron sur le noeud 4 (NP). La tête lexicale de ce noeud est *table* propagée à travers le noeud fils 9 (repéré par un " !").

| Trigramme |
|---|
| STOP DET N ! |
| DET N ! PP |
| N ! PP STOP |

TAB. 2.16: Application du patron *Trigrams* sur le noeud 4.

**Performances de notre implémentation du réordonnanceur sur le FTB-UC**

Dans cette section, nous indiquons les résultats obtenus par notre implémentation du réordonnanceur dans le cadre de plusieurs expériences sur l'analyse syntaxique du français. Une première expérience consistera à évaluer l'impact du réordonnanceur lorsqu'il est combiné à différents analyseurs en constituants. Nous effectuerons également une comparaison des performances avec les analyseurs en dépendances après conversion automatique des arbres. Dans une deuxième expérience, nous évaluerons l'impact de chaque patron de traits lorsqu'ils sont utilisés seuls. Nous verrons que leur force provient du fait qu'ils sont en grand nombre. Et enfin, la dernière expérience consiste à déterminer si le réordonnanceur discriminatif est une technique utile et fiable lorsque le corpus d'apprentissage disponible pour la langue est de petite taille. Pour ce faire, nous ferons varier la taille du corpus FTB-UC.
Pour la première expérience, nous avons travaillé avec deux analyseurs en constituants différents, BKY[69] (Petrov & Klein, 2007) et Brown (Charniak, 2000). Tout d'abord, pour chacun d'eux, nous avons généré les $n$ analyses les plus probables pour chaque phrase du corpus d'apprentissage du FTB-UC, ceci afin de pouvoir appliquer les patrons de traits, puis d'entraîner le réordonnanceur. Le nombre de traits obtenus à partir de chaque ensemble de données d'apprentissage est indiqué dans le tableau 2.28a selon différentes valeurs de $n$[70]. Le nombre de traits obtenus semble linéaire au nombre $n$. De plus, avec $n = 100$, moins de 10 millions de traits sont extraits, ce qui reste raisonnable notamment en terme de temps de calcul nécessaire à l'analyse[71].
Après avoir entraîné le réordonnanceur, nous l'avons évalué sur les corpus de développement et d'évaluation du FTB-UC. De même qu'à l'apprentissage, il est nécessaire de générer, pour chaque phrase brute de ces corpus, la liste $n$ des meilleures analyses pour chaque analyseur. Une métrique d'évaluation particulière, appelée *oracle*, permet d'évaluer la qualité d'un système d'analyse et de connaître également les performances maximales qu'il pourrait obtenir. Pour chaque phrase, il faut tout d'abord extraire l'analyse, parmi les $n$, qui maximise un score ($F_1$, UAS,...) obtenu par rapport à l'analyse correcte. Ensuite, avec une seule analyse par phrase, le score global est calculé sur l'ensemble des phrases du corpus et ce score correspond à l'oracle. Les scores oracle $F_1$ obtenus par les différents analyseurs sur les deux corpus sont indiqués dans le tableau 2.28b selon les diverses valeurs de $n$. On peut noter que $n = 1$ correspond aux scores $F_1$ de base des différents analyseurs.
Dans la figure 2.29, nous avons indiqué les résultats ($F_1$, UAS et LA) obtenus par le réordonnanceur lorsqu'il est combiné aux analyseurs BKY et Brown. Contrairement à de nombreux travaux, nous n'avons pas filtré les traits avant d'effectuer l'apprentissage du modèle car, comme énoncé auparavant, le nombre de traits extraits est raisonnable pour la totalité des valeurs de $n$ testées. Comparativement aux résultats de base, l'ensemble des scores obtenus avec le réordonnanceur sont significatifs ($p < 10^{-4}$). Les meilleurs résultats sont obtenus avec $n = 50$. On

[69] Version 1.1 optimisée pour le français. Bien que l'analyseur $BKY_p$ (Petrov *et al.*, 2010) obtiennent les meilleurs résultats sur le FTB-UC (section 2.3.2), nous sommes dans l'incapacité de l'utiliser en combinaison du réordonnanceur car la version actuelle ne permet pas de générer la liste des $n$ analyses.

[70] L'ensemble des patrons décrit dans la section précédente a été utilisé.

[71] L'analyse du corpus de développement est effectuée en 12 minutes. Avec $n = 5$, ce temps est de moins de 1 minute.

| $n$ / Analyseur | 5 | 20 | 50 | 100 |
|---|---|---|---|---|
| BKY | 1686315 | 3760168 | 5787343 | 8563481 |
| Brown | 1563859 | 3635498 | 5486490 | 8127927 |

(a) Nombre de traits obtenus selon l'analyseur et la valeur de $n$.

| $n$ / Analyseur | 1 | 5 | 20 | 50 | 100 |
|---|---|---|---|---|---|
| **Corpus de développement** | | | | | |
| BKY | 82.83 | 88.04 | 90.75 | 92.10 | 92.90 |
| Brown | 79.56 | 84.56 | 89.16 | 90.84 | 92.00 |
| **Corpus d'évaluation** | | | | | |
| BKY | 84.15 | 89.48 | 92.26 | 93.60 | 94.22 |
| Brown | 80.86 | 86.79 | 90.42 | 92.00 | 92.97 |

(b) Scores oracle $F_1$ obtenus selon l'analyseur et la valeur de $n$.

FIG. 2.28: (a) Nombres de traits obtenus en appliquant l'ensemble des patrons de traits décrit dans la section précédente sur les corpus d'apprentissage produits par les différents analyseurs. (b) Scores oracle $F_1$ obtenus par les analyseurs sur les corpus de développement et d'évaluation selon différentes valeurs de $n$.

peut néanmoins constater que la différence entre les gains de $n = 20$ et $n = 50$ est très faible et n'est pas significative (p>0.5). De plus, avec $n = 100$, on observe même une légère dégradation des résultats. Il semblerait donc que $n$ fixé à 20 soit un bon compromis entre performances et rapidité d'exécution.

## 2.5 Adaptation des analyseurs à de petits corpus et à des textes de genres différents

Suite à la création de nombreux corpus annotés manuellement pour diverses langues, nous avons vu un regain d'intérêt pour les analyseurs dont le modèle est issu d'un apprentissage supervisé. Ces analyseurs ont de bonnes performances, notamment pour l'anglais avec un score $F_1$ d'environ 92% sur le WSJ (Charniak & Johnson, 2005) et 89% pour le français sur le FTB-UC (LeRoux *et al.*, 2011). Cependant, on voit rapidement les limites d'utilisation de ces analyseurs car d'une part, les corpus sont souvent de petite taille et donc inadaptés la plupart du temps à des traitements lourds que sont les apprentissages supervisés. En général, on a recours à ce qu'on appelle du lissage de probabilités afin de pallier le problème de la dispersion des données (voir section 2.3.2). Et d'autre part, les expériences sur l'analyse syntaxique sont souvent effectuées dans un seul genre textuel (journalistique, littéraire, médical...). L'analyseur est entraîné sur une partie d'un corpus et les évaluations sont effectuées sur la partie restante de ce même corpus. Or, de nombreuses expériences font état de mauvais résultats lorsqu'on essaye d'appliquer un analyseur sur un texte dont le genre est différent de celui du corpus d'entraînement.

| $n$ / Analyseur | 1 | 5 | 20 | 50 | 100 |
|---|---|---|---|---|---|
| **Corpus de développement** | | | | | |
| BKY | 82.83 | 83.66 | 83.87 | **84.20** | 84.18 |
| Brown | 79.56 | 80.46 | 80.82 | **80.89** | 80.81 |
| **Corpus d'évaluation** | | | | | |
| BKY | 84.15 | 85.20 | 85.38 | **85.49** | 85.50 |
| Brown | 80.86 | 81.92 | 82.08 | **82.10** | 82.05 |

(a) Score $F_1$.

| $n$ / Analyseur | 1 | 5 | 20 | 50 | 100 |
|---|---|---|---|---|---|
| **Corpus de développement** | | | | | |
| BKY | 86.8 | 87.4 | 87.5 | **87.7** | 87.6 |
| Brown | 83.8 | 84.1 | **84.4** | **84.4** | 84.3 |
| **Corpus d'évaluation** | | | | | |
| BKY | 86.8 | 87.3 | **88.1** | **88.1** | 87.9 |
| Brown | 84.5 | 85.0 | **85.2** | **85.2** | 85.0 |

(b) Score LA.

| $n$ / Analyseur | 1 | 5 | 20 | 50 | 100 |
|---|---|---|---|---|---|
| **Corpus de développement** | | | | | |
| BKY | 87.74 | 88.18 | 88.44 | **88.50** | 88.41 |
| Brown | 85.41 | 85.91 | 86.17 | **86.25** | 86.24 |
| **Corpus d'évaluation** | | | | | |
| BKY | 88.23 | 88.64 | **89.15** | 89.10 | 89.12 |
| Brown | 85.96 | 86.40 | 86.57 | **86.59** | 86.49 |

(c) Score UAS.

FIG. 2.29: Scores obtenus par le réordonnanceur combiné aux analyseurs Brown et BKY et selon le nombre variable $n$ : (a) $F_1$, (b) LA, (c) UAS.

Par exemple, l'analyseur PCFG de (Charniak, 2000) obtient un score $F_1$ de 89.7% lorsqu'il est entraîné et évalué sur le WSJ. Mais, ses performances chutent considérablement lorsqu'il est évalué sur le Brown Treebank [BT] avec un score $F_1$ de seulement 82.9%, sachant que ces deux corpus diffèrent relativement peu en terme de genre[72]. Une autre expérience, décrite dans (Candito *et al.*, 2011), indique une chute de cinq points du score $F_1$ lorsque BKY est appliqué sur un corpus bio-médical alors qu'il a été entraîné sur le corpus journalistique FTB-UC.
Les travaux de (Sekine, 1997) tentent de donner une réponse à ces problèmes. En explorant les productions de grammaires extraites de corpus annotés de genres différents, il a remarqué que les structures syntaxiques contenues dans ces grammaires ainsi que leur lexique propre sont très différents entre les corpus. De plus, il a déterminé qu'un corpus d'apprentissage composé de données annotées du même genre que le texte à analyser permet de produire les meilleures analyses, suivi de données d'un genre approchant et enfin, de données d'un genre différent. Il conclut sur le fait qu'il faudrait créer des corpus annotés manuellement pour un grand nombre de genres textuels. Cependant, le coût non-négligeable de création d'un corpus fait que cette solution n'est pas adoptée par la plupart des chercheurs. Ces derniers se sont plutôt penchés sur des algorithmes dits *semi-supervisés* qui exploitent à la fois des données annotées manuellement mais également des données brutes. La quantité de données brutes disponible actuellement est presque illimitée (web, blogs, corpus...) et ces données ont des genres textuels divers (corpus médicaux, corpus de journaux,...).
Dans les sections qui suivent, nous décrivons les principales méthodes semi-supervisées permettant de résoudre au mieux les deux problèmes que nous venons d'évoquer, à savoir comment améliorer l'analyse syntaxique quand le corpus d'entraînement de la langue est de petite taille, et comment améliorer l'analyse de textes dont le genre diffère de celui du corpus d'entraînement. Dans ce qui suit, lorsque nous faisons référence à des scores ou à des gains il s'agit du score $F_1$ sauf mention contraire.

### 2.5.1 Auto-Apprentissage

L'approche appelée *auto-apprentissage* (self-training en anglais), dont l'architecture générale est montrée dans la figure 2.30, peut se résumer en trois points. Tout d'abord, un modèle probabiliste est appris de manière classique sur un corpus annoté manuellement. Ce modèle est ensuite utilisé pour analyser les phrases d'un grand corpus brut, puis, les analyses produites sont ajoutées au corpus annoté original. Enfin, un autre modèle est appris sur le nouveau corpus. L'hypothèse sous-jacente à l'auto-apprentissage est que l'ajout massif de phrases annotées automatiquement au corpus d'apprentissage permettrait d'améliorer l'estimation des probabilités du modèle. Dans le cadre de l'adaptation d'un analyseur à différents genres textuels, l'intérêt de cette technique provient du fait que le corpus brut puisse être d'un genre différent du corpus annoté. Les expériences de cette section traitant d'un genre textuel unique ont leurs résultats indiqués dans le tableau 2.17. Quant aux expériences d'adaptation à plusieurs genres textuels, les résultats sont signalés dans le tableau 2.18.
Les premières expériences (Charniak, 1997; Steedman *et al.*, 2003; Clark *et al.*, 2003) ont d'abord montré les faiblesses de cette approche pour un unique genre textuel. Le modèle appris

[72]En terme de lexique, seulement 6% des mots du BT ne font pas partie du vocabulaire du WSJ.

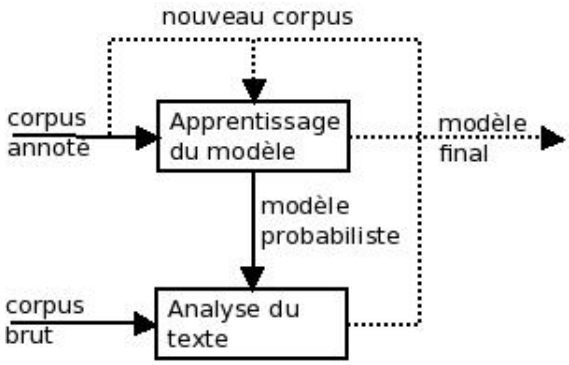



FIG. 2.30: Architecture générale de l'approche appelée *auto-apprentissage*.

sur le corpus enrichi reproduirait les erreurs du modèle de base car ces mêmes erreurs seraient présentes dans les analyses ajoutées au corpus d'apprentissage. Dans (Charniak, 1997), l'analyseur PCFG est d'abord entraîné sur le WSJ, pour ensuite analyser les 30 millions de mots d'un corpus brut correspondant à d'autres années du WSJ. Ces nouvelles données annotées sont ajoutées au corpus d'apprentissage et l'analyseur est réentraîné. Le gain obtenu sur le corpus d'évaluation du WSJ est de +0.1 par rapport au score obtenu sans auto-apprentissage. Ce résultat mitigé est confirmé par (Steedman *et al.*, 2003), même si l'approche d'auto-apprentissage utilisée est quelque peu différente. L'algorithme itératif démarre par l'entraînement d'un analyseur sur un corpus annoté composé de seulement 500 phrases tirées du WSJ. Ensuite, à chaque itération, 30 phrases issues des phrases restantes du WSJ sont analysées par le modèle et les 20 meilleures analyses, d'après une formule de score (probabilité jointe), sont ajoutées au corpus d'apprentissage initial. Puis, l'analyseur est réentraîné sur ce nouveau corpus. L'hypothèse sous-jacente à cet algorithme est qu'en employant un score afin de sélectionner uniquement des analyses de qualité supérieure, les erreurs contenues dans ces analyses auraient un impact moins important. Cependant, les gains obtenus par l'analyseur PCFG de (Collins & Singer, 1999) ne sont pas significatifs et ils constatent même une nette dégradation sur un autre analyseur basé sur une grammaire LTAG (Sarkar, 2001). (Clark *et al.*, 2003) ont appliqué cet algorithme pour l'étiquetage morpho-syntaxique et ont également observé des résultats mitigés.
En revanche, d'autres travaux plus récents (Bacchiani *et al.*, 2006; Reichart & Rappoport, 2007; Huang, 2009; Sagae, 2010) ont réussi à exploiter avec succès l'auto-apprentissage. L'hypothèse développée dans (Bacchiani *et al.*, 2006) est que la cause des faibles performances d'un analyseur sur un texte de genre différent serait le manque de couverture lexicale. Ce problème pourrait être partiellement résolu par l'ajout de données issues d'un texte, du même genre que le texte à analyser, contenant des éléments lexicaux et des structures syntaxiques propres à ce genre. Afin d'analyser le corpus d'évaluation du WSJ, l'analyseur de (Roark, 2001) a été entraîné sur un corpus annoté composé du BT (genre différent) et 200000 phrases brutes préalablement analysées du WSJ (même genre). Ils ont montré qu'en limitant l'influence des analyses provenant du corpus brut, les performances s'en trouvent améliorées. Ainsi, ils ont pondéré plus fortement les analyses du BT. Cette pondération signifie d'une manière abstraite que le WSJ a été concaténé à plusieurs copies du BT. Le gain obtenu est de +4.2 par rapport à un modèle appris sur le BT uniquement. (Bacchiani *et al.*, 2006) ont également effectué une

expérience similaire avec une inversion dans le rôle des corpus. Le corpus d'apprentissage annoté est maintenant le WSJ et le corpus brut ainsi que le corpus d'évaluation proviennent du BT. Le gain significatif de +4.5 par rapport à un modèle appris uniquement sur le WSJ semble confirmer l'hypothèse du manque de couverture lexicale. Ces résultats concluants sont à mettre en parallèle de ceux obtenus par (Gildea, 2001; Reichart & Rappoport, 2007; Sagae, 2010). (Gildea, 2001) se base sur un protocole d'expérimentation relativement similaire et obtient un gain de +3.7. Quant à (Reichart & Rappoport, 2007), ils arrivent aux mêmes conclusions en se limitant volontairement à un petit corpus d'apprentissage annoté. (Sagae, 2010) utilise un analyseur PCFG (Charniak, 2000) et le WSJ comme corpus d'apprentissage annoté. Le corpus brut de 320000 phrases provient du projet Gutenberg[73] dont le genre textuel se rapproche plus du BT que du WSJ. Comme on peut s'y attendre, les performances se dégradent fortement lorsque l'auto-apprentissage est évalué sur le corpus d'évaluation du WSJ[74]. En revanche, cette méthode permet d'améliorer significativement les performances sur le corpus d'évaluation du BT avec un gain de +1.9, validant une fois de plus l'hypothèse du manque de couverture lexicale.

L'approche d'auto-apprentissage décrite dans (Huang, 2009) vise à améliorer l'analyse d'un genre textuel unique en utilisant la puissance des annotations latentes de BKY. Leur hypothèse est que la combinaison d'analyses de phrases brutes avec celles de même genre annotées manuellement aiderait l'entraînement du modèle PCFG-LA lors de la phase d'ajout des annotations latentes. Afin de réduire l'impact des erreurs d'analyse, ils proposent de limiter l'influence des analyses des phrases brutes dans le calcul des probabilités du modèle. Contrairement à la technique employée par (Bacchiani *et al.*, 2006), les corpus ne subissent aucune pondération. En revanche, ils ont appliqué une pondération équilibrée des probabilités postérieures du modèle, avec un poids différent selon l'origine de l'analyse. Cette approche a été évaluée sur deux langues, l'anglais en prenant le WSJ comme corpus annoté et 210000 phrases brutes du corpus journalistique BLLIP[75], ainsi que le chinois en prenant le Penn Chinese TreeBank 6.0 [CTB6][76] (Nianwen Xue & Palmer, 2005) comme corpus annoté et 210000 phrases brutes d'articles de presse chinoise. Le CTB6 est un corpus journalistique de taille moyenne d'environ 28000 phrases et 781000 mots. Les gains obtenus sont significatifs, que l'on utilise la totalité du WSJ, ou du CTB6, comme donnée annotée (+0.8 sur le WSJ et +1.03 sur le CTB6) ou seulement une partie (avec 20%, +2.0 sur le WSJ et +1.8 sur le CTB6).

### 2.5.2 Co-Apprentissage

L'approche appelée *co-apprentissage* (co-training en anglais) est un algorithme itératif qui peut être décrit de la manière suivante, d'après la définition donnée par (Blum & Mitchell, 1998) :

– Prendre deux (ou plus) *vues* d'un même problème de classification.

– Construire un modèle pour chacune de ces vues en l'entraînant sur un même corpus de

[73] `http://www.gutenberg.org`

[74] Le gain est de -1.0 par rapport à l'expérience sans auto-apprentissage.

[75] `http://bllip.cs.brown.edu/resources.shtml`

[76] Le Penn Chinese TreeBank est disponible à l'adresse `http://www.cis.upenn.edu/~chinese/ctb.html`

| Corpus d'apprentissage | Corpus d'évaluation | Charniak | Gildea | Steedman | Sagae | Huang |
|---|---|---|---|---|---|---|
| WSJ | WSJ | 86.4 | 86.4 | 75.0 | 89.13 | 90.63 |
| WSJ+WSJ⋆ | WSJ | 86.5 (+0.1) | - | 75.70 (+0.7) | - | - |
| WSJ+BT⋆ | WSJ | - | 86.6 (+0.2) | - | 88.06 (-1.07) | - |
| WSJ+BLLIP⋆ | WSJ | - | - | - | - | 91.46 (+0.83) |
| CTB6 | CTB6 | - | - | - | - | 84.15 |
| CTB6+CNews⋆ | CTB6 | - | - | - | - | 85.18 (+1.03) |

TAB. 2.17: Auto-apprentissage sur un unique genre textuel. ⋆ signifie que les analyses proviennent d'un corpus brut.

| Corpus d'apprentissage | Corpus d'évaluation | Gildea | Bacchiani | Sagae |
|---|---|---|---|---|
| WSJ | WSJ | - | 87.0 | - |
| BT | WSJ | - | 77.0 | - |
| BT+WSJ⋆ | WSJ | - | 81.2 (+4.2) | - |
| WSJ | BT | 80.6 | 81.1 | 83.56 |
| WSJ+BT⋆ | BT | 84.3 (+3.7) | 85.6 (+4.5) | - |
| WSJ+Gutenberg⋆ | BT | - | - | 85.42 (+1.86) |

TAB. 2.18: Auto-apprentissage sur plusieurs genres textuels. ⋆ signifie que les analyses proviennent d'un corpus brut.

données annotées.
– Pour chacun des modèles précédemment calculés, analyser un petit ensemble de données brutes provenant d'un corpus de plus grande taille.
– Pour chaque vue, les analyses de bonne qualité (d'après une fonction de score) sont extraites et sont ajoutées aux corpus d'apprentissage des autres vues.
– Les modèles sont recalculés sur leur corpus respectif et la procédure est répétée jusqu'à ce que le corpus de données brutes soit épuisé.

Cet algorithme peut être vu comme une entraide entre différents processus (vues) ayant le même but. Les avantages des uns peuvent profiter aux autres mais également les inconvénients. Cette technique a été employée dans divers processus comme la désambiguisation sémantique (Yarowsky, 1995), la reconnaissance d'entités nommées (Collins & Singer, 1999), la reconnaissance des syntagmes nominaux (Pierce & Cardie, 2001), l'étiquetage morpho-syntaxique (Clark *et al.*, 2003), la classification de pages web (Blum & Mitchell, 1998) et l'analyse syntaxique (Sarkar, 2001; Steedman *et al.*, 2003; McClosky *et al.*, 2006a,b; McClosky & Charniak, 2008; McClosky *et al.*, 2008, 2010). Les expériences de cette section traitant d'un genre textuel unique ont leurs résultats indiqués dans le tableau 2.19. Quant aux expériences d'adaptation à plusieurs genres textuels, les résultats sont signalés dans le tableau 2.20.

Dans (Steedman *et al.*, 2003), les deux vues sont deux analyseurs syntaxiques, un analyseur PCFG (Collins & Singer, 1999) et un analyseur LTAG (Sarkar, 2001). L'algorithme itératif, dont l'architecture est montrée dans la figure 2.31, démarre par l'entraînement de chaque analyseur sur un corpus annoté composé de 500 phrases tirées du WSJ. À chaque itération, 30

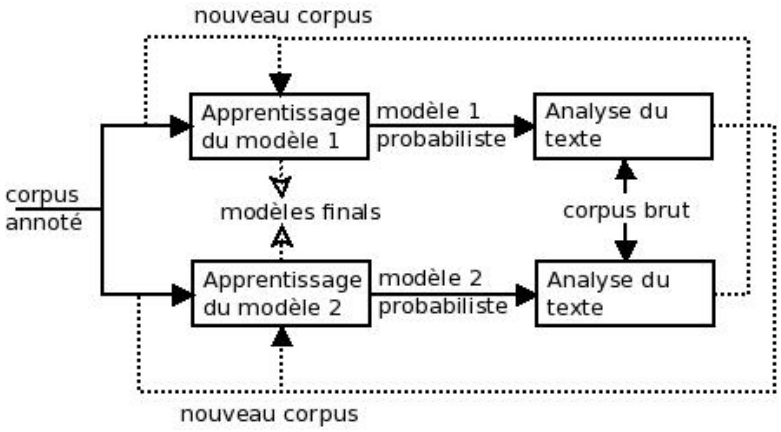



FIG. 2.31: Architecture générale de l'approche *co-apprentissage*.

phrases brutes issues des phrases restantes du WSJ sont analysées par l'analyseur PCFG et les 20 meilleures analyses, d'après une formule de score (probabilité jointe), sont ajoutées au corpus d'apprentissage de l'analyseur LTAG. Cette étape est également effectuée pour l'analyseur LTAG et les analyses produites sont ajoutées au corpus de l'analyseur PCFG. Chaque analyseur est ensuite réentraîné sur leur corpus d'apprentissage propre et l'algorithme continue tant qu'il reste des données brutes à analyser. Nous avons pu voir que cet algorithme cause une dégradation des performances lorsqu'il est utilisé avec une seule vue (auto-apprentissage). En revanche, la méthode de co-apprentissage permet d'obtenir un gain d'environ +3 sur le corpus d'évaluation du WSJ avec l'analyseur PCFG. Des expériences similaires décrites dans (Sarkar, 2001) confirment l'efficacité de cet algorithme.

Ces expériences ont été menées avec un petit corpus de phrases annotées manuellement (provenant du WSJ) et un grand corpus de phrases brutes. Elles montrent que des langues ayant de petits corpus annotés peuvent probablement obtenir de meilleures performances. Mais cela ne prouve en aucun cas que cet algorithme fonctionne lorsqu'un grand corpus annoté est exploité. De plus, il repose essentiellement sur une fonction de score permettant d'évaluer la qualité relative d'une analyse produite par les différents analyseurs syntaxiques. En ce sens, les expériences décrites dans (McClosky *et al.*, 2006a) diffèrent de ces travaux, tant au niveau de l'algorithme de co-apprentissage que de la masse de données annotées utilisées. La procédure, dont l'architecture est montrée dans la figure 2.32, consiste tout d'abord à produire les $n$ meilleures analyses pour chaque phrase d'un corpus brut avec un analyseur génératif (Jiménez & Marzal, 2000; Huang & Chiang, 2005). Ensuite, un réordonnanceur discriminatif (Charniak & Johnson, 2005) reclasse les analyses produites. Puis, la meilleure analyse de chaque phrase est ajoutée au corpus d'apprentissage initial. L'analyseur génératif est ensuite réentraîné sur ce nouveau corpus. Bien que les auteurs classent cet algorithme dans l'auto-apprentissage, les deux analyseurs peuvent être identifiés comme les deux vues sur les données d'entrées, excepté qu'ici, la vue du réordonnanceur est restreinte aux analyses de l'analyseur génératif.

Le corpus annoté utilisé dans leurs expériences est le WSJ et le corpus brut est un corpus journalistique, le North American News Text Corpus [NANC] (Graff, 1995) contenant 24 millions de phrases. Ils obtiennent un gain significatif de +0.7 pour l'analyseur génératif en incorpo-

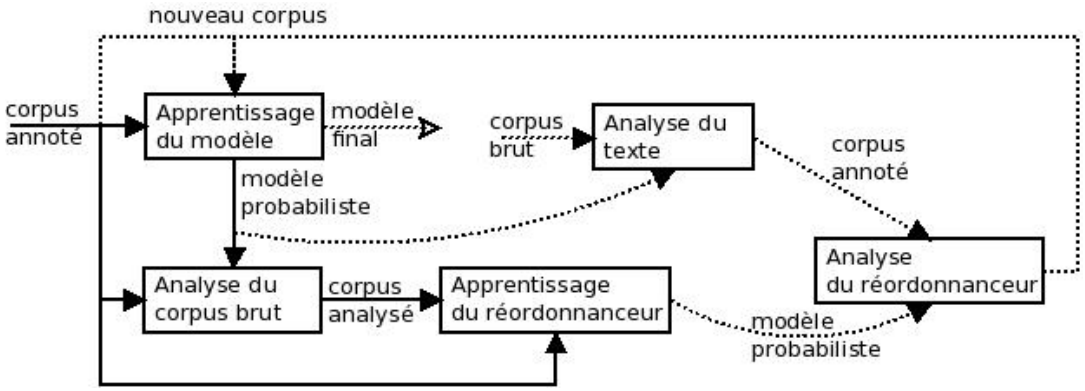



FIG. 2.32: Architecture générale de l'approche de co-apprentissage décrite dans (McClosky *et al.*, 2006a).

rant 750000 phrases du NANC au corpus d'apprentissage[77]. En reprenant l'idée développée par (Bacchiani *et al.*, 2006), ils ont montré qu'une pondération différente des arbres du corpus d'apprentissage selon leur origine permet d'améliorer significativement les résultats. Ainsi, en fixant un poids relatif de 5 aux arbres du WSJ par rapport à ceux du NANC, le gain est de +1.3 pour l'analyseur génératif en comparaison de l'expérience précédente. De plus, l'amélioration de la qualité des analyses produites par ce dernier a également un impact sur les performances du réordonnanceur. Les gains obtenus, sans avoir réentraîné le réordonnanceur, sont de +0.8 et +1.6 respectivement avec ou sans pondération des arbres[78].
Quelques travaux ont par la suite repris cet algorithme lors d'expériences d'adaptation sur des corpus de genres différents. Ainsi, (McClosky *et al.*, 2006b) ont évalué le protocole d'expérimentation précédent sur un corpus d'évaluation composé d'arbres du BT. Les résultats sont très proches de ceux obtenus par un analyseur et un réordonnanceur entraînés sur le BT et confirment donc l'efficacité de cette approche. Puis, (McClosky & Charniak, 2008) se sont intéressés à l'analyse syntaxique de documents bio-médicaux. Dans leur article, le corpus NANC est remplacé par 270000 phrases brutes choisies au hasard dans le corpus bio-médical Medline[79]. Le corpus évalué est le Genia TreeBank[80] (Ohta *et al.*, 2005) composé de phrases issues de Medline qui ont été annotées manuellement. Le co-apprentissage conduit à une réduction de 20% des erreurs (gain de +4.1) par rapport aux meilleurs résultats des travaux précédents sur ce même corpus (Lease & Charniak, 2005). Plus récemment, (Foster *et al.*, 2007) ont observé un gain significatif de +1.7 en prenant le British National Corpus [81] [BNC] (genre journalistique et littéraire) comme corpus brut et comme corpus d'évaluation.

[77]Ils observent, sans surprise, une dégradation des résultats lorsque les analyses des phrases du NANC proviennent de l'analyseur et non du réordonnanceur, ce qui correspond à de l'auto-apprentissage.
[78]Le réentraînement du réordonnanceur sur le nouveau corpus n'a pas d'effets positifs sur les performances.
[79]`http://www.ncbi.nlm.nih.gov/PubMed/`
[80]`http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/home/`
[81]`http://www.natcorp.ox.ac.uk/`

| Corpus d'apprentissage | Corpus d'évaluation | Sarkar | Steedman | (McClosky *et al.*, 2006a) |
|---|---|---|---|---|
| WSJ | WSJ | 70.59 | 75.0 | 90.5 |
| WSJ+WSJ⋆ | WSJ | 79.82 (+9.23) | 77.8 (+2.8) | - |
| WSJ+NANC⋆ | WSJ | - | - | 92.1 (+1.6) |

TAB. 2.19: Co-apprentissage sur un unique genre textuel. ⋆ signifie que les analyses proviennent d'un corpus brut.

| Corpus d'apprentissage | Corpus d'évaluation | (McClosky *et al.*, 2006b) | (McClosky & Charniak, 2008) | Foster |
|---|---|---|---|---|
| WSJ | BT | 85.2 | - | - |
| WSJ+NANC⋆ | BT | 88.2 (+3.0) | - | - |
| WSJ | GENIA | - | 80.2 | - |
| WSJ+MEDLINE⋆ | GENIA | - | 84.3 (+4.1) | - |
| WSJ | BNC | - | - | 83.9 |
| WSJ+BNC⋆ | BNC | - | - | 85.6 (+1.7) |

TAB. 2.20: Co-apprentissage sur plusieurs genres textuels. ⋆ signAdaptation des analyseurs à de petits corpus et à des textes de genres différentsifie que les analyses proviennent d'un corpus brut.

### 2.5.3 Autre approches

Le constat fait par (Foster, 2010) souligne que la plupart des expériences faites sur l'adaptation des analyseurs sont évaluées sur des types de textes ayant un formatage particulier, généralement des articles de journaux. Or, de nos jours, les documents textuels ne se résument pas uniquement à ce genre de textes. Le contenu Web 2.0 généré par les utilisateurs d'internet (blogs, forums,...) est par exemple une source textuelle riche et de nombreuses applications requièrent une analyse, même partielle, de ces documents. Aussi, deux discussions à propos du sport tirées d'un forum ont été annotées manuellement afin de former deux petits corpus, un de développement et un autre d'évaluation (environ 200 phrases chacun). Les premiers résultats obtenus par BKY, entraîné sur le WSJ, sont logiquement faibles ($F_1$ de 77.56 et 78.37). Afin d'améliorer les performances générales, (Foster, 2010) propose d'homogénéiser les données annotées des différents corpus en appliquant des règles de transformation, crées manuellement, sur les arbres de ces corpus (Aw *et al.*, 2006; van der Plas *et al.*, 2009). Dans le but de déterminer un ensemble de règles performant, celles-ci ont été systématiquement évaluées sur le corpus de développement. Cette sélection préalable a permi de ne conserver que quatre règles pour le corpus d'apprentissage et trois règles pour les autres corpus. Par exemple, les discussions de forum contiennent de nombreux acronymes particuliers permettant de réduire une suite de mots à un mot simple (*lol*, *mdr*,...). Ces acronymes n'étant généralement pas présents dans des articles de presse comme ceux du WSJ, ils sont donc supprimés. Les gains obtenus, après l'application de ces règles de transformation, sont significatifs sur les deux corpus (respectivement +2.71 et +1.57). Malgré tout, on voit rapidement les limites de cette approche qui sont les limites habituelles liées à la création manuelle de règles, à savoir la maintenance et la portabilité.

(Petrov *et al.*, 2010) se sont penchés sur l'analyse automatique de questions. En général, les questions sont sous-représentées dans les corpus annotés manuellement. Le WSJ ne contient par exemple que 334 *questions* parmi les 40000 phrases. Or, les structures syntaxiques liées aux questions sont différentes de celles que l'on trouve dans les phrases déclaratives. Ce cas de figure est donc un problème d'adaptation de l'analyseur à un genre textuel particulier. Ils ont observé une chute des performances lorsque divers analyseurs en constituants (Charniak & Johnson, 2005; Petrov *et al.*, 2006) et en dépendances (McDonald *et al.*, 2006; Nivre *et al.*, 2006) sont entraînés sur le WSJ puis évalués sur un corpus de questions/réponses QuestionBank[82] [QB] (Judge *et al.*, 2006). Ils ont remarqué que les analyseurs en constituants semblent moins souffrir du changement de genre textuel. Par exemple, les performances de BKY indiquent une baisse de seulement 5% contre 14% pour Malt en terme de UAS par rapport au score obtenu sur le corpus d'évaluation du WSJ. Cependant, les analyseurs les plus performants en terme d'adaptation de domaine sont aussi ceux qui ont la plus grande complexité en temps d'analyse (au minimum cubique en la longueur de la phrase). Cela les rend donc inexploitables par des processus lourds qui les utilisent en tant que prétraitement d'un texte (moteur de recherche basé sur les questions/réponses, intelligence artificielle, jeopardy,...). Ils proposent donc une méthode appelée *uptraining* consistant à entraîner un analyseur rapide mais peu efficace sur les données de sortie d'un autre analyseur plus lent mais plus performant. Cette approche a de fortes similitudes avec les techniques de co-apprentissage (Steedman *et al.*, 2003; McClosky *et al.*, 2006a) excepté qu'ici (Petrov *et al.*, 2010) ne cherchent pas à améliorer les performances du meilleur des deux analyseurs. Concrètement, ils ont tout d'abord utilisé le meilleur analyseur, BKY, pour annoter automatiquement une grande quantité de données brutes du même genre que le corpus à évaluer (une partie du QB). Les arbres générés sont transformés en dépendances puis ajoutés au corpus d'apprentissage, composé du WSJ en dépendances. L'apprentissage du modèle de l'analyseur le plus mauvais (une variante de Malt) est ensuite effectué sur ce nouveau corpus. Grâce à cette méthode, les performances de cet analyseur sur le corpus d'évaluation du QB passent de 72.23 à 88.63 (UAS).
Plus récemment, (Candito *et al.*, 2011) ont observé une réduction d'erreurs de 21% sur un corpus bio-médical du français analysé par un analyseur PCFG-LA (Petrov & Klein, 2007; Attia *et al.*, 2010) entraîné sur le FTB-UC. Pour ce faire, ils se sont penchés sur le problème de dispersion des données lié aux PCFG. Leur méthodologie consiste à remplacer les mots des différents corpus par des classes lexicales. Ces classes ont été apprises de manière non-supervisée sur un grand corpus brut contenant, en partie, des phrases du même genre que le corpus d'évaluation. Nous décrivons plus précisément cette approche dans le chapitre 6.

## 2.6 Conclusion

Dans ce chapitre, nous avons effectué un tour d'horizon des différents systèmes d'analyse syntaxique probabiliste. Actuellement, les systèmes les plus performants sont basés soit sur un modèle génératif, soit sur un modèle discriminant. Les grammaires PCFG font partie de la première catégorie de modélisation. Elles permettent aux analyseurs en constituants d'obtenir des

[82] `http://www.computing.dcu.ie/~jjudge/qtreebank/`

performances élevées, et ce pour diverses langues comme l'anglais ou le français. Ces performances sont néanmoins obtenues en appliquant des traitements afin de palier la faiblesse de la modélisation, qui est principalement liée à deux problèmes : (a) des hypothèses d'indépendances trop fortes entre les règles. (b) le faible conditionnement lexical. Ces traitements sont divers mais consistent généralement à modifier le corpus annoté (markovisation horizontale, verticale,...) ou la grammaire (divisions de symboles, lexicalisation...). Il faut noter que la complexification des règles d'une grammaire PCFG conduit, malgré tout, à un éventuel problème de dispersion des données. En effet, les corpus d'apprentissage sont généralement de petite taille, ce qui rend certains phénomènes très rares. De nombreux travaux récents ont proposé des solutions efficaces pour résoudre ce problème, et notamment une approche consistant à remplacer les mots du corpus par des classes lexicales. Ces classes contiennent des mots partageant certaines propriétés lexicales ou syntaxiques. La réduction de la taille du lexique du corpus d'apprentissage et des textes à analyser a potentiellement pour effet de réduire la taille du lexique tout en améliorant les performances générales des analyseurs. Dans le chapitre 6, nous verrons en détail cette approche par classe de mots, et nous proposerons nos propres algorithmes permettant d'obtenir des classes lexicales à partir de lexiques syntaxiques du français.
En ce qui concerne les modèles discriminants, ils ont l'avantage de ne pas poser d'hypothèses d'indépendances entre les variables d'entrées. D'un point de vue mathématique, ils sont donc plus performants que les modèles génératifs, car ils peuvent modéliser de nombreuses informations (traits), et ce, sans recourir à des traitements sur le corpus comme pour les grammaires PCFG. Actuellement, ce type de modèle permet aux analyseurs en dépendances, comme Malt et MST, d'obtenir des scores très élevés, que ce soit en terme de dépendances typées ou non typées. En revanche, il apparaît clairement que la complexité, pour modéliser efficacement des arbres de constituants, est grande car les analyseurs en constituants discriminatifs sont majoritairement moins performants que les analyseurs génératifs comme BKY. Quant à la méthode de réordonnancement discriminatif, lorsqu'elle est utilisée en combinaison d'un analyseur lambda, elle permet d'améliorer significativement la qualité des analyses en constituants ou en dépendances produites par ce même analyseur. De plus, à travers l'algorithme de co-apprentissage, c'est une méthode très efficace pour résoudre partiellement la problématique liée à l'adaptation des analyseurs à de petits corpus et à des textes de genres différents.
On peut néanmoins noter que la plupart des évaluations concernant l'analyse syntaxique probabiliste ont été réalisées avec une segmentation parfaite du texte, car identique à celle du corpus évalué. Or, dans les cas réels d'application, la segmentation d'un texte est très rarement disponible et les segmenteurs automatiques actuels sont loin de proposer une segmentation de bonne qualité, et ce, à cause de la présence de nombreuses unités multi-mots (mots composés, entités nommées,...). Dans le cadre de notre travail, nous nous sommes donc penché sur l'identification automatique des unités multi-mots, et leur impact sur l'analyse syntaxique. Pour ce faire, nous proposons, dans le chapitre 5, d'évaluer deux stratégies discriminantes d'intégration de ces expressions dans un contexte réel d'analyse syntaxique en constituants : (a) pré-segmentation lexicale au moyen d'un étiqueteur-segmenteur de mots composés basé sur un modèle CRF ; (b) analyse basée sur une grammaire incluant l'identification des mots composés, suivie d'une phase de réordonnancement des analyses à l'aide d'un modèle maximum d'entropie intégrant des traits concernant les mots composés.

# 3

# Étiquetage morpho-syntaxique

## 3.1 Introduction

Dans le domaine du traitement automatique des langues, l'étiquetage morpho-syntaxique a pour but d'identifier la classe morpho-syntaxique de chaque mot d'une phrase à partir de son contexte et de connaissances lexicales. L'architecture d'un étiqueteur, illustrée par la figure 3.1, est généralement formée de trois étapes successives.


FIG. 3.1: Architecture générale d'un étiqueteur morpho-syntaxique.

La phase de tokenisation segmente le texte brut en tokens simples et parfois composés[83]. Ensuite, l'analyse lexicale ambigüe assigne, pour chacun des tokens, les étiquettes morpho-syntaxiques possibles à partir d'un lexique de corpus annoté ou de dictionnaires externes. Parfois, cette étape consiste simplement à affecter, pour chaque token, l'ensemble des étiquettes du jeu d'étiquettes. Les ambiguités lexicales inhérentes à la langue naturelle sont généralement représentées par un automate acyclique ambigu de la phrase. Chaque état de l'automate est une étiquette possible pour un mot donné de la phrase. Les états correspondant à des mots différents sont reliés par des transitions entrantes et sortantes (graphe ordonné acyclique). La figure 3.2 montre un automate possible pour les mots de la phrase *L'immeuble a une taille immense* après une tokenisation en mots simples[84]. On peut voir, par exemple, que le mot *taille* est associé à deux états et donc deux étiquettes morpho-syntaxiques possibles, l'une verbale `V` et l'autre nominale `NC`. De plus, ces états sont reliés aux états associés aux mots *une* et *immense* situés dans l'environnement immédiat gauche et droit de *taille*.

[83]On considère qu'un mot de la phrase est assimilé à un token, e.g. une séquence de caractères alpha-numériques.

[84]Un état initial est ajouté afin d'obtenir un automate émondé. De plus, le jeu d'étiquettes utilisé est celui du corpus FTB-UC (section 1.2.1)

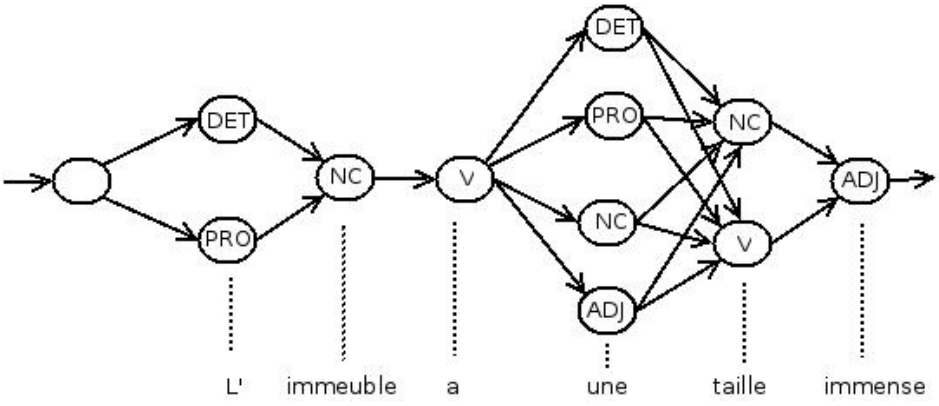



FIG. 3.2: Automate ambigu de la phrase *L'immeuble a une taille immense*.

L'étape finale, appelée linéarisation, consiste à désambiguiser l'automate afin de trouver une séquence d'étiquettes optimale pour la phrase en entrée. En sortie, l'étiqueteur associe une unique étiquette morpho-syntaxique à chaque mot du texte. Par exemple, la linéarisation correcte de l'automate montré dans la figure 3.2 donne la séquence d'étiquettes suivante :

| L' | immeuble | a | une | taille | immense |
|---|---|---|---|---|---|
| DET | NC | V | DET | NC | ADJ |

Il existe trois approches permettant de réaliser l'étape de linéarisation. Une première approche, dite *symbolique*, consiste à appliquer des règles de désambiguisation écrites manuellement sur l'automate afin de supprimer des ambiguités lexicales. On pourra citer notamment l'étiqueteur EngCG (Karlsson *et al.*, 1995; Voutilainen, 1995) ou encore le système Elag[85] (Laporte & Monceaux, 1999). La figure 3.3 montre une règle Elag du français représentée sous la forme d'un automate[86]. D'après cette règle, après l'adverbe *si* (`ADV`), on ne peut trouver qu'un adjectif (`A`), un adverbe, un verbe au participe passé (`V :K`), ou une ponctuation. Le système retire de l'automate les chemins dans lesquels la règle n'est pas respectée. Les phrases d'exemple suivantes illustrent les cas tolérés par la règle :

*elle est si belle*

*si intimement mêlée*

*si attachée à son mari*

*Tu n'en veux pas ? Si !*

Généralement, les systèmes à base de règles n'effectuent qu'une désambiguisation partielle de l'automate laissant la linéarisation complète à d'éventuelles heuristiques ou à un processus stochastique. Pour ce dernier cas, on parle souvent d'approche hybride (Sigogne, 2010).
La deuxième approche, dite *statistique*, consiste à modéliser le problème de manière probabiliste. Le but est de sélectionner la séquence d'étiquettes de l'automate ayant la meilleure

[85]Inclus dans l'outil *Unitex* disponible à l'adresse `http://igm.univ-mlv.fr/~unitex/`
[86]Cette règle Elag a été créée par Olivier Blanc et Anne Dister.

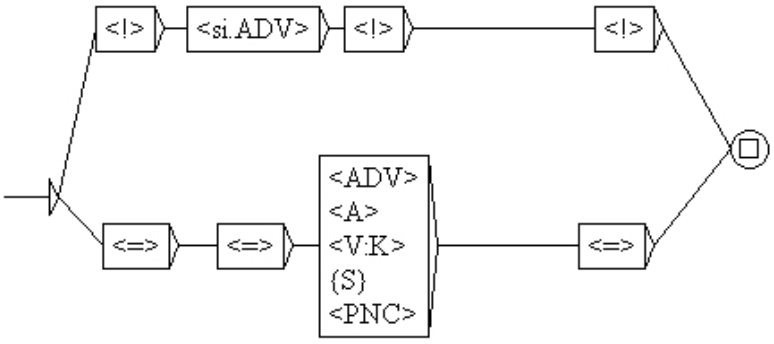



FIG. 3.3: Règle de désambiguisation du système Elag concernant l'adverbe *si*.

probabilité d'après un modèle probabiliste appris sur un corpus. L'utilisation des probabilités permet de développer des étiqueteurs efficaces qualitativement mais sous la condition qu'un corpus annoté de la langue soit disponible. Les étiqueteurs les plus simples intègrent un modèle génératif de Markov caché (Brill, 1995; Brants, 2000; Thede & Harper, 1999; Schmid, 1995). D'autres étiqueteurs plus récents et plus performants se basent sur des modèles discriminants comme le Maximum d'Entropie (Ratnaparkhi, 1996; Toutanova & Manning, 2000; Toutanova *et al.*, 2003; Denis & Sagot, 2009), les Champs Conditionnels Aléatoires (Lafferty *et al.*, 2001; Tsuruoka *et al.*, 2009; Constant *et al.*, 2011; Constant & Sigogne, 2011), les Séparateurs à Vaste Marge (Giménez & Màrquez, 2004) ou encore le Perceptron (Collins, 2002; Shen *et al.*, 2007; Spoustová *et al.*, 2009).
Dans les sections qui suivent, nous décrivons, pour chaque approche, le ou les formalismes mathématiques sous-jacents, ainsi que quelques étiqueteurs parmi la multitude de systèmes disponibles actuellement. Puis, nous indiquons les résultats de diverses expériences sur l'étiquetage morpho-syntaxique.

## 3.2 Modèles génératifs markoviens

La linéarisation stochastique consiste à déterminer la séquence d'étiquettes $y^\star$ qui maximise la probabilité d'avoir une séquence d'étiquettes $y$ étant donnée une séquence de tokens $x$, parmi les séquences d'étiquettes possibles $Y$. Le calcul des probabilités des séquences présentes dans l'automate $p(y|x)$ peut être effectué par l'intermédiaire d'un modèle génératif de Markov caché [MMC] (Rabiner, 1989). Les MMC sont une extension des modèles génératifs bayésiens pour la modélisation efficace de structures linéaires séquentielles. Dans un modèle bayésien, les variables d'entrées $x$ sont considérées comme indépendantes. Dans le cadre d'un MMC, ces hypothèses sur les variables conduisent à modéliser la probabilité jointe $p(y, x)$ des variables d'entrées $x$ et des prédictions $y$. Un MMC est défini par les équations suivantes :

$$y^\star = \underset{y \in Y}{argmax}\ p(y|x) \simeq \underset{y \in Y}{argmax}\ p(y, x) \tag{3.1}$$

$$p(x, y) = p(x_1|y_1) \prod_{i=2}^{n} p(x_i|y_i)p(y_i|y_{i-n}...y_{i-1}) \tag{3.2}$$

D'après l'équation 3.2, la probabilité globale d'une séquence $y$ est égale à un produit de probabilités locales permettant de passer d'un état à un autre de l'automate du texte. Ces probabilités locales sont obtenues par le produit de deux types de probabilités calculées à partir de statistiques extraites d'un corpus annoté. La probabilité *d'émission* $p(x_i|y_i)$, modélisant la dépendance entre un token $x_i$ et une étiquette $y_i$, est obtenue en divisant le nombre d'occurrences de la paire $(x_i, y_i)$ par le nombre d'occurrences de $y_i$ (équation 3.3). La probabilité *de transition* $p(y_i|y_{i-n}...y_{i-1})$, modélise la dépendance entre une étiquette $y_i$ et son contexte composé des $n$ étiquettes précédentes. Elle est obtenue en divisant le nombre d'occurrences de la séquence d'étiquettes $(y_{i-n}, ..., y_{i-1}, y_i)$, appelé aussi *n-gramme*, par le nombre d'occurrences du contexte $y_{i-n}, ..., y_{i-1}$ (équation 3.4).

$$p(x_i|y_i) = \frac{occ(x_i, y_i)}{occ(y_i)} \tag{3.3}$$

$$p(y_i|y_{i-n}...y_{i-1}) = \frac{occ(y_{i-n}, ..., y_{i-1}, y_i)}{occ(y_{i-n}, ..., y_{i-1})} \tag{3.4}$$

L'*ordre* d'un MMC correspond à la taille du contexte utilisé pour le calcul des probabilités de transitions (paramètre $n$ dans l'équation 3.4). Une taille de contexte élevée permet théoriquement d'améliorer les performances d'un étiqueteur. En effet, un modèle de second ordre, se basant sur un contexte de deux étiquettes, intègre plus d'informations qu'un modèle de premier ordre qui se base sur un contexte d'une seule étiquette. En revanche, dans la pratique, l'ordre 2 est généralement utilisé car un ordre élevé implique également que le corpus d'apprentissage soit de grande taille. Si tel n'est pas le cas, les performances chutent. Un MMC d'ordre $n$ peut être répresenté graphiquement par un automate. La figure 3.4 montre un exemple d'automate pour un MMC de premier ordre (un contexte d'une étiquette). Dans cet automate, on peut distinguer deux types d'états : chaque étiquette est inscrite dans un noeud rectangulaire et chaque mot est sous forme d'ovale. Les flèches pleines représentent la probabilité de transition de passer d'une étiquette à une autre. Quant aux flèches en pointillés, elles représentent la probabilité d'émission de générer un mot étant donné une étiquette. Un noeud spécial `START` est ajouté à l'automate et relié à chaque noeud rectangulaire afin de modéliser la probabilité de transition du premier mot d'une phrase.

Un modèle de Markov permet de donner une probabilité à une séquence d'étiquettes. Cependant, l'automate ambigu d'une phrase contient hypothétiquement une multitude de séquences possibles. Aussi, pour extraire la meilleure séquence (linéarisation), il est d'usage d'appliquer sur l'automate un algorithme de programmation dynamique appelé *Viterbi* fonctionnant en deux passes (Viterbi, 1967; Rabiner, 1989). Une première passe parcourt l'ensemble des états de l'automate et pour chacun d'eux calcule la meilleure probabilité, dite *partielle*, d'arriver dans cet état étant donné les précédents états possibles. La deuxième passe démarre de l'état final et remonte l'automate en ne conservant que les états qui font parti du chemin maximisant la probabilité globale obtenue à l'état final. La sortie de l'algorithme est la séquence d'étiquettes ayant la meilleure probabilité. L'algorithme de Viterbi a une complexité en $O(n^2)$ avec $n$ le nombre de transitions de l'automate. La probabilité partielle $\delta_t$ d'un état $i$ associé à un token

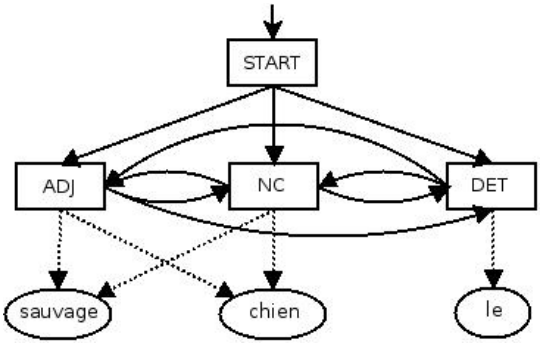



FIG. 3.4: Représentation graphique sous forme d'automate d'un modèle de Markov caché de premier ordre.

$w_t$ à la position $t$ de la phrase est calculée par la formule suivante :

$$\delta_t(i) = \underset{j}{argmax}(\delta_{t-1}(j) \cdot p_{emission}(i, w_t) \cdot p_{transition}(j, i)) \tag{3.5}$$

où $\delta_{t-1}(j)$ est la probabilité partielle d'atteindre l'état $j$ du mot précédent (position t-1), et $p_{transition}(j, i)$ la probabilité de transition et $p_{emission}(i, w_t)$ la probabilité d'émission. À l'initialisation ($t = 0$) et pour tous les états reliés au noeud *START*, l'équation est la suivante :

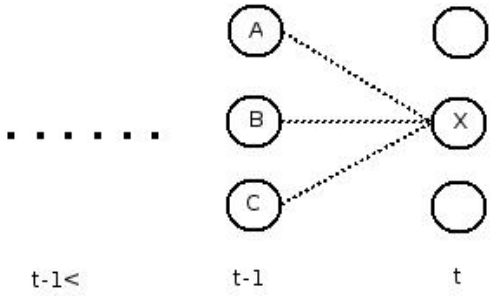
$$\delta_0(i) = p_{emission}(i, w_t) \cdot p_{transition}(START, i) \tag{3.6}$$

La figure 3.5 illustre un cas de calcul d'une probabilité partielle pour un état $X$ d'un automate ambigu, partiellement representé dans la figure 3.5. Trois état ($A$, $B$ et $C$) à la position t-1 permettent d'atteindre $X$, ce qui revient à choisir parmi ces états, celui qui maximise la probabilité partielle de $X$ (équation 3.5b).
Comme la plupart des formalismes génératifs, les MMC souffrent d'un problème de dispersion des données causé principalement par les hypothèses d'indépendances appliquées pour le calcul des probabilités. Un MMC d'ordre élévé, bien que théoriquement performant, nécessite un corpus d'apprentissage de taille conséquente. Or, si tel n'est pas le cas, il en résulte une estimation difficile des probabilités des phénomènes rares ou inconnus[87]. En effet, si un n-gramme ou un mot rencontrés dans une phrase est inconnu, alors les probabilités de transition ou d'émission sont égales à 0. Dans la section suivante, nous verrons succinctement quelques algorithmes, parmi les plus efficaces, permettant d'atténuer la dispersion des données.

### 3.2.1 Étiqueteurs basés sur un modèle de Markov caché

Les étiqueteurs basés sur un MMC intègrent en général un algorithme permettant de résoudre certains problèmes liés au formalisme génératif. Quelques-uns comme (Brill, 1995) ont tenté de réduire la complexité offerte par l'analyse lexicale ambigüe. D'autres comme (Schmid, 1995;

[87]Nous rappelons que par phénomène rare (ou inconnu), nous entendons qu'il est rare (ou absent) dans le corpus ayant servi à l'apprentissage des analyseurs.




(a) Automate partiellement représenté.

$$\delta_t(X) = argmax( \\ \delta_{t-1}(A) \cdot p_{emission}(X, w_t) \cdot p_{transition}(A, X), \\ \delta_{t-1}(B) \cdot p_{emission}(X, w_t) \cdot p_{transition}(B, X), \\ \delta_{t-1}(C) \cdot p_{emission}(X, w_t) \cdot p_{transition}(C, X) \\ )$$

(b) Probabilité partielle de l'état $X$.

FIG. 3.5: Calcul de la probabilité partielle d'un état $X$ de l'automate.

Thede & Harper, 1999; Brants, 2000) se sont penchés sur la dispersion des données. Dans cette section, nous décrivons succinctement ces différents algorithmes et nous indiquons également, dans le tableau 3.1, les performances de chaque étiqueteur sur le PTB[88]. Afin d'établir une base de comparaison, nous avons indiqué les résultats obtenus par un étiqueteur basé sur un simple MMC de second ordre (Weischedel *et al.*, 1993). Nous pouvons déjà remarquer qu'au vu de la simplicité du modèle, les performances sont étonnament élevées.
Dans son étiqueteur, (Brill, 1995) exploite une approche à base de règles de transformation d'étiquettes. Ces règles sont destinées à être appliquées sur un étiquetage initial généré pour un texte brut afin de modifier les étiquettes assignées à certains mots. L'exemple suivant est une règle de transformation possiblement exploitable par l'étiqueteur de Brill pour le français :

*Changer l'étiquette ADV en une étiquette NC si l'étiquette précédente est DET*

Les règles sont automatiquement créées par un algorithme récursif, montré dans la figure 3.6, à partir d'erreurs d'étiquetage initiales. Tout d'abord, la version sans annotations du corpus d'apprentissage est annotée par un système d'étiquetage quelconque. Puis, les données produites sont comparées à l'étiquetage correct du corpus et des règles de transformation sont calculées à partir des erreurs observées. Cette étape de comparaison est récursive et consiste à déterminer un ensemble de règles optimal minimisant le nombre d'erreurs d'étiquetage. Les gains obtenus par l'étiqueteur de Brill sont de +0.2 pour les mots inconnus et +0.3 pour l'ensemble des mots par rapport aux performances de (Weischedel *et al.*, 1993).
L'étiqueteur TreeTagger[89] (Schmid, 1995) propose de calculer les probabilités grâce à un arbre

[88]Les étiqueteurs ont été entraînés sur le corpus d'apprentissage du PTB et évalués sur la partie évaluation de ce même corpus. Les scores indiqués proviennent principalement de l'article de (Thede & Harper, 1999).
[89]`http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

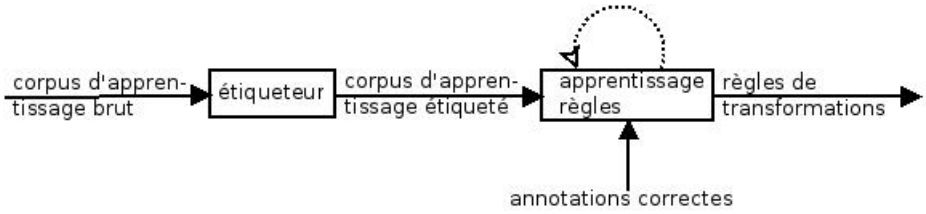



FIG. 3.6: Algorithme récursif de Brill permettant de déterminer un ensemble optimal de règles de transformation.

binaire de décision. Cet arbre, construit automatiquement à partir d'un corpus, détermine la taille appropriée du contexte devant être utilisé pour calculer les probabilités de transition. La figure 3.7 montre un exemple d'arbre binaire de décision. La probabilité d'un n-gramme est obtenue en suivant le chemin correspondant à ce n-gramme dans l'arbre jusqu'à arriver à une feuille. Par exemple, nous souhaitons déterminer la probabilité d'un adjectif précédé d'un nom commun et d'un déterminant, $p(ADJ|DET, NC)$. La première étape consiste à répondre au test contenu dans le noeud racine de l'arbre ($t_{-1} = NC?$). L'étiquette du mot précédent est `NC`, on suit donc le chemin étiqueté par $oui$. Le test suivant ($t_{-2} = DET?$) est positif également, ce qui nous amène à une feuille de l'arbre contenant une table de probabilités. La dernière étape consiste à extraire de cette table la probabilité associée à l'étiquette courante `ADJ`.

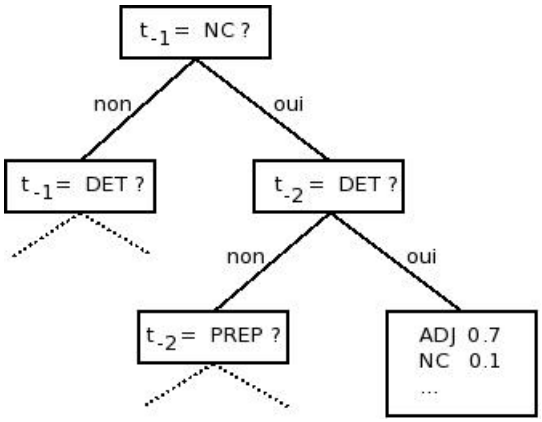



FIG. 3.7: Exemple d'arbre binaire de décision créé et utilisé par TreeTagger.

La construction de l'arbre de décision est effectuée grâce à une version modifiée de l'algorithme *ID3-algorithm* (Quinlan, 1986). Cet algorithme récursif crée un test à chaque itération qui divise l'ensemble des n-grammes d'apprentissage en deux sous-ensembles d'après la distribution de probabilités de l'étiquette à prédire. Les tests créés par cet algorithme sont de la forme suivante :

$$y_{-i} = t?; i \in 1, 2, ..., n; t \in T \tag{3.7}$$

où $t$ est une étiquette morpho-syntaxique du jeu $T$. Malgré l'originalité de cette approche, TreeTagger a des performances légèrement supérieures à (Weischedel *et al.*, 1993) (gain global de +0.24).

(Thede & Harper, 1999; Brants, 2000) tentent de répondre au problème de dispersion en appliquant des méthodes de lissage des probabilités du modèle. Ils proposent un algorithme d'interpolation linéaire, dans le cadre d'un modèle de second ordre, qui consiste à combiner des statistiques provenant d'unigrammes, de bigrammes et de trigrammes afin de calculer les probabilités de transition. Ces probabilités sont estimées par l'équation suivante :

$$p(y_i|y_{i-2}y_{i-1}) = \lambda_1\hat{p}(y_i) + \lambda_2\hat{p}(y_i|y_{i-1}) + \lambda_3\hat{p}(y_i|y_{i-2}y_{i-1}) \tag{3.8}$$

où $\hat{p}$ est une fonction calculant le maximum de vraisemblance (section 2.4.3) et $\lambda_1+\lambda_2+\lambda_3 = 1$. Les paramètres $\lambda$ sont estimés grâce à une méthode appelée *deleted interpolation* basée sur des statistiques d'occurrences. Afin d'être capable de donner une probabilité aux mots inconnus, (Thede & Harper, 1999; Brants, 2000) proposent d'utiliser le principe de la signature du mot. Cette approche, déjà employée avec succès pour l'analyse syntaxique (section 2.3.2), consiste à remplacer un mot inconnu par un autre mot (signature) calculé la plupart du temps à partir d'informations morphologiques comme les suffixes (Samuelsson, 1993). La probabilité d'émission $p(x_i|y_i)$ d'un mot inconnu $x_i$ est considérée comme égale à la probabilité $p(S(x_i)|y_i)$ avec $S$ une fonction renvoyant la signature du mot. Les statistiques concernant les signatures sont extraites au moment de l'apprentissage du modèle. Sur l'anglais, cette technique semble efficace car le gains obtenu par l'étiqueteur TnT de (Brants, 2000)[90] sur les mots inconnus est de +0.86 par rapport à (Weischedel *et al.*, 1993).

| Étiqueteur | Performances globales | Mots inconnus |
|---|---|---|
| (Weischedel *et al.*, 1993) | 96.3 | 82 |
| (Brill, 1995) | **96.6** | 82.2 |
| (Schmid, 1995) | 96.54 | - |
| (Brants, 2000) | 96.46 | **85.86** |

TAB. 3.1: Performances des étiqueteurs génératifs sur le corpus d'évaluation du PTB.

## 3.3 Modèles discriminants

Dans la section précédente, nous avons pu observer que les systèmes d'étiquetage basés sur un MMC permettent d'obtenir des performances élévées malgré la simplicité du modèle génératif. Cependant, on a pu également constater que les performances sont presque identiques pour l'ensemble des étiqueteurs. En effet, ils exploitent sensiblement les mêmes informations car les hypothèses d'indépendances des MMC rendent complexe la modélisation d'informations autres que des n-grammes. À l'opposé, les modèles discriminants ont l'avantage de ne pas poser d'hypothèses d'indépendances, ce qui permet de faire intervenir des informations diverses dans le calcul des probabilités (section 2.4). La modélisation discriminative d'un problème peut être vu comme un processus à deux étapes distinctes. La première étape consiste à tirer une série d'observations d'un corpus, que l'on appelle *traits*, extraits généralement par des patrons de

[90] `http://www.coli.uni-saarland.de/~thorsten/`

traits[91]. Ces traits sont ensuite intégrés au modèle stochastique de manière à obtenir un modèle uniforme, sous contraintes de ces traits, pouvant refléter la distribution initiale du corpus.
Il existe trois différences majeurs entre les étiqueteurs discriminatifs qui influent plus ou moins sur les performances générales. La première différence concerne le type de formalisme utilisé. Parmi les plus populaires, on peut citer les modèles de Maximum d'Entropie [ME] (Jaynes, 1957; Berger *et al.*, 1996), les Champs Conditionnels Aléatoires [CRF] (Lafferty *et al.*, 2001), les Séparateurs à Vaste Marge [SVM] (Boser *et al.*, 1992) ou encore le Perceptron (Rosenblatt, 1958). La deuxième différence est à propos des méthodes mathématiques d'estimation des paramètres du modèle. On peut rappeler que (Malouf, 2002) a montré, à travers une évaluation comparative, que les méthodes de second ordre (Cauchy, Quasi-Newton,...) semblent les plus performantes en terme de nombre d'itérations avant convergence des paramètres. Et enfin, dans la plupart des travaux, le point central concerne la sélection des informations à inclure au modèle. Bien qu'il y ait en général un socle de patrons de traits commun à tous les systèmes d'étiquetage, nous verrons que pour se différencier en terme de gains, il est nécessaire d'étendre le spectre des données exploitables à l'utilisation de ressources externes au corpus et notamment des dictionnaires (Denis & Sagot, 2009; Constant & Sigogne, 2011).

### 3.3.1 Étiqueteurs basés sur un modèle discriminant

Les premiers travaux concernant l'utilisation d'un modèle discriminant pour l'étiquetage sont décrits dans (Ratnaparkhi, 1996). Ce dernier se base sur un modèle ME et exploite l'ensemble des patrons de traits montré dans le tableau 3.2, avec $x_1, x_2, ..., x_n$ le vecteur de tokens de la phrase et $y_1, y_2, ..., y_n$ la séquence d'étiquettes à prédire. Ratnaparkhi a classé les patrons en trois catégories distinctes. Certains sont appliqués uniquement si le mot courant à étiqueter n'est pas rare, c'est à dire s'il apparaît au moins $M$ fois dans le corpus d'apprentissage ($M$ étant un paramètre à définir manuellement). D'autres patrons sont au contraire applicables exclusivement sur des mots rares ou inconnus, ceci afin d'améliorer l'étiquetage de ce type de mot. Et enfin, plusieurs patrons sont applicables sur tout type de mot indépendamment du nombre d'occurrences de ce mot dans le corpus. Nous avons indiqué dans le tableau 3.2 les patrons utilisés selon les différents types de mots (rare, courant, tous)[92]. La fonction $occ(w_i)$ renvoie le nombre d'occurrences du mot $w_i$ dans le corpus d'apprentissage.
De la même façon qu'il y a deux types de probabilité dans la modélisation MMC (émission et transition), on peut distinguer deux types de patron de traits. Les patrons dits *lexicaux* extraient des informations concernant le mot courant comme par exemple les préfixes, les suffixes ou encore le fait qu'il contienne un trait d'union. Quant aux patrons *contextuels*, ils extraient des informations à propos du contexte du mot courant comme les $n$ mots précédents ou encore l'étiquette du mot précédent.
La figure 3.3 montre une séquence d'étiquettes possible pour les mots de la phrase *Le chat sauvage a griffé Bob*. Chaque mot est associé à une étiquette morpho-syntaxique et à sa posi-

[91] Il est néanmoins possible de modéliser des traits qui ont été créés de façon manuelle.

[92] On peut noter que cet ensemble de patrons a été réutilisé par (Tsuruoka *et al.*, 2009) pour la phase d'étiquetage de leur analyseur syntaxique basée sur un modèle CRF. Nous sommes cependant dans l'incapacité d'indiquer les résultats obtenus car ils ne sont pas précisés dans leur article.

| Type de mot | Patron | |
|---|---|---|
| occ($w_i$)$\geq M$ | $w_i = X$ | & $t_i = T$ |
| occ($w_i$)$< M$ | prefixes de $w_i$ de taille $\leq 4$ | & $t_i = T$ |
| | suffixes de $w_i$ de taille $\leq 4$ | & $t_i = T$ |
| | $w_i$ contient un chiffre ? | & $t_i = T$ |
| | $w_i$ contient une lettre capitale ? | & $t_i = T$ |
| | $w_i$ contient un trait d'union ? | & $t_i = T$ |
| $\forall w_i$ | $t_{i-1} = X$ | & $t_i = T$ |
| | $t_{i-2}t_{i-1} = XY$ | & $t_i = T$ |
| | $w_{i-1} = X$ | & $t_i = T$ |
| | $w_{i-2} = X$ | & $t_i = T$ |
| | $w_{i+1} = X$ | & $t_i = T$ |
| | $w_{i+2} = X$ | & $t_i = T$ |

TAB. 3.2: Patrons de traits utilisés par l'étiqueteur décrit dans (Ratnaparkhi, 1996).

tion dans la phrase. En supposant que les mots *Bob* et *sauvage* soient respectivement rare et courant, la figure 3.8 montre les traits extraits pour chacun de ces deux mots après l'application des patrons.

| Mots | Le | chat | sauvage | a | griffé | Bob |
|---|---|---|---|---|---|---|
| Étiquettes | DET | NC | ADJ | V | VPP | NPP |
| Positions | 1 | 2 | 3 | 4 | 5 | 6 |

TAB. 3.3: Séquence d'étiquettes possible pour la phrase *Le chat sauvage a griffé Bob*. Chaque mot est également associé à sa position dans la phrase.

La recherche de la meilleure séquence peut être effectuée par un algorithme de programmation dynamique à base d'heuristiques appelé *beam search* (Lowerre, 1976). Cet algorithme construit itérativement un arbre ordonné des solutions partielles possibles (sous-séquences d'étiquettes) et a la particularité de ne conserver à chaque itération que les $N$ meilleures solutions en terme de probabilités partielles. Bien que le formalisme ME soit adapté aux tâches automatiques comme l'étiquetage morpho-syntaxique, les performances de cet étiqueteur sont décevantes (tableau 3.4) car légèrement mais non significativement supérieures à celles des étiqueteurs MMC vus dans la section précédente[93]. Les systèmes d'étiquetage que nous verrons dans la suite de cette section tendent à montrer que l'ensemble des patrons exploité par Ratnaparkhi n'est pas assez complexe alors qu'il s'agit de l'élément central des modèles discriminants. En effet, les traits extraits sont des informations relativement similaires à celles modélisées par les MMC.
(Toutanova & Manning, 2000) ont par la suite effectué une analyse d'erreurs poussée à partir des résultats obtenus par (Ratnaparkhi, 1996). Ils ont observé que les sources fréquentes d'erreurs étaient liées à trois phénomènes particuliers que sont les mots inconnus démarrant par

[93]On peut noter que les performances sur les mots inconnus sont mêmes moins bonnes que celles de l'étiqueteur TnT.

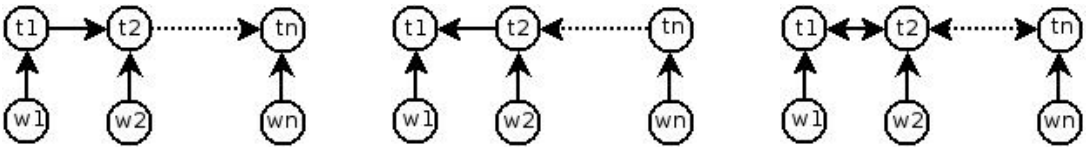
| | |
|---|---|
| $w_i$ = sauvage | & $t_i$ = ADJ |
| $w_{i-1}$ = chat | & $t_i$ = ADJ |
| $w_{i-2}$ = Le | & $t_i$ = ADJ |
| $w_{i+1}$ = a | & $t_i$ = ADJ |
| $w_{i+2}$ = griffé | & $t_i$ = ADJ |
| $t_{i-1}$ = NC | & $t_i$ = ADJ |
| $t_{i-2}$ = DET | & $t_i$ = ADJ |

(a) Traits extraits pour le mot courant $sauvage$.

| | |
|---|---|
| $w_{i-1}$ = griffé | & $t_i$ = NPP |
| $w_{i-2}$ = a | & $t_i$ = NPP |
| $t_{i-1}$ = VPP | & $t_i$ = NPP |
| $t_{i-2}$ = V | & $t_i$ = NPP |
| $prefixe(w_i)$ = B | & $t_i$ = NPP |
| $prefixe(w_i)$ = Bo | & $t_i$ = NPP |
| $prefixe(w_i)$ = Bob | & $t_i$ = NPP |
| $suffixe(w_i)$ = b | & $t_i$ = NPP |
| $suffixe(w_i)$ = ob | & $t_i$ = NPP |
| $suffixe(w_i)$ = Bob | & $t_i$ = NPP |
| $w_i$ contient une lettre capitale | & $t_i$ = NPP |

(b) Traits extraits pour le mot rare $Bob$.

FIG. 3.8: Application des patrons sur deux mots de la phrase de la figure 3.3.




(a) Modélisation ME *gauche vers droite*

(b) Modélisation ME *droite vers gauche*

(c) Modélisation RD *bidirectionnelle*

FIG. 3.9: Différences graphiques entre les modèles ME de premier ordre et les réseaux de dépendances bidirectionnelles.

une lettre capitale, les ambiguités entre étiquettes verbales et les ambiguités entre étiquettes de prépositions. Ils en ont déduit qu'il serait possible de minimiser ces erreurs par la création de nouveaux patrons spécifiques. Par exemple, pour le premier cas, un trait est activé quand un mot inconnu contient une lettre capitale et que ce mot n'est pas situé en début de phrase. Avec un protocole d'expérimentation identique à (Ratnaparkhi, 1996), ils obtiennent, avec leur nouvel ensemble de patrons, de faibles gains de +0.23 pour les performances globales et +1.35 pour les mots inconnus uniquement (tableau 3.4).

Généralement, les approches utilisées pour l'étiquetage morpho-syntaxique sont unidirectionnelles, c'est-à-dire que la prédiction d'une étiquette est conditionnée soit par les $n$ précédentes étiquettes ou par les $n$ étiquettes suivantes (Church, 1988). (Toutanova *et al.*, 2003) ont posé l'hypothèse que les processus d'étiquetage pourraient tirer avantage d'un conditionnement bidirectionnel en utilisant à la fois les contextes gauches et les contextes droits. Pour modéliser ce type de conditionnement, ils utilisent une approche basée sur les réseaux de dépendances [RD] (Heckerman *et al.*, 2001) qui sont une extension des modèles ME. La figure 3.9 illustre graphiquement les différences entre les modélisations discriminatives unidirectionnelles classiques ME (*gauche vers droite* ou *droite vers gauche*) et les réseaux de dépendances bidirectionnels.

La liste non exhaustive des patrons bidirectionnels utilisés par (Toutanova *et al.*, 2003) est montrée dans la figure 3.10a. L'application de ces patrons sur le mot $sauvage$ de l'analyse de la figure 3.3 génère l'ensemble de traits indiqué dans la figure 3.10b. Le patron $t_i, t_{i-1}, t_{i+1} = TXY$ permet par exemple de calculer des traits composés de l'étiquette courante et des éti-

| | |
|---|---|
| $t_{i-1}t_{i+1} = XY$ | & $t_i = T$ |
| $t_{i+1}t_{i+2} = XY$ | & $t_i = T$ |
| $t_{i-2}t_{i-1} = XY$ | & $t_i = T$ |
| $t_{i-2}t_{i-1}t_{i+1} = XYZ$ | & $t_i = T$ |

(a) Quelques patrons bidirectionnels.

| | |
|---|---|
| $t_{i-1}$ = NC, $t_{i+1}$ = V | & $t_i$ = ADJ |
| $t_{i+1}$ = V, $t_{i+2}$ = VPP | & $t_i$ = ADJ |
| $t_{i-2}$ = DET, $t_{i-1}$ = NC | & $t_i$ = ADJ |
| $t_{i-2}$ = DET, $t_{i-1}$ = NC, $t_{i+1}$ = V | & $t_i$ = ADJ |

(b) Traits correspondant extraits pour le mot $sauvage$.

FIG. 3.10: Liste non exhaustive des patrons bidirectionnels utilisée par (Toutanova *et al.*, 2003) et leur application sur le mot $sauvage$ de l'analyse lexicale de la figure 3.3.

quettes des deux mots situés à gauche et à droite du mot courant. L'algorithme de Viterbi (section 3.2) dans une version optimisée permet de rechercher efficacement la meilleure séquence dans un RD (Cowell *et al.*, 1999, 2007). Les résultats de leur étiqueteur, appelé StanfordTagger[94] (Toutanova *et al.*, 2003; Manning, 2011), sur le corpus d'évaluation du PTB semblent valider l'hypothèse initiale (tableau 3.4). En effet, par rapport à (Toutanova & Manning, 2000), les gains sont d'environ +0.4 pour l'ensemble des mots et +2.13 pour les mots inconnus uniquement. Ces résultats sont également légèrement meilleurs que ceux obtenus par (Collins, 2002) qui avait expérimenté en premier cette hypothèse en se basant sur un modèle perceptron. L'approche de bidirectionnalité des patrons est reprise par (Giménez & Màrquez, 2004) pour l'étiqueteur SVMTool[95] basé sur un modèle SVM. Ils intègrent également des patrons permettant d'extraire les classes d'ambiguité des mots. Une classe d'ambiguité est composée des différentes étiquettes possibles pour un mot. Par exemple, en supposant que le mot *sauvage* soit associé à deux étiquettes dans le corpus d'apprentissage, `ADJ` et `NC`, la classe du mot est `[NC,ADJ]`. Au vu du peu de modifications par rapport à l'étiqueteur de (Toutanova *et al.*, 2003), les résultats obtenus par SVMTool sont relativement similaires à ces derniers.

| Étiqueteur | Type de patrons | Performances globales | Mots inconnus |
|---|---|---|---|
| (Ratnaparkhi, 1996) | unidirectionnel | 96.63 | 85.56 |
| (Toutanova & Manning, 2000) | unidirectionnel | 96.86 | 86.91 |
| (Collins, 2002) | bidirectionnel | 97.11 | - |
| (Toutanova *et al.*, 2003) | **bidirectionnel** | **97.24** | **89.04** |
| (Giménez & Màrquez, 2004) | bidirectionnel | 97.16 | 89.01 |

TAB. 3.4: Performances des étiqueteurs discriminatifs sur le corpus d'évaluation du PTB.

### 3.3.2 Utilisation de lexiques externes au corpus

Plus récemment, (Denis & Sagot, 2009, 2010) ont constaté que la plupart des systèmes d'étiquetage effectuent l'analyse lexicale de la phrase en se basant uniquement sur le dictionnaire du corpus. Ils ont donc posé l'hypothèse que l'intégration de données issues de dictionnaires

[94] `http://nlp.stanford.edu/software/tagger.shtml`
[95] `http://www.lsi.upc.es/~nlp/SVMTool/`

externes permettraient d'améliorer les performances générales d'un étiqueteur et plus particulièrement l'étiquetage des mots inconnus. En effet, un mot inconnu du corpus d'apprentissage pourrait en revanche être hypothétiquement présent dans un lexique externe. Afin d'expérimenter cette hypothèse sur le français, l'étiqueteur proposé par (Denis & Sagot, 2009, 2010), appelé *MElt*[96], s'inspire des précédents travaux principalement réalisés et évalués sur l'anglais. Aussi, ils se basent sur un modèle ME et des patrons de base similaires à ceux de (Toutanova & Manning, 2000) qui peuvent être considérés comme relativement indépendants de la langue. Le dictionnaire externe utilisé, le Le*fff*, est un lexique morphologique et syntaxique à large couverture du français (Sagot, 2010) (section 4.3). Il contient 110477 formes lemmatisées (simples et composées) pour 536375 formes fléchies et traite plusieurs catégories grammaticales, à savoir les noms, les adjectifs, les prépositions, les verbes et les adverbes. Les étiquettes morpho-syntaxiques des entrées du Le*fff* ont été préalablement transformées en celles du corpus afin d'harmoniser les jeux d'étiquettes du dictionnaire externe et du corpus[97].
(Denis & Sagot, 2009, 2010) proposent deux approches d'intégration des données du Le*fff* dans le processus d'étiquetage. Une première méthode, initialement proposée par (Hajič, 2000), consiste à utiliser les étiquettes du dictionnaire externe comme des contraintes de filtrage. Pour chaque mot à étiqueter, l'ensemble des étiquettes possibles pour ce mot est composé de l'union des étiquettes du dictionnaire du corpus et de celles du dictionnaire externe associées à ce mot. La deuxième méthode exploite la puissance des modèles discriminants en incorporant au modèle ME les informations du Le*fff* sous forme de traits. Aussi, un nouveau patron lexical crée un trait pour chaque étiquette possible du mot présente dans le Le*fff*. Un autre patron lexical extrait la classe d'ambiguités du mot en concaténant ces étiquettes. Ces deux types d'information sont également utilisées de façon contextuelle pour les mots qui sont dans une fenêtre de taille 2 autour du mot courant ($w_{i-2}$,...,$w_{i+2}$). Les patrons additionnels sont formellement décrits dans le tableau 3.11a. Les fonctions *Lefff* et *Lefff_cls* renvoient respectivement les étiquettes du mot et la classe d'ambiguité de celui-ci. Le tableau 3.11b montre les différents traits obtenus par l'application de ces patrons sur le mot $sauvage$ de l'analyse lexicale de la figure 3.3. Dans le Le*fff*, le mot *sauvage* possède deux entrées, l'une adjectivale et l'autre nominale. Cela revient donc à créer trois traits lexicaux, un pour chaque étiquette possible du mot `ADJ` et `NC`, et un pour sa classe d'ambiguité `NC_ADJ`.
Les résultats obtenus par MElt, indiqués dans (Denis & Sagot, 2009, 2010), sur le français et plus particulièrement sur le corpus d'évaluation du FTB-UC sont indiqués dans le tableau 3.5. Les expériences notées MElt$_f$ et MElt$_c$ correspondent aux deux approches d'intégration du lexique externe Le*fff* dans le processus d'étiquetage : respectivement en tant que traits ou en tant que filtrage des étiquettes. À titre de base de comparaison, nous indiquons également les résultats de MElt utilisant uniquement les traits de base (expérience notée simplement MElt). Les meilleurs résultats sont obtenus par MElt$_f$ avec 97.70 (+0.7) pour l'ensemble des mots et 90.01 (+3.91) pour les mots inconnus. Plusieurs raisons sont avancées pour expliquer de tels gains. D'une part, le nombre moindre d'erreurs sur les mots inconnus est une conséquence directe de l'utilisation d'un lexique morpho-syntaxique. Moins d'erreurs sur les mots inconnus entraîne une réduction de l'effet de dispersion des données ce qui améliore logiquement l'étiquetage des mots aux alentours. Cette remarque est valable également pour MElt$_c$ car le

[96] `https://gforge.inria.fr/frs/download.php/27240/melt-0.6.tar.gz`
[97] On peut rappeler que le corpus FTB-UC a un jeu de 28 étiquettes.

| **Patrons lexicaux** | |
|---|---|
| Le*fff*$(w_i) = X$ | & $t_i = T$ |
| Le*fff*_cls$(w_i) = X_0...X_n$ | & $t_i = T$ |
| **Patrons contextuels** | |
| Le*fff*$(w_{i+j}) = X$, $j \in \{-2, -1, 1, 2\}$ | & $t_i = T$ |
| Le*fff*_cls$(w_{i+j}) = X_0...X_n$, $j \in \{-2, -1, 1, 2\}$ | & $t_i = T$ |

(a) Patrons lexicaux et contextuels basés sur le lexique externe Le*fff*, utilisés par l'étiqueteur MElt.

| | |
|---|---|
| Le*fff*(sauvage) = NC | & $t_i$ = ADJ |
| Le*fff*(sauvage) = ADJ | & $t_i$ = ADJ |
| Le*fff*_cls(sauvage) = NC_ADJ | & $t_i$ = ADJ |
| Le*fff*(chat) = NC | & $t_i$ = ADJ |
| Le*fff*(Le) = DET | & $t_i$ = ADJ |
| Le*fff*(Le) = CLO | & $t_i$ = ADJ |
| Le*fff*_cls(Le) = DET_CLO | & $t_i$ = ADJ |

(b) Liste non exhaustive des traits (lexicaux et contexuels) obtenus à partir du mot $sauvage$.

FIG. 3.11: Liste des patrons (basés sur le dictionnaire externe Le*fff*) utilisés par (Denis & Sagot, 2009) et leur application sur le mot $sauvage$ de l'analyse lexicale de la figure 3.3.

gain sur les mots inconnus est de +2.90. D'autre part, les traits bidirectionnels tirent parti des informations disponibles pour le contexte droit des mots (étiquettes, classes d'ambiguité,...) ce qui rend le modèle beaucoup plus robuste. En ce qui concerne l'expérience MElt$_c$, les faibles résultats proviennent du fait que l'approche à base de contraintes sur les étiquettes est trop brutale pour être gérée efficacement par le modèle, contrairement aux traits.

| Étiqueteur | Performances globales | Mots inconnus |
|---|---|---|
| MElt | 97.00 | 86.10 |
| MElt$_c$ | 97.10 | 89.00 |
| MElt$_f$ | **97.70** | **90.01** |

TAB. 3.5: Performances de MElt sur le corpus d'évaluation du FTB-UC selon les différentes approches d'intégration du lexique externe Le*fff*.

Dans le chapitre 5, nous proposons une approche à base de lexiques similaire à MElt. Cependant, contrairement à ce dernier, des patrons de traits seront spécifiques aux mots composés. Nous verrons que les informations morpho-syntaxiques disponibles dans les lexiques sur ce type de mots permettent d'améliorer significativement un étiqueteur discriminatif. De plus, grâce à ces informations, nous serons également capable de proposer une segmentation performante des phrases en mots simples et composés.

## 3.4 Éxpériences d'étiquetage réalisées sur le corpus du français FTB-UC

Dans cette section, nous indiquons les résultats de plusieurs expériences réalisées sur le FTB-UC. La première expérience est une évaluation comparative des étiqueteurs présentés précédemment. Nous tenterons de répondre notamment à la question de savoir si, comme pour l'anglais, les étiqueteurs basés sur un modèle discriminant sont meilleurs que ceux utilisant un modèle génératif. Une deuxième expérience évalue l'approche développée dans (Denis & Sagot, 2009, 2010), à savoir qu'un lexique externe permettrait d'améliorer significativement les performances générales d'un étiqueteur. On peut noter que l'ensemble de ces expériences ont été réalisées sur le FTB-UC avec des textes considérés comme déjà segmentés en mots simples et composés. Il s'agit pour l'instant de se conformer au protocole des travaux actuels afin de montrer les performances générales des systèmes d'étiquetage.

### 3.4.1 Évaluation comparative des étiqueteurs

L'évaluation comparative a porté sur l'entraînement et l'évaluation de six étiqueteurs morpho-syntaxiques sur le corpus du français FTB-UC. Trois sont basés sur un modèle génératif MMC de second ordre : TnT (version 2.2), TreeTagger (version 3.2) et un simple modèle trigramme dont les résultats sont donnés à titre de base de comparaison[98]. Les trois autres étiqueteurs sont basés sur un modèle discriminant : SVMTool (version 1.3.2), StanfordTagger (version 3.1.1) et MElt (version 0.6). Pour l'ensemble des étiqueteurs, nous utilisons les paramètres fournis par défaut dans la distribution. De plus, dans cette première expérience, seul le lexique du corpus d'apprentissage est utilisé[99]. Dans le tableau 3.6, nous indiquons, pour chaque étiqueteur, les performances globales ainsi que les performances pour les mots inconnus, en utilisant la méthode de validation croisée ($p$=10) afin d'établir la significativité des résultats.
Bien que les performances du modèle trigramme de base soient étonnamment élevées, elles sont surpassées par celles d'algorithmes plus évolués comme TreeTagger ou TnT. De plus, contrairement aux résultats observés sur le PTB, TnT est significativement meilleur que TreeTagger tant au niveau global qu'au niveau des mots inconnus. Quant aux étiqueteurs discriminatifs, ils sont significativement meilleurs que les étiqueteurs génératifs. Comme on a pu le remarquer également sur l'anglais, leurs performances sont relativement similaires car ils sont tous basés sur un ensemble de patrons de traits similaire.

### 3.4.2 Impact de ressources externes

Nous avons étendu l'expérience précédente par la prise en compte d'un lexique externe, le Le*fff*. En cela, nous avons suivi le protocole d'expérimentation décrit dans (Denis & Sagot, 2009).

[98] Nous avons utilisé l'étiqueteur implémenté dans l'outil *nltk* en langage Python, disponible à l'adresse `http://www.nltk.org/`. Un simple algorithme à base de suffixes permet de gérer l'étiquetage mots inconnus.

[99] En d'autres termes, MElt ne fait pas donc pas usage des patrons de traits liés au lexique Le*fff*.

| Étiqueteur | Performances globales | Mots inconnus |
|---|---|---|
| **Modèle génératif** | | |
| Trigramme | 95.30 | 75.68 |
| TreeTagger | 96.06 | 75.87 |
| TnT | 96.78 | 85.89 |
| **Modèle discriminant** | | |
| SVMTool | 97.10 | 89.85 |
| StanfordTagger | 97.39 | 90.01 |
| MElt | 97.35 | 89.70 |

TAB. 3.6: Performances des étiqueteurs sur le FTB-UC sans l'utilisation de ressources lexicales externes (validation croisée).

Pour l'étiqueteur génératif TnT, le lexique a été simplement combiné avec celui du corpus d'apprentissage. Pour l'étiqueteur discriminatif MElt, les patrons de traits liés au lexique externe ont été activés. Les résultats sont indiqués dans le tableau 3.7. On peut voir que l'étiquetage des mots inconnus par TnT est grandement amélioré (+1.41) car il profite directement de l'intégration du lexique. Pour MElt, on constate également une forte hausse des performances. Cela montre que l'utilisation de patrons spécifiques au lexique permet d'améliorer significativement l'étiquetage des mots inconnus et par la même occasion de réduire l'effet de dispersion des données.

| Étiqueteur | Performances globales | Mots inconnus |
|---|---|---|
| **Modèle génératif** | | |
| TnT | 96.93 | 87.30 |
| **Modèle discriminant** | | |
| MElt | 97.75 | 91.30 |

TAB. 3.7: Performances des étiqueteurs sur le FTB-UC en faisant appel à des ressources lexicales externes (validation croisée).

Les expériences précédentes ont été réalisées avec un corpus de taille relativement importante. Or, certaines langues ne disposent pas toujours d'un corpus annoté de taille conséquente. Aussi, il est intéressant d'observer l'impact des ressources lexicales externes sur les performances d'étiquetage selon la quantité de données fournie à l'apprentissage du modèle probabiliste. Pour ce faire, nous avons évalué ces mêmes étiqueteurs, avec ou sans ressource externe, sur le corpus d'évaluation du FTB-UC en faisant varier la taille du corpus d'apprentissage[100] par palier de 10%. Les résultats globaux (tous les mots) sont illustrés graphiquement dans la figure 3.12. Les résultats pour les mots inconnus uniquement sont montrés dans la figure 3.13. Les courbes rouges et bleues correspondent aux scores des étiqueteurs entraînés respectivement avec ou sans l'aide du Le*fff*. Tout d'abord, on peut constater qu'en se basant sur seulement 10% du corpus d'apprentissage et en utilisant le Le*fff*, le gain pour l'ensemble des mots est d'environ +0.6 pour TnT et MElt. Globalement, les courbes rouges et bleues pour les deux étiqueteurs

[100]Nous avons inclus le corpus de développement au corpus d'apprentissage.

sont relativement parallèles, avec un avantage significatif à chaque palier pour la rouge (Le*fff*).

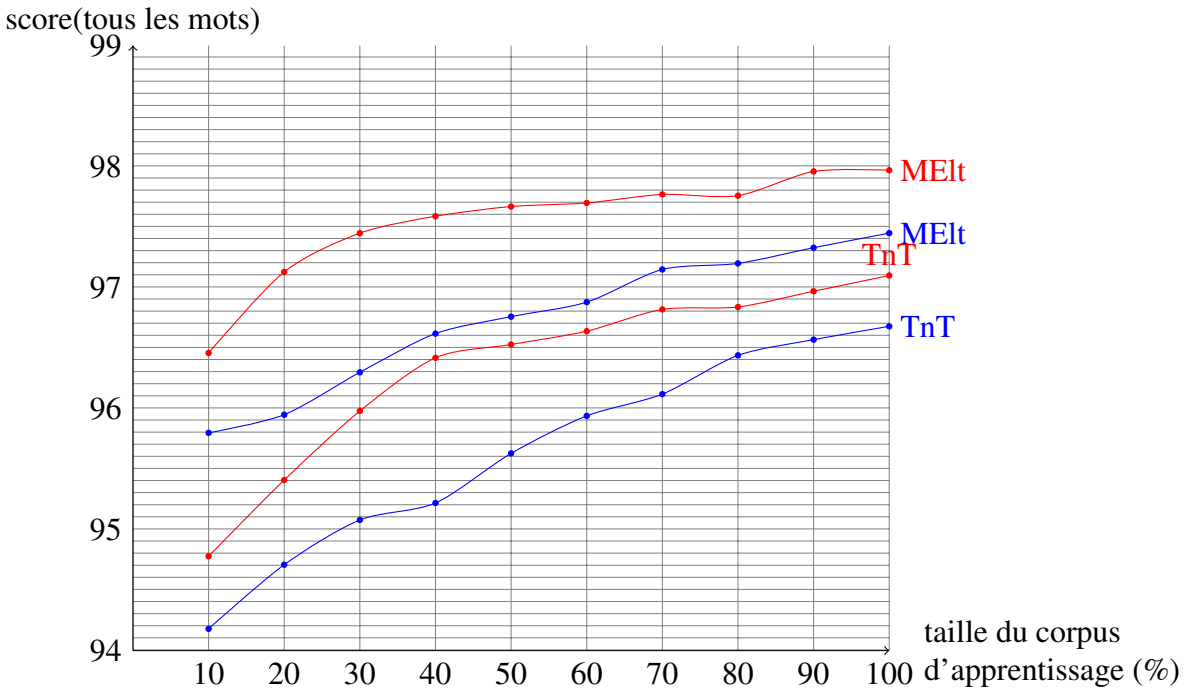



FIG. 3.12: Courbes d'apprentissage des étiqueteurs TnT et MElt avec ou sans l'aide du Le*fff* (courbes de couleur rouge ou bleue). Le score est indiqué pour tous les mots.

## 3.5 Conclusion

Au cours de ce chapitre, nous avons décrit les algorithmes des principaux systèmes d'étiquetage. La différence majeure entre les systèmes se situe au niveau de la modélisation du problème. D'un côté, les modèle génératifs (markoviens) posent des hypothèses d'indépendances fortes entre les variables d'entrées. Le but est de simplifier mathématiquement le problème en décomposant le calcul de la probabilité d'une séquence en sous-produits de probabilités d'émissions et de transitions. Des étiqueteurs basés sur ce type de modélisation, comme TnT, obtiennent des scores inférieurs à 97% dans le cadre du français. D'autre part, il existe à présent de nombreux systèmes basés sur une modélisation discriminative. Ces modèles ont la particularité de ne pas poser d'hypothèses d'indépendances, ce qui leur permet de prendre en compte de nombreuses informations (traits). L'élément central de ce type de modèle est l'ensemble de patrons permettant de générer les traits. Les patrons utilisés par une majorité de systèmes sont dits indépendants de la langue car ils extraient des informations uniquement à partir du corpus annoté ou des textes à étiqueter. Le but est de permettre aux systèmes de s'adapter efficacement à diverses langues, et ce, sans hypothèses sur la langue traitée. Les scores d'étiqueteurs comme SVMTool ou StanfordTagger sont supérieurs à 97%.

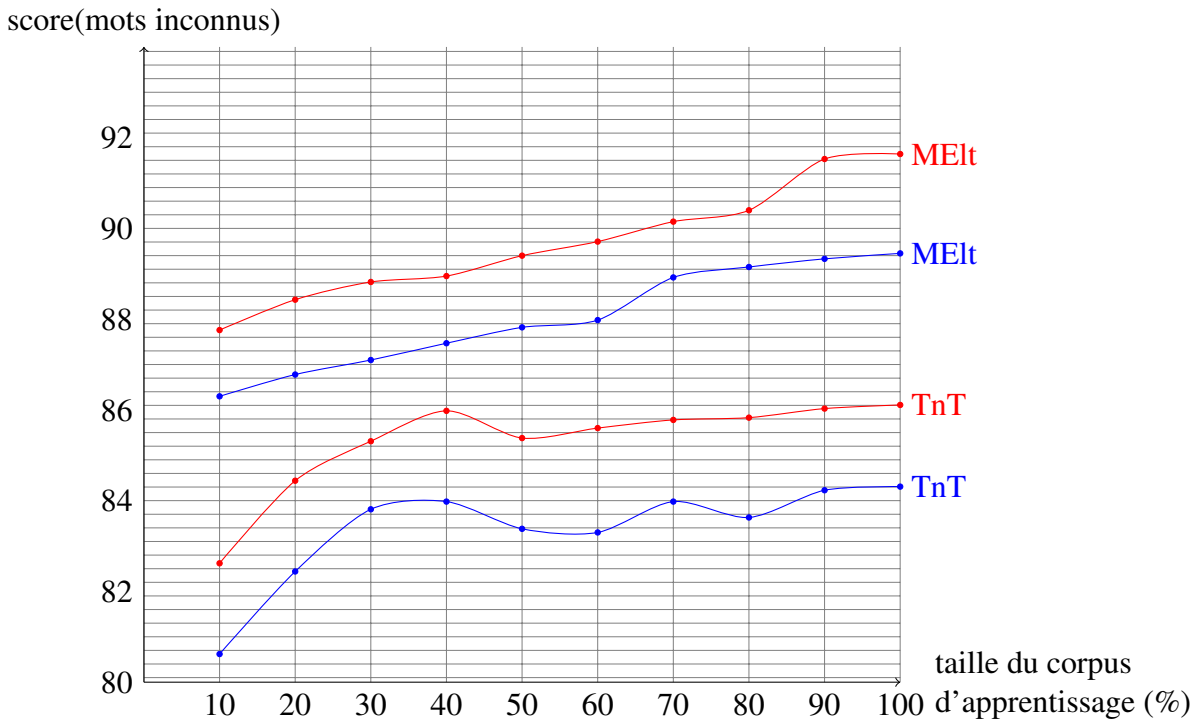



FIG. 3.13: Courbes d'apprentissage des étiqueteurs TnT et MElt avec ou sans l'aide du Le*fff* (courbes de couleur rouge ou bleue). Le score est indiqué pour uniquement pour les mots inconnus.

Plus récemment, (Denis & Sagot, 2009, 2010) ont montré l'intérêt d'intégrer des ressources lexicales externes au processus d'étiquetage. Selon le type de modélisation utilisé, ils ont proposé plusieurs solutions d'intégration des données externes. Pour les modèles génératifs, le dictionnaire du corpus est complété par l'ajout d'entrées provenant des lexiques externes. Pour les modèles discriminants, deux solutions sont possibles : (a) filtrage des étiquettes incompatibles avec la ressource. (b) création de nouveaux patrons de traits (par exemple, les classes d'ambiguités). Ces trois méthodes ont prouvé leur efficacité sur le FTB-UC, avec des gains absolus variant de +2 à +3 pour les mots inconnus, et des gains moins importants pour la totalité des mots.
On peut néanmoins noter que la plupart des évaluations ont été réalisées avec une segmentation parfaite du texte, car identique à celle du corpus évalué. Or, dans les cas réels d'application, la segmentation d'un texte est très rarement disponible et les segmenteurs automatiques actuels sont loin de proposer une segmentation de bonne qualité, et ce, à cause de la présence de nombreuses unités multi-mots (mots composés, entités nommées,...). Dans le cadre de notre travail, nous nous sommes donc penché sur l'identification automatique des unités multi-mots, et leur impact sur l'étiquetage morpho-syntaxique. Nous proposons, dans la partie 5, d'intégrer les deux tâches de segmentation et d'étiquetage dans un seul modèle CRF couplé à des ressources lexicales externes sur les mots composés. Cet étiqueteur-segmenteur permet d'identifier et d'étiqueter les mots composés d'un texte grâce à des traits consacrés aux mots composés.

# 4

# Ressources lexicales et syntaxiques

## 4.1 Introduction

Un lexique syntaxique est une ressource qui contient l'information sur le potentiel combinatoire d'un prédicat, mais aussi sur le type de ses arguments. Par exemple, le verbe *dormir* régit un seul argument, le sujet, et ce dernier peut être un groupe nominal. De plus, ces différentes informations varient selon la langue. Il est donc nécessaire de développer au moins un lexique pour chaque langue devant être traitée par les applications du TAL. Ces lexiques sont traditionnellement développés par des experts humains (le Lexique-Grammaire (Gross, 1975), par exemple), ce qui en fait des ressources de bonne qualité et à large couverture, mais qui sont, en général, difficilement adaptables aux traitements automatiques. Il existe actuellement de plus en plus de lexiques développés de manière automatique, ou semi-automatique, par des algorithmes statistiques appliqués sur des textes ou des corpus annotés (Carroll & Fang, 2004; Bourigault & Frérot, 2006; Kupsc, 2007).
Dans le cadre d'applications comme l'analyse syntaxique, il a été prouvé par diverses expériences que les informations lexicales et syntaxiques contenues dans les lexiques pouvaient potentiellement améliorer la qualité des analyses produites. (Briscoe & Carroll, 1993) ont par exemple estimé qu'environ 50% des erreurs des analyseurs reposent sur des informations insuffisantes concernant la structure argumentale. Quant à (Carroll & Fang, 2004), ils ont montré qu'un analyseur HPSG pouvait être significativement amélioré grâce à des informations de sous-catégorisation présentes dans un lexique de verbes acquis automatiquement. Plus récemment, (Tolone, 2011a) a intégré avec succès les données d'un lexique du français, le Lexique-Grammaire, dans un analyseur symbolique FRMG (Thomasset & de La Clergerie, 2005). On peut noter que les lexiques jouent un rôle essentiel dans d'autres domaines comme l'extraction d'information (Surdeanu *et al.*, 2003), la traduction automatique (hye Han *et al.*, 2000), ou encore la génération automatique de textes (Danlos, 1985).
Dans ce chapitre, nous présentons les différents lexiques syntaxiques du français que nous utilisons pour nos expériences, à savoir *Dicovalence* (section 4.2), le *Lefff* (section 4.3), le *Lexique-Grammaire* (section 4.4) et *LexSchem* (section 4.5). Pour chacun de ces lexiques, nous décrivons le principe général, l'architecture ainsi que les différentes informations codées pour

chaque entrée.

## 4.2 Dicovalence

Le dictionnaire Dicovalence[101] est un lexique syntaxique qui répertorie les cadres de valence des verbes simples du français (Mertens, 2010). Ce lexique fait suite au développement du dictionnaire Proton (Eynde & Piet, 2003). Il contient 5011 formes verbales lemmatisées pour 8254 entrées. Un même lemme peut présenter plusieurs emplois d'où plusieurs entrées distinctes avec leur propre cadre valenciel. Par exemple, on peut distinguer deux emplois distincts pour le verbe *écumer* :

(1) *les bières anglaises n'écument pas.*

(2) *le garçon écumait les bières.*

Pour chaque entrée, le lexique indique le cadre valenciel (cadre de sous-catégorisation) qui énumère l'ensemble des compléments valenciels (arguments) d'un verbe dans un emploi donné. Pour chaque argument, il est précisé le caractère obligatoire ou facultatif, la fonction syntaxique associée, les réalisations syntagmatiques possibles ainsi que des traits sémantiques (humain, abstrait,...). Dicovalence se base sur un ensemble de quinze fonctions syntaxiques, parmi lesquelles :

- `subj`, sujet (*Max* dans *Max mange une pomme*).
- `obj`, objet direct (*une pomme* dans *Max mange une pomme*).
- `objà`, objet indirect introduit canoniquement par la préposition *à* (*à Sophie* dans *Max donne une pomme à Sophie*).
- `objde`, objet indirect introduit par la préposition *de* (*de son courage* dans *je me suis émerveillé de son courage*).
- `objp<x>`, objet indirect (*avec son partenaire* dans *il a composé avec son partenaire*).
- `loc<x>`, argument locatif (*dans leur maison* dans *les nouveaux voisins ont emménagé dans leur maison*).
- `deloc<x>`, argument délocatif introduit par la préposition *de* (*de son bureau* dans *ils ont enlevé la photo de son bureau*).
- `attr_obj<x>`, attribut de l'objet (*comme chauffeur* dans *je peux vous employer comme chauffeur*).
- `attr_subj<x>`, attribut du sujet (*comme agent de police* dans *je me suis engagé comme agent de police*).
- `quant<x>`, complément de quantification (*pour cinq euros* dans *je vous laisse ce livre pour cinq euros*).

Les fonctions syntaxiques contenant le symbole `<x>` sont introduites par une préposition remplaçant *x* dans les entrées concernées. Les réalisations syntagmatiques possibles d'une fonction syntaxique sont de trois types :

- clitique : *y, en, celui-ci,...*
- syntagme direct : pronominal `pron`, nominal `n`, proposition infinitive `inf`, proposition complétive `compl`, proposition interrogative `qcompl`.

[101] Disponible à l'adresse `http://bach.arts.kuleuven.be/dicovalence/`

• syntagme prépositionnel : syntagme précédé d'une préposition *à_inf, à_ce_que_compl,....*
Si la réalisation d'une fonction est facultative, un symbole `?` est placé devant la fonction.
La particularité de ce lexique est que les cadres de sous-catégorisation sont également représentés selon les principes de l'approche pronominale en syntaxe (Eynde & Blanche-Benveniste, 1978). Pour chaque place de valence, appelée aussi paradigme, le dictionnaire précise le paradigme de pronoms qui y est associé et qui couvre les lexicalisations possibles. La délimitation d'un cadre de valence, appélé formulation, repose non seulement sur la configuration (nombre, nature, caractère facultatif, composition) de ces paradigmes pronominaux, mais aussi sur les autres propriétés de construction associées à cette configuration, comme les reformulations passives.

### 4.2.1 Exemple tiré du lexique

La Figure 4.1 montre un extrait du lexique pour une entrée du verbe *chérir*. Dans cet exemple, le champ `FRAME` fournit le cadre de sous-catégorisation et nous indique que l'entrée possède deux arguments obligatoires, un sujet (fonction syntaxique `subj`) et un objet direct (fonction syntaxique `obj`). Les réalisations syntagmatiques de ces deux fonctions sont soit un syntagme nominal humain soit un pronom humain. Le champ `VAL` décrit le cadre de sous-catégorisation selon l'approche pronominale en syntaxe. Les deux arguments sont associés aux deux ensembles de pronoms acceptés, `P0` et `P1`.

VAL chérir : P0 P1
VTYPE predicator simple
VERB CHERIR/chérir
NUM 15170
EG elle chérit son bébé
TR_DU (innig) liefhebben, (teer) beminnen, innig gehecht zijn (aan), veel houden (van)
TR_EN love
FRAME subj :pron|n :[hum], obj :pron|n :[hum]
P0 qui, je, nous, elle, il, ils, on, celui-ci, ceux-ci
P1 que, qui, te, vous, la, le, les, se réfl., se réc., en Q, ça, celui-ci, ceux-ci, l'un l'autre
RP passif être
AUX avoir

FIG. 4.1: Extrait de Dicovalence pour une entrée du verbe *chérir*.

## 4.3 Le*fff*

Le Le*fff*[102] (Lexique des formes fléchies du français) est un lexique morphologique et syntaxique à large couverture du français (Sagot, 2010). Il contient 110477 formes lemmatisées

[102] `http://atoll.inria.fr/~sagot/lefff.html`

(simples et composées) pour 536375 formes fléchies. Ce lexique traite plusieurs catégories grammaticales, à savoir les noms, les adjectifs, les prépositions, les verbes et les adverbes. Le lexique repose sur le formalisme Alexina[103] (Architecture pour les LEXiques INformatiques et leur Acquisition) qui est une architecture d'acquisition et de modélisation lexicale permettant de décrire les niveaux morphologiques et syntaxiques d'éléments lexicaux. Plusieurs lexiques reposant sur Alexina sont en cours de construction, notamment pour l'espagnol (le Le*ff*e) ou encore pour le polonais. On peut noter que les informations lexicales disponibles dans le Le*ffff* proviennent de plusieurs sources, dont Dicovalence et le Lexique-Grammaire (section 4.4).
Dans le Le*ffff*, les éléments lexicaux sont des léxèmes, c'est à dire des lemmes possédant des propriétés morphologiques et syntaxiques. Chaque entrée est associée à des informations morphologiques, un lemme, une catégorie grammaticale et une classe de flexion. À partir du lemme, cette classe permet de générer l'ensemble des formes fléchies possibles. Quant aux informations syntaxiques, elles sont regroupées dans un cadre de sous-catégorisation, dont la représentation est presque identique à celle de Dicovalence. Ainsi, il nous renseigne sur les possibles arguments syntaxiques d'une entrée, chacun étant lié à une fonction syntaxique et ses réalisations syntagmatiques possibles. Le Le*ffff* se base sur un ensemble de neuf fonctions syntaxiques, dont les critères définitoires sont proches de ceux de Dicovalence (Danlos & Sagot, 2007) : `Suj` (sujet), `Obj` (objet direct), `Objà` (objet indirect introduit par la *à*), `Objde` (objet indirect introduit par *de*), `Loc` (argument locatif), `Dloc` (argument délocatif), `Att` (attribut sujet, objet ou à-objet), `Obl` et `Obl2` (autres arguments obliques). De même que Dicovalence, les réalisations syntagmatiques possibles d'une fonction syntaxique sont de trois types :

- clitique : nominal `cln`, accusatif `cla`, datif `cld`, réfléchi `seréfl`, réciproque `seréfl`, `y`, `en`.
- syntagme direct : nominal `sn`, adjectival `sa`, proposition infinitive `sinf`, proposition complétive `scompl`, proposition interrogative `qcompl`.
- syntagme prépositionnel, syntagme précédé d'une préposition `pour-sinf`, `de-scompl`, `à-sn`.

Si la réalisation d'une fonction est facultative, la liste des réalisations est mise entre parenthèses.
Dans la version actuelle du Le*ffff*[104], seuls les verbes, les adjectifs et les prépositions ont des cadres de sous-catégorisation exploitables. Dans le tableau 4.2, nous avons indiqué le nombre d'entrées fléchies distinctes ainsi que le nombre de lemmes correspondant pour les différentes catégories grammaticales présentes dans le Le*ffff*.

| Catégorie grammaticale | Entrées distinctes | Lemmes distincts |
|---|---|---|
| verbes | 337960 | 6873 |
| noms communs | 78417 | 37525 |
| noms propres | 52575 | 52190 |
| adjectifs | 58699 | 16558 |
| adverbes | 4336 | 3716 |
| prépositions | 651 | 226 |

FIG. 4.2: Statistiques tirées du Le*ffff* pour différentes catégories grammaticales.

[103]Pour le projet Alexina, voir `https://gforge.inria.fr/projects/alexina`
[104]Version 3.0.3

### 4.3.1 Architecture sur deux niveaux

La modélisation d'un lexique Alexina est faite sur deux niveaux. Un lexique intensionnel décrit pour chaque entrée lemmatisée (une entrée par sens) un cadre de sous-catégorisation canonique et liste les redistributions (transformations élémentaires) possibles à partir de ce cadre. Un lexique extensionnel est construit automatiquement par la compilation du lexique intensionnel. Pour chaque forme fléchie du lemme d'une entrée intensionnelle et pour chaque redistribution de cette entrée, une ou plusieurs entrées extensionnelles sont construites. Les redistributions sont des transformations élémentaires devant être appliquées sur les cadres de sous-catégorisation initiaux afin de produire les cadres finaux. Concrètement, une transformation peut assigner un argument syntaxique à une autre fonction syntaxique que celle initiale. Elle peut également changer des propriétés comme la liste des réalisations. Le *Lefff* se base sur un ensemble de dix redistributions possibles pour les verbes, parmi lesquelles :

- `%actif`, redistribution de base, sans effet sur le cadre initial.
- `%passif`, passivation standard avec *par* (*Max mange une pomme* devient *Une pomme est mangée par Max*).
- `%passif_de`, passivation avec *de* (*Sophie aime Max* devient *Max est aimé de Sophie*).
- `%impersonnel`, constructions actives impersonnelles à sujet inversé.
- `%passif_impersonnel`, constructions passives impersonnelles à sujet inversé.

Les adjectifs ont deux redistributions possibles supplémentaires :

- `%adj_impersonnel`, un adjectif est la tête lexicale d'une construction impersonnelle (*il est tentant de manger ce gâteau*).
- `%adj_personnel`, pour les autres cas des constructions personnelles.

Les autres catégories grammaticales n'ont pas de redistributions possibles (marqué `%default` dans le lexique). Les cadres finaux sont donc identiques aux cadres initiaux. Parfois, des incompatibilités sont détectées lors de la génération des entrées. Chaque transformation élémentaire est donc associée à une balise de contrôle qui vérifie le résultat de cette transformation et le corrige si nécessaire. Par exemple, la balise `Obj-cla` supprime la réalisation `cla` (clitique accusatif) de la liste des réalisations d'une fonction syntaxique en position d'objet direct.

### 4.3.2 Exemple tiré du lexique

La figure 4.3 montre une entrée du lexique intensionnel pour le lemme *manger*. Cette entrée verbale (`v`) transitive possède deux arguments réalisés canoniquement par les fonctions syntaxiques `Suj` et `Obj`. Les réalisations possibles de la fonction sujet sont `cln, scompl, sinf, sn` et celles, facultatives[105], de la fonction objet sont `cla, sn`. Elle accepte deux redistributions, `%actif` et `%passif`. De plus, sa classe de flexion est `v-er :std` qui est la classe des verbes du premier groupe (finissant par `-er`).

*manger v-er :std Lemma ;v ;<Suj :cln|scompl|sinf|sn,Obj :(cla|sn)> ;%actif,%passif*

FIG. 4.3: Entrée intensionnelle du Le*fff* pour le lemme *manger*.

[105]La présence de parenthèses, comme dans Dicovalence, indique que les réalisations sont facultatives.

La figure 4.4 montre une entrée du lexique extensionnel pour la forme fléchie *mangées* de l'entrée du lemme *manger*. Cette forme fléchie est la forme au participe passé du verbe et résulte de l'application de la redistribution au passif sur l'entrée de ce verbe. On obtient un cadre de sous-catégorisation final dont l'objet direct initial (`Obj`) a été transformé en un sujet passif (`Suj`). Un complément d'agent (`Obl2`), réalisé par un syntagme nominal précédé d'une préposition `par-sn`, a été ajouté.

*mangées v [pred="manger_1";<Suj :cln|sn,Obl2 :(par-sn)>,@passive,@pers,@Kfp] %passif*

FIG. 4.4: Entrée extensionnelle du L*efff* pour le verbe *manger*.

## 4.4 Lexique-Grammaire

Les tables du Lexique-Grammaire[106] constituent aujourd'hui une des principales sources d'informations lexicales et syntaxiques pour le français. Ce lexique traite non seulement des verbes mais aussi des noms prédicatifs, des adverbes ainsi que des phrases figées. Leur développement a été entamé dès les années 1970 par Maurice Gross et son équipe (Gross, 1975, 1994), d'abord au LADL puis au LIGM. C'est un lexique syntaxique à large couverture qui se présente sous la forme de tables. Chaque table représente une classe syntaxique contenant des éléments lexicaux partageant certaines propriétés syntaxiques, qui relèvent généralement de la sous-catégorisation. Un élément lexical est associé à une unique forme lemmatisée et n'appartient qu'à une classe. De plus, si une forme lemmatisée a plusieurs sens distincts, il possède autant d'entrées lexicales puisque chaque sens n'accepte pas le même ensemble de propriétés. Par exemple, pour le cas du verbe *démolir*, on peut distinguer au moins deux sens distincts :

(1) *Max démolit un bâtiment à coups de pioche.*

(2) *Max démolit l'argumentation de Sophie.*

Le premier sens représente la destruction d'un objet concret alors que le deuxième sens se rapproche plus de la destruction d'un objet abstrait. Ces deux sens correspondent à deux entrées lexicales distinctes dans les classes verbales *32D* et *32R1*.
Chaque ligne d'une table correspond à un élément lexical et chaque colonne correspond à une propriété (i.e. les constructions syntaxiques, la distribution des arguments,...). Une cellule de la table encode le fait qu'un élément lexical accepte ou non une propriété. La figure 4.5 montre un extrait de la table *12* des verbes. Dans cette table, nous pouvons observer que le verbe *chérir* accepte un sujet *humain*, indiqué par un "+" dans la cellule pointée par la propriété *N0 = : Nhum* (`N0` étant le sujet du verbe et `Nhum`, un nom humain). En revanche, ce verbe ne peut être intransitif, indiqué par un "−" dans la cellule pointée par la propriété *N0 V* (construction sans compléments). Dans le tableau 4.6, nous avons indiqué le nombre d'entrées lemmatisées ainsi que le nombre de lemmes distincts pour les différentes catégories grammaticales présentes dans le LG.

[106] `http://infolingu.univ-mlv.fr/`

| N0 = : Nhum | N0 = : le fait Qu P | N0 = : Vi-inf W | <ENT>Ppv | Ppv = : Neg | <ENT>V | Neg | N0 V | N1 = : Qu Pind | Qu P = V0-inf W | N1 = : Qu P = Aux V0-inf W | N1 = Ppv | [passif par] | [passif de] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + | - | - | <E> | - | chérir | - | - | - | - | - | + | + | + |
| + | - | - | <E> | - | comprendre | - | + | - | - | - | + | + | + |
| + | - | - | <E> | - | critiquer | - | + | - | - | - | + | + | + |
| + | - | - | <E> | - | débiner | - | - | - | - | - | + | + | + |

FIG. 4.5: Extrait de la table *12* des verbes du LG.

| Catégorie grammaticale | Classes | Lemmes distincts | Entrées |
|---|---|---|---|
| verbes | 61 | 5923 | 13872 |
| noms | 79 | 10112 | 13926 |
| adverbes | 33 | 9507 | 10529 |
| phrases figées | 70 | 39627 | 39627 |

FIG. 4.6: Statistiques tirées du LG pour différentes catégories grammaticales.

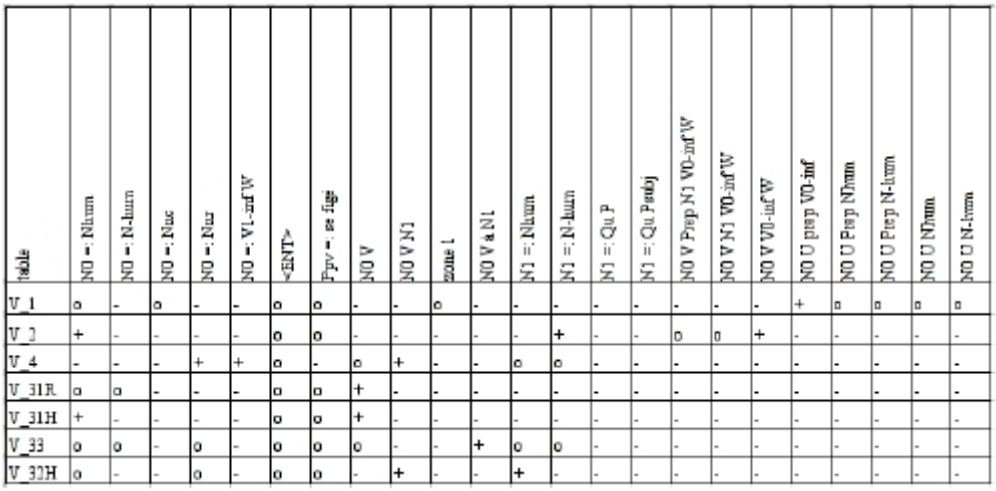
### 4.4.1 Table des classes

Les tables du LG ont récemment été rendues plus cohérentes et plus explicites dans le cadre du travail de (Tolone, 2011b), notamment au moyen d'une table des classes (Paumier *et al.*, 2003). Cette table particulière code les propriétés qui sont communes aux entrées d'une même classe. Ces propriétés n'étaient présentes initialement que dans la littérature. Chaque ligne de cette table correspond à une classe du LG et chaque colonne correspond à une propriété (i.e. les constructions syntaxiques, la distribution des arguments,...). Une cellule code le fait qu'une classe accepte ou non une propriété. On peut distinguer deux cas possibles :

- ∘ la valeur de la propriété varie selon les entrées de la classe (symbole ∘ dans la cellule).
- ∘ la valeur de la propriété est identique pour l'ensemble des entrées de la classe. Une valeur + signifie que la propriété est acceptée par l'ensemble des entrées, et un − signifie au contraire que cette propriété n'est valable pour aucune de ces entrées.

Par exemple, la table des classes de verbes est composée des 61 classes et de 490 propriétés. La figure 4.7 montre un extrait de cette table[107]. Dans cette table, nous pouvons observer que, pour la classe *V_31R*, la construction *N0 V* est vraie (+), alors que la construction *N0 V N1* n'est pas acceptée (−). La propriété *N0 = : Nhum* ayant pour valeur ∘, elle dépend donc des différentes entrées lexicales.

[107]Cet exemple est directement tiré de (Constant & Tolone, 2010).



| Table | N0 = Nhum | N0 = N-hum | N0 = Nhuc | N0 = Nhar | N0 = V1-inf W | <ENT> | Ppv = se figé | N0 V | N0 V N1 | zone 1 | N0 V à N1 | N1 = Nhum | N1 = N-hum | N1 = Qu P | N1 = Qu Pasubj | N0 V Prep N1 V0-inf W | N0 V N1 V0-inf W | N0 V V0-inf W | N0 Ø prep V0-inf | N0 Ø Prep Nhum | N0 Ø Prep N-hum | N0 Ø Nhum | N0 Ø N-hum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| V_1 | o | - | o | - | - | o | o | - | - | o | - | - | - | - | - | - | - | - | + | o | o | o | o |
| V_2 | + | - | - | - | - | o | o | - | - | - | - | - | + | - | - | o | o | + | - | - | - | - | - |
| V_4 | - | - | - | + | + | o | - | o | + | - | - | o | o | - | - | - | - | - | - | - | - | - | - |
| V_31R | o | o | - | - | - | o | o | + | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| V_31H | + | - | - | - | - | o | o | + | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| V_33 | o | o | - | o | - | o | o | o | - | - | + | o | o | - | - | - | - | - | - | - | - | - | - |
| V_33H | o | - | - | o | - | o | o | - | + | - | - | + | - | - | - | - | - | - | - | - | - | - | - |

FIG. 4.7: Extrait de la table des classes de verbes du LG.

### 4.4.2 Hiérarchie des classes de verbes

Pour les verbes, nous disposons d'une hiérarchie des classes du LG sur plusieurs niveaux créée manuellement, décrite pour la première fois dans (Sigogne *et al.*, 2011). Chaque niveau contient plusieurs classes qui regroupent des classes du LG qui ne partagent pas forcément toutes leurs propriétés mais dont les entrées ont un comportement syntaxique relativement similaire. Un extrait de cette hiérarchie est montré dans la figure 4.8. Ici, les classes *4, 6* et *12* du LG sont regroupées dans une classe *QTD2* (transitifs directs à deux arguments pouvant être sous la forme de complétives). Puis cette classe est elle-même regroupée avec d'autres classes au niveau supérieur de la hiérarchie pour former une classe *TD2* (transitifs directs à deux arguments).

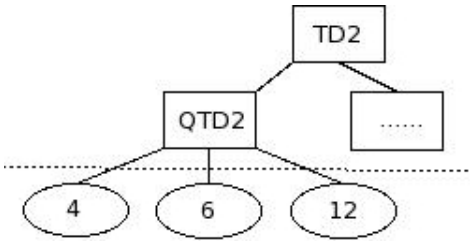



FIG. 4.8: Extrait de la hierarchie des classes de verbes du LG.

La hiérarchie complète des classes de verbes est disponible dans l'annexe 1. Les caractéristiques de chaque niveau sont indiquées dans le tableau 4.1. Le niveau 0 représente l'ensemble des classes présentes initialement dans le Lexique-Grammaire. La colonne `#classes` précise le nombre de classes distinctes. Quant aux colonnes `AVG_1` et `AVG_2`, elles indiquent le nombre moyen d'entrées par classe et le nombre moyen de classes par forme verbale distincte[108].

[108]On peut également préciser que 3121 verbes pour 3195 entrées, possèdent toutes leurs entrées dans une seule et même table.

| Niveau | #classes | AVG_1 | AVG_2 |
|---|---|---|---|
| 0 | 61 | 227 | 2,34 |
| 1 | 13 | 1067 | 1,82 |
| 2 | 10 | 1386 | 1,75 |
| 3 | 4 | 3465 | 1,44 |

TAB. 4.1: Caractéristiques de la hiérarchie des classes de verbes du LG.

### 4.4.3 Le LG au format XML, Lglex

Afin d'être exploité de façon plus aisée par des processus informatiques, de nombreux travaux (Hathout & Namer, 1998; Gardent *et al.*, 2006; Sagot & Fort, 2007) ont tenté de transformer le LG en d'autres formats. Dans ces travaux, un script de configuration, assigné à chaque classe, permet d'effectuer des opérations de restructuration des informations ou de codage d'informations manquantes. Cependant, le LG est un lexique en constante évolution ce qui pose un problème de maintenance évident. Par exemple, si une propriété est supprimée d'une ou plusieurs classes, les scripts de l'ensemble de ces classes doivent être révisés pour prendre en compte cette nouvelle modification. Afin de réduire au maximum les opérations de maintenance, un outil appelé LGExtract (Constant & Tolone, 2010) se base sur une approche globale plutôt que locale à chaque classe, en s'aidant notamment de la table des classes du LG. Le Lglex est un lexique au format XML généré à partir des tables du LG grâce à l'outil LGExtract[109]. Dans ce lexique, chaque entrée est associée à des informations lexicales et syntaxiques divisées en trois sections :
– informations lexicales spécifiant le prédicat (e.g. le verbe *mange*) et des contraintes lexicales (la distribution des prépositions et des déterminants acceptés dans les constructions).
– nature des arguments des prédicats (types de syntagme, traits sémantiques).
– constructions acceptées par le prédicat (constructions absolues et transformations).
La figure 4.9 montre un extrait du Lglex pour une entrée simplifiée du verbe *achever* de la classe *1*. Cette entrée accepte plusieurs constructions absolues dont *N0 V Prép V0-inf W*, comme dans *Max achève de peindre le mur*. L'infinitive contenue dans cette construction est introduite par la préposition *de*. De plus, l'argument 0 doit être un syntagme nominal humain. La transformation autorisée *Prép N2hum = Ppv = : lui* indique que l'argument *Prép N2hum*, dénotant une personne, peut être pronominalisé sous la forme du clitique *lui*.

### 4.4.4 Le LG au format Alexina, Lglex-Le*ffff*

Le Lglex-Le*ffff* (Tolone & Sagot, 2011) est un lexique au format Alexina résultant de la conversion automatique du lexique Lglex. Chaque entrée de ce dernier est convertie en une ou plusieurs entrées possédant les mêmes caractéristiques que celles du Le*ffff*, à savoir un ensemble d'arguments liés à leur fonction syntaxique et leurs possibles réalisations syntagmatiques. Pour chaque entrée à construire, les fonctions syntaxiques sont calculées à l'aide d'heuristiques

[109] `http://igm.univ-mlv.fr/~mconstan/research/software/`

```
<entry id="V_1_1">
 <lexical-info cat="verb">
  <lemma value="achever" />
  <aux avoir value="true" />
  <prepositions>
   <preposition id="1">
    <prep value="de" />
   </preposition>
  </prepositions>
 </lexical-info>
 <arguments>
  <constituent pos="0">
   <component>
    <cat value="NP" />
    <hum value="true" />
   </component>
  </constituent>
 </arguments>
```

```
<all-constructions>
 <absolute-constructions>
  <construction value="o::N0 V N1hum" />
  <construction value="base::N0 V Prép V0-inf W" />
  ...
 </absolute-constructions>
 <relative-constructions>
  <construction value="Prép N2hum = Ppv =: lui" />
 </relative-constructions>
</all-constructions>
</entry>
```

FIG. 4.9: Entrée simplifiée du Lglex pour le verbe $achever$ de la table *1*.

utilisant la position, la nature et certaines propriétés des arguments. Quant aux réalisations, elles peuvent être obtenues soit directement, soit par l'intermédiaire des champs `prep` et `loc` lorsque la construction mentionne un argument en *Prép X*, soit grâce à des propriétés spécifiques (par exemple, *N1 = Ppv = : lui*). La figure 4.10 montre un extrait du Lglex-Le*ffff* pour une entrée lemmatisée du verbe *déferler* de la table des verbes *V_2*. Ce verbe accepte deux arguments, un sujet et un complément locatif facultatif[110], comme dans *Les étudiants déferlent dans les rues*.

*déferler_V_2_45 v-er :std 100 ;Lemma ;v ;*
*<Suj :cln|sn,Loc :(à-sn|de_chez-sn|de-sn|dans-sn)> ;cat=v ;%actif*

FIG. 4.10: Entrée du Lglex-Le*ffff* pour le verbe *déferler* de la table *V_2*.

Dans la version actuelle du Lglex-Le*ffff*[111], seuls les lexiques Lglex des verbes et des noms prédicatifs ont été convertis. Dans le tableau 4.11, nous avons indiqué le nombre d'entrées distinctes ainsi que le nombre de lemmes distincts pour ces deux catégories grammaticales.

| Catégorie grammaticale | Entrées distinctes | Lemmes distincts |
|---|---|---|
| verbes | 22130 | 5737 |
| noms prédicatifs | 30445 | 10069 |

FIG. 4.11: Statistiques tirées du Lglex-Le*ffff* pour différentes catégories grammaticales.

[110]Comme dans Dicovalence et le Le*ffff*, la présence de parenthèses autour des réalisations indique le caractère facultatif.
[111]Version 3.4

## 4.5 LexSchem

LexSchem[112] est un lexique des verbes du français automatiquement acquis à partir de corpus grâce à l'outil ASSCI (Messiant *et al.*, 2008; Messiant, 2008). Il contient 4656 formes lemmatisées pour 9476 entrées (environ 2.2 entrées par forme). La procédure d'acquistion du lexique est composée de trois modules distincts :

– Les verbes et leur environnement immédiat (contexte de mots) sont extraits d'un grand corpus. Ce corpus aura été préalablement étiqueté automatiquement par TreeTagger (section 3.2.1) et annoté syntaxiquement par Syntex (Bourigault *et al.*, 2005). Syntex est un analyseur syntaxique de surface basé sur des statistiques et des heuristiques qui extrait des relations de dépendances bilexicales typées (adjectif/nom, verbe/nom,...). Seuls les verbes apparaissant plus de 200 fois dans le corpus ont été analysés par le système ASSCI. Les relations de dépendances concernant les verbes sont ensuite extraites.
– Pour chaque verbe analysé, les cadres de sous-catégorisation sont construits dynamiquement à partir de l'ensemble des dépendances extraites. Les arguments peuvent prendre une fonction syntaxique parmi : sujet (`SUJ`), objet direct (`OBJ`), objet indirect (`A-OBJ, DE-OBJ, P-OBJ`) et pronom réfléchi (`REF`). L'ensemble des réalisations syntagmatiques possibles est presque identique à ceux de Dicovalence et du L*efff*.
– La dernière étape a pour but de filtrer les résultats obtenus. En effet, ASSCI est basé sur un système séquentiel dont chaque étape peut provoquer de potentielles erreurs. L'opération de filtrage consiste à calculer le score de maximum de vraisemblance pour les cadres de sous-catégorisation extraits pour chaque verbe (Korhonen *et al.*, 2000). Un cadre est conservé si son score est supérieur à un certain pallier fixé manuellement.

Dans le cadre du français, le corpus utilisé est composé d'articles de journaux provenant de 10 ans du journal *Le Monde* (environ 200 millions de mots), et 107 schémas différents sont répertoriés dans le lexique pour l'ensemble des entrées.

### 4.5.1 Exemple tiré du lexique

La figure 4.12 montre un extrait de LexSchem pour une entrée du verbe *abîmer*. Ce verbe accepte trois arguments : un sujet, un pronom réfléchi et un complément d'objet indirect introduit par *dans* (comme dans *L'avion s'est abîmé dans la mer.*). On peut voir que les lemmes têtes les plus fréquents pour chaque argument sont également disponibles avec leur fréquence associée. Par exemple, *Europe* est le sujet du verbe avec une fréquence de 0.02.

## 4.6 Couverture des lexiques

La description syntaxique des entrées verbales est une particularité commune à l'ensemble des lexiques que nous venons de décrire. Aussi, nous avons indiqué, dans le tableau 4.2, différentes informations statistiques à propos des verbes (lemmatisés), et ceci pour chaque lexique. Les

[112] `http://www-lipn.univ-paris13.fr/~messiant/lexschem.html`

** Entrée 00497 ( 87 occurences, fréquence relative : 23.9 ) :
Verbe : abîmer ( 364 occurences )
Schéma de sous-catégorisation : [<SUJ :SN,REF :refl,P-OBJ :SP<dans+SN> >]
Lemmes têtes de l'argument 0 : il (5 :0.06), on (4 :0.05), boeing (4 :0.05), Europe (2 :0.02),...
Lemmes têtes de l'argument 1 : se (87 :1.00)
Lemmes têtes de l'argument 2 : Atlantique (6 :0.07), océan (5 :0.05), contemplation (5 :0.05),...

FIG. 4.12: Entrée simplifiée de LexSchem pour le verbe *abîmer*.

colonnes `entrées`, `formes` et `#e.p.f` indiquent respectivement le nombre d'entrées verbales, le nombre de formes verbales distinctes et le nombre moyen d'entrées par forme verbale. La colonne `couverture` indique la couverture du lexique sur les formes verbales distinctes du FTB-UC. Le L*efff* a une couverture presque parfaite sur le corpus car il possède un nombre de formes verbales distinctes fortement supérieur à celui des autres lexiques. En ce qui concerne le nombre moyen d'entrées par forme, on peut voir qu'à format identique les lexiques L*efff* et Lglex-L*efff* ont des valeurs très différentes. Le principe de la séparation en sens des éléments lexicaux introduite par le LG se répercute sur le nombre conséquent d'entrées du Lglex-L*efff* et du Lglex. On peut noter que le Lglex-L*efff* possède légèrement moins d'entrées que le Lglex car, au cours de la génération automatique du lexique, certaines entrées ont été supprimées pour cause de d'erreurs de conversion.

| Lexique | entrées | formes | #e.p.f | couverture |
|---|---|---|---|---|
| Dicovalence | 8254 | 5011 | 1.64 | 88.4 |
| L*efff* | 7310 | **6808** | 1.07 | **99.0** |
| Lexschem | 7239 | 3123 | 2.31 | 91.0 |
| Lglex | 13862 | 5922 | 2.34 | 96.5 |
| Lglex-L*efff* | 22130 | 5737 | 3.85 | 96.4 |

TAB. 4.2: Couvertures des lexiques sur les formes verbales du FTB-UC.

Concernant les autres catégories grammaticales, le L*efff* décrit des cadres de sous-catégorisation exploitables pour les adjectifs. Quant au LG, il contient des tables de noms prédicatifs d'où la présence d'informations concernant cette catégorie grammaticale dans les lexiques Lglex et Lglex-L*efff*. Nous avons indiqué, dans le tableau 4.3, la couverture des lexiques sur le FTB-UC pour chacune de ces catégories.

| Lexique | entrées | formes | #e.p.f | couverture |
|---|---|---|---|---|
| adjectifs$_{lefff}$ | 58699 | 28864 | 2.03 | 87.2 |
| noms$_{lglex}$ | 13926 | 10079 | 1.38 | 48.9 |
| noms$_{lglex-lefff}$ | 30445 | 10069 | 3.02 | 48.9 |

TAB. 4.3: Couvertures des lexiques sur les formes nominales et adjectivales du FTB-UC.

# Deuxième partie

# Stratégies d'exploitation de ressources lexicales et syntaxiques pour l'étiquetage morpho-syntaxique et l'analyse syntaxique

# 5

# Les unités multi-mots dans l'étiquetage morpho-syntaxique et l'analyse syntaxique

## 5.1 Introduction

L'intégration des expressions multi-mots[113] [MWE] dans des applications réelles, comme la traduction automatique ou l'extraction d'information, est cruciale car de telles expressions ont la particularité de contenir un certain degré de figement. En particulier, elles forment des unités lexicales complexes qui, si elles sont prises en compte, pourraient non seulement améliorer l'étiquetage morpho-syntaxique et l'analyse syntaxique, mais aussi faciliter les analyses sémantiques qui en découlent.
Ces dernières années, l'étiquetage morpho-syntaxique a atteint d'excellents niveaux de performance grâce à l'utilisation de modèles probabilistes discriminants (chapitre I.3), comme les modèles de maximum d'entropie [ME] (Ratnaparkhi, 1996; Toutanova *et al.*, 2003), les séparateurs à vaste marge [SVM] (Giménez & Màrquez, 2004) ou encore les champs conditionnels aléatoires [CRF] (Tsuruoka *et al.*, 2009). Il a par ailleurs été montré que le couplage de ces modèles avec des lexiques externes augmente encore la qualité de l'annotation, comme l'illustre (Denis & Sagot, 2009, 2010) pour les modèles ME (section I.3.3.2). Néanmoins, les évaluations réalisées considèrent toujours en entrée un texte avec une segmentation lexicale parfaite, c'est-à-dire que les MWEs, qui forment par définition des unités linguistiques, ont été parfaitement reconnues au préalable. Or cette tâche de segmentation est difficile car elle nécessite des ressources lexicales importantes. On notera que les systèmes tels que Macaon (Nasr *et al.*, 2010) et Unitex (Paumier, 2011) intègrent une analyse lexicale avec segmentation multi-mots ambiguë avant levée d'ambiguité par l'utilisation d'un modèle de Markov caché [HMM].
L'intégration des MWEs dans un processus d'analyse syntaxique probabiliste a déjà été envisagée dans quelques études. Toutefois, comme pour la phase d'étiquetage, elles reposent pour la majorité sur un corpus au sein duquel l'ensemble des MWEs a été parfaitement identifié au préalable. Bien qu'artificielles, ces études ont montré une amélioration des performances

[113] Aussi appelé *MultiWord Expressions* en anglais.

d'analyse avec, par exemple, (Nivre & Nilsson, 2004; Eryigit *et al.*, 2011) pour l'analyse en dépendance et (Arun & Keller, 2005; Hogan *et al.*, 2011) pour l'analyse en constituants. Plus récemment, (Green *et al.*, 2011) ont intégré la reconnaissance des MWEs au sein de la grammaire et non plus dans une phase préalable. La grammaire est entrainée sur un corpus arboré où les MWEs sont annotées avec des noeuds non-terminaux spécifiques.
Dans ce mémoire, nous nous intéressons à un type de MWE, les mots composés, que nous souhaitons incorporer, de manière pertinente, aux processus d'étiquetage morpho-syntaxique et d'analyse syntaxique. D'une part, nous proposons d'intégrer les deux tâches de segmentation et d'étiquetage dans un seul modèle CRF couplé à des ressources sur les mots composés. Cet étiqueteur-segmenteur permet d'identifier et d'étiqueter les mots composés d'un texte grâce à des traits consacrés aux mots composés. D'autre part, nous proposons d'évaluer deux stratégies discriminantes d'intégration de ces expressions dans un contexte réel d'analyse syntaxique en constituants : (a) pré-segmentation lexicale au moyen d'un reconnaisseur discriminatif de mots composés basé sur un modèle CRF, suivie d'une analyse ; (b) analyse basée sur une grammaire incluant l'identification des mots composés, suivie d'une phase de réordonnancement des analyses à l'aide d'un modèle maximum d'entropie intégrant des traits consacrés aux mots composés. (a) est une implémentation réaliste de l'approche classique de pré-regroupement des MWEs. Nous souhaitons évaluer si la reconnaissance automatique des MWEs a toujours un impact positif sur l'analyse syntaxique, c'est-à-dire, si une segmentation lexicale imparfaite ne provoque pas trop d'effets de bord sur les constituants supérieurs. L'approche (b) est innovante pour la reconnaissance des MWEs : nous sélectionnons la segmentation lexicale finale après l'analyse syntaxique afin d'explorer le plus d'analyses possibles (contrairement à la méthode (a)). Cette approche ressemble à celle proposée par (Wehrli *et al.*, 2010) qui reclasse les hypothèses d'analyses générées par un analyseur symbolique en se basant sur la présence ou non de collocations.
La suite de ce chapitre est organisée comme suit : la section 5.2 présente les problématiques du repérage des MWEs et de leur intégration dans les tâches fondamentales que sont l'étiquetage et l'analyse syntaxique. La section 5.3 décrit plus en détail les deux stratégies proposées et les modèles sous-jacents. Nous décrivons ensuite, dans la section 5.4, les traits concernant les MWEs intégrés dans nos deux modèles. Enfin, la section 5.5 rapporte et analyse les résultats obtenus lors de nos expériences.

## 5.2 Unités lexicales multi-mots

Dans le consensus actuel du Traitement Automatique des Langues (TAL), les expressions multi-mots forment des unités linguistiques aux comportements lexicaux, syntaxiques et/ou sémantiques particuliers. Elles regroupent les expressions figées et semi-figées, les collocations, les entités nommées, les verbes à particule, les constructions à verbe support, les termes, etc... (Sag *et al.*, 2002). Leur identification est donc cruciale avant toute analyse sémantique. Elles apparaissent à différents niveaux de l'analyse linguistique : certaines forment des unités lexicales contigues à part entière, comme par exemple *cordon bleu*, *San Francisco*, *par rapport à*, d'autres composent des constituants syntaxiques comme les phrases figées, telles que

*N0 prendre le taureau par les cornes*, *N0 prendre N1 en compte*, ou les constructions à verbe support, comme *N0 donner un avertissement à N1* et *N0 faire du bruit*.
Dans ce mémoire, nous nous focalisons sur les MWEs continues qui forment des unités lexicales auxquelles on peut associer une étiquette morpho-syntaxique. Par exemple, *tout à fait* est un adverbe, *à cause de* est une préposition, *table ronde* est un nom. Les variations morphologiques et lexicales sont très limitées, et les variations syntaxiques très souvent interdites[114]. De telles expressions sont généralement analysées au niveau lexical. Par exemple, le mot *vin*, quand il désigne la boisson, peut être associé à un mot représentant une couleur prise parmi la liste restreinte suivante : *rouge, blanc ou rosé*. Le mot *orange* n'est en revanche pas accepté. Le mot *caisse noire* est un autre exemple de limitation lexicale car, dans le cas présent[115], *caisse* ne peut être associé qu'au mot *noire*. Par la suite, nous utilisons le terme *mot composé* ou *unité polylexicale*.
Les unités polylexicales peuvent être recensées dans des dictionnaires électroniques ou des grammaires locales. Les dictionnaires électroniques sont des listes qui associent des formes lexicales à des informations linguistiques comme les catégories grammaticales ou certains traits sémantiques tels que *humain* ou *concret*. Les grammaires locales (Gross, 1997; Silberztein, 2000) sont des réseaux récursifs de transitions décrits sous la forme de graphes d'automates finis. Chaque transition est étiquetée par un élément lexical (*mange* par exemple), un masque lexical correspondant à un ensemble de formes lexicales encodées dans un dictionnaire (`<manger>` symbolisant toutes les formes fléchies dont le lemme est *manger*) ou un élément non-terminal référant à un autre automate. Elles sont très utiles pour décrire de manière compacte des MWEs acceptant des variations lexicales. Un système de transduction permet d'annoter les expressions décrites, comme la catégorie grammaticale ou l'analyse des composants internes pour les entités nommées (Martineau *et al.*, 2009). La figure 5.1 montre une grammaire locale sous forme de graphe présente dans la distribution d'Unitex. Cette grammaire permet d'identifier des nombres en toutes lettres simples (*cent*) et composés (*cent et un*). En sortie, les mots reconnus se voient attribuer une ou plusieurs étiquettes morpho-syntaxiques grâce à une opération de transduction (inscriptions en gras). Dans le cas présent, 4 entrées lexicales sont créées pour les catégories grammaticales suivantes : `D`, `NC`, `ADJ` et `PRO`.

### 5.2.1 Identification des mots composés

La reconnaissance automatique des mots composés est, la plupart du temps, réalisée à l'aide de ressources lexicales construites manuellement ou apprises automatiquement. La méthode la plus simple est fondée sur la consultation de lexiques comme dans (Silberztein, 2000). Le plus grand désavantage est que cette procédure se base entièrement sur des dictionnaires, et est donc incapable de découvrir de nouveaux mots composés. L'utilisation d'extracteurs automatiques de collocations peut donc s'avérer utile. Par exemple, (Watrin & François, 2011) calculent à la volée pour chaque collocation candidate dans le texte traité, son score d'association au

[114] De telles expressions acceptent très rarement des insertions, souvent limitées à des modifieurs simples comme dans *à court terme* et *à très court terme*.

[115] Wikipedia : *"Une caisse noire est une réserve d'argent, le plus souvent illicite, servant à financer des actions souvent illicites comme le versement de pots-de-vin, par exemple pour la conclusion d'une vente."*

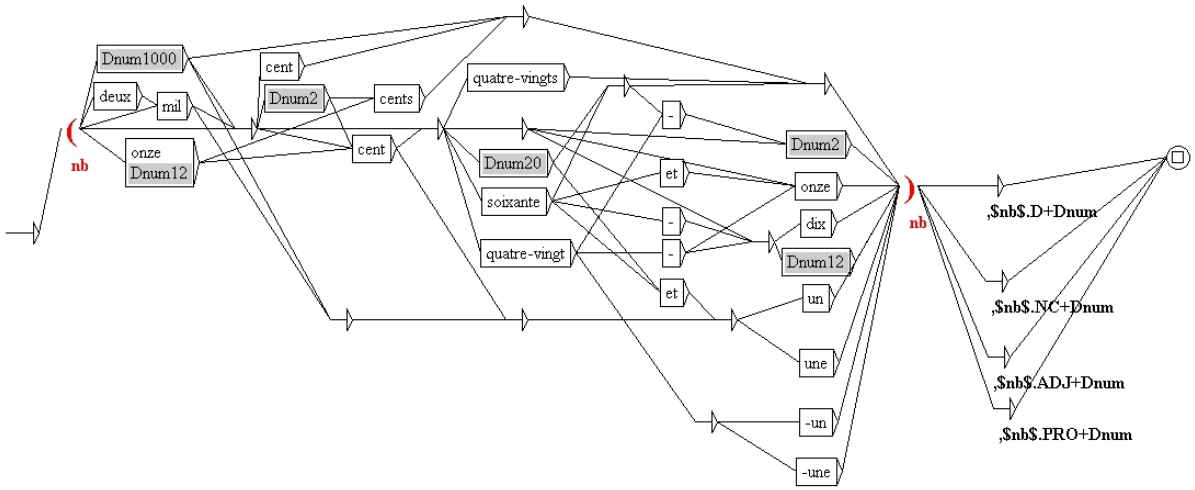



FIG. 5.1: Grammaire locale d'identification de mots composés (des nombres) présente dans la distribution d'Unitex.

moyen d'une base externe de n-grammes apprise sur un grand corpus brut. L'expression est ensuite étiquetée comme MWE si son score d'association est plus grand qu'un seuil donné. Ils obtiennent d'excellents résultats dans le cadre d'une tâche d'extraction de mots-clés. Dans le cadre d'une évaluation sur corpus de référence, (Ramisch *et al.*, 2010) ont développé un classifieur basé sur un séparateur à vastes marges intégrant des traits correspondant à différentes mesures d'associations des collocations. Les résultats sont plutôt faibles sur le corpus GENIA. (Green *et al.*, 2011) ont confirmé ces mauvais résultats sur le FTB. Ceci s'explique par le fait que de telles méthodes ne font aucune distinction entre les différents types de MWEs, et que les types de MWEs annotés dans les corpus sont souvent limités.
L'identification de telles expressions est une tâche très difficile car les unités non décrites dans les ressources sont difficilement reconnaissables. Elle est d'autant plus difficile qu'elle dépend du contexte d'occurrence. En effet, une expression reconnue est souvent ambigue avec l'analyse en combinaison libre, comme par exemple :

*il **en fait** une priorité* (combinaison libre)
*j'ai **en fait** beaucoup travaillé* (combinaison figée)

On observe parfois des chevauchements avec d'autres unités polylexicales, comme dans la séquence *une pomme de terre cuite*, où *pomme de terre* et *terre cuite* sont des mots composés. C'est pourquoi les outils existants de segmentation en unités polylexicales, comme dans INTEX (Silberztein, 2000) ou SxPipe (Sagot & Boullier, 2008), produisent une segmentation ambiguë sous la forme d'automates finis acycliques pour éviter de prendre une décision définitive trop hâtive. Cette analyse ambiguë peut alors être intégrée dans des traitements linguistiques tels que l'étiquetage morpho-syntaxique (Nasr *et al.*, 2010; Paumier, 2011)) ou l'analyse syntaxique superficielle (Blanc *et al.*, 2007; Nasr *et al.*, 2010) et profonde (Sagot, 2006).
À présent, nous allons voir en détail deux approches pour l'identification des MWEs consistant à combiner cette tâche avec un processus automatique : l'étiquetage morpho-syntaxique ou

l'analyse syntaxique.

### Combinaison avec l'étiquetage morpho-syntaxique

La tâche d'un étiqueteur-segmenteur consiste en premier lieu à identifier, dans un texte donné en entrée, les unités simples et composées. Puis, ces unités sont annotées par des étiquettes morpho-syntaxiques. On peut voir l'étape d'identification des unités multi-mots comme une tâche de segmentation similaire au chunking ou à la reconnaissance d'entités nommées, qui identifient les limites des segments (chunks ou entités nommées) et les annotent. En utilisant la représentation BIO (Ramshaw & Marcus, 1995), la tâche de segmentation revient à annoter des unités minimales du texte, et ce avec trois étiquettes possibles : (a) `B`, le mot est le premier d'une unité polylexicale. (b) `I`, le mot est à l'intérieur d'une unité polylexicale. (c) `O`, le mot est en dehors d'une unité polylexicale.
Pour combiner étiquetage morpho-syntaxique et reconnaissance d'unités multi-mots, il suffit de concaténer les deux étiquetages en associant à chaque unité minimale une étiquette de la forme `X+B` ou `X+I`, où $X$ est la catégorie grammaticale de l'unité, et le suffixe indique si elle se trouve au début d'une unité multi-mots (`B`) ou dans une position interne (`I`). Le suffixe `O` est inutile car la fin d'un segment lexical correspond au début d'un autre (suffixe `B`), ou à une fin de phrase. Cette procédure d'annotation permet de déterminer à la fois les limites des unités lexicales, mais aussi leur étiquette morpho-syntaxique. L'application de cette procédure sur un corpus consiste donc à transformer les unités multi-mots de ce corpus en une représentation BIO. Le jeu d'étiquettes initial est ainsi doublé car chaque étiquette se dédouble en une variante `B` et une variante `I`. À titre d'illustration, nous avons appliqué cette méthode sur la phrase étiquetée ci-dessous, tirée du FTB-UC :

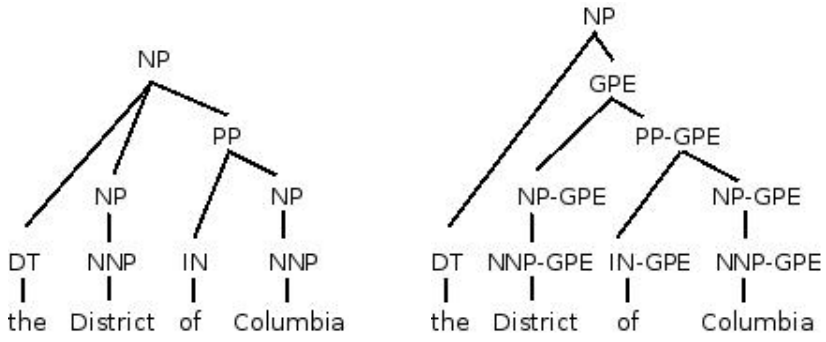
$$\underset{DET}{\underline{Une}}\ \underset{NC}{\underline{guerre}}\ \underset{V}{\underline{pourrait}}\ \underset{VINF}{\underline{conduire}}\ \underset{P}{\underline{à}}\ \underset{DET}{\underline{une}}\ \underset{NC}{\underline{prise\_de\_contrôle}}\ \underset{P}{\underline{des}}\ \underset{NC}{\underline{guerilleros}}\ \underset{PONCT}{\underline{.}}$$

Dans cette phrase, le nom commun *prise_de_contrôle* est la seule unité multi-mots présente. D'après la procédure d'annotation, cette unité est d'abord découpée en unités minimales, ce qui a pour effet d'extraire 3 unités distinctes. La première unité, *prise*, est étiquetée par `NC+B`. La deuxième et la troisième unités sont étiquetées par `NC+I` car elles sont dans une position interne. Les autres mots de la phrase se voient attribuer le suffixe `+B`. En sortie, la phrase étiquetée sous le format BIO est la suivante :

$$\underset{DET+B}{\underline{Une}}\ \underset{NC+B}{\underline{guerre}}\ \underset{V+B}{\underline{pourrait}}\ \underset{VINF+B}{\underline{conduire}}\ \underset{P+B}{\underline{à}}\ \underset{DET+B}{\underline{une}}\ \underset{NC+B}{\underline{prise}}\ \underset{NC+I}{\underline{de}}\ \underset{NC+I}{\underline{contrôle}}\ \underset{P+B}{\underline{des}}\ \underset{NC+B}{\underline{guerilleros}}\ \underset{PONCT+B}{\underline{.}}$$

### Combinaison avec l'analyse syntaxique

Une approche récente consiste à coupler, dans un même modèle, l'annotation des mots composés avec un analyseur syntaxique (Cafferkey *et al.*, 2007; Finkel & Manning, 2009; Green *et al.*,




(a) Arbre original (b) Arbre modifié

FIG. 5.2: Modification du sous-arbre lié à l'entité nommée *District of Columbia*.

2011). Les travaux de (Finkel & Manning, 2009) ont pour but de combiner analyse syntaxique et reconnaissance des entités nommées dans un modèle discriminant d'analyse syntaxique basé sur les CRF. Leurs expériences sont basées sur la version 2.0 du corpus en constituants de l'anglais, OntoNotes (Hovy *et al.*, 2006). Ce corpus a la particularité de contenir plusieurs niveaux d'annotations, incluant les constituants, des structures prédicatives, des informations sémantiques, ainsi que le repérage des entités nommées. La figure 5.2a montre un sous-arbre extrait du corpus pour la séquence de mots *the District of Columbia* (le District de Columbia). La séquence *District of Columbia* est identifiée comme une entité nommée associée au symbole `GPE`[116]. Afin de coupler la tâche d'identification des entités nommées avec l'analyse syntaxique, les arbres du corpus ont subi des modifications structurelles, ainsi qu'au niveau des annotations. Aussi, pour un sous-arbre dont la séquence continue de mots (feuilles) est identifiée comme une entité nommée, un noeud est ajouté en tant que racine du sous-arbre. Ce noeud a pour étiquette syntaxique le symbole de l'entité nommée. Puis, ce symbole est propagé dans tous les noeuds, terminaux et non-terminaux, du sous-arbre en l'ajoutant à l'étiquette originale. L'illustration de cette phase de modification sur l'arbre précédent est montrée dans la figure 5.2b. La séquence *the District of Columbia* étant reconnue comme une entité nommée de symbole `GPE`, un noeud racine d'étiquette `GPE` est ajouté, et ce symbole est bien propagé dans le sous-arbre (`NNP-GPE`, `PP-GPE`,...). L'analyseur syntaxique utilisé par (Finkel & Manning, 2009) est basé sur un modèle discriminant CRF-CFG (section I.2.4.2). Lors de l'apprentissage et de l'évaluation, ce type de modèle requiert des traits pour fonctionner. Pour ce faire, ils ont exploité deux ensembles de patrons de traits décrits dans de précédents travaux : l'un extrayant des informations générales (Finkel *et al.*, 2008), et l'autre des informations spécifiques aux sous-arbres liés aux entités nommées (Finkel *et al.*, 2005). Les évaluations ont montré que l'ajout d'informations sur les entités nommées, dans les arbres et donc dans le modèle, permet d'améliorer significativement les performances de l'analyseur : gain absolu de +1.36 pour le score $F_1$, et +9 pour le score $F_1$ associé à la reconnaissance des entités nommées.
Plus récemment, (Green *et al.*, 2011) ont adapté le principe de modification structurelle pour les arbres du corpus du français FTB. Leur but premier étant l'identification des MWEs, ils ont conservé la représentation syntaxique des mots composés (chapitre I.1). On peut rappeler que les mots composés sont représentés sous la forme de sous-arbres plats, où l'ensemble des

[116]Geo-political entity (entité géo-politique).

noeuds préterminaux sont reliés à un même noeud père. Dans le cas présent, ce noeud non-terminal a pour étiquette le préfixe `MW` combiné à la catégorie grammaticale du mot composé (`MWN`, `MWA`,...). Cela conduit à augmenter de 11 la taille du jeu d'étiquettes syntaxiques. Le corpus modifié de cette façon est appelé FTB-STF. La figure 5.3 montre le sous-arbre associé au nom composé *tour de passe-passe* dans le FTB-STF. Les évaluations ont montré que les scores d'identification des MWEs étaient maximisés avec des grammaires à substitution d'arbres[117], bien que les meilleurs scores globaux soient obtenus avec des analyseur PCFG.

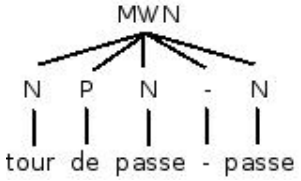



FIG. 5.3: Représentation du nom composé *tour de passe-passe* dans le FTB-STF.

En parallèle de ces travaux, les expériences de (Cafferkey *et al.*, 2007) ont consisté à coupler des annotateurs réels de MWEs et différents types d'analyseurs probabilistes pour l'anglais. Ils ont travaillé sur un corpus de référence non annoté en MWEs. Les MWEs sont reconnues et pré-groupées automatiquement à l'aide de ressources externes et d'un reconnaisseur d'entités nommées. Ils appliquent, ensuite, un analyseur syntaxique et réinsèrent finalement les sous-arbres correspondants aux MWEs pour faire l'évaluation. Ils ont montré des gains faibles mais significatifs.

### 5.2.2 Ressources lexicales

Même s'il existe de plus en plus d'études sur l'extraction automatique d'unités multi-mots, en particulier les collocations ou les termes (Daille, 1995; Dias, 2003; Seretan *et al.*, 2003), les ressources les plus riches et les plus précises ont été construites manuellement. Nous avons notamment utilisé deux dictionnaires de mots simples et composés de la langue générale : le Dela (Courtois & Silberztein, 1990; Courtois *et al.*, 1997) et le Le*fff* (Sagot, 2010). Le Dela a été manuellement développé dans les années 80-90 par l'équipe de linguistes du LADL à Paris 7. Nous utilisons la version libre intégrée à la plateforme Unitex. Il est composé de 840813 entrées lexicales, incluant 104350 entrées composées (dont 91030 noms). Les mots composés présents dans la ressource respectent, en général, les critères syntaxiques définis dans (Gross, 1986). Une ligne, pour un mot composé du Dela, se présente comme suit :

eau de vie,.N+NDN :fs

Dans cet exemple, le nom composé *eau de vie* (`N`) a le code `NDN` car sa structure interne est de la forme : nom suivi d'une préposition *de*, elle-même suivie d'un nom. Quant au Le*fff*, il s'agit d'une ressource lexicale qui a été accumulée automatiquement à partir de diverses sources, et

[117]Tree Substitution Grammar (Cohn *et al.*, 2009).

qui a ensuite été validée manuellement (section I.4.3). Elle comprend 553138 entrées lexicales, incluant 26311 entrées composées (dont 22673 noms). Leurs différents modes de construction rendent ces deux ressources complémentaires. Pour toutes les deux, les entrées lexicales possèdent une forme fléchie, un lemme et une catégorie grammaticale. Le Dela possède un trait supplémentaire pour la plupart des mots composés, à savoir leur struture interne. Une ligne du Le*fff* se présente comme suit :

bottes 100 nc [pred=botte 1<Objde :(de-sinf de-sn),Objà :(à-sinf)>, cat=nc,@fp botte 1 fp

On retrouve dans une telle ligne : le mot (`bottes`), son étiquette associée (`nc`), la forme lemmatisée du mot (`botte`), mais aussi le genre et le nombre (`fp`). En complément, nous disposons aussi de lexiques spécifiques comme Prolex (Piton *et al.*, 1999) composé de toponymes, et d'autres incluant des noms d'organisation et des prénoms (Martineau *et al.*, 2009). En terme de collocations, (Watrin & François, 2011) ont présenté un système retournant, pour toute phrase, la liste des collocations nominales potentielles accompagnées de leur mesure d'association[118]. Leur algorithme est constitué de trois étapes : (a) le texte ciblé est étiqueté. (b) des filtres linguistiques sont appliqués sur le texte étiqueté. Ces filtres sont des patrons qui identifient les MWEs candidates ayant une structure interne bien spécifique (`NA, AN, NDN`,...). (c) les candidats sont donnés en entrée d'une phase de validation statistique, basée sur les mesures d'association. Pour le FTB-UC, nous obtenons 15385 collocations nominales candidates associées à leur log-vraisemblance et leur structure interne. Le nombre d'entrées de ces divers dictionnaires est donné dans le tableau 5.1.

| Dictionnaire | #mots simples | #mots composés |
|---|---|---|
| DELA | 690619 | 272226 |
| Le*fff* | 553140 | 26,311 |
| Prolex | 25190 | 97925 |
| Organisations | 772 | 587 |
| Prénoms | 22074 | 2220 |
| Collocations | - | 15385 |

TAB. 5.1: Dictionnaires morpho-syntaxiques de mots simples et composés exploités pour nos expériences.

Cet ensemble de dictionnaires est complété par une bibliothèque de grammaires locales qui reconnaissent différents types d'unités multi-mots comme les entités nommées (dates, noms d'organisation, de personne et de lieu), prépositions locatives, déterminants numériques et nominaux. En pratique, nous avons utilisé une bibliothèque de 211 automates développée à partir de la bibliothèque en-ligne GraalWeb (Constant & Watrin, 2008).

[118]Les mesures d'association permettent de déterminer si une séquence de mots est une unité composée. Il s'agit de fonctions mathématiques ayant pour but de mesurer le degré de cohésion, ou d'association, entre les mots de la séquence.

### 5.2.3 Couverture des lexiques sur les MWEs du corpus

Dans le corpus original de développement du FTB-UC, nous avons observé qu'environ 97.8% des unités lexicales[119] sont présentes dans nos ressources lexicales (en particulier, 97.4% sont présentes dans les dictionnaires). Alors que 4% des unités sont inconnues (i.e. absentes du corpus d'apprentissage), 1.2% sont à la fois inconnues et absentes des ressources lexicales, ce qui montre que 70% des unités inconnues sont couvertes par nos ressources. On observe également qu'environ 11% des unités sont multi-mots. En décomposant toutes les unités multi-mots du texte en unités minimales, on s'aperçoit qu'à peu près 15% d'entre elles sont incluses dans une unité multi-mots. Parmi les unités multi-mots codées dans le corpus de développement, 81.3% d'entre elles sont présentes dans nos ressources (89.5% en incluant le lexique du corpus d'apprentissage). Ceci montre que 10.5% des unités multi-mots sont totalement inconnues et, par conséquent, seront sans doute très difficilement reconnaissables.
On observe, par ailleurs, que le corpus FTB-UC ne couvre pas la reconnaissance de toutes les unités multi-mots. Tout d'abord, certains déterminants ou certaines entités nommées ne sont pas identifiés, comme les déterminants nominaux (*un certain nombre de mesures*), les dates (*31 décembre 1992*), les noms de personne (*Jacques Chirac*), ou encore les adresses postales (*rue de la Liberté*). Par ailleurs, de nombreux noms composés sont manquants. Par exemple, après avoir appliqué nos ressources lexicales de manière non contextuelle (en excluant les collocations et les grammaires locales reconnaissants des types d'entités nommées ou des déterminants nominaux non codés dans le FTB), nous avons manuellement observé sur le corpus de développement qu'environ 25% des unités polylexicales de nos ressources *adaptées* ne sont pas prises en compte dans le corpus.

## 5.3 Stratégies discriminantes pour l'intégration de ressources lexicales

Au cours de la section 5.2.1, nous avons décrit deux approches d'identification des MWEs basées sur la combinaison de la tâche de segmentation d'un texte avec un processus automatique, à savoir l'étiquetage morpho-syntaxique (représentation BIO) ou l'analyse syntaxique. Pour ces deux approches, nous proposons, dans cette section, plusieurs stratégies permettant d'incorporer des ressources lexicales. Pour l'étiquetage morpho-syntaxique, nous réalisons cette tâche grâce à un modèle discriminant, et nous considérons deux stratégies d'utilisation des ressources lexicales : (a) pré-filtrage ou post-filtrage des étiquetages possibles pour une phrase donnée, et (b) création de nouveaux patrons dont les traits sont générés à partir des lexiques. Quant à l'analyse syntaxique, nous considérons également deux stratégies d'intégration des ressources : (a) une pré-identification des mots composés par un reconnaisseur discriminatif dont les traits proviennent en partie de lexiques, suivie d'une analyse, et (b) une analyse syntaxique incorporant l'identification des mots composés, suivie d'un réordonnanceur intégrant des traits générés à partir des lexiques.

[119] Les unités lexicales sont les unités autres que les nombres et les ponctuations.

### 5.3.1 Couplage d'un segmenteur et d'un étiqueteur

Pour réaliser automatiquement la tâche combinée d'étiquetage et de segmentation (représentation BIO), nous utilisons le modèle des champs conditionnels aléatoires linéaires (Tellier & Tommasi, 2011) [CRF] introduits par (Lafferty *et al.*, 2001) pour l'annotation de séquences. Bien que ce modèle soit décrit dans la section I.2.4.2, nous faisons ici un bref rappel sur les points essentiels, dont la définition mathématique. Etant donné une séquence de mots en entrée $x = (x_1, x_2, ..., x_N)$ et une séquence d'étiquettes en sortie $y = (y_1, y_2, ..., y_N)$, le modèle est défini comme suit :

$$P_\lambda(y|x) = \frac{1}{Z(x)}.\sum_{t}^{N}\sum_{k}^{K} log\lambda_k.f_k(t, y_t, y_{t-1}, x)$$

où $Z(x)$ est un vecteur de normalisation dépendant de $x$. Il est basé sur $K$ traits définis par des fonctions binaires $f_k$ dépendant de la position courante $t$ dans $x$, l'étiquette courante $y_t$, l'étiquette précédente $y_{t-1}$ et toute la séquence en entrée. Chaque mot $x_i$ de $x$ intègre non seulement sa propre valeur lexicale mais aussi toutes les propriétés du mot (e.g. ses suffixes, ses étiquettes dans un lexique externe, il commence par une majuscule, etc.). Les traits sont activés si une configuration particulière entre $t$, $y_t$, $y_{t-1}$ et $x$ est satisfaite (i.e. $f_k(t, y_t, y_{t-1}, x) = 1$). Chaque trait est associé à un poids $\lambda_k$. Ces poids sont les paramètres du modèle et sont estimés lors de la phase d'apprentissage. Les traits utilisés pour notre tâche sont décrits dans la section 5.4. Ils sont générés à partir de patrons qui sont instanciés à chaque position dans la séquence à annoter. Chaque instance $z$ correspond à une fonction binaire $f_k$ qui retourne 1 si l'instance à la position courante correspond à $z$, 0 sinon.
L'intérêt et l'efficacité des CRF proviennent de ce qu'ils prennent en compte des dépendances entre étiquettes reliées les unes aux autres dans le graphe. En cherchant le meilleur $y$, c'est-à-dire la meilleure séquence d'étiquettes associée à une donnée complète $x$, ils se comportent en général mieux qu'une série de classifications d'unités isolées. Mais cette prise en compte a un prix : la phase d'apprentissage d'un CRF peut être longue. Une fois cette phase réalisée, annoter une nouvelle séquence $x$ de $n$ mots en entrée revient alors à trouver le $y$ qui maximise $p(y|x)$. L'espace théorique de recherche de ce meilleur étiquetage $y$ est $|Y|^n$, où $|Y|$ est le nombre d'étiquettes distinctes possibles pour chaque nœud. Mais, grâce à des techniques de programmation dynamique, ce calcul peut être factorisé à l'intérieur des cliques et ramené à $K * n * |Y|^c$ où $c$ est la taille de la plus grande clique ($c = 2$ pour les CRF linéaires) et $K$ le nombre de fonctions caractéristiques. Une fois appris, l'étiqueteur est donc performant.
À présent, il s'agit de déterminer la ou les meilleures stratégies d'intégration des ressources lexicales externes dans un étiqueteur-segmenteur, dont la phase d'apprentissage est réalisée par un CRF. Pour cela, nous nous sommes inspiré des travaux réalisés par (Denis & Sagot, 2009, 2010) pour l'étiqueteur MElt (section I.3.3.2). On peut rappeler que ces derniers ont testé deux approches possibles :

– Des propriétés tirées d'un lexique (étiquettes, classe d'ambiguité,...) sont assignées à chaque mot des corpus et des textes. Concrètement, cela consiste à créer, pour chaque propriété, une nouvelle colonne. De cette manière, les valeurs de ces propriétés peuvent être intégrées aux fonctions caractéristiques du modèle d'apprentissage grâce à des patrons de traits spéci-

fiques.
– Un filtrage des étiquettes incompatibles est effectué à partir des informations présentes dans les ressources lexicales. Concrètement, lors de l'analyse lexicale, la liste des étiquettes attribuées à un mot est l'union des étiquettes du jeu du corpus et des dictionnaires pour ce mot.

Nous discutons, ci-dessous, de la mise en place de chacune de ces solutions, ainsi que des éventuels avantages et inconvénients, par rapport à notre problématique d'identification et d'étiquetage des MWEs.

**Les ressources comme filtrage *a priori* ou *a posteriori*** Les ressources peuvent être vues comme un moyen de contraindre, ou encore de *filtrer* les étiquetages possibles. Concrètement, ce filtrage peut s'opérer avant ou après l'appel à la phase d'apprentissage du modèle CRF. Le filtrage *a priori* consiste à réduire l'espace de recherche des étiquetages possibles $y$ d'une chaîne $x$, et ceci via un prétraitement fondé sur une ressource. Dans le chapitre I.3, nous avons montré qu'à partir d'une phrase brute en entrée, les étiqueteurs morpho-syntaxiques produisent généralement un DAG (graphe orienté acyclique), dont chaque chemin correspond à une séquence possible d'étiquettes. Cette étape est effectuée préalablement à une phase de décodage permettant de déterminer le chemin de ce graphe maximisant la probabilité conditionnelle. Les MWEs peuvent être identifiées lors de l'analyse lexicale, et ainsi figurer dans le DAG. Pour une phrase constituée de $n$ unités minimales, il est évidemment plus facile et rapide de chercher le $y$ qui maximise $p(y|x)$ parmi l'ensemble des étiquetages du DAG plutôt que sur l'espace de tous les $|Y|^n$ étiquetages possibles. Quant au filtrage *a posteriori*, le but est de déterminer les $m$ meilleurs étiquetages possibles de $x$ afin de choisir, ensuite, celui qui est le plus en accord avec la ressource. On peut noter que le filtrage requiert que les étiquettes qui figurent dans la ressource soient identiques à celles qui sont la cible de l'apprentissage. Au cas où les conventions d'étiquetage ne sont pas les mêmes, une fonction de correspondance doit être préalablement appliquée. Malgré la pertinence de ces techniques de filtrage, plusieurs obstacles, principalement techniques, nous ont dissuadé d'explorer cette voie.

**Les ressources comme aide à l'apprentissage.** La puissance des modèles discriminants provient de la souplesse d'intégration et de manipulation de données, pouvant potentiellement provenir de différentes sources, dont des lexiques externes. Il existe au moins deux façons d'agir sur les performances de sortie de tels modèles, dont fait partie le CRF :
– Le choix des propriétés des unités est crucial. En effet, un jeu de propriétés trop simple conduit à une faible entropie du modèle généré, causée par le peu d'observations tirées du corpus d'apprentissage. Concrètement, il s'agit d'ajouter une ou plusieurs colonnes aux données tabulaires.
– Un jeu de patrons de traits pertinents est également essentiel. Ils permettent d'extraire l'ensemble des traits devant être intégrés au modèle. Aussi, ils doivent tirer le maximum d'informations à partir des données, afin d'obtenir des traits suffisamment discriminants pour déterminer une analyse maximisant la probabilité. La définition d'un jeu de patrons dépend fortement du point précédent, à savoir le choix des propriétés à exploiter.

À l'opposé du filtrage *a priori* ou *a posteriori*, il nous paraît plus naturel d'insérer les informations des ressources en tant que propriétés des unités d'un exemple $x$, c'est à dire en jouant sur le jeu de colonnes $x_i^2, ..., x_i^p$. Pour l'ensemble des unités, on peut soit concaténer les différentes propriétés des lexiques associées à une unité pour en faire une seule colonne, soit définir autant de colonnes qu'il n'y a de propriétés à intégrer. Pour ce dernier cas, la combinatoire des conjonctions possibles de plusieurs critères peut être explosive. Ces deux possibilités nécessitent évidemment un jeu de patrons différent, et génèrent donc différents ensembles de traits. Malgré tout, nous avons choisi d'exploiter la deuxième solution qui est beaucoup plus aisée à manipuler et à évaluer. Afin de prendre en compte les nouvelles propriétés, nous avons défini de nouveaux patrons de traits. Ces patrons, consacrés aux MWEs, sont décrits dans la section 5.4.

### 5.3.2 Couplage de la segmentation et de l'analyse syntaxique

Nous rappelons que, pour l'analyse syntaxique, nous souhaitons évaluer deux stratégies innovantes : (a) une pré-identification des mots composés par un reconnaisseur discriminatif dont les traits proviennent en partie de lexiques, suivie d'une analyse, et (b) une analyse syntaxique incorporant l'identification des mots composés suivie d'un réordonnanceur intégrant des traits générés à partir des lexiques.
L'algorithme de la première stratégie est illustré par la figure 5.4. Tout d'abord, un reconnaisseur discriminatif est entraîné sur le corpus d'apprentissage du FTB-UC sous sa représentation BIO. L'analyseur syntaxique est, quant à lui, entraîné de manière classique sur le corpus d'apprentissage original du FTB-UC. Pour les évaluations, les corpus bruts de développement et d'évaluation subissent, en premier lieu, un prétraitement qui consiste à décomposer les MWEs en unités minimales. Ensuite, le reconnaisseur est appliqué sur ces textes bruts. Pour finir, l'analyseur syntaxique est appliqué sur ces textes pré-segmentés. Dans le cadre de nos expériences, le reconnaisseur est l'étiqueteur-segmenteur décrit dans la section précédente. On peut rappeler que les analyseurs syntaxiques PCFG que nous utilisons disposent de leur propre phase d'étiquetage. Aussi, l'étiquetage produit par l'étiqueteur-segmenteur est supprimé afin de ne conserver que la segmentation en unités simples et composées. L'intérêt de cette approche est de pouvoir observer le comportement de l'analyseur syntaxique lorsque la segmentation des textes proposée varie selon l'ensemble de patrons exploité par le reconnaisseur, et en exploitant notamment des informations tirées de lexiques externes.
L'algorithme de la deuxième stratégie est illustré par la figure 5.6, pour l'étape d'apprentissage des analyseurs, et par la figure 5.7 pour l'étape de prédiction. En ce qui concerne les données d'entrée, les corpus et les textes bruts à analyser contiennent maintenant des MWEs décomposées en unités minimales. Dans les corpus, la structure syntaxique associée à une MWE est représentée de façon identique à (Green *et al.*, 2011). Aussi, les noeuds préterminaux associés aux mots internes de la MWE sont reliés à un même noeud père, ayant pour étiquette `MWX`, où $X$ est la catégorie grammaticale de l'expression. Il existe 11 symboles de type MWE, ce qui conduit à obtenir un jeu de 23 étiquettes spécifique aux noeuds non-terminaux. Quant aux noeuds préterminaux des mots internes aux MWEs, ils prennent un symbole parmi le jeu de 28 étiquettes du FTB-UC, et ceci contrairement à (Green *et al.*, 2011) qui faisaient usage d'un jeu de taille moindre composé uniquement des catégories grammaticales (nom, verbe, conjonc-

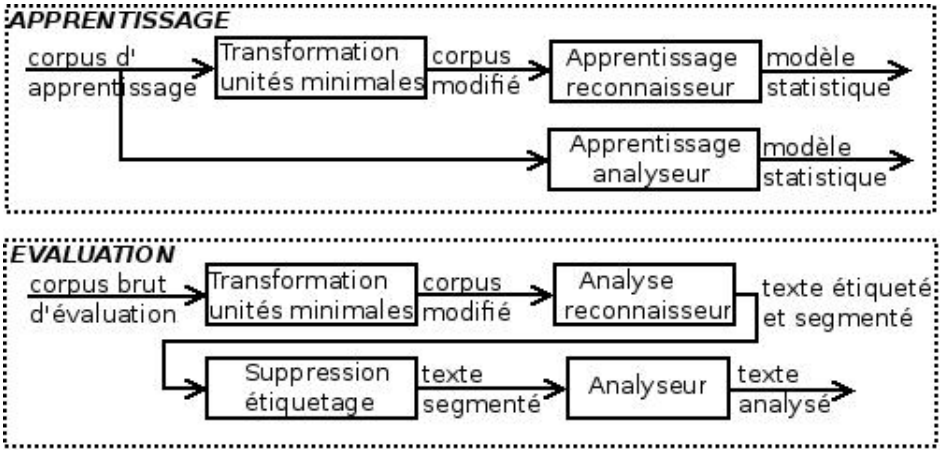
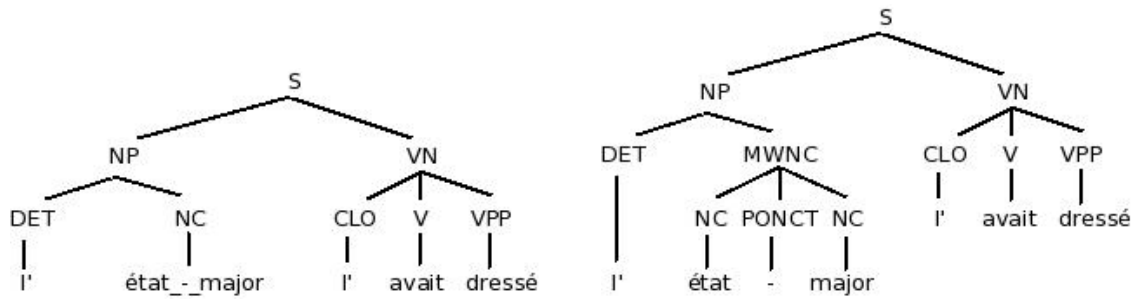



FIG. 5.4: Algorithme de la première stratégie de couplage de la segmentation et de l'analyse syntaxique.

tion,...). Afin d'attribuer ces symboles automatiquement, il est nécessaire d'employer un étiqueteur morpho-syntaxique. Dans la figure 5.5, nous mettons en opposition l'arbre original tiré du FTB-UC de la phrase *l'état-major l'avait dressé depuis quelque temps déjà* (représenté partiellement), et la version où les MWEs sont décomposées. Le nom *état-major* est formé de 3 éléments internes, ce qui conduit à générer un sous-arbre de noeud racine `MWNC`, et possédant trois noeuds préterminaux fils liés aux mots internes d'étiquettes respectives, `NC` (*état*), `PONCT` (-) et `NC` (*major*).




(a) Arbre original. (b) Arbre modifié.

FIG. 5.5: Application de la méthode de décomposition des MWEs sur l'arbre syntaxique de la phrase *l'état-major l'avait dressé depuis quelque temps déjà* (représenté partiellement).

Ensuite, l'apprentissage d'un analyseur et d'un réordonnanceur[120] est effectué sur ce corpus modifié. De cette manière, la tâche d'identification des MWEs est repoussée jusqu'à la dernière étape, à savoir le réordonnancement. De fait, parmi une liste d'analyses candidates renvoyée par l'analyseur, le réordonnanceur choisit celle qui maximise la probabilité. L'intéret est double car, d'une part, les analyses candidates peuvent être potentiellement différentes en terme d'analyse des MWEs. D'autre part, et contrairement à la première stratégie, le fait de ne pas pré-segmenter un texte permet aux analyseurs de moins souffrir de l'effet de dispersion des

[120]Dans notre cas, il s'agit du réordonnanceur de Brown (section I.2.4.6).

données lexicale. En effet, une pré-segmentation peu efficace pourrait générer des MWEs erronées, qui seraient donc possiblement inconnues du modèle, perturbant ainsi la phase d'analyse tout en dégradant les performances générales. Comme dans la section précédente et dans les travaux de (Finkel & Manning, 2009), nous avons exploité deux ensembles de patrons de traits destinés au modèle discriminant du réordonnanceur. L'un est composé de patrons calculant des traits généraux (section I.2.4.6), et l'autre est composé de patrons consacrés aux MWEs. Ces derniers sont décrits dans la section suivante, et sont presque identiques à ceux utilisés par l'étiqueteur-segmenteur.

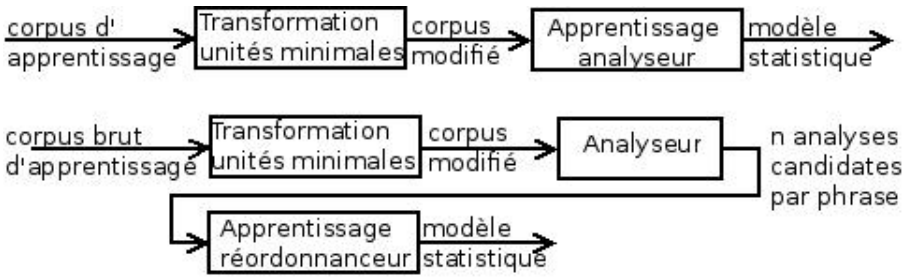



FIG. 5.6: Étape d'apprentissage de la deuxième stratégie d'intégration des ressources de MWE.

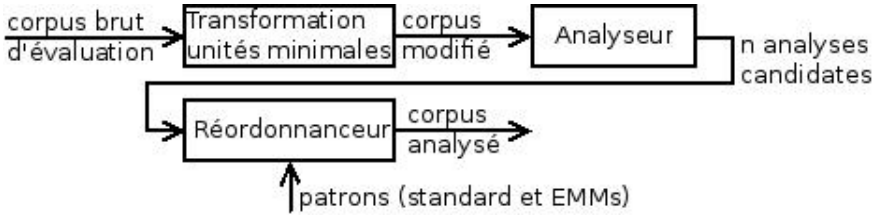



FIG. 5.7: Étape de prédiction de la deuxième stratégie d'intégration des ressources de MWE.

## 5.4 Les traits concernant les mots composés

Dans la section précédente, nous avons décrit des stratégies discriminatives permettant d'utiliser les ressources comme aide à l'apprentissage. Ces stratégies nécessitent des traits concernant les mots composés. Les traits que nous proposons sont générés à partir de patrons. Dans le but de rendre ces modèles comparables, nous avons mis en place deux jeux comparables de patrons de traits : l'un adapté à l'annotation séquentielle et l'autre adapté au réordonnancement. Les patrons pour l'annotation séquentielle sont instanciés à chaque position de la séquence en entrée. Les patrons pour le réordonnanceur sont seulement instanciés aux feuilles des analyses candidates, qui sont dominées par un noeud de type MWE (c'est-à-dire qui ont un ancêtre de type MWE). Nous définissons un patron $T$ comme suit :

– Annotation séquentielle : à chaque position $n$ dans la séquence en entrée $x$,

$$T = (f(x, n), y_n)$$

– Réordonnancement : à chaque feuille (à la position $n$) dominée par un noeud de type MWE $m$ dans l'analyse candidate $p$,

$$T = (f(p, n), label(m), pos(p, n))$$

où $f$ est une fonction à définir et $y_n$ est une étiquette de sortie à la position $n$. $label(m)$ est l'étiquette du noeud $m$, et $pos(p, n)$ indique la position relative du mot à l'indice $n$ dans l'unité polylexicale, avec `B` la position initiale et `I` les autres positions. Afin d'illustrer les différents types de traits, nous définissons, dans la figure 5.8, un étiquetage de la phrase d'exemple suivante :

*La part de marché du produit chute*

Dans cette phrase, *part de marché* est un nom composé car les éléments internes (*de* et *marché*) sont tous annotés par le symbole `+I`. La représentation syntaxique correspondante de ce nom composé dans le corpus FTB-UC modifié est montrée dans la figure 5.9. Chaque unité interne possède sa propre étiquette grammaticale, et les noeuds préterminaux ont tous le même père, annoté par `MWN`.

$$\underset{DET+B}{\underline{La}}\ \underset{NC+B}{\underline{part}}\ \underset{NC+I}{\underline{de}}\ \underset{NC+I}{\underline{marché}}\ \underset{P+B}{\underline{du}}\ \underset{NC+B}{\underline{produit}}\ \underset{V+B}{\underline{chute}}$$

FIG. 5.8: Un étiquetage selon la représentation BIO de la phrase d'exemple.

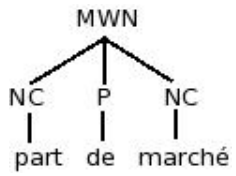



FIG. 5.9: Représentation du nom composé *part de marché* dans le corpus FTB-UC modifié.

### 5.4.1 Traits endogènes

Les traits endogènes sont des traits calculés directement à partir des mots eux-mêmes, de la séquence d'étiquettes, des arbres, ou encore d'un outil appris sur le corpus d'apprentissage comme un étiqueteur morpho-syntaxique. Dans le tableau 5.2, nous avons défini formellement les valeurs prises par les fonctions $f$ de $T$ utilisées à la fois par l'annotateur séquentiel et le réordonnanceur. Pour chacune de ces fonctions, $n$ est la position courante dans la phrase, $w(i)$ est le mot à la position $i$, et $t(i)$ est l'étiquette morpho-syntaxique de $w(i)$. Ces fonctions se basent sur deux types de données que nous décrivons par la suite : n-grammes de mots et n-grammes d'étiquettes morpho-syntaxiques. Nous avons également défini des fonctions spécifiques aux deux tâches.

| **n-grammes de mots** |
|---|
| $w(n+i), i \in \{-2,-1,0,1,2\}$ |
| $(w(n+i), w(n+i+1)), i \in \{-2,-1,0,1\}$ |
| **n-grammes d'étiquettes** |
| $t(n+i), i \in \{-2,-1,0,1,2\}$ |
| $(t(n+i), t(n+i+1)), i \in \{-2,-1,0,1\}$ |
| **couples de mots et d'étiquettes** |
| $(w(n+i), t(n+j)), (i,j) \in \{(1,0),(0,1),(-1,0),(0,-1)\}$ |

TAB. 5.2: Fonctions utilisées par les patrons endogènes qui sont communes à l'annotateur séquentiel et au réordonnanceur.

**n-grammes de mots**. Nous utilisons les bigrammes et unigrammes de mots pour apprendre les mots composés présents dans le corpus d'entrainement et pour extraire des indices lexicaux afin d'en découvrir de nouveaux. Par exemple, le bigramme *coup de* est souvent le préfixe d'unités polylexicales comme *coup de pied*, *coup de foudre* ou encore *coup de main*. Dans les tableaux 5.3 et 5.4, nous avons indiqué quelques traits obtenus par l'application du patron des bigrammes de mots sur les analyses d'exemple.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(x,n) = (w(n+i), w(n+i+1))$ | $n=0, i=0$ | $\rightarrow$ | (La, part, DET+B) |
| $\rightarrow T = (w(n+i), w(n+i+1), y_n)$ | $n=0, i=1$ | $\rightarrow$ | (part, de, DET+B) |
| avec $i \in \{-2,-1,0,1,2\}$ | $n=1, i=1$ | $\rightarrow$ | (de, marché, NC+B) |

TAB. 5.3: Quelques traits obtenus par l'application du patron des bigrammes de mots sur la séquence étiquetée d'exemple.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(p,n) = (w(n+i), w(n+i+1))$ | $n=0, i=0$ | $\rightarrow$ | (part, de, MWN, B) |
| $\rightarrow T = (w(n+i), w(n+i+1), label(m), pos(p,n))$ | $n=0, i=1$ | $\rightarrow$ | (de, marché, MWN, B) |
| avec $i \in \{-2,-1,0,1,2\}$ | $n=1, i=0$ | $\rightarrow$ | (de, marché, MWN, I) |

TAB. 5.4: Quelques traits obtenus par l'application du patron des bigrammes de mots sur l'arbre syntaxique d'exemple.

**n-grammes d'étiquettes morpho-syntaxiques**. Nous utilisons les unigrammes et bigrammes d'étiquettes morpho-syntaxiques dans le but d'apprendre des structures syntaxiques irrégulières souvent caractéristiques de présence de mots composés. Par exemple, la séquence *préposition, adverbe* associée à l'adverbe composé *depuis peu* est très inhabituelle. Nous avons aussi intégré des bigrammes mélangeant mots et étiquettes morpho-syntaxiques. Dans les tableaux 5.5 et 5.6, nous avons indiqué quelques traits obtenus par l'application du patron des bigrammes d'étiquettes sur les analyses d'exemple. On peut noter que, dans le cadre de la tâche d'étiquetage morpho-syntaxique d'un texte, il est nécessaire d'appliquer, au préalable, un étiqueteur sur

ce texte afin d'être capable d'extraire les n-grammes d'étiquettes[121]. Il est d'usage d'entraîner cet étiqueteur sur le même corpus d'apprentissage que l'étiqueteur-segmenteur.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(x,n) = (t(n+i), t(n+i+1))$ | $n=0, i=0$ | $\rightarrow$ | (DET+B, NC+B, DET+B) |
| $\rightarrow T = (t(n+i), t(n+i+1), y_n)$ | $n=0, i=1$ | $\rightarrow$ | (NC+B, NC+I, DET+B) |
| avec $i \in \{-2,-1,0,1,2\}$ | $n=1, i=1$ | $\rightarrow$ | (NC+I, NC+I, NC+B) |

TAB. 5.5: Quelques traits obtenus par l'application du patron des bigrammes d'étiquettes sur la séquence étiquetée d'exemple.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(p,n) = (w(n+i), w(n+i+1))$ | $n=0, i=0$ | $\rightarrow$ | (NC, P, MWN, B) |
| $\rightarrow T = (t(n+i), t(n+i+1), label(m), pos(p,n))$ | $n=0, i=1$ | $\rightarrow$ | (P, NC, MWN, B) |
| avec $i \in \{-2,-1,0,1,2\}$ | $n=1, i=0$ | $\rightarrow$ | (P, NC, MWN, I) |

TAB. 5.6: Quelques traits obtenus par l'application du patron des bigrammes d'étiquettes sur l'arbre syntaxique d'exemple.

**Traits spécifiques**. Chaque type de modèle intègre des traits particuliers car chacun s'attèle à des tâches différentes. Les fonctions correspondantes sont indiquées dans le tableau 5.7. On incorpore dans le CRF des traits spécifiques pour gérer les mots inconnus et les mots spéciaux (nombres, traits d'union,...) : le mot en lettres minuscules, les préfixes et suffixes de taille 1 à 4, l'information si un mot commence par une majuscule, s'il contient un chiffre, si c'est un trait d'union. Nous ajoutons en plus les bigrammes des étiquettes de sortie. Les modèles liés au réordonnanceur intègrent des traits associés aux noeuds de type MWE, dont les valeurs sont les formes lexicales des mots composés correspondants. Par exemple, dans l'arbre d'exemple, l'application de ce patron sur le noeud étiqueté par `MWN` produit un trait composé de sa forme lexicale, *part_de_marché*.

### 5.4.2 Traits exogènes

Les traits exogènes sont des traits calculés, totalement ou en partie, à partir de données externes. Dans notre cas, il s'agit de nos ressources lexicales. Ces ressources peuvent être utiles pour découvrir de nouvelles expressions. Généralement, les mots composés, en particulier les noms, suivent un schéma régulier, ce qui les rend très difficilement repérables à partir de traits endogènes uniquement. Nos ressources sont appliquées au corpus à l'aide d'une analyse lexicale qui produit, pour chaque phrase, un automate fini qui représente l'ensemble des analyses possibles. Les traits exogènes sont calculés à partir de cet automate. Dans le tableau 5.8, nous avons définis formellement les valeurs prises par les fonctions $f$ de $T$ utilisées à la fois par

[121]Le réordonnanceur dispose déjà de ces informations dans les analyses candidates.

| **Annotation séquentielle** |
|---|
| Forme en minuscule de $w(n)$ |
| Préfixe de $w(n) = P$ avec $\|P\| < 5$ |
| Suffixe de $w(n) = S$ avec $\|S\| < 5$ |
| $w(n)$ contient un tiret |
| $w(n)$ contient un chiffre |
| $w(n)$ commence par une majuscule |
| $w(n)$ est tout en majuscule |
| $w(n)$ commence par une majuscule et est en début de phrase |
| $t(n-1)/t(n)$ |
| **Réordonnancement** |
| Forme lexicale($n$) |

TAB. 5.7: Fonctions utilisées par les patrons endogènes qui sont spécifiques aux deux tâches.

l'annotateur séquentiel et le réordonnanceur. Ces fonctions se basent sur deux types de données que nous décrivons par la suite : des traits basés sur un lexique, et des traits basés sur les collocations.

| **lexique** |
|---|
| ac($n$) |
| (mwt($n$),mwpos($n$)) |
| (mws($n$),mwpos($n$)) |
| **collocations** |
| c($n$)/cs($n$)/cpos($n$) |

TAB. 5.8: Fonctions utilisées par les patrons exogènes qui sont communes à l'annotateur séquentiel et au réordonnanceur.

**Les traits basés sur un lexique**. Nous associons à chaque mot l'ensemble des étiquettes morpho-syntaxiques trouvées dans notre lexique morphologique externe. Cet ensemble forme une classe d'ambiguité $ac$. Si un mot appartient potentiellement à une unité polylexicale dans son contexte d'occurrence, l'étiquette correspondante au mot composé est aussi intégrée à la classe d'ambiguité. Par exemple, le mot *de* dans le contexte *part de marché* est associé à la classe `DET_PREP_N+I`. En effet, le mot simple *de* est soit un déterminant (`DET`) soit une préposition (`PREP`) dans le Dela. Par ailleurs, il se trouve dans une position interne (`I`) du nom *part de marché*. Ce trait est directement calculé à partir de l'automate généré par l'analyse lexicale, montrée dans la figure 5.10. Nous utilisons également cet automate afin de réaliser une segmentation lexicale préliminaire en appliquant un algorithme du plus court chemin pour favoriser les analyses polylexicales. Cette segmentation préliminaire est source d'indices pour la segmentation finale, donc source de nouveaux traits. On peut associer à tout mot appartenant à un segment composé différentes propriétés : l'étiquette morpho-syntaxique $mwt$ du segment, sa structure interne $mws$, ainsi que sa position relative $mwpos$ dans le segment (`B` ou `I`). Par exemple, le mot *de* dans le contexte du mot composé *eau de vie*, présent dans nos ressources,

sera associé au trait NC+NPN+I, composé de la catégorie grammaticale NC (nom commun), la structure interne NPN (nom, préposition, nom), et la position relative I car il est en 2ème position du segment composé.

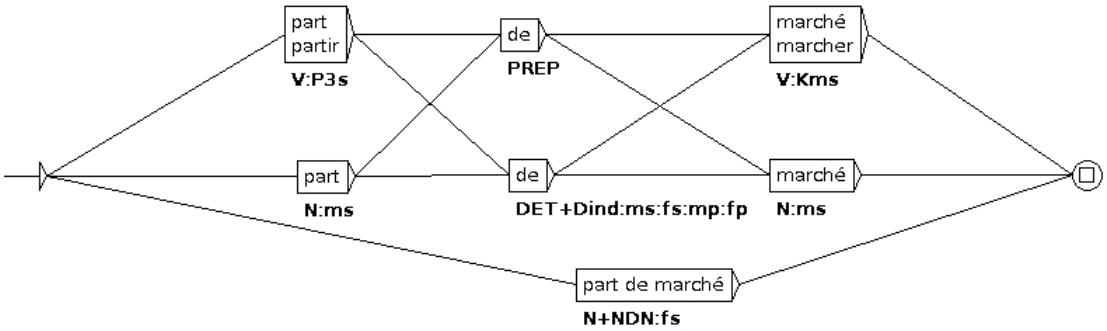



FIG. 5.10: Analyse lexicale générée par Unitex pour la séquence de mots *part de marché*.

**Traits basés sur les collocations**. Dans notre ressource de collocations, chaque candidat du FTB-UC est accompagné de sa structure syntaxique interne et de son score d'association (log-vraisemblance). Nous avons divisé ces candidats en deux classes : ceux qui ont un score supérieur à un certain seuil et les autres. Ainsi, tout mot du corpus peut être associé à un certain nombre de propriétés s'il appartient à une collocation candidate : la classe de la collocation $c$ ainsi que sa structure interne $cs$, la position relative $cpos$ du mot dans la collocation (B ou I). Nous avons manuellement fixé le seuil à une valeur de 150 après une phase de réglage sur le corpus de développement.

## 5.5 Évaluations de l'étiqueteur-segmenteur

Dans cette section, nous indiquons les résultats des expériences qui ont porté sur la stratégie d'identification et d'étiquetage des MWEs par un étiqueteur-segmenteur (Constant *et al.*, 2011; Constant & Sigogne, 2011). Cette méthode, décrite dans la section 5.3.1, consiste à faire varier l'ensemble des traits utilisés dans le modèle CRF : les notations endo et exo correspondent respectivement aux traits endogènes et exogènes (section 5.4). Ces deux ensembles peuvent être utilisés séparemment ou en combinaison (all). Le but est de déterminer le comportement et les performances de l'annotateur séquentiel en fonction de ces divers ensembles de patrons, et selon que le texte soit pré-segmenté (segmentation du corpus) ou non. Pour les expériences basées sur un texte non pré-segmenté, nous avons défini, en plus, deux autres ensembles de patrons : endo est composé de la totalité des patrons endogènes, plus les patrons exogènes basés sur un lexique. coll est, quant à lui, composé de la totalité des patrons endogènes, plus les patrons exogènes basés sur les collocations. Ces divers ensembles ont un tronc commun de 16 patrons exploitant des propriétés classiques des unités (forme lexicale, préfixes, suffixes, commence par une majuscule,...). À titre de comparaison, ces 16 patrons sont utilisés seuls dans l'expérience dénotée tronc. Pour les patrons endogènes basés sur les n-grammes d'étiquettes, on peut rappeler qu'il est nécessaire d'appliquer au préalable un étiqueteur sur le texte. Aussi,

pour chaque configuration concernée, nous avons réutilisé l'étiqueteur-segmenteur avec une configuration identique (moins les patrons liés aux n-grammes d'étiquettes).
Dans la section suivante, nous évaluons l'étiquetage morpho-syntaxique sur une segmentation multi-mots parfaite. Pour l'évaluation globale, nous avons précision = rappel = $F_1$. En effet, tous les mots ayant une unique étiquette, une erreur de précision sur une classe C1 correspond à une erreur de rappel sur une classe C2 et vice versa. Dans la section 5.5.2, nous évaluons la tâche d'étiquetage intégrant la reconnaissance des unités multi-mots. Dans ce cas précis, l'évaluation correspond à un score $F_1$ sur les segments lexicaux.

### 5.5.1 Évaluation de l'étiquetage avec segmentation parfaite

Pour cette expérience, nous avons comparé les résultats avec d'autres outils d'étiquetage que nous avons décrits dans le chapitre I.3. Nous avons évalué TreeTagger (Schmid, 1995) basé sur des arbres de décision probabilistes, ainsi que TnT (Brants, 2000) implantant un modèle de Markov caché d'ordre 2. Nous avons également comparé notre outil avec deux étiqueteurs de type discriminant : SVMTool (Giménez & Màrquez, 2004) basé sur les SVM et utilisant des traits standards, ainsi que MElt (Denis & Sagot, 2009) basé sur un modèle ME incorporant en plus des traits dépendants de la langue issus de lexiques externes[122]. Le système appelé MElt$_{base}$ correspond à ce même étiqueteur sans les traits spécifiques aux lexiques. Les scores obtenus par les différents systèmes sont donnés en pourcentage dans la table 5.9. Ces scores ont été calculés à partir des corpus de développement (colonne `DEV`) et d'évaluation (colonne `EVAL`). Afin d'établir la significativité des résultats, nous avons également employé la technique de validation croisée (colonne `VALID`), où le paramètre $n$ est fixé à 5[123].

| | DEV | EVAL | VALID |
|---|---|---|---|
| TreeTagger | 95.97 | 96.21 | 96.06 |
| TnT | 96.65 | 97.10 | 96.78 |
| SVMTool | 97.18 | 97.34 | 97.10 |
| MElt$_{base}$ | 97.05 | 97.32 | 97.00 |
| MElt | 97.52 | 97.78 | 97.75 |
| tronc | 97.12 | 97.25 | 97.20 |
| endo | **97.62** | **97.95** | **97.90** |
| exo | 97.56 | 97.76 | 97.79 |
| all | 97.61 | 97.94 | 97.87 |

TAB. 5.9: Comparaison de systèmes d'étiquetage pour le français.

Tout d'abord, on peut remarquer que les performances de base de notre étiqueteur (`tronc`) sont du même niveau que celles des autres étiqueteurs discriminatifs (MElt$_{base}$ et SVMTool). Avec des gains absolus supérieurs à +0.5, le système `endo` montre que des patrons pertinents

[122]Le lexique utilisé est le L*efff*.

[123]Ce paramètre détermine à la fois le nombre d'itérations effectuées mais aussi la partition du corpus. Dans le cas présent, le corpus d'apprentissage est formé, à chaque itération, des 4/5 du corpus, et le corpus d'évaluation de la partie restante.

permettent d'améliorer significativement les performances générales. De même, des traits basés sur des lexiques externes (`exo`) semblent être d'une aide utile car les gains sont similaires à `endo`. Cela confirme l'intérêt d'exploiter de telles ressources, comme l'ont montré auparavant (Denis & Sagot, 2009, 2010) pour MElt. En revanche, la combinaison des traits provenant des deux ensembles (`all`) ne permet pas de gagner des points supplémentaires.
Comme énoncé dans le chapitre I.3, il est en général intéressant d'observer les performances des systèmes d'étiquetage lorsque la quantité de données fournie à l'apprentissage varie. Il s'agit principalement de déterminer si un système est capable de s'adapter à un corpus de petite taille. Pour ce faire, nous avons évalué nos systèmes sur les corpus de développement et d'évaluation du FTB-UC, et ceci en faisant varier la taille du corpus d'apprentissage selon cinq paliers (10, 20, 40, 80 et 100%). Les résultats globaux sont illustrés graphiquement dans la figure 5.11, et ceci pour le corpus d'évaluation uniquement[124]. Les courbes bleue, orange, noire et violette correspondent respectivement aux systèmes `tronc`, `endo`, `exo` et `all`. À titre de comparaison, nous avons également montré la courbe d'apprentissage de MElt (courbe verte).

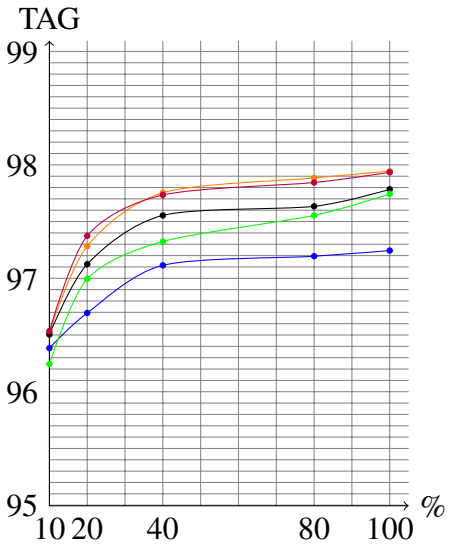



FIG. 5.11: Courbes d'apprentissage des systèmes d'étiquetage obtenues sur le corpus d'évaluation. Le code couleur des courbes est le suivant : `tronc`=bleue, `endo`=orange, `exo`=noire, `all`=violette et `MElt`=verte.

Alors qu'à 10% du corpus d'apprentissage l'ensemble des étiqueteurs font jeu égal, les différences sont plus nettes aux paliers suivants. En effet, comme observé lors de l'expérience précédente, les systèmes `endo` et `all` ont des courbes situées strictement au dessus des autres systèmes. Légèrement en dessous se trouvent celles de `exo` et MElt. Quant à notre étiqueteur de base, la pente de départ de la courbe est plus faible que les autres, ce qui conduit à des performances logiquement moindres. On peut donc en conclure que, dans le cadre d'une segmentation parfaite, l'utilisation de patrons consacrés aux mots composés permet de : (a) améliorer les performances générales d'étiquetage, que ce soit avec ou sans l'utilisation de

[124]Les mêmes phénomènes apparaissent sur le corpus de développement

lexiques externes. (b) conserver des performances élevées malgré un corpus d'apprentissage de taille moindre.

### 5.5.2 Évaluation de l'étiquetage avec identification des unités multi-mots

Pour évaluer la tâche d'étiquetage intégrant la reconnaissance des unités multi-mots, nous avons entraîné cinq modèles CRF sur le corpus d'apprentissage du FTB-UC : `tronc`, `endo`, `lex`, `coll` et `all`. On peut rappeler que les unités multi-mots de ce corpus ont été décomposées en séquences d'unités minimales étiquetées dans la représentation de type BIO (section 5.3.1). Ces cinq systèmes ont été comparés avec SVMTool entraîné sur le même corpus. Pour chaque étiqueteur-segmenteur, nous avons calculé le score $F_1$ sur le corpus brut d'évaluation, également décomposé en unités minimales. La précision et le rappel sont calculés par rapport aux segments lexicaux trouvés et non aux unités minimales simples. Les résultats sont synthétisés dans le tableau 5.12. La colonne `MWSEG` indique le score $F_1$ de la segmentation qui ne prend en compte que les limites des segments. La colonne $F_1$`(MWEs)` prend aussi en compte la catégorie grammaticale. Les scores inscris sous la colonne `TAG` indiquent la qualité de l'étiquetage des unités minimales avec les catégories intégrant `B` et `I`, et non la qualité de l'identification des unités multi-mots.
Tout d'abord, le passage d'une segmentation parfaite à une segmentation automatique induit une baisse significative des scores d'étiquetage globaux (colonne `TAG`). En effet, on constate une chute d'au minimum 2.7 points pour l'ensemble des systèmes et sur les trois corpus évalués. L'ajout de patrons endogènes améliore significativement les résultats, avec des gains absolus d'environ +1 pour l'étiquetage global et +10 pour la segmentation en MWEs (validation croisée). Bien que l'ajout de traits exogènes tirés des lexiques (`lex`) a relativement peu d'impact sur l'étiquetage global, on peut remarquer que cet impact se situe au niveau de la segmentation, avec des gains absolus d'environ +2 pour la validation croisée. En revanche, l'exploitation des patrons exogènes basés sur les collocations (`coll`) conduit à obtenir des gains similaires aux patrons endogènes, lorsqu'ils sont utilisés seuls, ce qui traduit une certaine incompatibilité entre les informations tirées de ces lexiques particuliers, et celles du corpus. Cette remarque peut expliquer pourquoi la combinaison des deux ensembles de traits, endogènes et exogènes (`all`), produit des scores similaires au système `lex`. Ces diverses observations montrent malgré tout que les gains sont obtenus majoritairement avec des traits acquis entièrement automatiquement à partir du corpus, en opposition aux traits provenant de lexiques externes.
De même que dans la section précédente, nous avons calculé les performances des systèmes d'étiquetage lorsque la quantité de données fournie à l'apprentissage varie. Le protocole d'expérimentation est identique. Nous avons donc évalué nos systèmes sur les corpus de développement et d'évaluation du FTB-UC, et ceci en faisant varier la taille du corpus d'apprentissage selon cinq paliers (10, 20, 40, 80 et 100%). Les résultats pour l'étiquetage global (`TAG`) et la segmentation en MWEs ($F_1$`(MWEs)`) sont illustrés graphiquement dans la figure 5.13, et ceci pour le corpus d'évaluation uniquement[125]. Les courbes bleue, orange, noire et violette correspondent respectivement aux systèmes `tronc`, `endo`, `lex` et `all`. À titre de comparaison, nous avons également montré la courbe d'apprentissage de SVMTool (courbe verte).

[125]Les mêmes phénomènes apparaissent sur le corpus de développement

| | TAG | $F_1$(MWEs) | MWSEG |
|---|---|---|---|
| SVMTool | 94.07 | 69.24 | 72.08 |
| tronc | 94.36 | 66.09 | 68.58 |
| endo | 95.63 | 76.79 | 78.76 |
| lex | **95.94** | **80.10** | **82.57** |
| coll | 95.66 | 77.31 | 79.37 |
| all | 95.91 | 80.09 | 82.27 |

(a) Corpus de développement

| | TAG | $F_1$(MWEs) | MWSEG |
|---|---|---|---|
| SVMTool | 94.93 | 72.73 | 75.70 |
| tronc | 94.65 | 66.70 | 69.59 |
| endo | 96.06 | 76.05 | 78.71 |
| lex | **96.25** | **79.33** | **81.80** |
| coll | 96.06 | 76.30 | 78.96 |
| all | 96.16 | 78.08 | 81.05 |

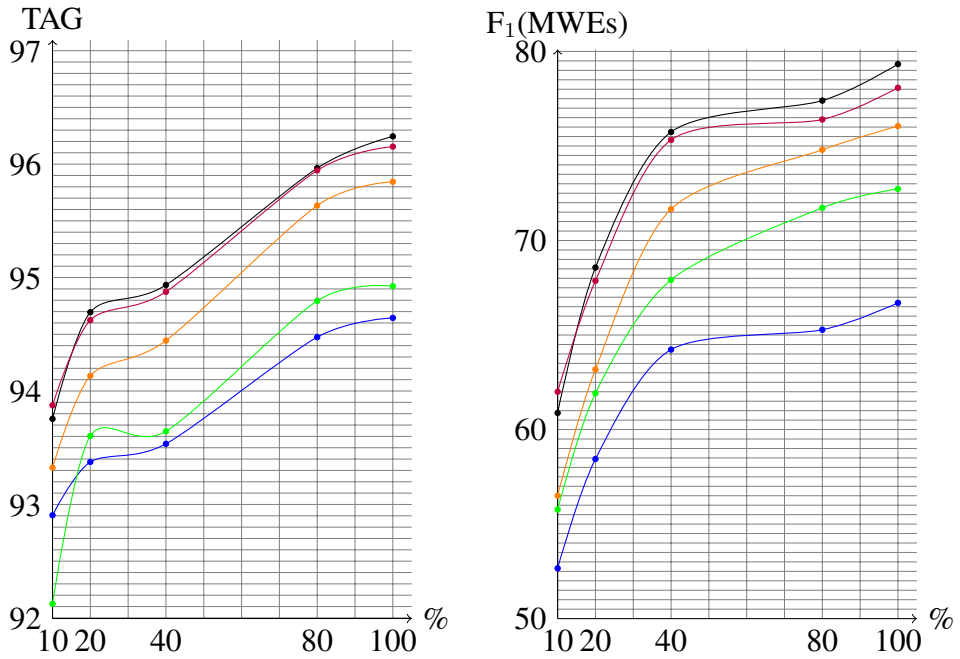
(b) Corpus d'évaluation

| | TAG | $F_1$(MWEs) | MWSEG |
|---|---|---|---|
| SVMTool | 94.67 | 71.03 | 74.51 |
| tronc | 94.50 | 66.75 | 69.01 |
| endo | 95.48 | 76.86 | 79.10 |
| lex | **95.60** | 78.92 | 81.21 |
| coll | 95.43 | 76.99 | 79.25 |
| all | **95.60** | **78.95** | **81.25** |

(c) Validation croisée

FIG. 5.12: Résultats obtenus sur le FTB-UC pour la stratégie d'apprentissage simultané de l'étiquetage et de la segmentation.

Des remarques similaires peuvent être tirées des deux graphiques. D'une part, le système `tronc` est associé à des courbes dont la pente est plutôt faible, ce qui n'est pas le cas des autres systèmes. D'autre part, les trois systèmes `lex`, `all` et `endo` ont des courbes parallèles, bien que, pour ce dernier, les courbes soient légèrement en dessous des deux autres. Ce que nous avons observé précédemment sur le corpus d'évaluation semble donc être valable dans le cas présent. En effet, quelle que soit la taille du corpus, l'introduction de traits tirés de lexiques externes augmente légèrement les résultats, alors que l'ajout de traits liés aux collocations est sans effet.
En conclusion, le principe d'un étiqueteur-segmenteur basé sur des traits consacrés aux mots composés permet d'améliorer significativement les performances d'étiquetage et d'identification des MWEs, et ceci quelle que soit la taille du corpus d'apprentissage. Nous discutons plus en détail des résultats dans la section 5.7, où nous proposons une comparaison entre systèmes, étiqueteur-segmenteurs et analyseurs syntaxiques, basés sur les différentes stratégies énoncées.




(a) Étiquetage global (`TAG`). (b) Identification des MWEs ($F_1$ (`MWEs`)).

FIG. 5.13: Courbes d'apprentissage des systèmes d'étiquetage obtenues sur le corpus d'évaluation. Le code couleur des courbes est le suivant : `tronc`=bleue, `endo`=orange, `exo`=noire, `all`=violette et `SVMTool`=verte.

Dans la suite de ce mémoire, nous ferons appel au nom *LGTagger* pour faire référence à notre étiqueteur-segmenteur[126].

## 5.6 Évaluation de l'analyse syntaxique avec identification des MWEs

Dans cette section, nous indiquons les résultats d'expériences qui ont porté sur deux stratégies d'exploitation de données sur les MWEs. La première d'entre elles, que nous appelerons à présent *pre*, consiste à effectuer une pré-identification des mots composés d'un texte brut par le reconnaisseur discriminatif décrit dans la section précédente[127]. Puis, un analyseur syntaxique est appliqué de manière classique sur ce texte dont la segmentation est considérée comme déjà faite. La deuxième stratégie, appelée *post*, consiste à effectuer une analyse syntaxique incorporant l'identification des mots composés, suivie de l'application d'un réordonnanceur intégrant

[126]LGTagger est disponible librement à l'adresse `http://igm.univ-mlv.fr/~mconstan/research/software/`. De plus, le fichier README inclus dans la distribution actuelle de l'outil est disponible en intégralité dans l'annexe 2.

[127]On peut rappeler que l'étiquetage est supprimé afin de ne conserver que la segmentation en unités simples et composées.

des traits consacrés aux MWEs.
L'ensemble des expériences décrites ci-dessous ont été réalisées avec trois analyseurs en constituants, Brown, BKY[128] et BKY$_p$[129]. Le premier est basé sur une stratégie lexicalisée, alors que les deux autres sont dits non-lexicalisés. Pour se prémunir contre une certaine instabilité dans les résultats, nous avons appris, pour BKY, 3 grammaires avec différentes graines aléatoires (1, 4 et 8). Les résultats indiqués sont la moyenne des résultats de l'application indépendante des trois grammaires. Ces mêmes grammaires sont utilisées par BKY$_p$ dans le cadre du produit de grammaires. Pour la deuxième stratégie, un réordonnanceur du français est nécessaire. Pour ce faire, nous avons utilisé notre implémentation du réordonnanceur de Brown, décrite dans la section I.2.4.6.
Les expériences sont évaluées à l'aide de plusieurs mesures classiques : $F_1$, UAS et LA. De plus, seuls les scores obtenus sur le corpus d'évaluation du FTB-UC sont indiqués dans les tableaux, car nous observons des phénomènes identiques sur le corpus de développement. Pour que les résultats des deux stratégies *pre* et *post* soient directement comparables, nous avons automatiquement converti les analyses générées par *pre* avec mots composés fusionnés en leurs analyses équivalentes avec des noeuds non-terminaux spécifiques pour les unités polylexicales. De cette manière, nous sommes capable d'évaluer l'identification des mots composés par le score $F_1$(MWEs) associé aux noeuds de type MWE. De plus, la significativité statistique entre deux expériences d'identification de mots composés est établie par le test de McNemar (Gillick & Cox, 1989). Les résultats de deux expériences sont considérés comme statistiquement significatifs si la valeur calculée lors du test est inférieure à 0.01.

### 5.6.1 Analyse syntaxique avec pré-identification des mots composés

Nous avons tout d'abord évalué la stratégie *pre* prenant en entrée un texte segmenté par notre reconnaisseur CRF de mots composés (sans les étiquettes). Les segmentations évaluées ont été réalisées avec les mêmes systèmes que précédemment, à savoir `tronc`, `endo`, `lex`, `coll` et `all`. Les résultats pour les trois analyseurs obtenus sur le corpus d'évaluation sont synthétisés dans les tableaux de la figure 5.14. À titre de comparaison, nous avons indiqué les scores des trois analyseurs lorsqu'ils sont entraînés et évalués sur le corpus dont les unités polylexicales sont annotées par des noeuds non-terminaux spécifiques `base`. L'expérience `référence` (généralement notée `gold` en anglais) signifie, dans le cas présent, que la segmentation lexicale avant analyse syntaxique est parfaite. De plus, le symbole $^*$ indique que le résultat n'est pas significatif par rapport à `base`. Tout d'abord, nous pouvons observer une forte chute des résultats lorsque le corpus n'est pas pré-segmenté (`base`/`référence`), et ceci quel que soit l'analyseur évalué : les différents scores subissent une baisse de 1 à 3 points, alors que l'identification des MWEs chute spectaculairement d'environ 20 points. Cette première remarque renforce l'intérêt d'utiliser des stratégies permettant de traiter les MWEs dans des conditions réelles d'utilisation des analyseurs syntaxiques (i.e sans pré-segmentation parfaite du texte). Cependant, les résultats du système `tronc` montrent qu'une stratégie lambda de segmentation du texte ne permet pas forcément d'améliorer la qualité des analyses produites. Les résultats

[128]Version améliorée pour le traitement des mots rares et inconnus pour l'analyse de textes en langue française.
[129]Version basée sur le produit de grammaires.

observés sont même moins bons que ceux de `base`. En revanche, l'introduction de traits endogènes consacrés aux mots composés (`endo`) améliore significativement l'ensemble des scores comparativement à `base`, et notamment $F_1$(MWEs) avec des gains absolus allant d'environ +2 à +5 selon l'analyseur évalué. L'ajout de traits exogènes tirés des lexiques (`lex`) n'a d'effets positifs que sur la reconnaissance des MWEs, avec des gains absolus significatifs d'environ +2 par rapport à `endo`. Et enfin, les traits exogènes basés sur les collocations ne permettent pas aux analyseurs d'améliorer les scores (`coll` et `all`). Les différentes observations faites à propos des MWEs sont similaires à celles constatées lors des expériences sur l'étiquetage morpho-syntaxique. Cela traduit donc une certaine constance dans les résultats selon le système de segmentation du texte utilisé, et que cette segmentation a, par la suite, un impact très important sur les performances d'un analyseur syntaxique lambda. Dans le cadre de la stratégie *pre*, il est donc nécessaire de fournir une segmentation de bonne facture, ce qui est partiellement résolu par l'ajout de traits basés sur les MWEs.

| | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| référence | 84.40 | 85.12 | 81.22 |
| base | 82.30 | 83.46 | 79.01 |
| tronc | 81.76* | 82.72* | 78.86* |
| endo | 82.93 | **83.88** | **80.03** |
| lex | **82.99** | 83.83 | 79.96 |
| coll | 82.90 | 83.81 | 80.00 |
| all | 82.93 | 83.85 | 79.90 |

(a) Score $F_1$

| | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| référence | 93.84 | 94.15 | 90.28 |
| base | 92.38 | 92.97 | 89.65 |
| tronc | 92.64 | 92.96 | 89.16 |
| endo | 93.12 | 93.46 | **89.60** |
| lex | **93.15** | **93.47** | 89.55 |
| coll | 93.11 | 93.43 | 89.59 |
| all | 93.10 | 93.42 | 89.53 |

(b) Score LA

| | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| référence | 91.10 | 91.19 | 89.34 |
| base | 87.47 | 88.15 | 85.76 |
| tronc | 87.28* | 87.77* | 85.78* |
| endo | 88.37 | 88.90 | 86.68 |
| lex | **88.57** | **89.11** | **86.83** |
| coll | 88.36 | 88.88 | 86.69 |
| all | 88.45 | 88.98 | 86.75 |

(c) Score UAS

| | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| référence | 91.78 | 91.85 | 89.08 |
| base | 72.66 | 73.55 | 70.46 |
| tronc | 66.57* | 66.79* | 65.80* |
| endo | 75.48 | 75.43 | 75.43 |
| lex | **77.80** | **77.75** | **77.45** |
| coll | 75.68 | 75.01 | 75.38 |
| all | 77.14 | 77.10 | 76.70 |

(d) Score $F_1$(MWEs)

FIG. 5.14: Intégration des mots composés dans l'analyse syntaxique par identification préalable : résultats obtenus sur le corpus d'évaluation du FTB-UC.

### 5.6.2 Analyse syntaxique avec réordonnancement

Nous avons ensuite évalué la stratégie *post* consistant à intégrer un réordonnanceur après l'application d'un analyseur lambda sur le corpus dont les MWEs ont été décomposées en unités minimales. Dans un premier temps, nous avons appliqué un modèle incorporant uniquement

les traits consacrés aux mots composés (section 5.4). Nous avons ensuite comparé avec un modèle intégrant aussi les traits généraux grâce aux 11 patrons décrits dans la section I.2.4.6 : *Rule*, *Word*, *Heavy*, *HeadTree*, *Bigrams*, *Trigrams*, *Edges*, *WordEdges*, *Heads*, *WProj*, *NGramTree* et *Score*. Les notations `std` et `tous` correspondent respectivement aux traits généraux et à tous les traits décrits. Pour chaque expérience, le réordonnanceur prend, en entrée, les 50 meilleures analyses de BKY ou de Brown. $\text{BKY}_p$ n'a pas été exploité pour cette expérience car nous rappelons que la version actuelle ne permet pas de générer la liste des $n$ analyses candidates d'une phrase. Les résultats obtenus sur le corpus d'évaluation sont synthétisés dans la table 5.15. L'expérience notée `référence` signifie que la segmentation lexicale avant analyse syntaxique est parfaite (analyseurs et réordonnanceur). Les symboles $^*$ et $^+$ indiquent que le résultat est significatif respectivement par rapport à `base` et à `std`. On peut noter que les catégories grammaticales des composants internes du corpus ont été déterminées à l'aide de LGTagger appris sur notre corpus d'apprentissage, et en se basant uniquement sur les traits endogènes (i.e. sans intégrer de ressources lexicales externes).

| | BKY | Brown |
|---|---|---|
| référence | 85.49 | 82.1 |
| base | 82.3 | 79.01 |
| std | **83.24**$^*$ | 80.16$^*$ |
| endo | 82.54$^*$ | 79.21 |
| lex | 82.67$^*$ | 79.32$^*$ |
| coll | 82.45 | 79.2 |
| all | 82.6$^*$ | 79.35$^*$ |
| tous | 83.23$^*$ | **80.38**$^*$ |

(a) Score $F_1$

| | BKY | Brown |
|---|---|---|
| référence | 94.5 | 90.73 |
| base | 92.38 | 89.65 |
| std | 93.36 | 91.06 |
| endo | 92.98 | 90.83 |
| lex | 93.02 | 90.87 |
| coll | 92.94 | 90.82 |
| all | 92.99 | 90.85 |
| tous | **93.39** | **91.14** |

(b) Score LA

| | BKY | Brown |
|---|---|---|
| référence | 92.37 | 90.12 |
| base | 87.47 | 85.76 |
| std | 87.9$^*$ | 86.07 |
| endo | 87.8 | 86.04 |
| lex | 87.94$^*$ | 86.14$^*$ |
| coll | 87.4 | 85.99 |
| all | 87.88$^*$ | 86.17$^*$ |
| tous | **87.99**$^*$ | **86.22**$^*$ |

(c) Score UAS

| | BKY | Brown |
|---|---|---|
| référence | 90.37 | 88.68 |
| base | 72.66 | 70.46 |
| std | 73.9$^*$ | 71.46 |
| endo | 75.86$^{*+}$ | 71.86$^*$ |
| lex | **77.19**$^{*+}$ | **73.29**$^{*+}$ |
| coll | 73.5 | 71.65 |
| all | 77.05$^{*+}$ | 73.12$^{*+}$ |
| tous | 73.54 | 73.07$^{*+}$ |

(d) Score $F_1$(MWEs)

FIG. 5.15: Intégration des mots composés dans l'analyse syntaxique en tant que traits d'un réordonnanceur discriminatif : résultats obtenus sur le corpus d'évaluation du FTB-UC.

Pour ce qui est de la reconnaissance des MWEs, des disparités importantes apparaissent. Avec `endo`, les gains sont de +3.2 pour BKY et +1.4 pour Brown. L'ajout de traits basés sur les collocations fait chuter ces gains, ce qui traduit la difficulté d'intégration de telles ressources. Quant aux traits tirés des lexiques (`lex`), ils permettent de gagner +1.4 supplémentaire pour

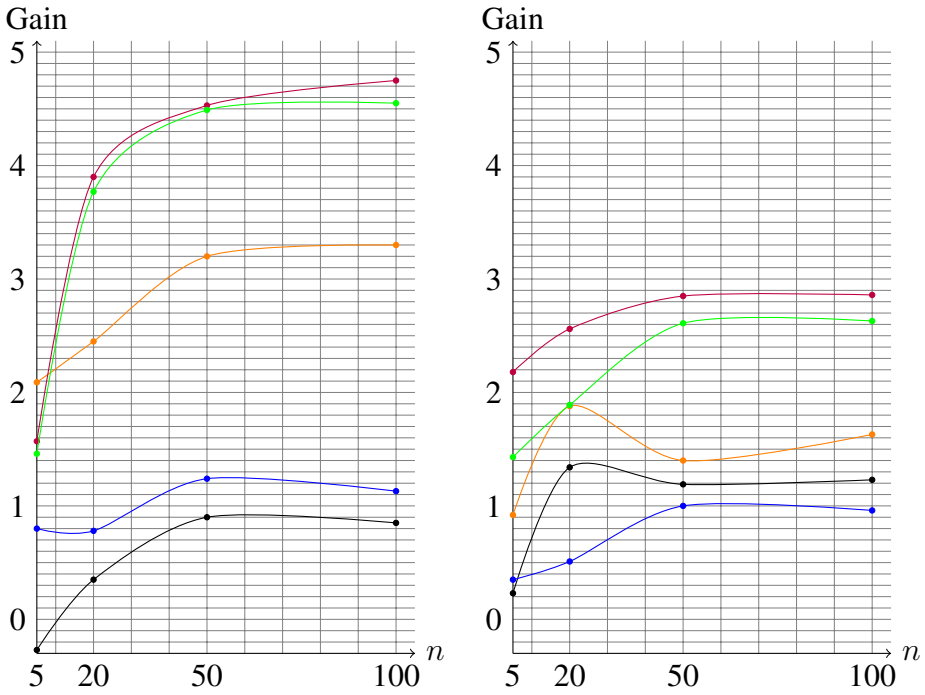
les deux analyseurs. La combinaison de tous les traits consacrés aux MWEs fait jeu égal avec `lex`, principalement à cause de l'impact négatif des collocations. Par ailleurs, les traits généraux seuls (`std`) sont plus efficaces que les traits concernant les mots composés pour ce qui concerne le parenthésage et le score LA. Comparativement à `lex`, les gains absolus respectifs sont d'environ +0.6 et +0.4. Par contre, ils font jeu égal pour le score UAS et dégradent fortement la reconnaissance des mots composés (-3.30 pour BKY et -1.80 pour Brown). Le mélange des deux types de traits (`tous`) n'est pas concluant car, pour l'ensemble des scores, aucune variation significative de l'écart par rapport aux traits généraux n'est observée. Cela montre que les traits consacrés aux MWEs ne sont pas associés à des poids assez discriminants pour influer sur les décisions du réordonnanceur. Cette faiblesse est renforcée par le fait qu'ils sont au final très minoritaires par rapport aux traits standards. Par exemple, avec $n$ fixé à 50, seuls 4% sont des traits concernant les MWEs. Et enfin, il faut noter que les patrons standards extraient, directement ou indirectement, de nombreuses informations lexicales et syntaxiques présentes dans les sous-arbres liés aux MWEs. Ces résultats et ces observations montrent qu'il est nécessaire de trouver un autre moyen de combiner ces deux types de traits.
Dans le tableau 5.10, nous avons indiqué les scores *Oracle* $F_1$ qui correspondent aux scores que l'on pourrait obtenir en sélectionnant, pour chaque phrase du corpus, l'analyse candidate la plus proche de la référence. Les scores $F_1$(MWEs) correspondants sont indiqués entre parenthèses. En ce qui concerne le score $F_1$, plus la valeur de $n$ augmente, plus ce score est élevé et ce pour les deux analyseurs. Les meilleurs scores potentiels sont atteints avec $n$ fixé à 100 : +5.16 pour BKY et +7.65 pour Brown par rapport à $n$=1. En revanche, le comportement du score $F_1$(MWEs) est quelque peu différent. À partir de $n$=20, ce score ne semble plus évoluer pour BKY et stagne aux alentours de 78%, soit un gain absolu potentiel d'environ +5.3. Par contre, pour Brown, l'augmentation continue, et le score obtenu avec $n$=100 dépasse même celui de BKY.

| | 1 | 5 | 20 | 50 | 100 |
|---|---|---|---|---|---|
| BKY | 82.30 (72.66) | 84.96 (75.89) | 86.75 (77.83) | 87.18 (77.73) | **87.46** (**77.97**) |
| Brown | 79.01 (70.46) | 82.51 (73.65) | 85.13 (76.02) | 85.97 (77.06) | **86.66** (**78.80**) |

TAB. 5.10: Évolution des scores *Oracle* $F_1$ selon le paramètre $n$ sur le corpus d'évaluation du FTB-UC. Les scores $F_1$(MWEs) correspondants sont indiqués entre paranthèses.

Dans la figure 5.16, les graphiques illustrent l'évolution des gains obtenus pour le score $F_1$(MWEs) par les différents systèmes (par rapport au score de base), et ceci selon 4 valeurs possibles du paramètre $n$ du réordonnanceur : 5, 20, 50 et 100. Les courbes orange, noire, violette, verte et bleue correspondent respectivement à `endo`, `coll`, `lex`, `all` et `std`. Tout d'abord, on peut constater que les scores sont de plus en plus élevés à mesure que la valeur de $n$ augmente. Pour les deux analyseurs, une majorité de gains sont maximisés avec $n$ fixé à 50. Ce point précis des courbes correspond géométriquement à la dérivée nulle. On peut observer que la pente de la courbe `std` est bien moins forte que d'autres comme `lex` et `all`. Cela montre qu'avec plus ou moins d'analyses candidates certains traits consacrés aux MWEs permettent au réordonnanceur de favoriser celles dont la reconnaissance des MWEs est qualitativement meilleure. On peut voir malgré tout que les courbes associées à l'analyseur Brown atteignent le point de la dérivée nulle plus rapidement que celles de BKY, ce qui traduit l'impact moins important de

nos patrons sur des analyses produites par un analyseur lexicalisé.




(a) Analyseur BKY. (b) Analyseur Brown.

FIG. 5.16: Évolution des gains pour le score $F_1$(MWEs) selon le paramètre $n$ sur le corpus d'évaluation du FTB-UC. Le code couleur des courbes est le suivant : `endo`=orange, `coll`=noire, `lex`=violette, `all`=verte et `std`=bleue.

Nous avons indiqué dans le tableau 5.11, la quantité de traits générés par les divers ensembles de patrons en faisant varier la valeur de $n$. Comme énoncé précédemment, le ratio traits MWEs/standards est d'environ 4%, et ce quelle que soit la valeur du paramètre. Bien que ce ratio soit très faible, nous pouvons rappeler, d'après les résultats précédents, que les traits spécifiques aux MWEs rivalisent avec les traits généraux en terme de score UAS et de reconnaissance des MWEs.

| | 5 | | 20 | | 50 | | 100 | |
|---|---|---|---|---|---|---|---|---|
| | BKY | Brown | BKY | Brown | BKY | Brown | BKY | Brown |
| endo | 51374 | 36023 | 124495 | 116519 | 190885 | 203902 | 244464 | 203902 |
| coll | 70537 | 48587 | 165231 | 152952 | 231827 | 263549 | 312373 | 265936 |
| lex | 72009 | 50760 | 170797 | 157680 | 258999 | 270458 | 329389 | 270458 |
| std | 1873085 | 1611443 | 4054543 | 3800534 | 6231817 | 5895253 | 9476146 | 9053710 |

TAB. 5.11: Quantité de traits extraits par les divers ensembles de patrons selon le paramètre $n$.

## 5.7 Discussion des résultats

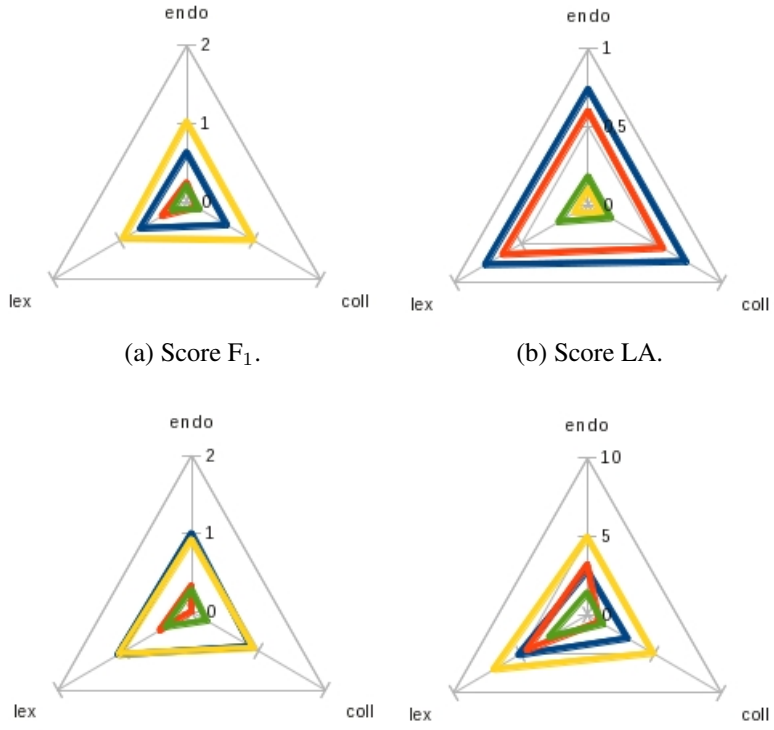
Dans cette section, nous indiquons, comparons et discutons des résultats obtenus par les diverses stratégies d'identification des MWEs, et ce dans le cadre de deux tâches fondamentales que sont l'étiquetage morpho-syntaxique et l'analyse syntaxique. On peut rappeler que, pour l'analyse syntaxique, nous avons évalué deux stratégies. La première *pre* consiste à pré-segmenter un texte brut avec un reconnaisseur discriminatif basé sur des traits consacrés aux MWEs, dans notre cas LGTagger. Puis, les phrases générées sont analysées syntaxiquement. La deuxième stratégie *post* consiste à décomposer le corpus en unités minimales et à appliquer ensuite une chaîne de traitements syntaxiques composée d'un analyseur lambda et d'un réordonnanceur discriminatif, lui aussi basé sur des traits concernant les MWEs. Afin de comparer directement les résultats de ces deux stratégies[130], nous avons représenté graphiquement, dans la figure 5.17, les gains pour les différentes mesures obtenus sur le corpus d'évaluation selon l'ensemble de traits exploité : `endo`, `lex` et `coll`[131]. Dans chaque graphique, ces trois ensembles sont liés à un axe propre, ce qui conduit à modéliser des courbes sous forme de triangles. Le code couleur des courbes est le suivant : bleu=BKY$_{pre}$, rouge=BKY$_{post}$, jaune=Brown$_{pre}$ et vert=Brown$_{post}$.
Globalement, on peut constater que la stratégie *pre* est plus efficace que la stratégie *post* car, pour la totalité des mesures et pour chaque analyseur, les triangles générés par *post* sont entièrement inclus dans les triangles générés par *pre*[132]. Une pré-segmentation pertinente semble donc plus adaptée à l'analyse syntaxique qu'un texte composé uniquement d'unités minimales dont la segmentation est effectuée en post-traitement. Pour la stratégie *post*, la complexité liée à plusieurs paramètres, dont le jeu d'étiquettes de taille plus conséquente, génère beaucoup de bruit réduisant l'impact positif potentiel du réordonnanceur sur les performances générales. Nous avons vu précédemment que les scores *oracle* de Brown et BKY évoluaient de façon identique. Il est donc intéressant de noter que les courbes de BKY liées au score $F_1$(MWEs) sont assez proches pour les deux stratégies, alors qu'il existe une nette différence pour Brown. L'impact de *post* est donc moins fort sur un analyseur lexicalisé comme Brown. Pour les deux stratégies, on peut néanmoins observer que les traits `lex` permettent de maximiser les scores $F_1$(MWEs).
Pour résumer, nous avons indiqué, dans le tableau 5.12, les meilleurs gains obtenus par les deux analyseurs ainsi que par LGTagger (en se basant sur les traits `lex`). Les scores de base sont indiqués dans la colonne `base`. Le score $F_1$(MWEs) de base de LGTagger est relativement bas par rapport aux analyseurs car il ne peut profiter d'informations à propos des structures syntaxiques. Par contre, l'ajout de traits concernant les MWEs permet de réaliser un gain d'environ +13, ce qui en fait le meilleur reconnaisseur de MWEs avec 79.33%. Il est, en pratique, meilleur que celui proposé par (Green *et al.*, 2011) qui est à 71.1% sur les phrases inférieures à 40 mots du FTB-STF[133]. LGTagger atteint, sur ce même corpus, 74% pour les traits endogènes

[130]Pour *post*, nous indiquons les résultats obtenus avec $n$ fixé à 50.

[131]Les résultats de `all` ne sont pas montrés car ils sont similaires à ceux de `lex`.

[132]Excepté pour le score LA de l'analyseur Brown. Ces différentes observations sont également valables sur le corpus de développement.

[133]On peut rappeler que (Green *et al.*, 2011) ont évalué leur système sur les phrases inférieures à 40 mots uniquement.




(a) Score $F_1$. (b) Score LA.

(c) Score UAS. (d) Score $F_1$(MWEs).

FIG. 5.17: Gains obtenus par les deux stratégies *pre* et *post* selon l'ensemble de traits exploité. Le code couleur des courbes est le suivant : bleu=BKY$_{pre}$, rouge=BKY$_{post}$, jaune=Brown$_{pre}$ et vert=Brown$_{post}$.

(soit un gain absolu de +2.9%) et 77.3% pour tous les traits (soit un gain absolu de +6.2%).
Nous avons également réitéré nos expériences sur la version de juin 2010 du corpus FTB (Constant *et al.*, 2012a,b), avec BKY uniquement. Cette version compte 3566 phrases supplémentaires et un nombre plus important d'unités polylexicales. Comme pour le FTB-UC, la qualité de l'identification des MWEs est maximisée avec *pre*$_{lex}$ : 77.4% soit un gain absolu de +6.3 par rapport aux résultats de base (71.1%). En revanche, la stratégie *post* a ici un impact plus faible comme le montre le gain de seulement +3.4 obtenu avec `lex`, gain comparable à celui de Brown sur le FTB-UC. Cela traduit une forte variation des performances selon l'analyseur utilisé et le corpus analysé, et ce contrairement à la stratégie *pre*.
Pour déterminer plus précisement ce qui est amélioré par la stratégie *pre*$_{lex}$ au niveau syntaxique, nous avons effectué une évaluation par catégorie syntaxique sur le corpus d'évaluation du FTB-UC grâce à l'outil `EvalbByCat` de l'analyseur de Stanford. Les scores sont indiqués dans le tableau 5.13 et sont calculés en absolus par rapport aux scores de base des analyseurs. La colonne `#ref` indique le nombre d'occurrences de chaque catégorie dans le corpus d'évaluation. Pour les deux analyseurs, il apparaît que cette stratégie améliore sensiblement la reconnaissance des noms communs, prépositions et adverbes composés. Pour les autres catégories de MWE, on peut observer des gains beaucoup plus faibles pour BKY que pour Brown, avec plusieurs cas de chute des performances : -6.96 pour `MWPRO`, -1.08 pour `MWCS` et -1.65

| | BKY | | | Brown | | | LGTagger | |
|---|---|---|---|---|---|---|---|---|
| | base | lex(pre) | lex(post) | base | lex(pre) | lex(post) | base | lex |
| $F_1$ | 82.30 | **+0.69** | +0.37 | 79.01 | **+0.95** | +0.31 | - | - |
| LA | 92.38 | **+0.77** | +0.64 | 89.65 | -0.10 | **+1.12** | - | - |
| UAS | 87.47 | **+1.10** | +0.47 | 85.76 | **+1.08** | +0.38 | - | - |
| $F_1$(MWEs) | 72.66 | **77.80** | 77.19 | 70.46 | **77.45** | 73.29 | 66.70 | **79.33** |

TAB. 5.12: Tableau récapitulatif des meilleurs gains obtenus avec les diverses stratégies sur le corpus d'évaluation du FTB-UC.

pour `MWNPP`. Avec une majorité de gains positifs ou nuls, aucun effet de bord n'est à constater sur les constituants de plus haut niveau. Ce n'est en revanche pas le cas sur le FTB 2010 où nous avons constaté une chute des performances au niveau des relatives (-3.5), des subordonnées (-1.1), et même des groupes nominaux (-0.8). L'un des principaux problèmes vient de l'identification des verbes composés à l'indicatif ou au subjonctif qui est précaire avec un score $F_1$(MWEs) d'environ 20%. Bien que ce score soit identique sur le FTB-UC avec BKY, il y a quatre fois plus de verbes composés dans le FTB 2010, ce qui explique l'impact négatif plus important sur ce corpus. Dans une moindre mesure, le repérage des noms communs composés et des conjonctions de subordination composées pose également des problèmes.

| Catégorie | #ref | BKY | | Brown | |
|---|---|---|---|---|---|
| | | base | lex | base | lex |
| MWV | 14 | 18.18 | +9.09 | 43.48 | -6.12 |
| MWCC | 16 | 88.24 | +2.08 | 76.47 | +11.03 |
| MWADJ | 22 | 55.00 | -16.67 | 52.00 | -26.0 |
| MWDET | 24 | 61.54 | +6.64 | 68.09 | +1.68 |
| MWPRO | 31 | 86.21 | -6.96 | 88.14 | +7.01 |
| MWNC | 55 | 60.34 | +13.75 | 58.33 | +8.98 |
| MWCS | 70 | 89.19 | -1.08 | 77.92 | +8.11 |
| MWNPP | 104 | 82.30 | -1.65 | 75.25 | +3.30 |
| MWP | 294 | 76.92 | +6.90 | 74.81 | +9.74 |
| MWADV | 357 | 67.49 | +9.80 | 68.27 | +9.81 |
| Adp | 86 | 46.45 | +6.25 | 49.06 | +3.44 |
| Ssub | 406 | 57.21 | +2.09 | 56.91 | +1.05 |
| Srel | 408 | 72.41 | +1.43 | 67.09 | +0.08 |
| VPpart | 541 | 65.69 | -1.98 | 64.71 | +1.57 |
| PP | 566 | 79.35 | +0.78 | 74.66 | +0.77 |
| Sint | 627 | 60.65 | +2.15 | 57.28 | +0.43 |
| COORD | 906 | 74.01 | +1.54 | 68.67 | +0.77 |
| VPinf | 781 | 75.22 | +2.11 | 72.08 | +1.81 |
| SENT | 1235 | 100 | -0.16 | 100 | -0.08 |
| AP | 1761 | 85.57 | +0.17 | 84.47 | +1.21 |
| NP | 9281 | 81.30 | +0.23 | 76.60 | +0.84 |
| VN | 3089 | 94.75 | +0.01 | 93.77 | -0.09 |

TAB. 5.13: Évaluation par catégorie syntaxique de la stratégie *pre* avec les traits `lex` sur le corpus d'évaluation du FTB-UC.

# 6

# Analyse syntaxique et algorithmes de regroupements lexicaux basés sur les lexiques

## 6.1 Introduction

Nous avons évoqué, dans le chapitre I.2.3, les deux problèmes principaux liés au formalisme PCFG. Le faible conditionnement lexical des règles d'une grammaire PCFG a été en partie résolu par les modèles lexicalisés proposés par (Charniak, 1997; Collins & Singer, 1999; Collins, 2003). Ces modèles annotent les noeuds non-terminaux par leur tête lexicale, ce qui entraîne une propagation des informations lexicales dans la totalité de l'arbre. Quant aux hypothèses d'indépendance trop fortes entre les règles de la grammaire, elles sont attenuées par l'introduction de symboles latents à travers l'utilisation de grammaires PCFG-LA (Matsuzaki *et al.*, 2005; Petrov *et al.*, 2006; Petrov & Klein, 2007; Petrov, 2010). Malgré le fait que les grammaires PCFG-LA soient dites *non lexicalisées*, certaines expériences, sur le français notamment (Crabbé & Candito, 2008), ont prouvé que le lexique joue un rôle important dans ces grammaires. Ils ont entraîné et évalué l'analyseur BKY sur une autre version du FTB[134] dont les formes fléchies ont été remplacées par leur étiquette morpho-syntaxique correcte. Ce corpus dont les mots sont maintenant des étiquettes conduit à créer une grammaire PCFG-LA totalement non lexicalisée obtenant un score $F_1$ de 86.28 sur le corpus d'évaluation. Ces performances ont ensuite été comparées avec une grammaire apprise sur un corpus dont les mots ont été remplacés par la combinaison correcte de l'étiquette morpho-syntaxique suivie du mot. Cette grammaire obtient un score $F_1$ de 87.79 ce qui correspond à un gain de +1.51. Ce résultat prouve qu'à travers les divisions des symboles préterminaux (étiquettes morpho-syntaxiques), les informations lexicales sont propagées dans l'arbre lors de la division des symboles des niveaux supérieurs.
La dispersion lexicale des données est un des problèmes liés aux PCFG, et causé par le degré

[134] Version décrite dans (Crabbé & Candito, 2008) et contenant 12351 phrases.

de complexité plus ou moins important de la flexion d'une langue. Par exemple, nous avons vu, dans le chapitre I.1, que le corpus du français FTB-UC possède une flexion plus importante que le PTB pour l'anglais (en moyenne 1.5 formes par lemme pour le FTB-UC contre seulement 1.3 pour le PTB). L'impact de cette flexion sur les performances d'analyseurs du français a été évalué dans (Candito & Crabbé, 2009), où les mots du corpus ont été remplacés par la combinaison correcte de l'étiquette et du lemme. BKY obtient un score $F_1$ de 88.18 sur le corpus d'évaluation, ce qui correspond à un gain de +0.39 par rapport à l'expérience précédente. Ainsi, ce résultat positif obtenu par une flexion moindre confirme l'impact négatif de la dispersion lexicale sur la grammaire PCFG-LA de BKY. Et cela montre également que le regroupement lexical est une technique efficace qui permet d'atténuer significativement ce problème.
En ce qui concerne l'analyse en dépendances, nous avons pu voir que la plupart des meilleurs analyseurs actuels sont basés sur un modèle discriminant. L'impact de la dispersion des données sur ces modèles est moindre que pour les modèles génératifs. Aussi, l'utilisation de regroupements lexicaux n'est pas aussi cruciale pour ce type d'analyse. Malgré tout, plusieurs expériences récentes (Koo *et al.*, 2008; Suzuki *et al.*, 2009) ont montré que des regroupements pertinents, intégrés sous la forme de traits au modèle, pouvaient participer à l'amélioration générale des performances.
Dans les sections qui suivent, nous décrivons, en premier lieu, les principaux types de regroupements lexicaux de la littérature, permettant de résoudre au mieux le problème de la dispersion lexicale des données, et ceci dans le cadre des PCFG (section I.6.2). Puis, nous ferons de même pour les regroupements qui améliorent les performances générales des analyseurs en dépendances (section I.6.3). Après avoir réalisé ce tour d'horizon, nous proposons et évaluons, dans la section 6.4, nos propres méthodes de regroupements de mots basées sur des données issues de lexiques syntaxiques.

## 6.2 Classes lexicales pour l'analyse en constituants

Dans la plupart des travaux liés aux PCFG, l'approche par regroupement lexical consiste à remplacer les mots des corpus et des textes par des symboles plus généraux (Agirre *et al.*, 2008; Candito & Crabbé, 2009; Candito & Seddah, 2010). Ces symboles sont des classes lexicales précalculées par des algorithmes semi-supervisés ou à partir de ressources syntaxiques et sémantiques. Une grammaire PCFG est ensuite apprise sur le nouveau corpus d'apprentissage, puis, cette grammaire est utilisée pour analyser les phrases brutes du nouveau corpus d'évaluation. La dernière étape consiste à réintroduire les tokens originaux dans les analyses produites afin d'obtenir la sortie finale de l'algorithme. (Agirre *et al.*, 2008) proposent de remplacer chaque mot par une classe sémantique, déterminée à partir d'un thésaurus. Quant à (Candito & Crabbé, 2009; Candito & Seddah, 2010), ils proposent trois types de regroupements lexicaux, où chaque mot est remplacé par :
– la combinaison de son étiquette et de son lemme,
– une forme fléchie dont certaines marques morphologiques sont supprimées,
– une classe lexicale obtenue par un algorithme de classification hiérarchique semi-supervisé

(Koo *et al.*, 2008).
Dans les sections suivantes, nous détaillons chaque méthode et nous indiquons les résultats obtenus sur le français quand ils sont disponibles.

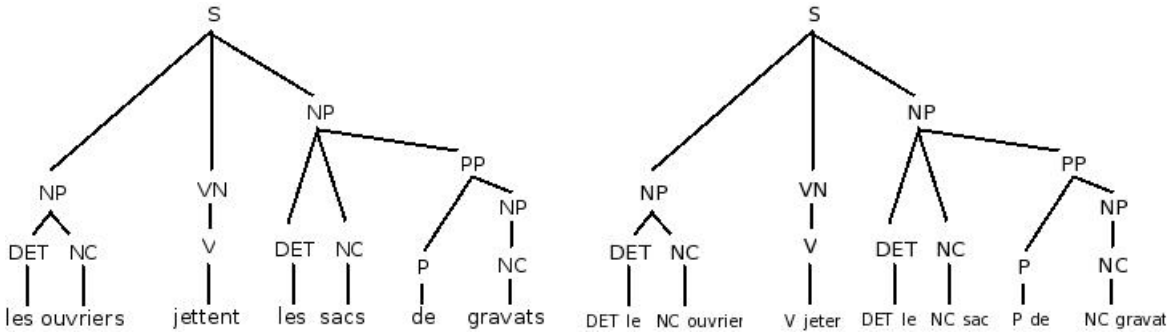
### 6.2.1 Exploitation d'informations sémantiques

Selon (Agirre *et al.*, 2008), la lexicalisation des analyseurs PCFG (Charniak, 1997; Collins & Singer, 1999; Collins, 2003) permet de modéliser de nombreuses affinités syntaxiques entre les mots, mais cela n'est pas suffisant pour résoudre certains problèmes comme les rattachements prépositionnels. Par exemple, supposons que le corpus d'apprentissage contienne de nombreuses occurrences de *four* en position de complément d'objet indirect de *cuire*, et introduit par *au*. Le modèle lexicalisé sera donc probablement capable de modéliser le fait que *au four* soit rattaché au verbe *cuire*, et non à un éventuel objet direct de celui-ci (e.g. *cuire un oeuf au four*). En revanche, cela ne donne pas d'informations sur la préférence de rattachement de *au micro-onde* dans *cuire un oeuf au micro-onde*. Bien que *four* et *micro-onde* soient sémantiquement similaires, une grammaire PCFG n'est pas capable de modéliser cette similarité et, ainsi, le fait qu'ils aient les mêmes affinités syntaxiques.
Aussi, (Agirre *et al.*, 2008) proposent d'utiliser des informations sémantiques afin de déterminer des similarités sémantiques entre les mots, et donc de suggérer des préférences de rattachement (Ratnaparkhi *et al.*, 1994). Leur approche, identique à celle décrite dans (Koo *et al.*, 2008; Candito & Crabbé, 2009; Candito *et al.*, 2010a), consiste à remplacer les mots du corpus d'apprentissage et des textes bruts par des symboles plus généraux qui sont, dans le cas présent, des classes sémantiques. Les mots *four* et *micro-onde* de l'exemple précédent peuvent être substitués par une classe sémantique *OUTIL_CUISSON*. (Agirre *et al.*, 2008) proposent plusieurs façons d'obtenir et d'utiliser les classes :

– un mot est remplacé par la classe la plus fréquemment associée à ce mot dans un thésaurus de l'anglais WordNet2.1[135] (Fellbaum, 1998). Dans ce thésaurus, les mots sont répartis dans des classes de synonymes, chacune ayant un sens principal (e.g. *outil tranchant utilisé comme instrument*), et ces classes sont elles-mêmes regroupées selon leur catégorie grammaticale et un sens plus général (noms dénotant une action, verbes de sentiments,...). Différentes combinaisons d'informations peuvent donc être utilisées pour calculer un ensemble de classes sémantiques à exploiter.
– la deuxième approche est identique au point précédent, excepté qu'ici, le thésaurus a été créé automatiquement grâce à un algorithme non-supervisé (Mccarthy *et al.*, 2004). Une méthode de classification (Lin, 1998), couplée à des mesures de similarité issues de WordNet, est appliquée sur le corpus de l'anglais BNC afin d'obtenir ce thesaurus.

Pour les expériences, le corpus qu'ils ont utilisé est composé de la partie sémantiquement annotée du Brown Corpus qui est incluse dans deux corpus distincts, SemCor (Landes *et al.*, 1998) et le PTB. Afin d'exploiter les informations sémantiques disponibles dans ces deux corpus, ils ont été fusionnés en effectuant un alignement des phrases et des mots. Il en résulte un corpus de 8669 phrases contenant 151928 mots. Les résultats ont montré que les performances de deux analyseurs PCFG lexicalisés (Bikel, 2004; Charniak, 2000) sont significativement améliorées,

[135] `http://wordnet.princeton.edu/`




(a) Arbre original (b) Arbre modifié

FIG. 6.1: Application de la méthode CatLemma sur l'arbre syntaxique de la phrase *les ouvriers jettent les sacs de gravats*.

tant globalement, avec environ 7% d'erreurs en moins par rapport aux résultats de base, qu'au niveau des rattachements prépositionnels (environ 20%).

### 6.2.2 Regroupements par combinaison de l'étiquette et du lemme

Afin de réduire l'effet de dispersion des données dû à la flexion importante du français, une méthode simple consisterait à remplacer un mot par son lemme. Cependant, la perte d'indices morphologiques provoquerait une dégradation des performances, causée par la forte hausse de l'ambiguité en terme d'étiquettes morpho-syntaxiques associées à chaque mot. Pour pallier ce problème, l'approche *CatLemma*, décrite dans (Candito & Seddah, 2010), combine le lemme à l'étiquette morpho-syntaxique du mot. D'après le jeu d'étiquettes du FTB-UC, les différentes formes fléchies du verbe *jouer* sont, par exemple, remplacées par un des 8 symboles possibles selon l'étiquette du verbe dans le texte : `V_jouer`, `VPP_jouer`, `VIMP_jouer`, `VPR_jouer`, `VINF_jouer` et `VS_jouer`. Cette approche a malgré tout un inconvénient majeur, à savoir que les informations sur le lemme et l'étiquette ne sont, initialement, pas présentes dans un texte brut. Avant de pouvoir analyser syntaxiquement ce texte, il faut donc appliquer sur chaque phrase un étiqueteur et un lemmatiseur. De fait, de potentielles erreurs d'étiquetage et de lemmatisation surviennent, car les performances actuelles des étiqueteurs et des lemmatiseurs ne sont pas parfaites[136]. La figure 6.1 montre l'arbre syntaxique de la phrase *les ouvriers jettent les sacs de gravats* avant et après l'application de CatLemma. On peut voir que l'occurrence du verbe *jeter* à l'indicatif a bien été remplacée par la classe `V_jeter`.

### 6.2.3 Hypothèses sur les marques morphologiques

Une autre méthode, décrite dans (Candito & Crabbé, 2009; Candito & Seddah, 2010), propose de laisser la tâche de désambiguisation des étiquettes à l'analyseur. Ils posent l'hypothèse

[136]Par exemple, l'étiqueteur Morfette a un taux d'erreurs d'environ 2% à la fois sur l'étiquetage et sur la lemmatisation sur le corpus d'évaluation du FTB-UC.

que certaines marques flexionnelles dites *peu importantes* ne sont pas nécessaires pour déterminer les projections syntaxiques en constituants. Le mode des verbes est, par exemple, une information utile car il nous renseigne sur le nombre et le type d'arguments du verbe. En revanche, d'autres marques, comme le genre ou le nombre pour les noms[137] ou la personne des verbes, ne sont probablement pas aussi cruciales. Le regroupement lexical, appelé *dé-flexion* [DFL], consiste donc à supprimer certaines marques flexionnelles aux mots, et ceci à l'aide d'un lexique morpho-syntaxique. Chaque mot d'un texte est remplacé par sa forme fléchie la plus réduite possible, tout en conservant les ambiguités initiales en terme d'étiquettes. Les marques concernées par une possible suppression sont le genre et le nombre pour les noms, les pronoms, les adjectifs et les déterminants, ainsi que la personne et le temps des verbes.
La figure 6.2 illustre l'application de cette méthode sur le mot *prises*. Ce mot est ambigu car il possède deux entrées distinctes dans le lexique, à savoir une entrée verbale conjuguée au participe passé au féminin pluriel, et une entrée nominale au féminin pluriel. L'algorithme va, tout d'abord, tenter de supprimer la marque du pluriel aux deux entrées, ce qui donne la même forme fléchie *prise* à la fois pour le nom et pour le verbe. Les ambiguités initiales sont donc conservées. Ensuite, l'algorithme fait de même avec la marque du féminin, ce qui, après suppression, correspond à la forme *pris* pour le verbe et à *prise* pour le nom car il n'y a pas de forme au masculin correspondante. La suppression de la marque du féminin génère deux formes réduites distinctes. À cette étape, les ambiguités initiales ne sont donc pas conservées, ce qui entraîne l'arrêt de l'algorithme. En sortie, le mot *prises* est remplacé par la forme fléchie commune la plus réduite possible, à savoir *prise*. La figure 6.3 montre l'arbre syntaxique de la phrase *les ouvriers jettent les sacs de gravats*, avant et après l'application de DFL[138]. Le verbe *jeter* a été remplacé par sa forme à la deuxième personne du pluriel de l'indicatif, et les autres mots ont été mis au singulier (excepté *dans*).

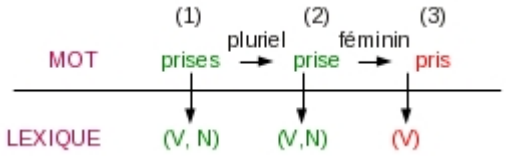
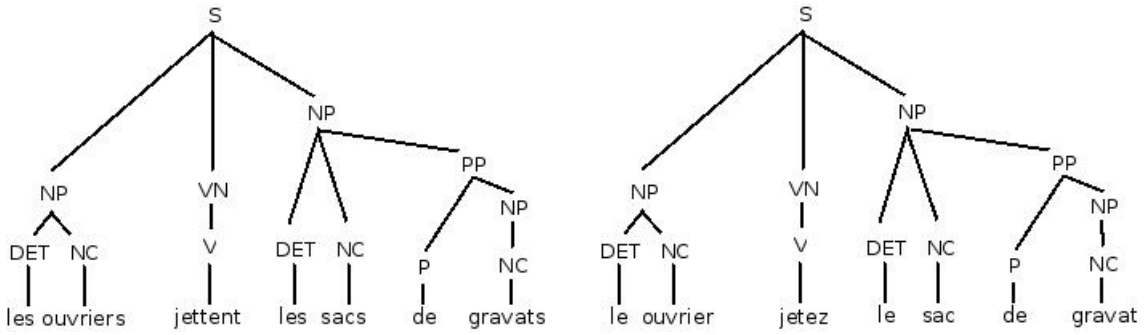



FIG. 6.2: Application de la méthode DFL sur le mot *prises*.

Formellement, pour un mot *w* connu du lexique, l'algorithme tente, tout d'abord, de supprimer la marque du pluriel à chaque entrée fléchie du lexique $w_{lex}$ morphologiquement identique à $w$. Si une forme au singulier $w_s$ est possible pour chaque entrée $w_{lex}$, et que $w_s$ est morphologiquement identique pour l'ensemble de ces entrées, alors $w$ est remplacé par $w_s$. Un processus similaire est ensuite appliqué sur $w_s$ dans le but de supprimer la marque du féminin. Si le premier mot de la phrase est inconnu du lexique, alors l'algorithme est appliqué sur la forme en minuscules du mot. De plus, les formes verbales, qui ne sont pas ambigües en terme d'étiquettes morpho-syntaxiques, sont systématiquement transformées à la deuxième personne du pluriel de l'indicatif. En général, la forme verbale correspondante a pour suffixe *-ez* qui est un

[137]On peut supposer que le mot *manivelle* et sa forme au pluriel *manivelles* apparaissent dans les mêmes structures syntaxiques même s'il s'agit bien de deux mots distincts d'après le formalisme PCFG.
[138]Dans cet exemple, le lexique utilisé est le Le*fff*.

suffixe peu commun pour les mots d'autres catégories grammaticales. Cela permet d'éviter de créer de l'ambiguité lexicale qui n'existait pas auparavant.




(a) Arbre original (b) Arbre modifié

FIG. 6.3: Application de la méthode dé-flection sur l'arbre syntaxique de la phrase *les ouvriers jettent les sacs de gravats*.

### 6.2.4 Algorithme de regroupement lexical non-supervisé

Une autre méthode consiste à remplacer les mots du corpus par des classes lexicales, obtenues automatiquement par l'application d'un algorithme de classification sur un grand corpus non annoté. Cette approche, appelée *StatClust*[139], fut explorée pour la première fois par (Miller *et al.*, 2004) dans le cadre de la reconnaissance d'entités nommées, puis reprise pour l'analyse syntaxique par (Koo *et al.*, 2008). La plupart des travaux exploitant cette approche (Liang, 2005; Koo *et al.*, 2008; Candito & Crabbé, 2009; Candito & Seddah, 2010; Candito *et al.*, 2010b) se basent sur l'algorithme de classification non-supervisé de Brown (Brown *et al.*, 1992)[140]. Cet algorithme itératif procède de la façon suivante pour déterminer $C$ classes (le paramètre $C$ étant à définir manuellement).
Initialement, chacun des $C$ mots les plus fréquents d'un corpus brut est associé à une classe propre. Puis, pour le $(C+1)^{eme}$ mot, on crée une classe. Ensuite, on fusionne la paire de classes, prise parmi les $C+1$ classes, qui minimise la perte de vraisemblance dans le corpus étant donné un modèle bigramme défini par les classes. On répète cette opération pour les mots suivants jusqu'à avoir couvert l'ensemble des mots du corpus. Afin de ne pas créer des classes contenant trop de mots, il est d'usage de filtrer les mots du corpus brut qui apparaissent moins de $x$ fois. En fusionnant une paire de classes à chaque itération de l'algorithme, on obtient en sortie une classification hiérarchique des mots qui peut être représentée sous la forme d'un arbre binaire. Chaque noeud de l'arbre est étiqueté par un identifiant unique représentant le chemin depuis la racine. Cet identifiant est une séquence binaire composée de `0` (branche gauche) et de `1` (branche droite). Par exemple, le mot *légumes* a pour identifiant `011` dans l'arbre de la

[139]Dans la littérature, cette méthode n'a pas de nom particulier. Pour faciliter la lecture, nous lui en avons donc donné un.

[140]Les expériences utilisent en général l'implémentation de l'algorithme décrite dans (Liang, 2005), disponible à l'adresse `http://cs.stanford.edu/~pliang/software/`

figure 6.4. Afin d'obtenir des classes de mots, il suffit de sélectionner un niveau de profondeur à partir de la racine de l'arbre. Le niveau de profondeur permet d'obtenir des ensembles de classes de différentes granularités. Par exemple, si la profondeur choisie est *deux*, alors, à partir de l'arbre de la figure 6.4, nous obtenons les classes suivantes : `[janvier, février]`, `[fruits, légumes]`, `[manger, boire]` et `[de, à]`.

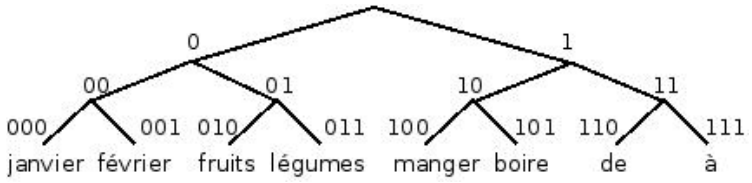



FIG. 6.4: Exemple d'arbre binaire créé par l'algorithme de classification de Brown.

Un mot est considéré comme inconnu lorsqu'il n'est associé à aucune classe. Comme dit précédemment, les mots apparaissant moins de $x$ fois dans le corpus brut ne sont pas pris en compte lors de la classification hiérarchique. Le traitement des mots inconnus consiste à les remplacer par la classe générique appelée `UNK`. De plus, un suffixe `_C` est ajouté à une classe lorsque le mot démarre par une lettre capitale. D'autres traits (9 au total) sont également pris en compte pour diviser les classes, comme le fait que le mot finisse par `-ant`, ou encore qu'il soit composé uniquement de chiffres. (Candito & Crabbé, 2009) ont montré que ces traits (indices morphologiques) sont nécessaires car les classes créées par l'algorithme contiennent du bruit. C'est à dire qu'elles regroupent des mots n'ayant pas forcément la même étiquette morpho-syntaxique. La conséquence immédiate liée à ce bruit est la multiplication des erreurs lors de la phase d'étiquetage morpho-syntaxique à cause de l'augmentation du nombre d'étiquettes possibles par classe.
Comme pour les deux méthodes précédentes, cette approche permet d'augmenter le vocabulaire connu, car le lexique du corpus est maintenant composé de classes lexicales. Dans la plupart des expériences réalisées sur le français, les différents corpus (bruts et annotés) ont d'abord été prétraités par l'une des deux méthodes, DFL ou CatLemma. Ce prétraitement permet de réduire un peu plus la part de mots inconnus tout en améliorant la complexité et la précision de l'algorithme de classification.

### 6.2.5 Impact des regroupements lexicaux sur le FTB-UC avec BKY

Nous avons indiqué, dans le tableau 6.1, les performances de BKY[141] entraîné et évalué sur le corpus FTB-UC modifié par les différentes méthodes de regroupement lexical énoncées précédemment. La totalité des chiffres de ce tableau sont directement tirés de l'article (Candito & Seddah, 2010). Nous indiquons, pour chaque méthode de regroupement lexical, la taille du lexique du corpus d'apprentissage du FTB-UC en mots distincts après modification (`#lexique`), ainsi que les scores $F_1$ et UAS obtenus sur le corpus d'évaluation. Dans le cadre de la méthode CatLemma, l'étiqueteur MElt couplé au lemmatiseur de l'outil Bonsaï ont été

[141] Version 1.1 modifiée pour le français avec les paramètres habituels, à savoir un nombre de cycles de $n = 5$, une markovisation horizontale de 0 et une markovisation verticale de 1.

utilisés afin de pouvoir remplacer tout mot des corpus d'évaluation par une étiquette morpho-syntaxique et un lemme. Quant à la méthode DFL, elle est basée sur le lexique Le*fff*. Le paramètre $C$ de la méthode StatClust a été fixé à 1000 et le nombre minimum d'occurrences $x$ à 200. Les classes de mots ont été automatiquement créées à partir du corpus brut de 125 millions de mots *L'Est Républicain*[142]. Ce corpus a préalablement subi plusieurs traitements dont une tokenisation et une segmentation en phrases. De plus, les mots composés les plus fréquents du FTB-UC[143] ont été repérés dans le corpus brut et systématiquement fusionnés pour former un seul token. Les ponctuations ne sont pas prises en compte dans les mesures. De plus, les gains précisés entre parenthèses sont calculés en absolu par rapport aux scores de base `base`. On peut voir que toutes les méthodes améliorent significativement les performances de BKY ($p < 0.05$), et ce, tout en réduisant la taille du lexique du corpus. `DFL` est légèrement meilleure que `CatLemma` mais l'intégration de `StatClust` dans la chaîne de traitement conduit à obtenir des performances identiques pour les deux méthodes ($p > 0.2$).

| Symboles terminaux | #lexique | $F_1$ | UAS |
|---|---|---|---|
| Formes fléchies (base) | 24110 | 84.10 | 89.57 |
| DFL | 18052 | 85.07 (+0.97) | 90.45 (+0.88) |
| DFL+StatClust | 1773 | **86.21** (+2.11) | 90.96 (+1.39) |
| CatLemma | 18654 | 84.83 (+0.73) | 90.30 (+0.73) |
| CatLemma+StatClust | 1298 | **86.20** (+2.10) | **91.22** (+1.65) |

TAB. 6.1: Performances de BKY sur le corpus d'évaluation du FTB-UC selon les regroupements lexicaux.

## 6.3 Classes lexicales pour l'analyse en dépendances

Dans le cadre de l'analyse en dépendances probabiliste, les classes lexicales ont été généralement utilisées comme de simples traits afin d'être intégrés dans un modèle discriminant. Ces modèles permettent aux analyseurs de moins souffrir de problèmes comme la dispersion des données. Dans ce but, (Koo *et al.*, 2008; Suzuki *et al.*, 2009) proposent d'exploiter conjointement deux ensembles de patrons de traits. Le premier ensemble est composé de patrons similaires à ceux utilisés habituellement dans la littérature (McDonald *et al.*, 2005; Carreras, 2007). Ils consistent à extraire des informations sur les gouverneurs et les étiquettes morpho-syntaxiques. Quant au deuxième ensemble, les patrons calculent des traits à partir d'informations tirées d'une ressource externe. Dans leur cas, cette ressource est un arbre hiérarchique de classification obtenu en appliquant l'algorithme de classification non-supervisé de Brown sur un grand corpus brut. Les patrons de traits utilisés par (Koo *et al.*, 2008) exploitent la granularité des regroupements en classes offerte par l'arbre de classification. Par exemple, le patron $hc_4, mc_4$ extrait deux informations pour un mot donné de la phrase, à savoir l'identifiant $hc_4$ à

[142] `http://www.cnrtl.fr/corpus/estrepublicain/`
[143] Les 240 mots composés les plus fréquents ont été sélectionnés.

une profondeur de 4 du gouverneur du mot, et l'identifiant $mc_4$ du mot courant à une profondeur de 4 également.
Cette approche a été évaluée sur deux langues, l'anglais et le tchèque. Pour l'anglais, le corpus annoté (apprentissage et évaluation) est le PTB transformé en dépendances à partir de la version en constituants, et ceci à l'aide de l'outil de conversion *Penn2Malt*[144]. Cet outil contient un ensemble de règles de sélection des têtes lexicales (Yamada & Matsumoto, 2003). Le corpus brut journalistique BLLIP est utilisé pour calculer les classes. Quant au tchèque, le corpus annoté est le Prague Dependency TreeBank 1.0 [PDT][145] (Hajič, 1998; Böhmová *et al.*, 2001), et une partie du PDT[146] contenant des phrases brutes est utilisée pour calculer les classes. Les gains obtenus (UAS) avec l'analyseur MST (McDonald *et al.*, 2006) sont significatifs pour les deux expériences comparativement à un analyseur entraîné uniquement avec l'ensemble des patrons de base (+1% pour les deux langues).

### 6.3.1 Impact des regroupements lexicaux sur le FTB-UC-DEP

Les performances des analyseurs syntaxiques sont généralement améliorées en incorporant au modèle des informations lexicales comme des traits morphologiques (Tsarfaty, 2006), les lemmes (Seddah *et al.*, 2010) ou encore des classes lexicales (Koo *et al.*, 2008; Candito & Crabbé, 2009; Candito & Seddah, 2010). Les expériences décrites dans (Candito *et al.*, 2010b) proposent, dans le cadre du français, une évaluation comparative de l'impact de ces différentes informations sur deux analyseurs en dépendances, Malt et MST, ainsi que sur un analyseur en constituants, BKY. Les traits morphologiques (mode, genre, nombre,...), ainsi que les formes lemmatisées, sont extraits du lexique L*efff*. Quant aux classes lexicales, elles sont calculées de la même façon que dans (Candito & Crabbé, 2009; Candito & Seddah, 2010), en appliquant l'algorithme de Brown sur le corpus brut *L'Est Républicain*. De plus, comme dans (Koo *et al.*, 2008), ils ont évalué l'impact de ces classes selon le degré de granularité offert par l'arbre hiérarchique de classification. Les analyseurs en dépendances étant basés sur un modèle discriminant, les informations lexicales peuvent être utilisées soit comme des nouveaux traits soit comme substitution d'un trait de base existant (mot, étiquette morpho-syntaxique,...). En revanche, BKY étant un analyseur PCFG, la seule solution est la substitution des mots dans les corpus d'apprentissage, de développement et d'évaluation.
Les résultats qu'ils ont obtenus sont indiqués dans le tableau 6.2. Pour les évaluations, ils se sont basés sur deux scores, UAS et LAS, ce qui induit une conversion des analyses de BKY vers le format en dépendances. Pour chaque analyseur, les résultats sont donnés sur le corpus de développement, selon que l'on utilise les traits de base uniquement[147] (`Base`), ou que l'on ajoute le trait qui correspond aux classes lexicales (`CLS`). Les scores de la combinaison de traits la plus performante (`Best`) sont également précisés (différente selon l'analyseur). Pour le corpus d'évaluation, seuls les résultats des expériences `Base` et `Best` sont donnés.

[144] `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`
[145] `http://ufal.mff.cuni.cz/pdt/`
[146] Cette partie du PDT contient environ 39 millions de mots et est disjointe du corpus d'apprentissage et d'évaluation.
[147] Pour BKY, il s'agit des mots originaux.

| | **Corpus de développement** | | | | | | **Corpus d'évaluation** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Base | | CLS | | Best | | Base | | Best | |
| | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS |
| BKY | 89.3 | 85.1 | 90.8 | 86.5 | **90.8** | 86.5 | 89.6 | 85.6 | **91.0** | 86.8 |
| MST | 90.0 | 87.2 | 90.1 | 87.2 | 90.3 | **87.5** | 90.3 | 87.6 | 90.9 | **88.2** |
| Malt | 89.0 | 86.2 | 89.2 | 86.5 | 89.4 | 86.9 | 89.3 | 86.7 | 89.7 | 87.3 |

TAB. 6.2: Impact de différentes informations lexicales, dont les classes lexicales, sur trois analyseurs syntaxiques.

Comme nous l'avons remarqué dans la section 6.2.5, la substitution des mots par les classes lexicales permet à BKY de réduire l'impact de la dispersion des données. Après conversion des arbres en dépendances, on peut voir que les résultats `Best` correspondent à ceux de `CLS`. BKY obtient également les meilleurs scores UAS. Pour MST et Malt, les meilleures performances (`Best`) sont obtenues en utilisant la combinaison du lemme, utilisé en tant que remplacement du mot, des classes et des traits morphologiques (mode, genre, nombre,...), qui sont eux ajoutés en tant que nouveaux traits. Malgré tout, l'ajout des classes en tant que trait, ou à titre de substitution, améliore très peu les performances des analyseurs en dépendances. Cet impact négatif est à l'opposé de ce que constatent (Koo *et al.*, 2008) sur l'anglais et le tchèque.

### 6.3.2 Autres travaux sur les regroupements lexicaux

(Versley & Rehbein, 2009) se sont penchés sur le traitement des mots inconnus dans le cadre de l'analyse syntaxique discriminative, et ont proposé un algorithme de création de classes lexicales. Tout d'abord, pour chaque mot d'un corpus étiqueté, les contextes linéaires gauches et droits sont extraits (mots frères et étiquettes). Puis, ces données sont exploitées par un algorithme de classification *k-moyennes*[148], qui regroupe les mots selon leur contexte. Les classes lexicales obtenues en sortie de l'algorithme sont ensuite utilisées comme traits dans l'analyseur discriminatif. Cependant, nous ne sommes pas en mesure d'indiquer l'impact de cette méthode, car seules les performances globales de l'analyseur sont précisées, incluant des traits d'origines très différentes.

Récemment, (Agirre *et al.*, 2011) ont expérimenté l'utilisation de patrons de traits sémantiques dans l'analyseur en dépendances Malt. Pour ce faire, ils ont repris les différentes méthodes d'obtention des classes sémantiques décrites dans (Agirre *et al.*, 2008). La granularité en terme de classes offerte par le thésaurus WordNet permet de créer des combinaisons riches de traits. Les résultats significatifs qu'ils observent sur les corpus SemCor et PTB-DEP prouvent encore une fois l'importance des informations sémantiques dans un processus comme l'analyse syntaxique[149].

(Sagae & Gordon, 2009) décrivent une autre technique permettant d'extraire des classes de

[148]L'algorithme de classification k-moyennes est aussi appelé k-means en anglais. (Versley & Rehbein, 2009) utilisent une implémentation particulière, *CLUTO* (Steinbach *et al.*, 2000).

[149]D'autres expériences font état de l'utilisation d'informations sémantiques afin d'aider à l'analyse d'un texte (Bikel, 2000; Lin *et al.*, 2009).

mots à partir d'un grand corpus brut. Ils ont remarqué que la plupart des algorithmes de classification utilisés dans les précédents travaux regroupent les mots grâce aux contextes linéaires lexicaux (Koo *et al.*, 2008; Versley & Rehbein, 2009). Leur approche diffère dans le sens où les classes sont créées à partir d'informations syntaxiques. Tout d'abord, un corpus brut[150] est analysé syntaxiquement par un analyseur en constituants. Puis, pour chaque mot de ce corpus, on extrait les chemins syntaxiques partant du noeud terminal jusqu'à la racine des arbres. Une mesure de similarité entre les chemins permet d'effectuer le regroupement en classes. Ces chemins étant des structures complexes, le regroupement n'a été réalisé que pour les 5000 mots les plus fréquents du corpus, afin d'obtenir des classes pertinentes. Elles sont ensuite utilisées comme traits d'un analyseur discriminatif basé sur un modèle shift-reduce (Sagae & Tsujii, 2008). L'utilisation des classes de mots ainsi générées permettent à un analyseur LR de gagner +1.6 en terme de score UAS par rapport aux résultats de base de cet analyseur sur le corpus d'évaluation du PTB-DEP.

## 6.4 Regroupements lexicaux générés à partir des lexiques syntaxiques du français

Au cours des sections précédentes, nous avons pu voir que l'utilisation de classes lexicales permettait d'améliorer significativement les performances de systèmes d'analyse syntaxique, et ceci pour diverses langues dont le français. Dans le cadre de l'analyse en constituants, la majeure partie des expériences ont pris comme formalisme d'étude les grammaires PCFG. Ces expériences ont consisté à remplacer les mots du corpus par des classes, calculées soit à partir d'un lexique (Candito & Crabbé, 2009; Candito & Seddah, 2010), soit par un algorithme de classification automatique basé sur des informations de coocurrences, tirées d'un grand corpus non annoté (Candito & Crabbé, 2009; Candito & Seddah, 2010; Candito *et al.*, 2010b). Ces algorithmes de création de classes ont été également exploités pour l'analyse en dépendances. Les meilleurs analyseurs étant basés sur un modèle discriminant, les classes sont utilisées sous forme de traits, soit en remplacement d'une propriété associée à un token (mot, lemme,...), soit en tant que nouvelle propriété du token (Koo *et al.*, 2008; Candito & Crabbé, 2009; Candito & Seddah, 2010). Dans le cadre de notre travail, les classes de mots, générées à partir des lexiques syntaxiques du français, sont grandement inspirées des travaux évoqués.
Les lexiques, que nous avons décrits dans le chapitre I.4, ont en commun la description syntaxique d'entrées verbales. Cependant, certains d'entre eux proposent également une description de mots d'autres catégories grammaticales. Pour le Lglex et le Lglex-Le*fff*, il s'agit des noms prédicatifs, et pour le Le*fff*, des adjectifs. Nous avons donc pu générer et évaluer des ensembles de classes pour ces trois catégories grammaticales. Il nous paraît important de rappeler que chaque lexique a ses spécificités en terme de propriétés codées. Seul le Lglex propose, par exemple, une classification des entrées sous forme de tables, chacune ayant un identifiant unique. Quant au Le*fff*, il contient des entrées dont les arguments du cadre de sous-catégorisation sont annotés par des fonctions syntaxiques, propriété qui est étrangère aux en-

[150](Sagae & Gordon, 2009) utilisent le corpus journalistique BLLIP.

trées du Lglex[151]. Bien qu'il y ait des différences conceptuelles plus ou moins prononcées entre les lexiques, nous pouvons néanmoins distinguer deux groupes distincts de lexiques, d'après les propriétés des entrées qu'ils renseignent. Le premier groupe est uniquement composé du Lglex pour la raison que nous venons d'évoquer. Tous les autres lexiques font partie du deuxième groupe, car ils représentent les cadres de sous-catégorisation sous la forme de fonctions syntaxiques liées aux arguments. Pour chacun de ces deux groupes de lexiques, nous avons donc généré des classes de mots à partir de propriétés bien spécifiques.
Dans la section suivante, nous décrivons notre algorithme de génération des classes à partir des lexiques. Ensuite, dans la section 6.4.3, nous détaillons et évaluons les divers ensembles de classes de verbes, obtenus en fonction des propriétés présentes dans les lexiques. Dans la section 6.4.4, nous faisons de même avec les classes d'autres catégories grammaticales, à savoir les adjectifs et les noms. Puis, dans la section 6.4.5, nous évaluons quelques expériences de combinaisons de classes de différentes catégories grammaticales, et provenant possiblement de différents algorithmes de classification, tels que DFL ou StatClust. Et enfin, section 6.5, nous discutons des résultats observés dans les sections précédentes grâce à une analyse d'erreurs poussée.

### 6.4.1 Méthodologie pour la création et l'utilisation des classes lexicales

Le processus de création des classes que nous proposons est dit *générique*, dans le sens où il peut être appliqué à tout lexique. Il suffit, en effet, de définir, à l'initialisation, la ou les propriétés des entrées à extraire afin de déterminer des affinités entre les mots, et donc de former, par la suite, des classes. Pour illustrer pas à pas l'algorithme, nous prenons comme exemple un petit lexique fictif composé de cinq entrées verbales (tableau 6.3), chacune étant associée à des propriétés ($p_1$, $p_2$, $p_3$), dont les valeurs sont, ici, des chiffres : `1` indique que la propriété est associée à l'entrée, sinon `0`. La colonne `Catégorie` précise l'étiquette morpho-syntaxique de l'entrée. Le but souhaité est de créer des classes de verbes à partir des entrées de ce lexique, en ne prenant en compte que les propriétés $p_1$ et $p_2$.

| | souper | soupes | manger | avoir |
|---|---|---|---|---|
| Catégorie | VINF | V | VINF | VINF |
| $p_1$ | 1 | 1 | 1 | 1 |
| $p_2$ | 1 | 0 | 1 | 0 |
| $p_3$ | 0 | 0 | 0 | 1 |

TAB. 6.3: Lexique d'exemple composé de cinq entrées verbales. Les valeurs des propriétés sont repérées par des chiffres : `1` indique que la propriété est associée à l'entrée, sinon `0`.

La première étape consiste à extraire toutes les entrées lemmatisées. Dans notre exemple, nous ne gardons donc que trois entrées car *soupes* est à l'indicatif. Puis, pour chaque lemme distinct présent parmi les entrées[152], un vecteur vide lui est attribué (tableau 6.5a). Ensuite, pour chaque

[151]Cependant, cette propriété est présente dans la version Lglex-Le*fff* du LG, générée à partir du Lglex.
[152]Un lemme peut avoir plusieurs entrées distinctes selon ses différents sens ou ses cadres de sous-catégorisation.

lemme et pour chaque entrée associée, la ou les valeurs des propriétés spécifiées sont ajoutées au vecteur (tableau 6.5b). Les propriétés que nous souhaitons utiliser étant $p_1$ et $p_2$, les valeurs de la propriété $p_3$ ne sont pas conservées. Et enfin, l'étape finale consiste à attribuer un même identifiant à des vecteurs identiques (tableau 6.5c). Ces identifiants uniques (`Id.`) représentent les classes de mots. On peut voir que les verbes *souper* et *manger* ont des vecteurs identiques, et appartiennent donc à la même classe.

| souper | manger | avoir |
|---|---|---|
| ∅ | ∅ | ∅ |

(a) Initialisation : Attribution de vecteurs vides aux lemmes.

| | souper | manger | avoir |
|---|---|---|---|
| $p_1$ | 1 | 1 | 1 |
| $p_2$ | 1 | 1 | 0 |

(b) Remplissage des vecteurs par les valeurs des propriétés.

| | souper | manger | avoir |
|---|---|---|---|
| Id. | 1 | 1 | 2 |

(c) Classes de mots finales.

FIG. 6.5: Étapes de création des classes de verbes à partir d'un lexique.

Les classes que nous venons de calculer doivent être ensuite appliquées sur les corpus du français, FTB-UC et FTB-DEP. Pour ce faire, nous nous sommes inspiré des précédents travaux sur les regroupements lexicaux. Aussi, dans le FTB-UC, chaque mot est remplacé par sa classe de mots. Si un mot n'est associé à aucune classe[153], alors le mot reste inchangé. Le remplacement d'un mot par une classe permet de réduire drastiquement la taille du lexique du corpus. En revanche, pour reprendre une observation faite dans (Candito & Seddah, 2010), une telle opération conduit à une perte importante d'informations morphologiques. Ces informations sont cruciales car elles permettent à l'analyseur de déterminer correctement les étiquettes morpho-syntaxiques, et donc les projections syntaxiques en constituants (section 6.2.2). Pour résoudre ce problème, (Candito & Seddah, 2010) ont créé des classes en combinant des informations lexicales, en l'occurrence les lemmes, avec les étiquettes morpho-syntaxiques. En suivant leur méthode, nous avons donc remplacé chaque mot[154] par sa classe lexicale combinée avec son étiquette morpho-syntaxique. La figure 6.6 montre un exemple d'application de notre algorithme sur l'arbre de la phrase *Paul a soupé avec ses nouveaux voisins*. On suppose que l'on connaît les lemmes des mots. Dans l'arbre original, montré dans la figure 6.6a, on peut distinguer deux verbes, l'un à l'indicatif de lemme *avoir* et l'autre, au participe passé de lemme *souper*. D'après le tableau 6.5c, ce dernier fait partie de la classe `1`, ce qui conduit à remplacer cette occurrence du verbe par le mot `1VPP`. De même, le verbe *avoir* a pour classe 2, alors il est remplacé par `2V`.

En ce qui concerne le FTB-DEP, qui est au format CoNLL (section I.1.3), il existe deux possibilités, à savoir le remplacement d'une propriété quelconque par sa classe, ou la création d'une propriété distincte. Le problème précédent de la perte d'indices morphologiques n'a pas

[153]En d'autres termes, le mot a soit une catégorie grammaticale différente des mots des classes, soit il n'est pas couvert par le lexique.

[154]Uniquement les mots présents dans la classification.

```
( (SENT (NP (NPP Paul))
    (VN (V a) (VPP soupé))
    (PP (P avec) (NP (DET ses) (ADJ nouveaux) (NC voisins)))
    (PONCT .)))
```

(a) Arbre original.

```
( (SENT (NP (NPP Paul))
    (VN (V 2V) (VPP 1VPP))
    (PP (P avec) (NP (DET ses) (ADJ nouveaux) (NC voisins)))
    (PONCT .)))
```

(b) Arbre modifié.

FIG. 6.6: Modification d'un arbre de constituants grâce aux classes de verbes générées à partir du lexique d'exemple.

lieu d'être ici car, pour chaque mot, ces informations sont incluses dans différentes propriétés. Aussi, l'étiquette morpho-syntaxique n'est pas combinée à la classe lexicale. La figure 6.7 montre les deux possibilités de modification de l'arbre de dépendances de la phrase *Paul a soupé avec ses nouveaux voisins*. Dans le premier arbre (figure 6.7a), les classes ont été ajoutées dans la colonne `Mot`, en remplacement des mots originaux[155]. Au contraire, dans le deuxième arbre (figure 6.7b), une nouvelle propriété, et donc une nouvelle colonne, a été créée et contient les classes de verbes. Dans le cadre d'expériences sur le français, (Candito *et al.*, 2010b) ont montré que la deuxième solution était plus adaptée aux modèles discriminants, car des traits peuvent être calculés à la fois sur le mot et sur les classes de mots, et non pas uniquement sur les classes. Aussi, par la suite, nous ne nous pencherons que sur cette solution pour intégrer nos classes de mots.

### 6.4.2 Protocole d'expérimentation

Les expériences que nous avons réalisées sont d'abord basées sur la génération d'ensembles de classes à partir des propriétés des lexiques, puis, sur l'évaluation de l'impact de ces classes dans des systèmes d'analyse syntaxique. Nous avons défini un protocole d'expérimentation presque identique pour chacune des expériences. Trois analyseurs en constituants, Brown, BKY[156] et $BKY_p$[157], et l'analyseur en dépendances Malt, sont entraînés sur le corpus d'apprentissage du FTB-UC altéré par les classes. Nous avons, dans un premier temps, évalué ces analyseurs sur les corpus de développement et d'évaluation. Pour chaque expérience, nous avons effectué deux séries d'évaluation. L'une permet de déterminer les scores de référence (les performances maxi-

[155]Pour l'exemple, nous avons ajouté le symbole # devant les numéros de classe afin d'écarter toute ambiguité avec les indices des mots.

[156]Version améliorée pour le traitement des mots rares et inconnus pour l'analyse de textes en langue française. Comme dans la section I.2.3.4, nous avons généré 8 grammaires, chacune ayant une graine aléatoire comprise entre 1 et 8.

[157]Version basée sur le produit de grammaires. Pour ce faire, les 8 grammaires générées pour BKY sont réutilisées.

| Indice | Mot | Lemme | ID_Gouv | Fct_Synt | T(FTB-UC) | Prop_Morph |
|---|---|---|---|---|---|---|
| 1 | Paul | Paul | 2 | SUJ | NPP | g=m\|n=s\|s=p |
| 2 | #1 | avoir | 3 | AUX_TPS | V | m=ind\|n=s\|p=3\|t=pst |
| 3 | #2 | souper | 0 | R00T | VPP | g=m\|m=part\|n=s\|t=past |
| 4 | avec | avec | 3 | MOD | P | - |
| 5 | ses | son | 7 | DET | DET | g=m\|n=s\|p=3\|s=poss |
| 6 | nouveaux | nouveau | 7 | MOD | ADJ | g=m\|n=p\|s=qual |
| 7 | voisins | voisin | 3 | OBJ | NC | g=m\|n=p\|s=c |
| 8 | . | . | 3 | PONCT | PONCT | - |

(a) Remplacement de la propriété *Mot*.

| Indice | Mot | Lemme | ID_Gouv | Fct_Synt | T(FTB-UC) | Prop_Morph | Classe |
|---|---|---|---|---|---|---|---|
| 1 | Paul | Paul | 2 | SUJ | NPP | g=m\|n=s\|s=p | - |
| 2 | a | avoir | 3 | AUX_TPS | V | m=ind\|n=s\|p=3\|t=pst | #2 |
| 3 | soupé | souper | 0 | R00T | VPP | g=m\|m=part\|n=s\|t=past | #1 |
| 4 | avec | avec | 3 | MOD | P | - | - |
| 5 | ses | son | 7 | DET | DET | g=m\|n=s\|p=3\|s=poss | - |
| 6 | nouveaux | nouveau | 7 | MOD | ADJ | g=m\|n=p\|s=qual | - |
| 7 | voisins | voisin | 3 | OBJ | NC | g=m\|n=p\|s=c | - |
| 8 | . | . | 3 | PONCT | PONCT | - | - |

(b) Ajout d'une nouvelle propriété.

FIG. 6.7: Deux possibilités de modification d'un arbre de dépendances (simplifié) par les classes de verbes générées à partir du lexique d'exemple.

males) que l'on pourrait observer avec un algorithme spécifique. Pour ce faire, les classes correctes sont attribuées aux mots des textes bruts des deux corpus à analyser. Quant à la deuxième série d'évaluation, les corpus bruts originaux sont donnés en entrée des analyseurs. Préalablement à l'analyse, il est donc nécessaire d'effectuer les étapes de reconnaissance, d'étiquetage et de lemmatisation de l'ensemble des mots de ces textes. Les gains bruts de nos algorithmes sont donc obtenus avec cette expérience. Dans un deuxième temps, les ensembles de classes obtenant les meilleurs résultats sont soumis à une analyse d'erreurs plus poussée. Nous utiliserons notamment la méthode de validation croisée afin de valider les phénomènes, observés les corpus de développement et d'évaluation, sur un échantillon d'évaluation aussi grand que le FTB-UC. Nous discutons des résultats dans la section 6.5. Dans cette section, nous avons également indiqué les scores obtenus par Malt. Le temps de calcul à l'apprentissage de Malt étant excessivement long[158], les évaluations que nous avons effectuées ne concernent qu'un certain nombre d'ensemble de classes.
L'étiquetage des mots des corpus bruts a été réalisé avec LGTagger. Comme nous l'avons vu au cours du chapitre précédent, cet étiqueteur possède des performances élevées sur le français grâce à des traits basés sur les mots composés, dont les informations proviennent de ressources

[158]L'apprentissage sur le corpus original, avec un jeu de 37 patrons de traits, prend par exemple environ 20 heures sur une machine à double coeur.

externes (dictionnaires, grammaires,...). Nous avons indiqué, dans le tableau 6.4, les performances générales de cet étiqueteur (avec les traits `lex`) sur les corpus de développement et d'évaluation du FTB-UC. Le score $F_1$ pour certaines catégories grammaticales (verbes, noms et adjectifs) est également indiqué[159]. Bien que les performances soient élevées pour l'ensemble des mots, nous verrons que les 2 à 3% d'erreurs sur les deux corpus ont un impact relativement important sur les résultats que nous obtiendrons avec nos méthodes de regroupement lexical. À propos des adjectifs, on peut constater un nombre plus grand d'erreurs, avec presque 5%, et ceci majoritairement à cause de l'ambiguité avec le participé passé[160].

| | Tous | Verbes | Noms | Adjectifs |
|---|---|---|---|---|
| Développement | 97.61 | 97.59 | 97.79 | 94.27 |
| Évaluation | 97.93 | 98.04 | 98.19 | 95.24 |

TAB. 6.4: Performances de l'étiqueteur LGTagger sur les corpus de développement et d'évaluation.

Pour remplacer chaque mot des corpus par sa classe, il est nécessaire d'avoir l'étiquette morpho-syntaxique, mais également le lemme. Le lemmatiseur que nous avons choisi est inclus dans l'outil Bonsaï. Les lemmes sont directement tirés du lexique Le*fff* et, pour le cas des mots inconnus, des heuristiques basées sur les suffixes permettent de prédire un lemme approchant. Les performances de cet algorithme sont montrées dans le tableau 6.5. Bien que les scores d'étiquetage des noms soient au même niveau que les verbes, on peut remarquer que la lemmatisation est en revanche beaucoup moins performante. Ceci est dû au nombre important de formes nominales inconnues dans le Le*fff*, avec environ 190 noms inconnus présents dans les deux corpus, qui doivent donc être lemmatisées avec les heuristiques. Comparativement, il y a 4 fois moins de formes verbales inconnues.

| | Tous | Verbes | Noms | Adjectifs |
|---|---|---|---|---|
| Développement | 96.70 | 97.07 | 95.87 | 95.79 |
| Évaluation | 96.85 | 97.34 | 95.72 | 95.95 |

TAB. 6.5: Performances du lemmatiseur inclus dans Bonsaï sur les corpus de développement et d'évaluation.

### 6.4.3 Classes de verbes

En ce qui concerne les classes de verbes, nous avons exploité, pour le cas du Lglex : (a) la classification des entrées offerte par ce lexique, (b) la hiérarchie des classes de verbes, (c) ainsi que deux autres types de propriétés, à savoir les constructions syntaxiques et les prépositions introduisant les arguments indirects. Pour tous les autres lexiques, nous avons utilisé les fonctions syntaxiques présentes dans les cadres de sous-catégorisation liés aux entrées (d).

[159]Nous sommes capable de calculer ce score car des mots peuvent, par exemple, être incorrectement étiquetés comme verbe, ou des verbes peuvent être étiquetés incorrectement.

[160]Environ 31% des erreurs sur les deux corpus

**(a) Lglex : Classification offerte par le LG**

On peut rappeler que, dans le Lglex, chaque entrée d'une forme lemmatisée, ici un verbe, est associée à un numéro de table. Il existe au total 61 tables de verbes. De plus, si une forme lemmatisée a plusieurs sens distincts, elle possède autant d'entrées lexicales puisque chaque sens n'a pas le même ensemble de propriétés. Afin d'obtenir des classes de verbes, la première propriété que nous avons exploitée est donc la classification des entrées du LG. Le processus de génération des classes de verbes, à partir de cette propriété, est illustré par le tableau 6.6. Dans cet exemple, plusieurs verbes tirés du Lglex sont associés à leurs tables du LG sous forme de vecteurs. On peut observer que seuls les verbes *sympathiser* et *souper* possèdent un vecteur identique, ce qui conduit à attribuer à ces deux verbes un même identifiant de classe (`Id.`).

| | abolir | badiner | souper | sympathiser |
|---|---|---|---|---|
| 10 | 1 | 0 | 0 | 0 |
| 32R2 | 1 | 0 | 0 | 0 |
| 15 | 0 | 1 | 0 | 0 |
| 35R | 0 | 1 | 1 | 1 |
| 35S | 0 | 0 | 1 | 1 |
| Id. | 1 | 2 | 3 | 3 |

TAB. 6.6: Illustration du processus de génération des classes de mots à partir des tables de verbes du Lglex.

En sortie, l'algorithme a produit un ensemble de 2023 classes de mots distinctes, ce qui équivaut à environ 3 verbes par classe. Ce nombre conséquent de classes est dû à l'ambiguité importante en terme de tables du LG associées aux verbes. En effet, le nombre moyen de tables par forme verbale distincte est de 2,34. Dans le tableau 6.7, nous avons indiqué la couverture du Lglex en terme de verbes lemmatisés distincts sur les corpus de la partition du FTB-UC (apprentissage, développement ou évaluation). La couverture sur les corpus de développement et d'évaluation est presque parfaite. De plus, on peut voir que 96,5% des verbes du FTB-UC sont présents dans le Lglex, ce qui correspond à exactement 1897 verbes sur un total de 1967. Parmi les quelques verbes ne faisant pas partie de la couverture, on observe des verbes courants comme *sembler* ou *falloir*, mais également de nombreux cas de verbes préfixés par *"re"* (rebaptiser, rebâtir,...) ou *"dé"* (désengager, déconcentrer,...)[161]. Dans le tableau 6.8, nous avons indiqué la taille du lexique en mots distincts (colonne `#lexique`) du corpus FTB-UC original et de sa version modifiée par les classes de verbes. L'opération de modification du corpus a permis de réduire la taille du lexique de 3248 mots. Les colonnes `#mots inconnus` et `#classes inconnues` sont spécifiques aux corpus de développement et d'évaluation, et correspondent respectivement au nombre de mots inconnus distincts et, parmi ces mots, lesquels sont des classes de verbes. Le nombre de mots inconnus baisse significativement dans les deux corpus (-12% par rapport au corpus original). En revanche, parmi les mots inconnus restant, on peut

[161]La solution pourrait être d'associer les classes d'une forme *X* à des formes *reX* ou encore *déX*. D'une point de vue linguistique, cette solution n'est pas acceptable car les cadres de sous-catégorisation des différentes formes ne sont généralement pas identiques.

signaler qu'environ 11% des mots, soit 165 mots, sont des classes de verbes non présentes dans le corpus d'apprentissage.

| Corpus | #verbes |
|---|---|
| Apprentissage | 96.2 |
| Développement | 99.0 |
| Évaluation | 99.3 |
| **Total** | 96.5 |

TAB. 6.7: Couverture du Lglex sur les verbes des corpus de la partition du FTB-UC.

| Corpus | #lexique | #mots inconnus | #classes inconnues |
|---|---|---|---|
| **Corpus original** | | | |
| Apprentissage | 24090 | - | - |
| Développement | 7221 | 1628 | - |
| Évaluation | 7052 | 1523 | - |
| **Total** | 27125 | - | - |
| **Classes de verbes** | | | |
| Apprentissage | 21035 | - | - |
| Développement | 6782 | 1416 | 165 |
| Évaluation | 6545 | 1297 | 158 |
| **Total** | 23877 | - | - |

TAB. 6.8: Taille du lexique des corpus de la partition du FTB-UC, avant et après l'utilisation des classes de verbes.

Les scores de référence $F_1$, UAS et LA obtenus par les analyseurs sur ces deux corpus sont indiqués dans les tableaux 6.8a et 6.8b. À titre de comparaison, nous avons également indiqué les résultats de base des analyseurs entraînés et évalués sur le corpus original. Les scores significatifs sont marqués par le symbole $\diamond$. Tout d'abord, on peut observer que les gains UAS sont significatifs pour l'ensemble des analyseurs ($p < 0.02$). L'introduction de classes de verbes pourrait donc améliorer la reconnaissance de dépendances syntaxiques. En ce qui concerne le score LA, les gains sont beaucoup plus faibles (+0.2 maximum), voires nuls. De même, les scores $F_1$ sont plutôt mitigés. Alors que les gains sont supérieurs à +0.8 sur le corpus d'évaluation ($p < 0.01$), et ceci pour les 3 analyseurs, sur le corpus de développement ils ne dépassent pas +0.5. On constate même une chute à +0.15 pour $\text{BKY}_p$. Et enfin, il est intéressant de noter que les scores de base et les gains avec nos classes sont significativement meilleurs sur le corpus d'évaluation que sur celui de développement, ce qui nous renforce dans l'idée d'utiliser la validation croisée pour déterminer plus précisement l'impact réel de nos gains.
En suivant le protocole d'expérimentation, dans notre deuxième expérience, les corpus bruts de développement et d'évaluation contiennent maintenant des classes à la place de verbes qui ont été repérés, étiquetés et lemmatisés automatiquement. Les gains bruts absolus des analyseurs sur ces deux corpus sont montrés dans la figure 6.9, et sont calculés par rapport aux scores de

| Analyseur | $F_1$ | LA | UAS |
|---|---|---|---|
| **Corpus original** | | | |
| BKY | 82.63 | 86.5 | 87.67 |
| $BKY_p$ | 84.05 | 87.6 | 88.62 |
| Brown | 79.56 | 83.8 | 85.41 |
| **Classes de verbes** | | | |
| BKY | 83.01◇ | 86.6 | 88.25◇ |
| $BKY_p$ | 84.20 | 87.6 | 89.06◇ |
| Brown | 80.03◇ | 84.0 | 86.20◇ |

(a) Corpus de développement.

| Analyseur | $F_1$ | LA | UAS |
|---|---|---|---|
| **Corpus original** | | | |
| BKY | 84.26 | 86.8 | 88.22 |
| $BKY_p$ | 85.51 | 87.7 | 89.04 |
| Brown | 80.86 | 84.5 | 85.96 |
| **Classes de verbes** | | | |
| BKY | 84.82◇ | 86.9 | 88.67◇ |
| $BKY_p$ | 86.35◇ | 87.9 | 89.52◇ |
| Brown | 81.79◇ | 84.6 | 86.89◇ |

(b) Corpus d'évaluation.

FIG. 6.8: Scores de référence des analyseurs sur les corpus de développement et d'évaluation modifiés par les classes de verbes. À titre de comparaison, les scores de base des analyseurs sont également indiqués.

base des analyseurs. On peut voir que les 2% d'erreurs d'étiquetage et de lemmatisation perturbent suffisamment les modèles pour que l'ensemble des gains soient, au minimum, divisés par deux par rapport à l'expérience de référence. Seuls les gains UAS de l'analyseur lexicalisé Brown restent significatifs ($p < 0.05$), ainsi que les gains $F_1$ sur le corpus d'évaluation pour $BKY_p$ et Brown ($p < 0.02$). Ces résultats mitigés peuvent être causés par le trop grand nombre de classes disponibles. En effet, malgré la réduction de la taille du lexique du corpus, on peut rappeler que 2023 classes sont créées à partir du Lglex, et que parmi les mots inconnus présents dans les deux corpus, environ 10% sont des classes.

| | $F_1$ | LA | UAS |
|---|---|---|---|
| BKY | +0.15 | +0.1 | +0.30 |
| $BKY_p$ | -0.06 | -0.1 | +0.14 |
| Brown | +0.14 | +0.2 | +0.47◇ |

(a) Corpus de développement.

| | $F_1$ | LA | UAS |
|---|---|---|---|
| BKY | +0.21 | +0.1 | +0.10 |
| $BKY_p$ | +0.49◇ | +0.2 | +0.19 |
| Brown | +0.51◇ | = | +0.56◇ |

(b) Corpus d'évaluation.

FIG. 6.9: Gains bruts absolus des analyseurs sur les corpus de développement et d'évaluation, calculés par rapport aux résultats de base.

### (b) Lglex : Exploitation de la hiérarchie des classes de verbes

Nous venons de voir que l'utilisation de la classification des verbes offerte par le LG ne permet pas d'obtenir des résultats stables selon l'analyseur utilisé et le corpus évalué. La problématique est potentiellement liée au trop grand nombre de classes de mots générées à partir du Lglex. Ce nombre conséquent est dû à l'ambiguité importante en terme de tables associées aux entrées du lexique. Dans la section I.4.4.2, nous avons fait mention de la hiérarchie des

tables de verbes[162]. Cette hiérarchie, constituée de quatre niveaux, regroupe les tables du LG en classes plus générales. L'utilisation de cette hiérarchie permettrait de limiter l'ambiguité, et ainsi réduire le nombre de classes générées. Le but de notre expérience est d'observer si des classes de mots plus générales, et donc syntaxiquement plus grossières, améliorent les performances des analyseurs syntaxiques en réduisant l'effet de dispersion des données.
Les vecteurs de classes liés aux verbes du LG sont maintenant calculés en fonction des tables originales du LG, mais également du niveau de la hiérarchie. De même qu'auparavant, des vecteurs identiques se voient attribuer le même identifiant représentant leur classe de mots. Le processus de génération des classes est illustré par la figure 6.10. Dans cet exemple, trois verbes du Lglex sont initialement associés à leur vecteur de tables du LG, ainsi qu'à leur identifiant (tableau 6.10a). Cette étape correspond au niveau 0 de la hiérarchie, et on peut voir que les vecteurs ainsi que leur identifiant sont tous différents. Au niveau 1 de la hiérarchie, le système ne change pas malgré la mise à jour des vecteurs avec les classes correspondantes. En revanche, au niveau 2, on peut s'apercevoir que les verbes *ruisseler* et *surgir* sont associés aux mêmes classes, à savoir `TI2` et `TI3` (transitives indirectes à 2 ou 3 arguments), et possèdent donc le même identifiant. Et enfin, au dernier niveau, les trois verbes possèdent le même vecteur et donc le même identifiant de classe.

| | poireauter | ruisseler | surgir |
|---|---|---|---|
| 2 | 0 | 0 | 1 |
| 8 | 0 | 0 | 1 |
| 34L0 | 0 | 1 | 0 |
| 35LR | 1 | 0 | 0 |
| 35L | 0 | 1 | 1 |
| Id. | 1 | 2 | 3 |

(a) Tables du LG associées à 3 verbes présents dans le Lglex (niveau 0).

| | poireauter | ruisseler | surgir |
|---|---|---|---|
| TI2L | 1 | 1 | 1 |
| TIN3 | 0 | 1 | 1 |
| QTI2 | 0 | 0 | 1 |
| Id. | 1 | 2 | 3 |

(b) Niveau 1 de la hiérarchie.

| | poireauter | ruisseler | surgir |
|---|---|---|---|
| TI2 | 1 | 1 | 1 |
| TI3 | 0 | 1 | 1 |
| Id. | 1 | 2 | 2 |

(c) Niveau 2 de la hiérarchie.

| | poireauter | ruisseler | surgir |
|---|---|---|---|
| INT | 1 | 1 | 1 |
| Id. | 1 | 1 | 1 |

(d) Niveau 3 de la hiérarchie.

FIG. 6.10: Évolution de la classification de 3 verbes du Lglex en fonction des niveaux de la hiérarchie des tables de verbes.

Dans le tableau 6.11a, nous avons indiqué le nombre de classes créées selon le niveau de la hiérarchie, ainsi que le nombre moyen de verbes par classe correspondant. Au niveau 0, on démarre avec un ensemble de classes fin, jusqu'à arriver au niveau 3 avec un jeu assez grossier, où seulement 15 classes sont générées. Plus l'ensemble est de petite taille, plus la taille du lexique et le nombre de mots inconnus baissent. On peut voir, dans le tableau 6.11b, que le lexique a été réduit de 3000 mots entre les niveaux 0 et 3, et que le nombre de classes inconnues est devenu

[162]En réalité, nous avons parlé de hiérarchie des *classes* de verbes. Cependant, pour éviter un amalgame avec les classes générés par notre algorithme, nous employons à présent le mot *table*.

nul (colonne `#cls inconnues`).

| Niveau | #classes | AVG |
|---|---|---|
| 0 | 2023 | 2.9 |
| 1 | 558 | 10.6 |
| 2 | 312 | 18.9 |
| 3 | 15 | 394.8 |

(a) Nombre de classes de mots générées et nombre moyen de mots par classe.

| Niveau | #lexique | #mots inconnus | | #cls inconnues | |
|---|---|---|---|---|---|
| | | Dev | Eval | Dev | Eval |
| 0 | 23877 | 1416 | 1297 | 165 | 158 |
| 1 | 22040 | 1316 | 1192 | 65 | 60 |
| 2 | 21474 | 1281 | 1162 | 30 | 30 |
| 3 | 20529 | 1251 | 1133 | 0 | 0 |

(b) Taille du lexique du FTB-UC, et nombre de mots et de classes inconnus.

FIG. 6.11: Statistiques à propos des classes de verbes générées en fonction des niveaux de la hiérarchie des verbes.

Les résultats selon les niveaux de la hiérarchie sont montrés dans les tableaux de la figure 6.12. Les scores significatifs sont marqués par le symbole $\diamond$. Seuls les gains $F_1$[163] sont indiqués car, au final, quel que soit le niveau utilisé, l'ensemble des scores varie très peu par rapport à l'expérience précédente. On peut néanmoins remarquer que les meilleures performances sont atteintes avec le 2ème niveau de la hiérarchie[164], bien que la différence de gains avec les autres niveaux ne soit pas significative. En conclusion, le nombre de classes ne semble pas être le facteur principal des faibles gains constatés, et peut donc être écarté des causes potentielles. La classification des entrées du LG pourrait, tout simplement, ne pas être adaptée aux données présentes dans le corpus. Il est donc nécessaire d'explorer les autres propriétés des entrées.

### (c) Lglex : Exploitation des propriétés *Constructions* et *Prépositions*

En plus de la classification des entrées, le Lglex nous renseigne également sur d'autres propriétés utiles comme les constructions syntaxiques acceptées, ainsi que les prépositions introduisant les arguments indirects. En ce qui concerne les constructions, on peut rappeler qu'il existe deux types distincts de constructions :

– les constructions absolues, telles que *N0 V* ou *N0 V N1*.
– les transformations applicables à la construction de base, comme la passivation ou l'extraposition.

De plus, parmi les constructions absolues, on peut distinguer 3 sous-types, chacun identifié par un préfixe propre dans la chaîne de caractères représentant la construction :

– `base`, construction absolue de base pour l'entrée courante. Il existe 20 constructions de base différentes dans le LG. Par exemple, l'entrée du verbe *assujettir* de la table *11* a pour construction de base *N0 V N1 à N2*, comme dans la phrase :

$$\underset{N0}{\underline{Cette\ loi}}\ \underset{V}{\underline{assujettit}}\ \underset{N1}{\underline{Max}}\ \underset{\grave{a}\ N2}{\underline{\grave{a}\ ce\ qu'il\ déclare\ ses\ profits}}$$

[163] En valeur absolue par rapport aux scores de base des analyseurs.
[164] Les scores UAS et LA sont également maximisés avec le niveau 2.

| Niveau<br>Analyseur | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **Corpus de développement** | | | | |
| BKY | +0.38 | +0.42 | +0.41 | **+0.54** |
| $BKY_p$ | +0.15 | +0.22 | **+0.40** | +0.35 |
| Brown | +0.48 | +0.45 | **+0.49** | +0.47 |
| **Corpus d'évaluation** | | | | |
| BKY | +0.56 | +0.47 | **+0.58** | +0.44 |
| $BKY_p$ | +0.84 | +0.55 | **+0.87** | +0.63 |
| Brown | **+0.93** | +0.83 | +0.81 | +0.81 |

(a) Gains $F_1$ de référence.

| Niveau<br>Analyseur | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **Corpus de développement** | | | | |
| BKY | +0.15 | +0.16 | +0.15 | **+0.26** |
| $BKY_p$ | -0.06 | +0.10 | **+0.14** | +0.09 |
| Brown | +0.14 | +0.13 | **+0.17** | +0.14 |
| **Corpus d'évaluation** | | | | |
| BKY | +0.19 | +0.06 | **+0.21** | +0.06 |
| $BKY_p$ | +0.49◇ | +0.20 | **+0.50**◇ | +0.17 |
| Brown | +0.51◇ | +0.51◇ | **+0.54**◇ | +0.45◇ |

(b) Gains $F_1$ bruts.

FIG. 6.12: Gains $F_1$ absolus des analyseurs sur les corpus de développement et d'évaluation en fonction des niveaux de la hiérarchie.

– `true`, construction acceptée par l'ensemble des entrées de la table. On peut dénombrer 12 constructions de ce type dans le LG (distinctes des constructions de base). L'ensemble des verbes de la table 2 (*accourir*, *abouler*,...) accepte, par exemple, la construction *N0 V Loc N1* :

$$\underset{N0}{\underline{Luc}}\ \textit{est}\ \underset{V}{\underline{accouru}}\ \underset{Loc}{\underline{au}}\ \underset{N1}{\underline{buffet}}$$
$$\underset{N0}{\underline{Max}}\ \textit{s'est}\ \underset{V}{\underline{aboulé}}\ \underset{Loc}{\underline{au}}\ \underset{N1}{\underline{buffet}}$$

– `o`, construction acceptée par l'entrée mais pas obligatoirement par l'ensemble des entrée de la table. Il existe 116 constructions qui sont de ce type pour au moins une table. Par exemple, le verbe *bécoter* de la table *32CL* accepte une construction *N0 V N1 de V-n*, mais pas *cogner*, bien qu'ils soient dans la même table :

$$\underset{N0}{\underline{Luc}}\ \underset{V}{\underline{bécote}}\ \underset{N1}{\underline{les\ joues}}\ \underset{de\ V-n}{\underline{de\ Max}}$$
**Luc cogne les joues de Max*

Ces trois types de construction permettent de générer des ensembles de classes de différentes granularités. Dans le cadre de notre travail, nous avons donc pu évaluer l'impact de multiples configurations de combinaison d'informations. Pour des raisons de clarté, nous indiquons, dans ce mémoire, uniquement les résultats de deux configurations, l'une basée sur des classes générées à partir des constructions de base, notée `Base`, et l'autre sur la totalité des constructions disponibles (`base`, `true`, `o` et les transformations), appelée `All`. Dans le tableau 6.9, nous avons indiqué le nombre de classes créées, ainsi que la taille du lexique du FTB-UC, en fonction de la configuration utilisée.

| | #classes | AVG | #lexique |
|---|---|---|---|
| Base | 761 | 7.78 | 22331 |
| All | 2304 | 2.57 | 23844 |

TAB. 6.9: Nombre de classes générées, et taille du lexique du FTB-UC correspondante, selon la configuration utilisée.

Les scores $F_1$ selon la configuration utilisée sont montrés dans les tableaux de la figure 6.12. On peut noter que tous les scores de référence sont significatifs ($p < 0.02$). Pour les scores bruts, les gains significatifs sont marqués par le symbole $\diamond$. On peut observer que, pour les deux configurations, les performances sont du même ordre que celles des expériences précédentes, bien que la configuration All ait des scores légèrement plus élevés.

| Configuration Analyseur | Base | All |
|---|---|---|
| **Corpus de développement** | | |
| BKY | +0.44 | **+0.55** |
| $BKY_p$ | **+0.45** | +0.37 |
| Brown | +0.36 | **+0.48** |
| **Corpus d'évaluation** | | |
| BKY | +0.32 | **+0.45** |
| $BKY_p$ | +0.42 | **+0.65** |
| Brown | +0.83 | **+0.89** |

(a) Gains $F_1$ de référence.

| Configuration Analyseur | Base | All |
|---|---|---|
| **Corpus de développement** | | |
| BKY | +0.18 | **+0.28** |
| $BKY_p$ | +0.13 | **+0.25** |
| Brown | +0.04 | **+0.19** |
| **Corpus d'évaluation** | | |
| BKY | -0.04 | **+0.11** |
| $BKY_p$ | +0.11 | **+0.35**$^\diamond$ |
| Brown | +0.46$^\diamond$ | **+0.53**$^\diamond$ |

(b) Gains $F_1$ bruts.

FIG. 6.13: Gains $F_1$ absolus des analyseurs sur les corpus de développement et d'évaluation en fonction de la configuration utilisée.

Nous allons nous pencher, à présent, sur les propriétés liées aux prépositions. Dans le Lglex, une liste de prépositions possibles est associée à tout complément indirect d'une entrée. Le Lglex contient un jeu de 52 prépositions différentes (à, de, pour,...). Par exemple, l'entrée de la table 18 du verbe *donner* nous indique que la préposition du premier complément est *à*, celle du deuxième est *comme*, et que la dernière est une préposition obligatoirement vide. La construction de base associée à cette entrée étant *N0 V Prép N1 Prép N2 Prép N3*, la phrase d'exemple suivante intègre l'ensemble des informations énoncées :

$$\underset{N0}{\underline{Max}}\ \underset{V}{\underline{a\ donné}}\ \underset{Prep\ N1}{\underline{à\ Luc}}\ \underset{Prep\ N2}{\underline{comme\ principe}}\ \underset{Prep\ N3}{\underline{qu'il\ devait\ avouer}}$$

De même que pour la propriété des constructions syntaxiques, il est possible d'évaluer de multiples configurations. Les scores les plus élevés et les plus stables sont obtenus avec une configuration qui associe, à chaque verbe, le vecteur de prépositions contenues dans ses entrées. On peut noter que les prépositions sont ajoutées aux vecteurs de manière indépendante du complément auquel elles étaient liées. Par exemple, en supposant que le verbe *donner* ne soit associé qu'à l'unique entrée décrite précédemment, le vecteur correspondant est composé des prépositions *à, comme* et *<E>*[165]. Dans le tableau 6.10, nous avons indiqué le nombre de classes créées par cette méthode (colonne `#classes`), ainsi que la taille du lexique correspondante (colonne `#lexique`).

| #classes | AVG | #lexique |
|---|---|---|
| 257 | 23 | 21278 |

TAB. 6.10: Nombre de classes générées à partir des prépositions, et taille du lexique du FTB-UC correspondante.

Les gains $F_1$ absolus obtenus avec cet ensemble de classes sont montrés dans la figure 6.14. Ces gains sont légèrement supérieurs, mais non significativement, à ceux de l'expérience précédente. Malgré tout, les résultats sont plus stables sur les deux corpus évalués qu'avec les autres propriétés. Les ensembles de classes profitent plus à des analyseurs lexicalisés comme Brown, et moins à ceux d'algorithmes non-lexicalisés comme ceux de BKY et BKY$_p$.

| | référence | brut |
|---|---|---|
| BKY | +0.65◇ | +0.40◇ |
| BKY$_p$ | +0.45◇ | +0.22 |
| Brown | +0.64◇ | +0.32◇ |

(a) Corpus de développement.

| | référence | brut |
|---|---|---|
| BKY | +0.58◇ | +0.19 |
| BKY$_p$ | +0.72◇ | +0.36◇ |
| Brown | +0.91◇ | +0.57◇ |

(b) Corpus d'évaluation.

FIG. 6.14: Gains $F_1$ absolus des analyseurs sur les corpus de développement et d'évaluation avec les classes générées à partir des prépositions.

Comme nous l'avions déjà remarqué pour la hiérarchie des classes de verbes, il semblerait que le nombre de classes joue peu sur les performances effectives des analyseurs car, pour des performances similaires, on peut constater qu'il y a 10 fois moins de classes générées à partir des prépositions qu'avec les constructions syntaxiques. À ce point précis des expériences, deux raisons pourraient expliquer ces résultats mitigés. D'une part, les regroupements lexicaux basés sur le Lglex pourraient être en contradiction avec les phénomènes syntaxiques rencontrés dans les corpus. Il y a une autre explication possible, le fait que le LG distingue les sens des verbes, et que ces sens ne sont pas annotés dans les corpus d'apprentissage. La prise en compte des sens ne peut donc provenir que d'un apprentissage de ceux-ci à partir des constructions syntaxiques ou du contexte, ce qui nécessiterait des masses énormes d'occurrences de chaque

[165]Le symbole <E> indique l'absence de préposition.

sens. Pour déterminer si ces deux raisons sont à l'origine des faibles gains, nous avons, dans la section suivante, effectué des regroupements lexicaux à partir d'autres lexiques du français.

**(d) Autres lexiques : Classes de verbes obtenues à partir des fonctions syntaxiques**

Le deuxième groupe de lexiques est composé de Dicovalence, du Lglex-Le*fff*, du Le*fff* et de LexSchem. Dans ces lexiques, chaque entrée est associée à un cadre de sous-catégorisation, dont les arguments possibles des verbes sont représentés sous la forme de fonctions syntaxiques. À titre d'illustration, nous avons indiqué, dans le tableau 6.11, le cadre de sous-catégorisation[166] d'entrées lemmatisées du verbe *adopter*. Pour chaque lexique, on peut trouver une entrée composée d'un cadre défini par deux arguments : un sujet et un objet direct.

| Lexique | Cadre de sous-catégorisation |
|---|---|
| Dicovalence | subj :pron\|n :[hum], obj :pron\|n :[hum] |
| Le*fff* | Suj :cln\|scompl\|sinf\|sn, Obj :cla\|scompl\|sn |
| Lglex-Le*fff* | Suj :cln\|sn, Obj :sn\|scompl\|de-sinf |
| LexSchem | SUJ :SN, OBJ :SN |

TAB. 6.11: Cadre de sous-catégorisation d'entrées du verbe *adopter* pour différents lexiques.

Ces lexiques possèdent un jeu de fonctions syntaxiques relativement différent, mais dont on peut extraire une base similaire composée de 5 fonctions[167] : `Suj` (sujet), `Obj` (objet direct), `Objde` (objet indirect introduit par la préposition *de*), `Objà` (objet indirect introduit par la préposition *à*) et `Attr` (attribut)[168]. Dans le tableau 6.12, nous avons indiqué les fonctions additionnelles propres à chaque lexique[169]. Tout d'abord, on peut remarquer que Dicovalence propose le jeu le plus fin, car il possède 8 étiquettes de fonctions en plus de celles de base. À l'opposé, LexSchem ne contient que deux étiquettes supplémentaires, et est le seul à ne pas introduire l'argument locatif (`Loc`). On peut rappeler que ce lexique est généré en partie grâce aux relations de dépendances fournies par l'analyseur Syntex (Bourigault *et al.*, 2005). Or, cet analyseur ne permet pas de déterminer si une préposition a une valeur locative. Et enfin, l'unique différence entre les jeux du Lglex-Le*fff* et du Le*fff* provient de la présence d'un troisième argument oblique possible dans les entrées du Lglex-Le*fff*[170].
Pour chaque lexique, la génération des classes est effectuée en fonction de la propriété des fonctions syntaxiques. Chaque lemme s'est vu attribué un vecteur de fonctions extraites des cadres de sous-catégorisations des différentes entrées associées à ce lemme. D'autres configurations sont possibles, notamment en ajoutant les réalisations syntagmatiques, mais les résultats sont beaucoup moins satisfaisants. Dans le tableau 6.13, nous avons indiqué le nombre de classes créées (colonne `#classes`) en fonction du lexique utilisé. Le nombre d'entrées par classe

[166]Pour l'exemple, les réalisations syntagmatiques associées aux fonctions sont conservées.
[167]Après normalisation des chaînes de caractères représentant les fonctions.
[168]On peut noter que, dans LexSchem, la fonction attribut est divisée en deux symboles, `ATTS` (du sujet) et `ATTO` (de l'objet).
[169]La description des fonctions est disponible dans la section I.4.
[170]Cet argument n'est présent que dans deux entrées du lexique, pour les verbes *payer* et *prévaloir*.

| Lexique | Fonctions |
|---|---|
| Dicovalence | Obj<P>, Temp, Loc, Px, Quant, Eval, DeLoc, Man |
| Le*fff* | Loc, DeLoc, Obl, Obl2 |
| Lglex-Le*fff* | Loc, DeLoc, Obl, Obl2, Obl3 |
| LexSchem | Obj<P>, Ref |

TAB. 6.12: Fonctions syntaxiques propres à chaque lexique syntaxique.

correspondant est indiqué dans la colonne `AVG`. Avec le jeu fin disponible dans Dicovalence, l'algorithme de génération des classes a produit un ensemble de grande taille, comparativement au nombre de lemmes. Il est intéressant de noter que le nombre d'entrées, deux fois plus important dans le Lglex-Le*fff* que dans le Le*fff*, se répercute directement sur le nombre de classes créées, qui est également deux fois plus conséquent. Avec une meilleure couverture des verbes du Le*fff*, on peut remarquer que le lexique du corpus est réduit d'environ 2% supplémentaires par rapport aux autres lexiques syntaxiques (colonne `#lexique`).

| Lexique | #classes | AVG | #lexique |
|---|---|---|---|
| Dicovalence | 189 | 27 | 21188 |
| Le*fff* | 43 | 158 | 20566 |
| Lglex-Le*fff* | 114 | 50 | 20904 |
| LexSchem | 34 | 92 | 20727 |

TAB. 6.13: Nombre de classes générées à partir des fonctions syntaxiques des cadres de sous-catégorisation, et taille du lexique du FTB-UC correspondante.

Les scores $F_1$ selon le lexique utilisé sont montrés dans les tableaux de la figure 6.12. On peut noter que tous les scores de référence sont significatifs ($p< 0.02$). Les meilleurs gains bruts sont obtenus en majorité par LexSchem[171], bien qu'ils soient au même niveau que ceux de l'expérience précédente basée sur les prépositions des entrées du Lglex. En conclusion, ces différents scores montrent que, quel que soit le lexique utilisé, notre méthode de regroupement lexical des verbes produit des résultats similaires. Les gains sont globalement significatifs pour Brown, et ceci indépendamment du corpus évalué, alors que pour BKY et $BKY_p$, ils sont beaucoup moins stables. Afin de montrer l'impact réel des classes de verbes sur le corpus, nous avons appliqué la méthode de validation croisée avec des analyseurs entraînés sur un corpus modifié par les classes (section 6.5).

### 6.4.4 Classes de mots non verbaux

Nous venons de montrer plusieurs façons d'obtenir des classes de verbes à partir des propriétés des entrées des lexiques. Comme dit précédemment, certains lexiques intègrent une description syntaxique d'éléments autres que verbaux. Pour le Lglex et le Lglex-Le*fff*, il s'agit des noms prédicatifs, et pour le Le*fff*, des adjectifs. Pour chacune de ces catégories grammaticales, nous

[171] C'est le cas également pour les scores UAS et LA.

| Analyseur / Lexique | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Dicovalence | +0.52 | +0.48 | +0.60 |
| Le*fff* | +0.61 | +0.47 | **+0.65** |
| Lglex-Le*fff* | +0.60 | +0.34 | +0.40 |
| LexSchem | **+0.80** | **+0.54** | +0.53 |
| **Corpus d'évaluation** | | | |
| Dicovalence | +0.55 | +0.63 | +0.81 |
| Le*fff* | +0.53 | **+0.68** | +0.79 |
| Lglex-Le*fff* | +0.56 | +0.63 | **+0.93** |
| LexSchem | **+0.63** | +0.67 | +0.87 |

(a) Gains $F_1$ de référence.

| Analyseur / Lexique | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Dicovalence | +0.25 | +0.17 | **+0.34**$^\diamond$ |
| Le*fff* | +0.31$^\diamond$ | +0.12 | +0.30$^\diamond$ |
| Lglex-Le*fff* | +0.32$^\diamond$ | +0.09 | +0.08 |
| LexSchem | **+0.52**$^\diamond$ | **+0.23** | +0.31$^\diamond$ |
| **Corpus d'évaluation** | | | |
| Dicovalence | +0.18 | +0.27 | +0.55$^\diamond$ |
| Le*fff* | +0.14 | +0.23 | +0.41$^\diamond$ |
| Lglex-Le*fff* | +0.14 | +0.21 | +0.60$^\diamond$ |
| LexSchem | **+0.29**$^\diamond$ | **+0.33**$^\diamond$ | **+0.61**$^\diamond$ |

(b) Gains $F_1$ bruts.

FIG. 6.15: Gains $F_1$ absolus des analyseurs sur les corpus de développement et d'évaluation en fonction des lexiques.

avons généré des classes grâce aux propriétés propres à chaque groupe de lexiques. On peut rappeler que, pour le Lglex, il s'agit de la classification proposée par le LG, les constructions syntaxiques et les prépositions. Pour les autres lexiques, nous utilisons les cadres de sous-catégorisation à base de fonctions syntaxiques.

### (a) Lglex : Classes de noms

Dans le LG, il existe actuellement 79 tables de noms pour 10112 lemmes distincts. Pour chaque entrée, diverses informations sont renseignées comme les constructions absolues, les réductions possibles du groupe nominal, ou encore une liste de verbes supports acceptés. À titre d'illustration, nous avons répertorié, dans le tableau 6.14, ces différentes propriétés pour chaque entrée du nom prédicatif *bienveillance*[172]. On peut voir, par exemple, que les trois entrées acceptent le verbe support *avoir*, mais aussi, que l'entrée de la table *N_aa* accepte la construction *N0 avoir Det N Prép N1*, comme dans :

$$\underset{N0}{\underline{Max}}\ \underset{avoir}{\underline{a}}\ \underset{Det}{\underline{une}}\ \textit{certaine}\ \underset{N}{\underline{bienveillance}}\ \underset{Prep}{\underline{(pour + envers + \grave{a}\ l'\acute{e}gard\ de)}}\ \underset{N1}{\underline{Luc}}$$

Cependant, à l'heure actuelle, les tables de noms prédicatifs sont grandement incomplètes, contrairement aux tables de verbes. En effet, un nombre important d'entrées de noms prédicatifs n'existent encore dans aucune table, comme par exemple *cinéaste* et *lecteur*. Les résultats des expériences que nous indiquons dans cette section sont donc fortement influencés par cette faible complétude des données.
Les propriétés que nous avons exploitées pour la création des classes de noms sont : la classification proposée par le LG, les verbes supports et les constructions syntaxiques de base. Le

[172]Pour l'exemple, seulement deux constructions par entrée ont été montrées bien qu'il y en ait plus en réalité.

| Classe du LG | Verbes supports | Constructions |
|---|---|---|
| N_aa | avoir, éprouver, ressentir | N0 avoir Det N Prép N1 ; N1 avoir Det N de N0 |
| N_an03 | avoir | N0 avoir Det N ; N0 être de Det N Modif |
| N_ape21 | avoir, perdre | N0 avoir Det N de N1 ; N1 avoir Det N Prép N0 |

TAB. 6.14: Propriétés, et leurs valeurs, des entrées du nom prédicatif *bienveillance* dans le Lglex.

processus de création des classes est illustré dans la figure 6.16 pour ces différentes propriétés, représentées sous la forme de vecteurs associés aux entrées. Tout d'abord, en ce qui concerne la classification du LG, on peut voir que les noms *imposition* et *recyclage* sont dans les mêmes tables *dr1* et *fs1*, et ont donc le même identifiant `1` (tableau 6.16a). Le nom *prorogation* est quant à lui uniquement dans la table *fs1*. De plus, on peut voir que la classification est identique lorsqu'on se base sur la propriété des constructions (tableau 6.16b). En revanche, avec les verbes supports, la classification est différente car *prorogation* et *recyclage* possèdent une liste de verbes identique, contrairement à *imposition*.

| | imposition | recyclage | prorogation |
|---|---|---|---|
| dr1 | 1 | 1 | 1 |
| fs1 | 1 | 1 | 0 |
| Id. | 1 | 1 | 2 |

(a) Classification du LG.

| | imposition | recyclage | prorogation |
|---|---|---|---|
| accorder | 0 | 1 | 1 |
| donner | 1 | 1 | 1 |
| faire | 1 | 1 | 1 |
| procéder à | 1 | 1 | 1 |
| Id. | 1 | 2 | 2 |

(b) Verbes supports.

| | imposition | recyclage | prorogation |
|---|---|---|---|
| N0 faire Det N Prép N1 | 1 | 1 | 0 |
| N0 donner Det N à N1 | 1 | 1 | 1 |
| Id. | 1 | 1 | 2 |

(c) Constructions syntaxiques de base.

FIG. 6.16: Classes de noms générées à partir du Lglex selon diverses propriétés : (a) classification du LG. (b) verbes supports. (c) constructions syntaxiques de base.

Ces diverses propriétés conduisent à obtenir différents ensembles de classes pour les mêmes données d'entrée. Dans le tableau 6.15, nous avons indiqué le nombre de classes créées (colonne `#classes`) en fonction de la propriété utilisée. Le nombre d'entrées par classe correspondant est indiqué dans la colonne `AVG`. En utilisant les classes fournies par le LG, 1210 classes ont été générées et, en se basant sur les 20 constructions de base possibles, environ 400

classes ont été produites. Bien qu'il y ait un jeu de 48 verbes supports, un certain nombre d'entre eux apparaissent souvent ensemble. En effet, on peut constater que quelques verbes très ambigus en terme de sens, comme *donner*[173] et *faire*[174], sont présents dans de nombreuses entrées. Cela conduit à créer peu de classes comparativement au nombre de verbes possibles. Au final, ces trois classifications nous offrent des ensembles de classes de granularité très différente, ce qui était le cas également pour les verbes avec la hiérarchie des classes. Cette granularité se répercute ensuite sur le corpus, sachant que le Lglex couvre environ 50% des noms du FTB-UC[175]. On peut voir, à travers la colonne `#lexique`, que la taille du lexique est réduite de 11% au minimum en utilisant les classes du LG, et à un maximum de 11,4% en utilisant les verbes supports. Dans le tableau 6.16, nous avons indiqué la couverture du LG en terme de noms pour les corpus de la partition du FTB-UC. La faible couverture observée est due au fait que le LG est incomplet et traite uniquement les noms prédicatifs. Aussi, les noms prédicatifs non codés, tels que *cinéastes* ou *lecteur*, et les noms non prédicatifs, tels que *lampe*, ne sont pas concernés.

| Propriété | #classes | AVG | #lexique |
|---|---|---|---|
| Classification du LG | 1210 | 8.4 | 24544 |
| Constructions syntaxiques | 398 | 25.3 | 23999 |
| Verbes supports | 154 | 65.5 | 23767 |

TAB. 6.15: Nombre de classes générées selon la propriété utilisée, et taille du lexique du FTB-UC correspondante.

| Corpus | #verbes |
|---|---|
| Apprentissage | 48.7 |
| Développement | 59.4 |
| Évaluation | 59.4 |
| **Total** | 48.9 |

TAB. 6.16: Couverture du Lglex sur les noms des corpus de la partition du FTB-UC.

Le protocole d'évaluation est identique aux expériences précédentes. Les trois analyseurs sont d'abord entraînés sur le corpus d'apprentissage où les noms, dont le lemme est présent dans le Lglex, sont remplacés par leur classe (identifiant et étiquette). Pour obtenir les résultats bruts des analyseurs, il est nécessaire, comme pour les verbes, d'effectuer les étapes d'étiquetage et de lemmatisation des noms communs qui apparaissent dans les corpus de développement et d'évaluation. Les résultats obtenus par les analyseurs, selon les propriétés utilisées, sont montrés dans les tableaux de la figure 6.17. Seuls les scores $F_1$ sont indiqués car, pour l'ensemble des mesures, ces scores obtenus sont très faibles et globalement inférieurs aux scores de base.

[173] 19 entrées distinctes dans les classes de verbes du LG.
[174] 31 entrées distinctes dans les classes de verbes du LG.
[175] Il y a exactement 2508 noms sur un total de 5120 qui sont dans le Lglex.

Les diverses observations faites au cours de cette section, et des précédentes, pourraient expliquer en partie cet échec. Tout d'abord, les nombreuses données incomplètes, ou erronées, en ce qui concerne les entrées du LG peuvent conduire à regrouper les noms en classes incohérentes. De plus, contrairement aux verbes, la couverture du lexique est faible. Ce problème se traduit concrètement par l'affaiblissement de la puissance des regroupements lexicaux. En effet, le modèle statistique doit prendre en compte à la fois les classes de noms, mais également la multitude de noms qui sont en dehors de celles-ci. Ce problème est également un "avantage" dans le cas présent car, contrairement aux verbes, la différence de gains entre les résultats de référence et bruts est minime. Les nombreuses erreurs d'étiquetage et de lemmatisation ne perturbent pas les analyseurs, car le modèle compense avec des statistiques calculées à la fois sur les classes, mais aussi sur des noms non modifiés.

| Analyseur / Propriété | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Classification du LG | +0.15 | +0.17 | -0.17 |
| Constructions syntaxiques | +0.03 | -0.41 | -0.20 |
| Verbes supports | +0.03 | -0.03 | -0.63 |
| **Corpus d'évaluation** | | | |
| Classification du LG | -0.03 | +0.01 | +0.25 |
| Constructions syntaxiques | +0.01 | -0.08 | +0.08 |
| Verbes supports | -0.23 | -0.30 | +0.14 |

(a) Gains $F_1$ de référence.

| Analyseur / Propriété | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Classification du LG | +0.04 | -0.30 | -0.20 |
| Constructions syntaxiques | -0.02 | -0.35 | -0.26 |
| Verbes supports | -0.01 | -0.08 | -0.75 |
| **Corpus d'évaluation** | | | |
| Classification du LG | -0.26 | -0.12 | +0.17 |
| Constructions syntaxiques | -0.23 | -0.20 | -0.01 |
| Verbes supports | -0.45 | -0.43 | +0.04 |

(b) Gains $F_1$ bruts.

FIG. 6.17: Gains $F_1$ absolus des analyseurs sur les corpus de développement et d'évaluation en fonction des classes de noms.

### (b) L*efff* : Classes d'adjectifs, et Lglex-L*efff* : Classes de noms

En plus des verbes, les lexiques L*efff* et Lglex-L*efff* fournissent une description syntaxique des adjectifs pour l'un (28864 lemmes), et des noms prédicatifs pour l'autre (10069 lemmes).

Ces entrées disposent des mêmes propriétés que pour les verbes, à savoir des cadres de sous-catégorisation sous la forme d'arguments associés à des fonctions syntaxiques. Les jeux de fonctions exploités par les deux lexiques, montrés dans le tableau 6.17, sont similaires à ceux utilisés pour la description syntaxique des verbes[176]. À titre d'illustration, nous avons indiqué, dans le tableau 6.18, divers cadres de sous-catégorisation associés aux entrées des deux lexiques. L'entrée de l'adjectif *absurde* accepte un seul argument, le sujet, comme dans l'emploi suivant :

$$\underset{Suj}{\underline{Ces\ blagues}}\ \textit{sont plus ou moins}\ \underset{ADJ}{\underline{absurdes}}.$$

De même, le nom prédicatif *antipathie* accepte deux arguments, un sujet et un argument oblique introduit par la préposition *pour*, comme dans :

$$\underset{Suj}{\underline{Je}}\ \textit{n'ai aucune}\ \underset{N}{\underline{antipathie}}\ \underset{Obl}{\underline{pour\ les\ chats}}.$$

| Lexique | Jeu de fonctions syntaxiques |
|---|---|
| Le*fff* $_{adjectifs}$ | Suj, Obj, Objde, Objà, Attr, Loc, Obl, Obl2 |
| Lglex-Le*fff* $_{noms}$ | Suj, Obj, Objde, Objà, Attr, Loc, Obl, Obl2 |

TAB. 6.17: Jeux de fonctions syntaxiques utilisés par les lexiques pour la description des adjectifs et des noms prédicatifs.

| Lexique | Mot | Cadre de sous-catégorisation |
|---|---|---|
| Le*fff* $_{adjectifs}$ | absurde | Suj :scompl\|sinf\|sn |
| Lglex-Le*fff* $_{noms}$ | antipathie | Suj :cln\|sn,Obl :pour-sn\|à_l'égard_de-sn |

TAB. 6.18: Cadre de sous-catégorisation associé à différentes entrées des lexiques.

L'algorithme de création et d'utilisation des classes de mots est identique à celui décrit en introduction (section 6.4.1). Les classes sont générées à partir des vecteurs de fonctions syntaxiques, et sont utilisées ensuite pour remplacer les noms, ou les adjectifs, des différents corpus. Dans le tableau 6.19, nous avons indiqué le nombre de classes créées (colonne `#classes`) en fonction du lexique. Le nombre d'entrées par classe correspondant, ainsi que la taille du lexique du FTB-UC, sont également indiqués. En utilisant les fonctions syntaxiques uniquement, on peut voir que l'ensemble des 49 classes de noms est relativement de petite taille par rapport à la taille des données d'entrées. Nous avons constaté qu'une majorité d'entrées, environ 83%, sont associées à un cadre de sous-catégorisation de un (un sujet) ou deux arguments (un sujet et une autre fonction). L'ambiguité en terme de fonctions est fortement réduite, et conduit donc à la création de vecteurs relativement similaires. Cependant, ce petit ensemble de classes réduit peu la taille du lexique du corpus, et ceci à cause de la faible couverture des noms. On peut

[176]Dans le cadre du Lglex-Le*fff*, on peut néanmoins noter que les fonctions *DeLoc* et *Obl3*, qui étaient utilisées pour les verbes, ne le sont plus pour les noms, ce qui revient à un jeu de 9 fonctions, au lieu de 11 précédemment.

rappeler que le Lglex-Le*ffff* est un lexique généré automatiquement à partir du Lglex. Aussi, les problèmes observés sur ce dernier sont inévitablement présents dans le Lglex-Le*ffff*.
En ce qui concerne les adjectifs, peu de classes ont également été créées. De plus, on peut constater que la couverture du Le*ffff*, sur les adjectifs du FTB-UC, est très élevée, comme montré dans le tableau 6.20. Pour les quelques adjectifs qui sont en dehors du lexique, il s'agit principalement de nombres (environ 85%), représentés par des caractères numériques ou alphabétiques.

| Lexique | #classes | AVG | #lexique |
|---|---|---|---|
| Le*ffff*$_{adjectifs}$ | 26 | 1110 | 23972 |
| Lglex-Le*ffff*$_{noms}$ | 49 | 205 | 23683 |

TAB. 6.19: Nombre de classes générées selon le lexique utilisé, et taille du lexique du FTB-UC correspondante.

| Corpus | #adjectifs |
|---|---|
| Apprentissage | 87.10 |
| Développement | 90.57 |
| Évaluation | 91.49 |
| **Total** | 87.84 |

TAB. 6.20: Couverture du Le*ffff* sur les adjectifs des corpus de la partition du FTB-UC.

Les résultats obtenus par les analyseurs selon le lexique utilisé sont montrés dans les tableaux de la figure 6.18. En ce qui concerne les noms, on peut voir que cette expérience n'est pas concluante, puisque la majorité des gains sont négatifs ou nuls. Les raisons que nous avons énoncées dans la section précédente sont probablement la cause de cet échec. À propos des adjectifs, le constat est plus nuancé. Tout d'abord, la plupart des gains de référence sont significatifs. En revanche, on s'aperçoit que les gains bruts sont positifs mais uniquement sur le corpus d'évaluation, et que, parmi eux, un seul est significatif (Brown). Ces résultats mitigés sont dûs aux faibles performances d'étiquetage et de lemmatisation, qui sont grandement en deça des verbes et des noms.

### 6.4.5 Combinaisons de classes lexicales

Les expériences que nous venons de décrire avaient pour but d'évaluer indépendamment, et avec plus ou moins de succès, des classes construites pour des mots d'une catégorie grammaticale particulière. Aussi, seuls les mots concernés par ces classes sont modifiés dans les textes et les corpus. Ce principe est différent d'autres méthodes de regroupement lexical que nous avons vu auparavant, comme DFL (section 6.2.3) ou encore l'algorithme semi-supervisé StatClust (section 6.2.4). En effet, ils se basent sur une stratégie globale aux mots, et ceci quel que soit

| Analyseur / Lexique | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Le*fff*$_{adjectifs}$ | +0.35$^\diamond$ | +0.30$^\diamond$ | +0.11 |
| Lglex-Le*fff*$_{noms}$ | +0.17 | -0.04 | -0.48 |
| **Corpus d'évaluation** | | | |
| Le*fff*$_{adjectifs}$ | +0.49$^\diamond$ | +0.50$^\diamond$ | +0.80$^\diamond$ |
| Lglex-Le*fff*$_{noms}$ | -0.15 | -0.25 | +0.11 |

(a) Gains $F_1$ de référence.

| Analyseur / Lexique | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Le*fff*$_{adjectifs}$ | -0.01 | -0.10 | -0.05 |
| Lglex-Le*fff*$_{noms}$ | +0.19 | -0.05 | -0.51 |
| **Corpus d'évaluation** | | | |
| Le*fff*$_{adjectifs}$ | +0.17 | +0.18 | +0.45$^\diamond$ |
| Lglex-Le*fff*$_{noms}$ | -0.29 | -0.32 | -0.03 |

(b) Gains $F_1$ bruts.

FIG. 6.18: Gains $F_1$ des analyseurs sur les corpus de développement et d'évaluation en fonction des lexiques.

leur nature. Pour explorer cette voie, nous avons effectué, dans un premier temps, plusieurs expériences de combinaisons des classes de mots de différentes catégories grammaticales. Puis, nous nous sommes intéressé à l'impact de ces classes sur les performances lorsqu'elles sont combinées aux méthodes DFL et StatClust.

### Classes générées à partir des lexiques

Au vu du nombre important de configurations possibles, nous nous sommes limité, dans notre première expérience, à l'utilisation d'un seul ensemble de classes par catégorie grammaticale. Notre critère de sélection pour déterminer un ensemble parmi les candidats est basé sur la maximisation des performances obtenues par les 3 analyseurs. Pour les verbes, il s'agit des classes du LexSchem, générées à partir des vecteurs de fonctions syntaxiques. Pour les adjectifs, seul le Le*fff* est en lice. Et pour les noms, nous avons sélectionné les classes créées à partir du Lglex et de la propriété de la classification du LG. Les trois ensembles obtenus en sortie de cette phase de sélection nous permettent d'évaluer quatre configurations possibles : `Verbes+Adjectifs`, `Adjectifs+Noms`, `Verbes+Noms` ainsi que `Verbes+Adjectifs+Noms`. La figure 6.19 montre un exemple de configuration `Verbes+Adjectifs` appliquée sur l'arbre de la phrase *Paul soupe avec ses nouveaux voisins*. On suppose que l'on connaît les lemmes des mots. Dans l'arbre original (figure 6.6a), on peut distinguer un verbe à l'indicatif de lemme *souper*, et un adjectif de lemme *nouveau*. En supposant que le verbe et l'adjectif font partie respectivement

des classes de mots `1` et `14`, ils sont donc remplacés, dans cette phrase, par les mots `1ADJ` et `14V`. Les caractéristiques des configurations sont montrées dans le tableau 6.21. La colonne `#lexique` indique la taille du FTB-UC après l'application des classes. Les autres colonnes font de même pour les corpus de la partition. La taille des corpus varie beaucoup selon la configuration utilisée. De plus, on peut noter que la taille totale du FTB-UC est réduite de 50% par rapport au corpus original avec la configuration `Verbes+Adjectifs+Noms`.

( (SENT (NP (NPP Paul))
(VN (V **soupe**))
(PP (P avec) (NP (DET ses) (ADJ **nouveaux**) (NC voisins)))
(PONCT .)))

(a) Arbre original.

( (SENT (NP (NPP Paul))
(VN (V **14V**))
(PP (P avec) (NP (DET ses) (ADJ **1ADJ**) (NC voisins)))
(PONCT .)))

(b) Arbre modifié.

FIG. 6.19: Modification d'un arbre de constituants avec des classes de la configuration `Verbes+Adjectifs`.

| Configuration | #lexique | Apprentissage | Développement | Évaluation |
|---|---|---|---|---|
| Noms+Adjectifs | 21265 | 18546 | 5474 | 5328 |
| Verbes+Adjectifs | 17292 | 15185 | 4683 | 4470 |
| Verbes+Noms | 17979 | 15723 | 4787 | 4545 |
| Verbes+Adjectifs+Noms | 14421 | 12603 | 3813 | 3608 |

TAB. 6.21: Tailles du lexique des corpus de la partition du FTB-UC selon les configurations de classes possibles.

Dans les tableaux de la figure 6.20, nous avons indiqué les performances des trois analyseurs selon la configuration utilisée (scores $F_1$ uniquement). Plusieurs constatations peuvent être tirées de ces chiffres. Tout d'abord, les faibles résultats de la configuration `Noms+Adjectifs` sont logiques dans le sens où ces deux ensembles de classes, lorsqu'ils sont utilisés indépendamment, ne permettent pas d'obtenir des performances significatives. En revanche, pour les autres configurations, il est difficile de déterminer laquelle améliore au mieux les performances. En effet, sur le corpus de développement, les meilleurs gains sont obtenus par deux configurations, `Verbes+Noms` et `Verbes+Adjectifs+Noms`. Mais, pour le corpus d'évaluation, il s'agit d'une autre configuration, à savoir `Verbes+Adjectifs`. Bien que la plupart de ces gains soient significatifs, on peut néanmoins noter qu'ils sont équivalents à ceux observés pour les classes de verbes du LexSchem. D'après ces observations, on peut donc en conclure que la faible puissance des classes d'adjectifs et de noms ne profite pas aux analyseurs, dont les performances reposent essentiellement sur les classes de verbes.

| Configuration | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Noms+Adjectifs | +0.35$^\diamond$ | +0.39$^\diamond$ | +0.03 |
| Verbes+Adjectifs | +0.72$^\diamond$ | +0.51 | **+0.59**$^\diamond$ |
| Verbes+Noms | +0.85$^\diamond$ | +0.86$^\diamond$ | +0.52$^\diamond$ |
| Verbes+Adjectifs+Noms | **+1.00**$^\diamond$ | **+0.99**$^\diamond$ | +0.48$^\diamond$ |
| **Corpus d'évaluation** | | | |
| Noms+Adjectifs | +0.39 | +0.57 | +0.44 |
| Verbes+Adjectifs | +0.74 | **+0.79** | **+1.05** |
| Verbes+Noms | +0.58 | +0.66 | +0.82 |
| Verbes+Adjectifs+Noms | **+0.76** | +0.72 | +0.87 |

(a) Gains $F_1$ de référence.

| Configuration | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| **Corpus de développement** | | | |
| Noms+Adjectifs | +0.03 | +0.01 | -0.26 |
| Verbes+Adjectifs | +0.34$^\diamond$ | +0.21 | **+0.43**$^\diamond$ |
| Verbes+Noms | +0.59$^\diamond$ | **+0.57**$^\diamond$ | +0.32$^\diamond$ |
| Verbes+Adjectifs+Noms | **+0.62**$^\diamond$ | +0.43$^\diamond$ | +0.32$^\diamond$ |
| **Corpus d'évaluation** | | | |
| Noms+Adjectifs | +0.01 | +0.20 | +0.11 |
| Verbes+Adjectifs | **+0.23** | **+0.30**$^\diamond$ | **+0.65**$^\diamond$ |
| Verbes+Noms | +0.13 | +0.21 | +0.35$^\diamond$ |
| Verbes+Adjectifs+Noms | +0.22 | +0.26 | +0.53$^\diamond$ |

(b) Gains $F_1$ bruts.

FIG. 6.20: Gains $F_1$ absolus des analyseurs sur les corpus de développement et d'évaluation en fonction des combinaisons de classes.

### Combinaison avec les classes de DFL et StatClust

Notre deuxième expérience concerne la combinaison de nos classes avec d'autres méthodes, à savoir DFL et StatClust. D'après l'état de l'art, ces méthodes permettent d'améliorer significativement les performances de divers analyseurs syntaxiques, en réduisant notamment l'effet de dispersion des données pour les grammaires PCFG. Afin de réaliser la combinaison des regroupements lexicaux, le corpus a d'abord été modifié par nos classes. Ensuite, pour les mots qui sont restés intacts, nous avons appliqué soit DFL, soit StatClust[177]. En sortie, le corpus contient donc un mélange de classes d'origines diverses. Comme énoncé précédemment, l'espace des configurations possibles est de grande taille. Aussi, par souci de clarté, nous avons fait le choix

[177]On peut rappeler que les classes obtenues par StatClust ont été générées à partir d'un grand corpus brut, préalablement analysé par DFL. Aussi, pour attribuer les classes de StatClust, il est nécessaire de remplacer, au préalable, les mots des textes bruts par les classes de DFL.

de n'indiquer que certains résultats selon des configurations bien spécifiques. Nous avons donc tenter de combiner les classes de DFL et StatClust avec les classes de verbes du LexSchem, les classes de noms du Lglex et les classes d'adjectifs du Le*fff*. Parmi les différentes combinaisons décrites dans l'expérience précédente, nous avons sélectionné une seule configuration qui est la combinaison des trois ensembles de classes `Verbes+Adjectifs+Noms`[178]. En partant de l'arbre modifié par nos classes et montré dans la figure 6.19b, l'application de DFL sur celui-ci a pour effet d'obtenir un nouvel arbre, montré dans la figure 6.21a. Seuls les mots qui ne sont pas des classes ont été modifiés par cette méthode, à savoir *Paul*, *avec*, *ses* et *voisins*. Le mot *ses* a, par exemple, été substitué par le mot *son*. La version de cet arbre modifiée par StatClust est montrée dans la figure 6.21b. La taille du lexique du FTB-UC après l'application de chaque configuration est montrée dans le tableau 6.22. Pour DFL, la réduction de la taille varie entre 10%, avec les classes de noms et d'adjectifs, et environ 40% lorsque tous nos ensembles de classes sont combinés. En revanche, pour StatClust, la taille ne varie que très peu à cause de la spécificité de cet algorithme, et est même supérieure à celle initiale avec les classes de noms.

```
( (SENT (NP (NPP Paul))
    (VN (V 14V))
    (PP (P avec) (NP (DET son) (ADJ 1ADJ) (NC voisin)))
    (PONCT .)))
```

(a) Arbre modifié par DFL.

```
( (SENT (NP (NPP 0101101111))
    (VN (V 14V))
    (PP (P 110011111110) (NP (DET 101111011) (ADJ 1ADJ) (NC 00101001011)))
    (PONCT .)))
```

(b) Arbre modifié par StatClust.

FIG. 6.21: Modification d'un arbre de constituants avec des classes de la configuration `Verbes+Adjectifs`.

| Configuration | #lexique$_{DFL}$ | #lexique$_{StatClust}$ |
|---|---|---|
| Base | 20178 | 1976 |
| Verbes | 16325 | 1901 |
| Noms | 18471 | 1987 |
| Adjectifs | 18324 | 1914 |
| Verbes+Adjectifs+Noms | 12674 | 1965 |

TAB. 6.22: Tailles du lexique du FTB-UC selon les configurations possibles, combinées à DFL ou à StatClust.

Les résultats de la combinaison de nos classes avec les méthodes StatClust et DFL sont montrés dans les tableaux de la figure 6.22. Pour des raisons de clarté, seuls les scores $F_1$ bruts obtenus

[178]On peut noter que les résultats obtenus par les autres combinaisons sont presque identiques.

sur le corpus d'évaluation sont indiqués, et sont représentés sous la forme de gains calculés en absolu par rapport aux scores de base de DFL ou StatClust. On peut voir que ces gains sont majoritairement négatifs, et dans le cas contraire, non significatifs. Ce constat sans appel montre que la combinaison de classes issues d'algorithmes distincts n'est pas chose aisée. Notre algorithme génère des ensembles de classes destinés à ne modifier que des mots d'une catégorie bien spécifique. On pourrait donc qualifier cet algorithme de local. Or, les algorithmes comme DFL ou StatClust ont pour but de remplacer, sans exception, tous les mots d'un texte avec des classes calculées indépendamment des catégories grammaticales. Aussi, l'introduction de nos classes semble perturber ces deux méthodes globales au texte. Il est intéressant de noter que les performances de base de la méthode StatClust ne sont pas aussi élevées que dans la littérature (section 6.2.5), et sont même très faibles pour Brown [179]. Nous voyons deux raisons à celà. D'une part, la version actuelle de BKY possède un algorithme significativement amélioré par rapport aux versions précédentes. D'autre part, les classes générées par cette méthode ne semblent pas être adaptées aux analyseurs lexicalisés comme Brown, bien que pour le prouver, il faudrait effectuer une évaluation comparative de plusieurs analyseurs lexicalisés. Pour finir, les performances de la configuration `Verbes+Adjectifs+Noms` associée à DFL sont plutôt étonnantes car, comme nous l'avons vu dans le tableau précédent, le lexique du corpus est réduit de 40% avec cette configuration. Or, nous constatons des performances presque identiques à celles de base.

| | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| Base | 84.73 | 86.03 | 81.66 |
| Noms | +0.01 | +0.06 | -0.28 |
| Adjectifs | -0.18 | -0.1 | -0.30 |
| Verbes | -0.08 | +0.07 | +0.27 |
| Verbes+Adjectifs+Noms | -0.02 | +0.11 | -0.08 |

(a) DFL.

| | BKY | BKY$_p$ | Brown |
|---|---|---|---|
| Base | 84.89 | 86.24 | 81.04 |
| Noms | -0.20 | -0.21 | +0.02 |
| Adjectifs | -0.19 | -0.39 | -0.08 |
| Verbes | -0.02 | +0.09 | +0.14 |
| Verbes+Adjectifs+Noms | -0.15 | -0.20 | -0.03 |

(b) StatClust.

FIG. 6.22: Gains $F_1$ absolus des analyseurs sur le corpus d'évaluation du FTB-UC par rapport aux résultats de base de StatClust et de DFL, combinés à nos ensembles de classes.

Il existe une deuxième manière de combiner nos classes avec la méthode StatClust. Nous rappelons que l'algorithme décrit dans (Candito & Crabbé, 2009; Candito & Seddah, 2010) est composée de deux étapes : (a) le grand corpus brut *L'est Républicain* est analysé par DFL

[179] C'est également le cas sur le corpus de développement.

pour réduire la complexité et améliorer les performances de l'algorithme d'apprentissage semi-supervisé. Les mots du corpus sont donc remplacés par une forme fléchie réduite. (b) l'algorithme de classification est appliqué sur ce corpus modifié. Dans le cadre de nos expériences, nous avons modifié l'étape (a) en incorporant nos classes lexicales. Les différentes étapes de notre approche sont illustrées graphiquement dans la figure 6.23. Le corpus est préalablement étiqueté avec LGTagger[180], et lemmatisé avec Bonsaï. Puis, les classes lexicales sont appliquées sur les mots concernés. Les mots restants (non remplacés par une classe) peuvent être ensuite modifiés par la méthode DFL.

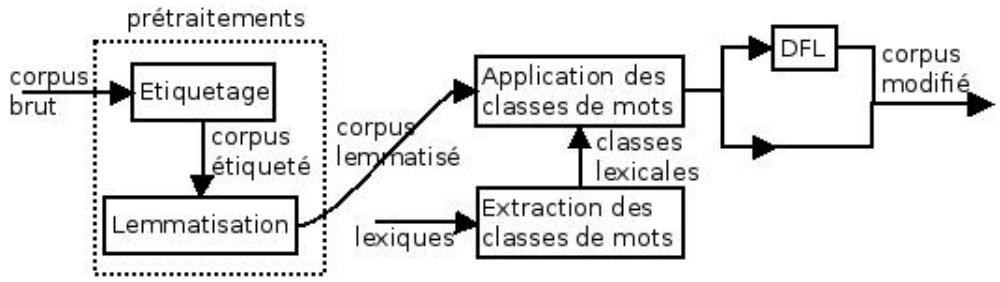



FIG. 6.23: Incorporation des classes de mots dans l'algorithme d'apprentissage de StatClust.

Nous avons réutilisé les mêmes paramètres que dans (Candito & Seddah, 2010) pour le programme d'apprentissage automatique des classes, à savoir un nombre de classes à créer fixé à 1000 et un nombre minimal d'occurrences d'un mot du corpus de 200[181]. On peut noter que l'exécution du programme est plutôt longue. Par exemple, il nous a fallu environ 2 jours pour apprendre une seule classification à partir du corpus. Aussi, pour l'étape (a), nous nous sommes limité à 6 configurations : `Base`[182], `Verbes`, `Verbes+Noms+Adjectifs`, `DFL`, `DFL+Verbes` et `DFL+Verbes+Noms+Adjectifs`. Les scores $F_1$ des diverses configurations sont montrés dans le tableau 6.23. On peut voir que l'utilisation des classes de verbes apportent des gains positifs mais non significatifs par rapport à `Base`. De même, la combinaison `DFL+Verbes` ne permet pas d'améliorer les performances de la configuration `DFL` utilisée seul. Les autres ensembles de classes (adjectifs et noms) sont sans effets.

| | BKY | $BKY_p$ | Brown |
|---|---|---|---|
| Base | 84.35 | 85.51 | 81.32 |
| Verbes | +0.17 | +0.25 | +0.19 |
| Verbes+Adjectifs+Noms | +0.23 | +0.29 | +0.27 |
| DFL | +0.56 | +0.82 | +0.43 |
| DFL+Verbes | +0.53 | +0.89 | +0.55 |
| Verbes+Adjectifs+Noms | +0.59 | +0.93 | +0.56 |

TAB. 6.23: Gains $F_1$ absolus des analyseurs sur le corpus d'évaluation du FTB-UC par rapport aux résultats de base `Base`.

[180]Cet étiqueteur a été entraîné sur le corpus d'apprentissage du FTB-UC.
[181]En dessous de ce palier, un mot n'est pas traité par l'algorithme et ne sera donc présent dans aucune classe.
[182]Le corpus brut n'a subi aucune modification.

## 6.5 Discussion des résultats

Dans cette section, nous discutons plus en détail des résultats obtenus lors des précédentes évaluations. Nous commencerons par approfondir l'étude des regroupements lexicaux lorsqu'ils sont utilisés de manière indépendante (section 6.5.1). Nous utiliserons notamment la techniques de validation croisée afin de déterminer si les scores obtenus par nos méthodes sur les corpus de de développement et d'évaluation sont confirmés sur un texte à analyser de taille plus conséquente. Ensuite, dans la section 6.5.2, nous nous intéresserons à l'adaptation des analyseurs lorsqu'ils sont entraînés sur des petits corpus. Pour ce faire, nous simulerons ce principe en faisant varier la taille du corpus d'apprentissage. Nous tenterons de déterminer l'impact des regroupements lexicaux lorsque la dispersion des données est de plus en plus forte, et ce à mesure que la taille du corpus d'apprentissage diminue. Dans la section 6.5.3, nous indiquons les scores obtenus par un réordonnanceur discriminatif utilisé en combinaison des analyseurs BKY et Brown. Nous évaluerons deux stratégies d'incorporation des données syntaxiques présentes dans les lexiques. L'une, que l'on pourrait qualifier d'*endogène*, génère des traits grâce à des informations extraites uniquement à partir des analyses candidates dont les mots sont remplacés par les classes lexicales, et l'autre, *exogène*, se base sur un ensemble de patrons consacrés aux ressources externes. Pour finir, nous proposons d'évaluer un analyseur en dépendance discriminatif, Malt, entraîné sur un corpus dont les données syntaxiques (classes, propriétés des entrées,...) sont ajoutées en tant que propriétés des mots (section 6.5.4). Nous comparerons notamment les scores UAS obtenus avec ceux des analyseurs en constituants.

### 6.5.1 Regroupements lexicaux

Au cours des diverses expériences sur les regroupements lexicaux, nous avons pu remarquer qu'il existe une certaine difficulté d'analyse du corpus de développement. D'une part, les scores obtenus par les trois analyseurs sont en deça de ceux sur le corpus d'évaluation. Et d'autre part, une majorité de regroupements lexicaux ont moins d'impact sur ce corpus. La problématique consiste donc à déterminer si ces observations sont locales au corpus de développement, ou valables pour la totalité du corpus. La solution à ce problème est d'utiliser la méthode de validation croisée qui permet de calculer des scores moyens sur un échantillon d'évaluation aussi grand que le corpus initial (section I.2.2.4). Cet algorithme d'évaluation étant très couteux en temps[183], nous avons fait le choix de n'évaluer que quelques ensembles de classes. Aussi, pour chaque catégorie grammaticale ayant une description syntaxique dans les lexiques (adjectifs, noms et verbes), l'ensemble sélectionné est celui qui maximise les scores obtenus par les analyseurs sur les corpus de développement et d'évaluation. Il s'agit donc des classes du LexSchem pour les verbes, du Le*fff* pour les adjectifs, et du Lglex pour noms[184]. On peut noter que l'analyseur $\text{BKY}_p$ n'a pas été exploité pour cette expérience, car il est nécessaire de rappeler que l'algorithme sous-jacent à l'analyse d'un texte est basé sur un produit de grammaires, dont nous avons fixé le nombre à 8. Or, la validation croisée nous impose déjà de générer 10 grammaires,

[183]En effet, avec le paramètre $n$ fixé à 10 par exemple, le corpus est découpé en 10 parties, ce qui conduit donc à entraîner 10 modèles probabilistes distincts sur des corpus d'apprentissage composés de 9 parties.

[184]Les classes sont générées à partir de la propriété de la classification du LG.

ce qui conduirait à entraîner 10×8 grammaires pour un seul ensemble de classes à évaluer. Pour la même raison, une seule grammaire a été générée pour BKY à chaque itération de la validation croisée[185].

Les résultats de la validation croisée, selon les différents ensembles de classes, sont indiqués dans les tableaux de la figure 6.24. Pour chaque analyseur, nous avons indiqué les scores $F_1$, UAS et LA pour l'expérience `Base`, et, comparativement à celle-ci, les gains absolus pour chaque configuration évaluée. À titre de comparaison, nous avons également indiqué les scores des méthodes DFL et StatClust. Au final, les résultats de la validation croisée ne font que confirmer les observations que nous avons formulées dans les sections précédentes. Tout d'abord, les classes de verbes, générées à partir du LexSchem, permettent à Brown d'obtenir des gains significatifs pour l'ensemble des mesures ($p < 0.01$). En revanche, pour BKY, seules les dépendances syntaxiques sont significativement améliorées. En ce qui concerne les classes d'adjectifs, on peut voir que l'ensemble des scores de référence sont significatifs, mais le passage aux scores bruts ne permet pas de conserver de tels gains. Et enfin, on ne peut que constater l'inefficacité des classes de noms pour les deux analyseurs. En conclusion, le bilan est en demi-teinte car d'un côté, un analyseur PCFG, basé sur un algorithme non lexicalisé comme BKY, ne semble pas tirer profit des informations lexicales pour réduire l'effet de dispersion des données. À l'opposé, un analyseur lexicalisé et moins évolué comme Brown est plus réceptif aux regroupements lexicaux.

| | $F_1$ | | LA | | UAS | |
|---|---|---|---|---|---|---|
| | BKY | Brown | BKY | Brown | BKY | Brown |
| Base | 83.02 | 79.83 | 86.1 | 83.7 | 87.55 | 85.45 |
| LexSchem$_{verbes}$ | +0.51 | +0.72 | +0.3 | +0.4 | +0.70 | +0.78 |
| Le*fff*$_{adjectifs}$ | +0.40 | +0.46 | +0.3 | +0.2 | +0.44 | +0.46 |
| Lglex-Le*fff*$_{noms}$ | +0.05 | -0.06 | +0.1 | +0.2 | +0.04 | +0.17 |

(a) Gains de référence.

| | $F_1$ | | LA | | UAS | |
|---|---|---|---|---|---|---|
| | BKY | Brown | BKY | Brown | BKY | Brown |
| LexSchem$_{verbes}$ | +0.14 | +0.38$^{\diamond}$ | +0.2 | +0.3 | +0.37$^{\diamond}$ | +0.55$^{\diamond}$ |
| Le*fff*$_{adjectifs}$ | +0.02 | +0.11 | +0.2 | +0.1 | +0.07 | +0.13 |
| Lglex-Le*fff*$_{noms}$ | -0.07 | -0.16 | = | +0.1 | -0.07 | +0.01 |
| DFL | +0.47$^{\diamond}$ | +0.37$^{\diamond}$ | +0.4 | +0.4 | +0.55$^{\diamond}$ | +0.47$^{\diamond}$ |
| StatClust | +0.58$^{\diamond}$ | +0.48$^{\diamond}$ | +0.4 | +0.3 | +0.7$^{\diamond}$ | +0.42$^{\diamond}$ |

(b) Gains bruts.

FIG. 6.24: Gains des analyseurs en fonction des ensembles de classes avec la méthode de validation croisée.

[185]La graine aléatoire utilisée est celle défini par défaut dans le programme, à savoir 8.

**Comparaison à deux stratégies aléatoires**

Nous avons comparé les performances de nos classes avec celles générées à partir de deux algorithmes aléatoires. L'algorithme d'apprentissage automatique des *k-moyennes* (aussi appelé *k-means* en anglais) a pour but de déterminer une partition, en $K$ classes, de $N$ éléments donnés en entrée, et ce grâce à des mesures statistiques. À l'instar de notre méthode de regroupement lexical, il faut définir au préalable un ensemble de propriétés à associer aux éléments et qui vont nous permettre de calculer les vecteurs de propriétés. L'initialisation de l'algorithme consiste à créer les $K$ classes, puis à assigner, à chacune d'entre elles, un élément pris parmi les $N$. Bien qu'il existe une multitude de techniques permettant de choisir au mieux les $K$ éléments, la technique la plus courante consiste simplement à les choisir de manière aléatoire. Chaque classe est maintenant représentée par un vecteur de propriétés (celui de l'élément). Chaque itération de l'algorithme consiste à recalculer les vecteurs des classes et est constituée de deux étapes :

– classer chacun des $N$ éléments en choisissant parmi les classes disponibles, celle qui maximise (ou minimise) le score d'une mesure mathématique. Il existe plusieurs mesures possibles (appelées aussi *distance*) : distance de Jaccard, coefficient de Dice, distance d'Euclide... Pour nos expériences, nous avons choisi la distance d'Euclide dont la formule mathématique est la suivante :

$$distance_{euclide} = \frac{X.Y}{|X| \times |Y|} \tag{6.1}$$

avec $X$ le vecteur associé à l'élément à classer, et $Y$ le vecteur de la classe courante. L'élément courant est donc ajouté à la classe qui maximise cette distance.

– dès que tous les éléments ont été classés, les vecteurs des classes sont mis à jour grâce aux vecteurs des éléments faisant partie de ces classes.

L'algorithme s'arrête quand le point de convergence est atteint, c'est-à-dire quand tous les éléments sont affectés aux mêmes classes que lors de l'itération précédente. Le deuxième algorithme que nous avons expérimenté consiste à assigner un élément, d'un ensemble de taille $N$, à une classe prise aléatoirement parmi un ensemble de taille $K$. Cette expérience permet de mesurer l'impact d'une méthode complètement aléatoire sur les performances des analyseurs syntaxiques.

Pour ces deux expériences, nous avons testé quatre valeurs de $K$ (nombre de classes) : 50, 100, 200 et 400. Sachant que les algorithmes sont basés sur de l'aléatoire, nous avons généré, pour chaque valeur de $K$, trois ensembles de classes. Les scores des analyseurs syntaxiques sont calculés en faisant la moyenne des résultats obtenus avec les trois ensembles afin d'obtenir des résultats stables et pertinents. Dans le tableau 6.24, nous avons uniquement indiqué les gains $F_1$ bruts des analyseurs obtenus avec les classes de verbes générées à partir du LexSchem. Ces gains sont calculés en absolus par rapport aux résultats de l'expérience $\texttt{LexSchem}_{verbes}$. Afin d'être directement comparable à cette expérience, nous avons exploité les mêmes propriétés afin de composer les vecteurs, à savoir les fonctions syntaxiques[186]. On peut constater que la méthode k-moyennes dégrade légèrement mais non significativement les scores, et ce, quelle que soit la valeur de $K$. Cet algorithme mathématique n'est donc pas plus puissant que notre

[186]Nous avons testé d'autres configurations mais les résultats se sont révélés identiques.

méthode initiale. En revanche, l'algorithme entièrement aléatoire fait chuter considérablement les scores. Cela montre qu'un regroupement lexical ne reposant sur aucune information syntaxique ne permet pas de corriger l'effet négatif de la dispersion des données.

| | k-moyennes | | aléatoire | |
|---|---|---|---|---|
| $K$ | BKY | Brown | BKY | Brown |
| 50 | = | -0.13 | -0.36 | -0.6 |
| 100 | -0.08 | -0.11 | -0.34 | -0.6 |
| 200 | -0.02 | -0.09 | -0.38 | -0.55 |
| 400 | -0.12 | -0.15 | -0.41 | -0.65 |

TAB. 6.24: Gains $F_1$ bruts des analyseurs en fonction des ensembles de $K$ classes générés avec deux stratégies aléatoires (validation croisée).

**Impact de l'étiquetage et de la lemmatisation**

D'après ces résultats et ceux des sections précédentes, on peut remarquer qu'il existe un écart de performances entre les scores de référence et bruts. Ces différences peuvent être minimes, comme pour le cas des classes de noms, ou significatives, comme avec les classes de verbes et d'adjectifs. Nous voyons trois raisons majeures pouvant expliquer un tel phénomène. D'une part, les performances d'étiquetage ne sont pas parfaites. Avec au minimum 2% d'erreurs pour chaque catégorie grammaticale (section 6.4.2), il n'est pas rare de voir, par exemple, un verbe qui n'est pas étiqueté comme tel, ou encore, qu'un mot soit étiqueté incorrectement comme verbe. Dans ce dernier cas, les conséquences peuvent être très négatives car cela change partiellement ou complètement la structure syntaxique de la phrase. La lemmatisation joue également un rôle très important car elle est nécessaire pour remplacer un mot par sa classe. Le lemmatiseur de Bonsaï étant basé sur un lexique, mais aussi sur des heuristiques, il y a de fortes chances qu'un verbe inconnu soit lemmatisé de la même façon quelle que soit sa forme fléchie. Cette remarque est également valable pour les cas d'ambiguité, lorsque plusieurs lemmes du lexique sont candidats pour un même verbe.
Cependant, ces deux causes, si elles sont vérifiées, ne seraient valables que pour les verbes et les adjectifs, mais pas pour les noms. En effet, on peut rappeler que l'étiquetage et la lemmatisation des noms sont au même niveau de performances que les verbes. Nous avons évoqué dans la section 6.4.4 que, pour les classes de noms, la faible différence dans les scores pourrait être due à la faible couverture des lexiques Lglex et Lglex-L*efff*. Le fait qu'un grand nombre de mots originaux soit mélangé avec des classes limite la puissance des regroupements lexicaux. Mais d'un autre côté, les statistiques disponibles pour les deux types de mots permettraient de mieux traiter d'éventuelles erreurs de classification. Cependant, nous ne voyons pas de solution acceptable pour démontrer l'exactitude de cette remarque. En revanche, pour montrer l'impact réel de l'étiquetage et de la lemmatisation, nous avons évalué par la validation croisée nos classes de verbes et d'adjectifs sur deux jeux de données particuliers. Dans le premier, les lemmes corrects ont été assignés aux mots des corpus d'évaluation. Quant à l'étiquetage, il est toujours réalisé par LGTagger. À l'opposé, dans le deuxième jeu, l'étiquetage correct est fourni

mais pas la lemmatisation. Nous avons indiqué dans la figure 6.25 les résultats des analyseurs selon les deux jeux de données. Les gains sont indiqués en absolu par rapport aux scores de référence. On peut voir que les écarts sont finalement très faibles entre les deux expériences, ce qui veut dire que l'étiquetage et la lemmatisation sont responsables à part égale des erreurs commises sur les textes.

| | $F_1$ | | LA | | UAS | |
|---|---|---|---|---|---|---|
| | BKY | Brown | BKY | Brown | BKY | Brown |
| LexSchem$_{verbes}$ | -0.19 | -0.24 | -0.1 | -0.1 | -0.25 | -0.15 |
| Le*fff*$_{adjectifs}$ | -0.17 | -0.23 | -0.1 | -0.1 | -0.22 | -0.11 |

(a) Lemmes corrects.

| | $F_1$ | | LA | | UAS | |
|---|---|---|---|---|---|---|
| | BKY | Brown | BKY | Brown | BKY | Brown |
| LexSchem$_{verbes}$ | -0.12 | -0.19 | -0.1 | -0.1 | -0.21 | -0.11 |
| Le*fff*$_{adjectifs}$ | -0.19 | -0.24 | -0.1 | -0.1 | -0.22 | -0.10 |

(b) Étiquettes correctes.

FIG. 6.25: Gains absolus obtenus lorsque l'étiquetage ou la lemmatisation est correct (par rapport aux résultats de référence).

**Scores par catégorie syntaxique**

Après l'observation des résultats globaux au corpus, nous allons à présent déterminer plus précisement ce qui a été éventuellement amélioré, et à l'opposé ce qui s'est dégradé, lorsque nous faisons appel à nos classes de mots. Dans le tableau 6.25, nous avons indiqué le score $F_1$ brut obtenu pour chaque constituant du jeu du FTB-UC, selon l'ensemble de classes utilisé. Ce score est, ici, un gain absolu calculé par rapport au score de base obtenu pour ce même constituant. Les scores d'intérêt sont mis en gras dans le tableau. Il est intéressant de noter que l'application de classes spécifiques à une catégorie de mot permet d'améliorer significativement la reconnaissance des syntagmes ayant pour noyau ces mots. En effet, pour les classes de verbes, les gains les plus importants sont obtenus pour les constituants verbaux `VN`, `VPpart` et `AdP`[187] qui représentent, à eux trois, environ 15% de la totalité des syntagmes. De même, pour les classes d'adjectifs, il s'agit du syntagme adjectival `AP`. Malgré tout, on peut voir que les classes de noms dégradent fortement les syntagmes nominaux et prépositionnels, ce qui ne fait que confirmer l'impact nul de ces classes.
Les faibles résultats constatés pour BKY, mais également pour BKY$_p$, ne sont pas encourageants. On peut rappeler que pendant la préparation de cette thèse, différentes versions de BKY ont vu le jour, chacune ayant un algorithme de plus en plus évolué. La première version est une implémentation de l'algorithme décrit dans (Petrov *et al.*, 2006), qui est basé sur les divisions successives de symboles (section I.2.3.2). Une deuxième version, issue de (Petrov

[187]On peut considérer que le syntagme `AdP` est généralement lié aux syntagmes verbaux.

| | #const | Base | | LexSchem$_{verbes}$ | | Le*fff*$_{adjectifs}$ | | Lglex$_{noms}$ | |
|---|---|---|---|---|---|---|---|---|---|
| | | BKY | Brown | BKY | Brown | BKY | Brown | BKY | Brown |
| NP | 40.4 | 82.17 | 78.34 | +0.02 | +0.30 | -0.06 | -0.03 | -0.09 | -0.22 |
| PP | 24.9 | 81.76 | 77.60 | -0.08 | +0.28 | -0.03 | -0.03 | -0.03 | -0.45 |
| VN | 12.5 | 94.95 | 93.71 | **+0.88** | **+1.40** | +0.25 | +0.40 | -0.14 | +0.04 |
| AP | 7.6 | 86.40 | 85.23 | +0.46 | +0.42 | **+0.80** | **+0.90** | +0.27 | -0.03 |
| COORD | 3.9 | 78.73 | 72.65 | -0.01 | -0.05 | = | +0.1 | -0.04 | +0.1 |
| VPinf | 3.1 | 78.18 | 74.68 | +0.10 | +0.22 | -0.27 | +0.4 | -0.12 | -0.05 |
| VPpart | 2.2 | 66.60 | 63.92 | **+2.84** | **+3.04** | +0.50 | +1.40 | +0.45 | -0.36 |
| Sint | 2.1 | 75.06 | 68.04 | -0.60 | +0.65 | +0.42 | +0.13 | +0.50 | +0.40 |
| Srel | 1.6 | 80.08 | 72.47 | -0.88 | -0.14 | -0.92 | +0.13 | -0.56 | +0.20 |
| Ssub | 1.5 | 76.90 | 72.45 | +0.51 | -0.14 | +0.50 | -0.02 | +0.56 | -0.04 |
| AdP | 0.3 | 50.11 | 51.00 | **+2.27** | **+1.50** | +1.54 | +0.85 | +1.09 | +0.51 |

TAB. 6.25: Gains $F_1$ par constituant des analyseurs BKY et Brown avec la validation croisée.

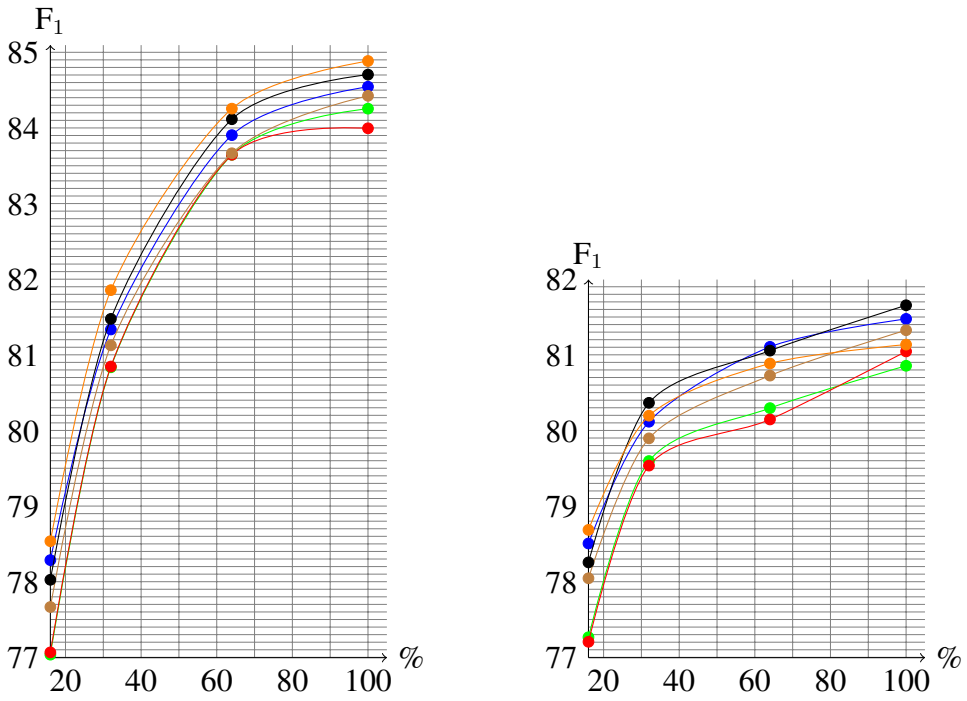
& Klein, 2007), introduit une phase de pré-filtrage des règles de la grammaire afin de limiter l'espace de recherche lors de l'analyse d'une phrase, réduisant ainsi le temps d'exécution et améliorant les performances générales. Et enfin, la dernière version est celle que nous utilisons dans l'ensemble des expériences de ce mémoire (Petrov, 2010). Chacune de ces versions dispose du même traitement à base de signature de mots permettant d'améliorer des mots rares et inconnus du français. Selon la version, nous avons pu constater de réelles différences en terme de performances, comme le prouve le tableau 6.26. Les scores $F_1$, UAS et LA sont indiqués en utilisant un protocole identique à la première expérience de cette section. En terme de scores de base, ceux de la version actuelle de BKY (sans produit de grammaires) sont significativement meilleurs que ceux des deux autres versions, qui obtiennent des scores identiques. On peut remarquer que les classes d'adjectifs et de noms sont toujours inefficaces. En revanche, avec les classes de verbes, la totalité des scores sont significatifs ($p<0.01$), et ceci pour les deux analyseurs. De plus, les scores observés sont globalement similaires à ceux obtenus par Brown. À partir de ces différentes observations, nous pouvons faire deux remarques. D'une part, le fait qu'un analyseur soit basé sur le paradigme lexicalisé, ou non lexicalisé, n'influe pas sur l'impact de nos regroupements lexicaux. D'autre part, les algorithmes sous-jacents aux analyseurs étant de plus en plus évolués, certains problèmes comme la dispersion des données sont mieux traités. Aussi, les classes de mots, et notamment de verbes, sont globalement moins efficaces.

| | $F_1$ | | | LA | | | UAS | | |
|---|---|---|---|---|---|---|---|---|---|
| | BKY$_{1.0}$ | BKY$_{1.1}$ | BKY | BKY$_{1.0}$ | BKY$_{1.1}$ | BKY | BKY$_{1.0}$ | BKY$_{1.1}$ | BKY |
| Base | 82.28 | 82.35 | 83.02 | 85.8 | 85.9 | 86.1 | 87.05 | 87.18 | 87.55 |
| LexSchem$_{verbes}$ | +0.59$^{\diamond}$ | +0.37$^{\diamond}$ | +0.14 | +0.3$^{\diamond}$ | +0.3$^{\diamond}$ | +0.02 | +0.54$^{\diamond}$ | +0.40$^{\diamond}$ | +0.37$^{\diamond}$ |
| Le*fff*$_{adjectifs}$ | +0.09 | +0.12 | +0.02 | = | = | +0.2 | +0.02 | +0.11 | +0.07 |
| Lglex-Le*fff*$_{noms}$ | +0.02 | -0.09 | -0.07 | +0.1 | -0.1 | = | +0.07 | -0.10 | -0.07 |

TAB. 6.26: Gains des différentes versions de BKY en fonction des regroupements lexicaux avec la validation croisée.

### 6.5.2 Variation de la quantité de données d'apprentissage

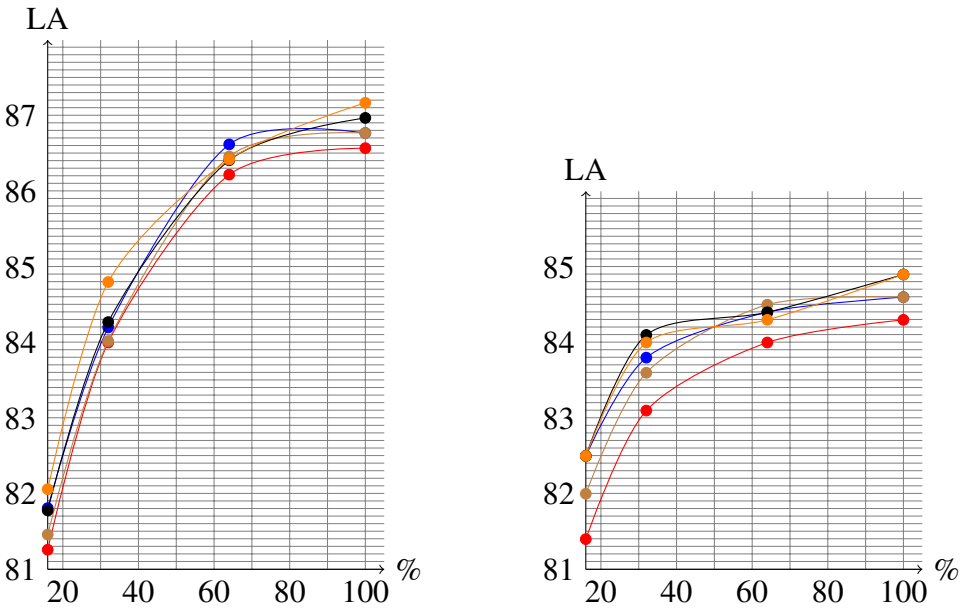
Nous avons vu dans la section I.2.5 que, dans le cas de corpus de petite taille, il existe de nombreux algorithmes permettant d'améliorer la qualité des analyses syntaxiques produites par un analyseur génératif. L'auto-apprentissage est une de ces méthodes, et consiste à effectuer l'apprentissage d'un analyseur sur un corpus composé d'analyses issues en partie d'un corpus annoté, et de phrases brutes préalablement analysées par ce même analyseur. Une autre méthode, appelée co-apprentissage, consiste à combiner les analyses de phrases brutes, générées par deux analyseurs distincts, afin de constituer un corpus d'apprentissage de plus grande taille. Il faut rappeler qu'un petit corpus d'apprentissage induit une forte dispersion des données. En effet, le nombre de règles rares à estimer est plus important qu'avec un corpus de plus grande taille. Pour contrer cet effet, et en parallèle des méthodes d'auto-apprentissage et de co-apprentissage, le regroupement lexical est une technique pertinente. (Candito *et al.*, 2011) ont, par exemple, amélioré avec succès la qualité des analyses de phrases provenant d'un petit corpus bio-médical du français. Le corpus d'apprentissage utilisé est le FTB-UC, dont les mots ont été remplacés par des classes plus générales. Dans leur expérience, ces classes ont été générées automatiquement par StatClust.
Ne disposant que du FTB-UC, nous avons simuler le fait d'avoir un plus petit corpus en limitant le nombre de phrases allouées au corpus d'apprentissage. Aussi, nous avons évalué l'impact de nos classes de verbes, noms et adjectifs sur trois corpus d'apprentissage de taille différente, composés de respectivement 2000, 4000 et 8000 phrases. Les corpus de développement et d'évaluation possèdent la même taille qu'initialement, soit 1235 phrases. Dans les graphiques montrés ci-dessous, nous avons indiqué les résultats bruts des analyseurs BKY et Brown sous forme de courbes d'apprentissage, où l'abscisse représente la taille du corpus d'apprentissage et l'ordonnée est le score observé. Les figures 6.26, 6.27 et 6.28 concernent respectivement le score $F_1$, le score LA, et le score UAS. Pour des raisons de clarté, seules les courbes obtenues sur le corpus d'évaluation sont montrées car des phénomènes identiques se produisent sur les courbes du corpus de développement. De même, les courbes de BKY et $BKY_p$ évoluent de manière parallèle, ce qui nous amène à ne montrer que celles de BKY. Dans les graphiques, les courbes possèdent une coloration spécifique selon le regroupement lexical évalué. Le bleu représente les classes de verbes, le marron les classes d'adjectifs, et le rouge les classes de noms. À titre de comparaison, les résultats de DFL (couleur noire) et de Statclust (couleur orange) sont également montrés. Les scores de base des analyseurs sont représentés par la courbe de couleur verte.
Avec un corpus d'apprentissage initial composé de seulement 2000 phrases, les gains obtenus par les deux analyseurs sont significatifs et relativement similaires. Pour les scores $F_1$ et UAS, ces gains se situent entre +1 et +1.5 pour les classes de verbes, et entre +0.5 et +1 pour les classes d'adjectifs. Quant au score LA, les gains sont moins importants, et oscillent entre 0 et 1. En ajoutant 2000 phrases, on observe que les gains restent significatifs malgré une légère baisse générale. À partir de 8000 phrases, des différences apparaissent entre les deux analyseurs. Alors que les gains de BKY continuent de décroître, ceux de Brown sont augmentés. Et enfin, en utilisant la totalité des phrases, on observe une stagnation des gains pour l'ensemble des courbes des deux analyseurs. En ce qui concerne StatClust et DFL, les courbes associées sont globalement au dessus des autres pour BKY. En revanche, pour Brown, l'écart avec les




(a) BKY (b) Brown

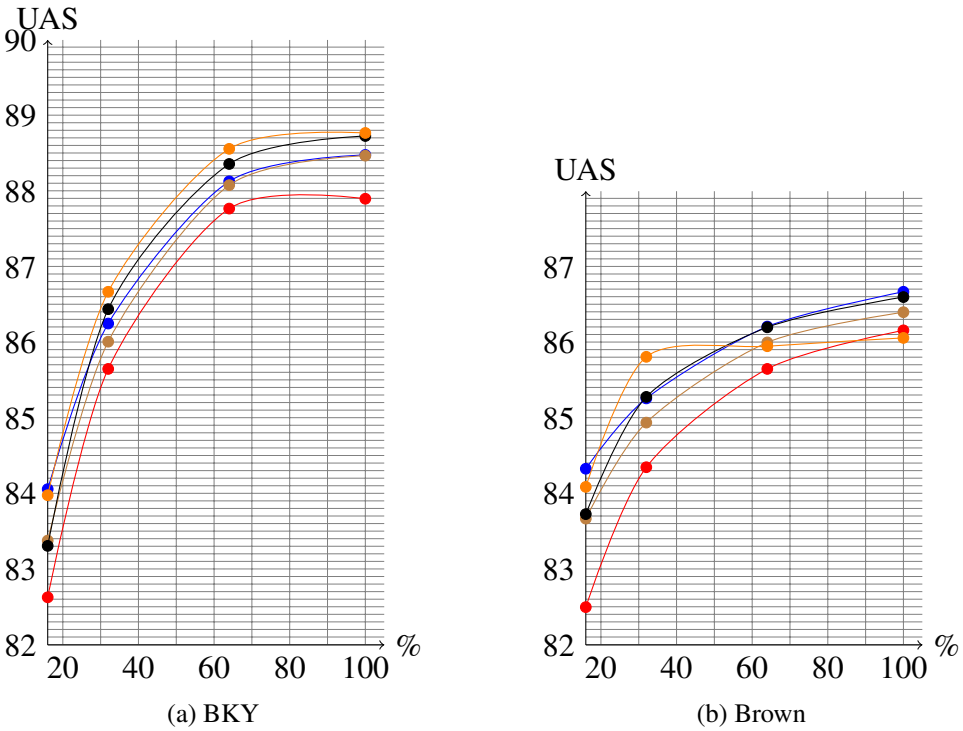
FIG. 6.26: Courbes d'apprentissage pour les scores $F_1$ bruts des analyseurs BKY et Brown selon les classes de mots. Le code couleur des courbes est le suivant : $\texttt{LexSchem}_{verbes}$=bleue, $\texttt{Le}fff_{adjectifs}$=marron, $\texttt{Lglex-Le}fff_{noms}$=rouge, DFL=noire et StatClust=orange.

résultats de nos classes est beaucoup moins prononcé.
D'après les constatations faites sur les courbes, nous pouvons formuler plusieurs remarques. Les gains importants observés avec un corpus d'apprentissage de 2000 phrases sont proportionnels à l'impact de la dispersion des données. Nos classes permettent de compenser le manque d'informations lexicales et syntaxiques, notamment au niveau des verbes et des adjectifs. Par exemple, au niveau lexical, le nombre de mots inconnus du corpus d'évaluation est réduit d'environ 25% grâce aux classes de verbes. Selon cette même logique, la chute des gains qui s'ensuit, avec 4000 phrases d'entraînement, est due à l'impact moindre de la dispersion. Les différences observées entre les deux analyseurs avec 8000 phrases peuvent trouver leurs explications dans la puissance respective des algorithmes des analyseurs. Grâce à l'algorithme de divisions de symboles, BKY est meilleur que l'analyseur lexicalisé Brown (Seddah *et al.*, 2009a). En effet, avec 2000 phrases, l'écart $F_1$ entre les deux analyseurs est de seulement 0.21, ce qui est non-significatif. Avec 4000 phrases, BKY creuse davantage l'écart (+1.10). Puis, à 8000 phrases, cet écart est d'environ 4 points. La croissance de cet écart va de pair avec le traitement de la dispersion des données, comme nous l'avons vu dans la section 6.5.1. Brown est plus sensible à ce phénomène et est donc plus réceptif aux classes de mots, et ceci indépendamment de la taille du corpus d'apprentissage.




(a) BKY (b) Brown

FIG. 6.27: Courbes d'apprentissage pour les scores LA bruts des analyseurs BKY et Brown selon les classes de mots. Le code couleur des courbes est le suivant : $\texttt{LexSchem}_{verbes}$=bleue, $\texttt{Lefff}_{adjectifs}$=marron, $\texttt{Lglex-Lefff}_{noms}$=rouge, DFL=noire et StatClust=orange.




(a) BKY (b) Brown

FIG. 6.28: Courbes d'apprentissage pour les scores UAS bruts des analyseurs BKY et Brown selon les classes de mots. Le code couleur des courbes est le suivant : $\texttt{LexSchem}_{verbes}$=bleue, $\texttt{Lefff}_{adjectifs}$=marron, $\texttt{Lglex-Lefff}_{noms}$=rouge, DFL=noire et StatClust=orange.

### 6.5.3 Réordonnancement discriminatif

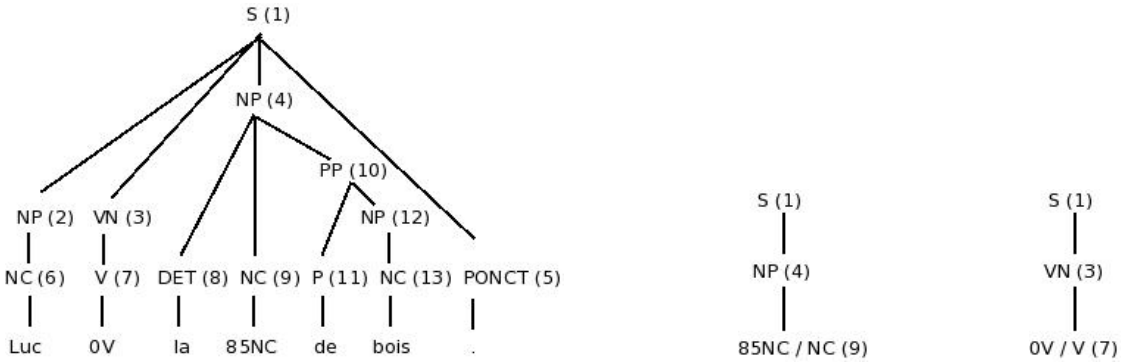
Jusqu'à présent, nous avons tenté, avec plus ou moins de succès, d'intégrer des données, provenant de lexiques, dans des systèmes d'analyse basés sur une grammaire PCFG. Les informations de sous-catégorisation sont utilisées de manière à générer des classes lexicales, dont le but est de remplacer les mots d'un corpus, et ainsi de réduire l'effet de dispersion des données. Nous avons vu dans la section I.2.4.6 une méthode de post-traitement, appelée réordonnancement discriminatif, permettant d'obtenir des scores élevés pour différentes langues. Cette méthode reclasse les $n$ analyses en sortie d'un analyseur syntaxique lambda[188] grâce à des observations, aussi appelées traits. Ces traits sont généralement obtenus automatiquement par l'intermédiaire de patrons de traits, qui sont appliqués sur le corpus d'apprentissage, préalablement analysé. La puissance du réordonnancement discriminatif provient du fait qu'il peut modéliser de nombreux traits sans avoir à poser d'hypothèses d'indépendances entre eux, et ceci contrairement aux grammaires PCFG. Il est important de définir un ensemble de patrons cohérent et pertinent dans le but d'extraire un maximum d'informations structurelles et lexicales à partir des arbres. Dans la section I.2.4.6, l'ensemble de patrons indiqué est utilisé par le réordonnanceur de Brown, mais également par notre réimplémentation de ce dernier. Il est composé de 12 patrons qui sont indépendants de la langue, car ils ne font pas d'hypothèses sur la langue traitée.
Nous avons défini deux stratégies d'intégration des informations des lexiques au processus de réordonnancement. L'une, que l'on pourrait qualifier d'*endogène*, génère des traits grâce à des informations extraites uniquement à partir des analyses candidates dont les mots sont remplacés par les classes lexicales, et l'autre, plutôt *exogène*, se base sur un ensemble de patrons consacrés aux ressources externes, à savoir les classes lexicales.

#### Stratégie "endogène"

La première solution consiste à appliquer les patrons de traits de base sur des analyses dont les mots ont été préalablement remplacés par des classes lexicales. Les traits potentiels contiennent maintenant des structures syntaxiques dont les feuilles peuvent être des classes. Il existe deux façons d'obtenir des analyses de ce type. Elles se différencient uniquement par la position où sont appliquées les étapes d'étiquetage et de lemmatisation, nécessaires au remplacement des mots par leur classe. Ces deux phases peuvent être effectuées, soit en tant que prétraitement, et, dans ce cas, l'analyseur PCFG est entraîné sur le corpus modifié par les classes[189], soit en tant que post-traitement, et, dans ce cas, les classes sont ajoutées après l'analyse du texte et donc avant le réordonnancement. La figure 6.29 illustre cette stratégie par l'exemple. L'analyse en constituants de la phrase *Luc recouvre la table de bois* a été modifiée par la configuration `Verbes+Noms`, et donc par le remplacement du verbe *recouvrir* et du nom *table* respectivement par les classes `0V`[190] et `85N`[191]. Puis, le patron `Word` a été appliqué sur les feuilles de

[188] L'unique condition est donc que l'analyseur puisse renvoyer une liste de $n$ analyses candidates.
[189] Cette solution est celle que nous avons expérimentée jusqu'à maintenant.
[190] Déterminée à partir du lexique L*efff* avec la configuration basée sur les fonctions syntaxiques.
[191] Déterminée à partir du lexique Lglex avec la configuration basée sur les classes du LG.

cet arbre. Nous rappelons que ce patron extrait les $n$ ancêtres d'un noeud terminal (ici, $n$ est fixé à 2). Le résultat est montré dans la figure 6.29b, et ce uniquement pour les deux terminaux associés aux classes.




(a) Analyse en constituants.

(b) Application du patron *Word*.

FIG. 6.29: (a) Une analyse en constituants de la phrase *Luc recouvre la table de bois* dont certains mots ont été remplacé par leur classe lexicale. (b) Résultat de l'application du patron `Word` sur les terminaux associés aux classes.

Au sujet du protocole d'évaluation, le réordonnanceur prend, en entrée, les 50 meilleures analyses de BKY ou de Brown, et applique les 12 patrons de base. Seule la configuration composée des trois meilleurs ensembles de classes a été évaluée (LexSchem$_{verbes}$, L*efff*$_{adjectifs}$ et Lglex-L*efff*$_{noms}$), et ce pour des raisons de performances. En effet, comme nous l'avons vu au cours de la discussion, les scores sont maximisés avec cette configuration lorsqu'elle est utilisé en tant que prétraitement des corpus et des textes. On peut noter que BKY$_p$ n'a pas été exploité car nous rappelons que la version actuelle ne permet pas de générer la liste des $n$ analyses candidates d'une phrase. De plus, sur les 8 grammaires traditionnellement utilisées par BKY, nous n'en avons exploité que trois, de graines 1, 4 et 8, et ce, pour des raisons de temps de calcul car il faut entraîner un modèle discriminant pour chaque grammaire de BKY. Pour cet analyseur, l'évaluation a donc été effectuée en calculant la moyenne des résultats obtenus par chaque réordonnanceur entraîné avec une des trois grammaires. Dans le tableau 6.30b, nous avons indiqué les scores oracle F$_1$ obtenus par les deux analyseurs sur les corpus de développement et d'évaluation. L'expérience notée `Base` correspond aux scores de base des analyseurs, alors que celle appelée `VNA` correspond à la configuration évaluée. Nous avons également indiqué, dans le tableau 6.30c, le nombre de traits générés à partir des corpus d'apprentissage préalablement analysés. On peut constater qu'avec un nombre moindre de traits générés, les scores oracle sont légèrement meilleurs avec la configuration `VNA`. Les gains absolus par rapport aux résultats de base varient entre +0.2 à +0.3.
Les résultats effectifs du réordonnanceur sur les corpus de développement et d'évaluation sont synthétisés dans la table 6.27. Les gains absolus sont calculés par rapport aux résultats de `Base`. Bien que la totalité des gains soient positifs, ils ne sont pas significatifs pour autant car ils ne varient qu'entre 0 et +0.24. Il s'agit des mêmes gains observés sur les scores oracle. On peut donc en conclure que les classes lexicales n'apportent au réordonnanceur que peu d'informations discriminantes supplémentaires.

| | BKY | Brown |
|---|---|---|
| Base | 92.35 | 90.84 |
| VNA | 92.57 | 91.11 |

(a) Scores oracle $F_1$ (DEV).

| | BKY | Brown |
|---|---|---|
| Base | 94.46 | 92.00 |
| VNA | 94.78 | 92.23 |

(b) Scores oracle $F_1$ (TEST).

| | BKY | Brown |
|---|---|---|
| Base | 5717126 | 5486490 |
| VNA | 5597541 | 5137298 |

(c) Nombre de traits générés.

FIG. 6.30: (b) Scores oracle $F_1$ obtenus par les analyseurs sur les corpus de développement (DEV) et d'évaluation (TEST), avec $n$ fixé à 50. (a) Nombres de traits obtenus en appliquant l'ensemble des patrons de traits de base sur les corpus d'apprentissage produits par les différents analyseurs.

| | BKY | | | Brown | | |
|---|---|---|---|---|---|---|
| | $F_1$ | LA | UAS | $F_1$ | LA | UAS |
| Base | 83.98 | 87.3 | 88.13 | 80.89 | 84.4 | 86.25 |
| VNA | +0.15 | = | +0.23 | +0.19 | +0.1 | +0.21 |

(a) Corpus de développement.

| | BKY | | | Brown | | |
|---|---|---|---|---|---|---|
| | $F_1$ | LA | UAS | $F_1$ | LA | UAS |
| Base | 85.67 | 88.3 | 89.10 | 82.10 | 85.2 | 86.59 |
| VNA | +0.21 | +0.1 | +0.24 | +0.17 | +0.1 | +0.15 |

(b) Corpus d'évaluation.

TAB. 6.27: Scores obtenus par le réordonnanceur combiné aux analyseurs Brown et BKY, avec $n$ fixé à 50, sur les corpus de développement et d'évaluation.

**Stratégie "exogène"**

La deuxième stratégie consiste à créer des patrons de traits consacrés aux classes lexicales. Les patrons sont seulement instanciés aux feuilles des analyses candidates, dont la valeur (le mot) est présente dans l'ensemble de classes utilisé en tant que ressource externe. Nous définissons un patron $T$ comme suit, pour chaque feuille (à la position $n$ de la phrase) de valeur $m$ (mot) dans l'analyse candidate $p$ :

$$T = (f(p, n), gp(n), cls(m))$$

où $f$ est une fonction à définir, $gp(n)$ est l'étiquette syntaxique du noeud père du préterminal de $m$, et $cls(m)$ est la classe lexicale associée au mot $m$. La fonction $f$ peut prendre une valeur parmi l'ensemble défini formellement dans le tableau 6.28. Nous utilisons trois types de fonctions : n-grammes de mots, n-grammes d'étiquettes morpho-syntaxiques, et les propriétés syntaxiques. Pour chacune de ces fonctions, $w(i)$ est le mot à la position $i$, $t(i)$ est l'étiquette morpho-syntaxique de $w(i)$, et $prop(w(i), p)$ est la valeur de la propriété syntaxique $p$ (d'un lexique) du mot à la position $i$.
Afin d'illustrer par l'exemple l'application des divers patrons, nous avons repris l'analyse en constituants de la phrase *Luc recouvre la table de bois* dont les feuilles contiennent les mots

| **n-grammes de mots** |
|---|
| $w(n+i), i \in \{-2,-1,0,1,2\}$ |
| $(w(n+i), w(n+i+1)), i \in \{-2,-1,0,1\}$ |
| **n-grammes d'étiquettes** |
| $t(n+i), i \in \{-2,-1,0,1,2\}$ |
| $(t(n+i), t(n+i+1)), i \in \{-2,-1,0,1\}$ |
| **couples de mots et d'étiquettes** |
| $(w(n+i), t(n+j)), (i,j) \in \{(1,0),(0,1),(-1,0),(0,-1)\}$ |
| **propriétés syntaxiques** |
| $(p, prop(w(n), p))$ |

TAB. 6.28: Fonctions utilisées par les patrons exogènes pour une feuille à la position $n$ de la phrase.

originaux. De plus, comme dans la stratégie précédente, nous disposons de deux ensembles de classes lexicales générés à partir des lexiques, l'un pour les verbes et l'autre pour les noms. On peut noter que la phase d'étiquetage des mots de la phrase est, ici, inutile car les étiquettes sont disponibles dans les noeuds préterminaux. Aussi, le verbe *recouvre* est associé à la classe `0`, et *table* à la classe `85`. Nous avons également à notre disposition des vecteurs de propriétés (et leur valeur) extraits des lexiques et qui sont associés aux deux mots[192]. Dans le cadre de notre exemple, le vecteur du verbe est `[(Suj,1) ; (Obj,1) ; (Objde,0)]`, et celui du nom est `[(Suj,1) ; (Obj,0) ; (Objde,1)]`. Les propriétés sont, ici, des fonctions syntaxiques, et les valeurs sont booléennes.

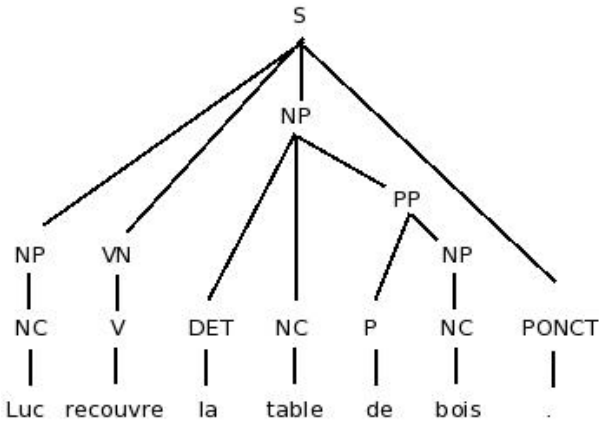



FIG. 6.31: Une analyse en constituants de la phrase *Luc recouvre la table de bois*.

Nous utilisons les bigrammes et unigrammes de mots et d'étiquettes pour extraire des indices lexicaux sur l'environnement direct d'un mot appartenant aux classes. Il s'agit principalement de déterminer des associations fréquentes de mots et d'étiquettes afin de faire correspondre une classe à un cadre de sous-catégorisation lexical, mais également syntaxique grâce aux proprié-

[192]Les propriétés incluses dans les vecteurs ne sont pas obligatoirement les mêmes que celles qui ont servies à la génération des classes, et peuvent même provenir de différents lexiques.

tés extraites des lexiques. Dans les tableaux 6.29, 6.30 et 6.31, nous avons indiqué quelques traits obtenus par l'application des différents patrons exogènes sur l'analyse d'exemple.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(x,n) = (w(n+i), w(n+i+1))$ | $n=1, i=0$ | $\rightarrow$ | (recouvre, la, VN, 0) |
| $\rightarrow T = (w(n+i), w(n+i+1), gp(n), cls(m))$ | $n=3, i=0$ | $\rightarrow$ | (table, de, NP, 85) |
| avec $i \in \{-2,-1,0,1\}$ | $n=3, i=1$ | $\rightarrow$ | (de, bois, NP, 85) |

TAB. 6.29: Quelques traits obtenus par l'application du patron exogène des bigrammes de mots sur l'analyse d'exemple.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(x,n) = (t(n+i), t(n+i+1))$ | $n=1, i=0$ | $\rightarrow$ | (V, DET, VN, 0) |
| $\rightarrow T = (t(n+i), t(n+i+1), gp(n), cls(m))$ | $n=3, i=0$ | $\rightarrow$ | (NC, P, NP, 85) |
| avec $i \in \{-2,-1,0,1\}$ | $n=3, i=1$ | $\rightarrow$ | (P, NC, NP, 85) |

TAB. 6.30: Quelques traits obtenus par l'application du patron exogène des bigrammes d'étiquettes sur l'analyse d'exemple.

| Patron | Variables | | Traits |
|---|---|---|---|
| $f(x,n) = (p, prop(w(n), p))$ | $n=1, i=0$ | $\rightarrow$ | (Obj, 1, VN, 0) |
| $\rightarrow T = (p, prop(w(n), p), gp(n), cls(m))$ | $n=3, i=0$ | $\rightarrow$ | (Objde, 1, NP, 85) |
| avec $i \in \{-2,-1,0,1\}$ | $n=3, i=0$ | $\rightarrow$ | (Obj, 0, NP, 85) |

TAB. 6.31: Quelques traits obtenus par l'application du patron exogène des propriétés syntaxiques sur l'analyse d'exemple.

Au sujet du protocole d'évaluation, le réordonnanceur prend en entrée les 50 meilleures analyses de BKY ou de Brown. Nous avons testé trois ensembles de patrons de traits : `std` est l'ensemble des 12 patrons de base, `regr` est l'ensemble des 6 patrons consacrés aux classes lexicales, et `all` est la combinaison des deux ensembles. Comme dans l'expérience précédente, les classes lexicales proviennent de trois lexiques, chacun lié à une catégorie grammaticale particulière (verbes, noms et adjectifs)[193]. Dans le tableau 6.32, nous avons indiqué le nombre de traits générés en fonction des divers ensembles de patrons. On peut constater que peu de traits sont extraits par les patrons `regr` (moins de 1% de la totalité des traits `all`).
Les résultats effectifs du réordonnanceur sur les corpus de développement et d'évaluation sont synthétisés dans le tableau 6.33. Les gains absolus sont calculés par rapport à `Base` qui correspond aux scores de base des analyseurs (sans réordonnancement). Lorsqu'ils sont utilisés seuls (`regr`), on peut constater que les gains sont négatifs ou nuls. L'inefficacité de ces traits est confirmée par l'expérience sur l'ajout des traits standards (`all`) dont les résultats sont très proches de `std`. Nous voyons deux raisons potentielles à cet échec : (a) le faible nombre de

[193]On peut noter que les vecteurs de propriétés exploités par l'un des patrons exogènes correspondent à l'identique à ceux utilisés pour calculer les classes.

| | BKY | Brown |
|---|---|---|
| std | 5717126 | 5486490 |
| regr | 31315 | 30874 |
| all | 5748441 | 5517364 |

TAB. 6.32: Nombres de traits obtenus en appliquant les divers ensembles des patrons sur les corpus d'apprentissage produits par les différents analyseurs.

traits exogènes générés à partir des données d'apprentissage, d'où un faible pouvoir discriminant. (b) les classes et les propriétés syntaxiques sont peut être des données trop abstraites pour être traitées correctement par un modèle discriminant.

| | BKY | | | Brown | | |
|---|---|---|---|---|---|---|
| | $F_1$ | LA | UAS | $F_1$ | LA | UAS |
| Base | 82.83 | 86.8 | 87.74 | 79.56 | 83.8 | 85.41 |
| std | +1.37 | +0.9 | +0.76 | +1.33 | +0.6 | +0.84 |
| regr | -0.12 | -0.1 | +0.05 | = | -0.2 | +0.1 |
| all | +1.41 | +0.92 | +0.88 | +1.37 | +0.61 | +0.90 |

(a) Corpus de développement.

| | BKY | | | Brown | | |
|---|---|---|---|---|---|---|
| | $F_1$ | LA | UAS | $F_1$ | LA | UAS |
| Base | 84.15 | 86.8 | 88.23 | 80.86 | 84.5 | 85.96 |
| std | +1.34 | +1.3 | +0.87 | +1.24 | +0.7 | +0.63 |
| regr | -0.20 | -0.2 | -0.10 | -0.11 | -0.1 | +0.02 |
| all | +1.39 | +1.35 | +0.96 | +1.26 | +0.72 | +0.68 |

(b) Corpus d'évaluation.

TAB. 6.33: Scores obtenus par le réordonnanceur combiné aux analyseurs Brown et BKY, avec $n$ fixé à 50, sur les corpus de développement et d'évaluation.

### 6.5.4 Analyse en dépendances

On peut rappeler que les scores de dépendance UAS, que nous avons indiqués jusqu'à présent, sont obtenus avec des analyseurs en constituants, dont la sortie a été automatiquement transformée en arbres de dépendances non typées. De plus, les performances de ces analyseurs, notamment BKY et $BKY_p$, sont au niveau de l'état de l'art. À travers nos précédentes expériences, nous avons montré que l'utilisation de regroupements lexicaux pouvaient améliorer significativement les performances générales. D'après les scores indiqués dans le tableau 6.24, les ensembles de classes ayant le plus d'impact sont ceux calculés à partir des lexiques de verbes, avec des gains de +0.55 pour Brown et +0.37 pour BKY. Les classes d'adjectifs et de noms n'ont, quant à elles, aucun effet positif et significatif. Dans la section 6.3, nous avons

énoncé les travaux sur l'intégration sous forme de traits de classes, générées avec StatClust, dans deux analyseurs en dépendances, Malt et MST (Koo *et al.*, 2008; Candito *et al.*, 2010b). Alors que cette approche est efficace pour des corpus de langue anglaise ou tchèque, elle l'est beaucoup moins dans le cadre du FTB-DEP pour le français.
En nous inspirant de ces travaux, nous allons évaluer, dans cette section, l'impact de nos classes sur le FTB-DEP avec l'analyseur Malt. On peut rappeler que, dans le FTB-DEP, les propriétés associées aux mots sont représentées sous la forme de colonnes, avec une colonne par propriété (section I.1.3). Ces propriétés sont le mot, son gouverneur (en général l'indice de ce mot dans la phrase) et la fonction syntaxique. (Candito *et al.*, 2010a) ont ajouté d'autres propriétés comme le lemme du mot, des éventuelles propriétés morphologiques (genre, nombre,...) ainsi que deux étiquettes morpho-syntaxiques, l'une étant tiré du jeu d'étiquettes du FTB-UC et une autre plus grossière provenant du jeu du FTB. De plus, chaque configuration, et donc chaque expérience, est basée sur tronc commun de patrons de traits, défini dans l'outil Bonsaï. Il est composé de 37 patrons calculant des traits à partir des données des différentes propriétés associées aux mots.
Comme dans la plupart des travaux précédents, nous avons défini une nouvelle propriété pour chaque mot d'un texte[194]. Concrètement, une nouvelle colonne est ajoutée, et a pour valeur la classe lexicale du mot, si elle est disponible. Par exemple, l'arbre de dépendances de la phrase *Luc recouvre la table fabriquée par Jean*, montré dans la figure 6.32, a été modifié par cette méthode, et avec une configuration de type `Verbes+Noms`. Dans la colonne `Classe`, les verbes *recouvre* et *fabriquée* ont pour classe `0`. Quant au nom *table*, la classe `85` lui a été attribuée.

| Mot | Lemme | ID_Gouv | Fct_Synt | $T_{FTB-UC}$ | $T_{FTB}$ | Prop_Morph | Classe |
|---|---|---|---|---|---|---|---|
| Luc | Luc | 2 | SUJ | NPP | N | g=m\|n=s\|s=p | - |
| recouvre | recouvrir | 0 | ROOT | V | V | m=ind\|n=s\|p=3\|t=pst | 0 |
| la | le | 4 | DET | DET | D | g=f\|n=s\|s=def | - |
| table | table | 2 | OBJ | NC | N | g=f\|n=s\|s=c | 85 |
| fabriquée | fabriquer | 4 | MOD | VPP | V | g=f\|m=part\|n=s\|t=past | 0 |
| par | par | 5 | P-OBJ | P | P | - | - |
| Jean | Jean | 6 | OBJ | NPP | N | g=m\|n=s\|s=p | - |

FIG. 6.32: Arbre de dépendances au format CoNLL de la phrase *Luc recouvre la table fabriquée par Jean*, modifié par l'ajout d'une colonne dont les valeurs sont les classes de la configuration `Verbes+Noms`.

Malt étant basé sur un modèle discriminant, la complexité induite rend l'apprentissage très long (en moyenne 20 heures). Aussi, nous nous sommes limité à quelques configurations de classes (8 au total), combinant nos classes avec celles de StatClust : `Verbes`, `Verbes+Noms`, `Verbes+Adjectifs`, `Noms+Adjectifs`, `Verbes+Noms+Adjectifs`, `StatClust`, `StatClust+Verbes` et `DFL`. Nous avons également évalué une configuration supplémentaire, appelée $\texttt{Verbes}_{prop}$, qui reprend le principe du patron exogène du réordonnanceur basé sur les propriétés syntaxiques. Concrètement, 7 colonnes (pour autant de fonctions syntaxiques possibles) sont ajoutées à chaque mot des textes et des corpus. Puis, les valeurs des propriétés des vecteurs associés aux verbes sont signalées dans les colonnes correspondantes.
Nous avons indiqué, dans le tableau 6.33a, les scores UAS obtenus par cet analyseur sur le

[194]Pour chaque nouvelle propriété, nous avons créé un nouveau patron associant le mot courant à sa classe.

corpus d'évaluation selon les différents ensembles de propriétés[195]. L'expérience notée `Base` correspond aux scores de base, alors que `Best` correspond aux scores de la meilleure configuration, à savoir `StatClust+Verbes`. À titre de comparaison, nous avons également indiqué les scores des analyseurs BKY, BKY$_p$ et Brown. Tout d'abord, on peut constater que Malt possède initialement les meilleurs performances, bien qu'elles soient du même ordre de grandeur que celles de BKY$_p$. Cependant, contrairement aux analyseurs en constituants, l'ajout de propriétés issues de ressources externes ne semble pas faire évoluer la reconnaissance des dépendances syntaxiques. En effet, le meilleur gain est de seulement +0.18 (tableau 6.33b). Cette expérience ne fait donc que confirmer les constatations faites précédemment dans (Candito *et al.*, 2010b), à savoir que les classes intégrées sous forme de traits d'un analyseur discriminatif ne permettent pas d'influer suffisamment sur les décisions à prendre. En effet, ces traits sont souvent minoritaires dans les modèles, et n'ont donc pas assez de pouvoir discriminant.

| | Malt | BKY | BKY$_p$ | Brown |
|---|---|---|---|---|
| Base | **89.17** | 88.22 | 89.04 | 85.96 |
| Best | 89.35 | 88.80 | **89.83** | 86.74 |

(a) Comparaison des scores UAS de base (`Base`) et des meilleurs scores (`Best`) de divers analyseurs, dont Malt.

| DFL | StatClust | V | V$_{prop}$ | N | A | N+V+A | StatClust+V |
|---|---|---|---|---|---|---|---|
| +0.02 | +0.11 | +0.07 | +0.10 | +0.09 | -0.05 | +0.02 | **+0.18** |

(b) Gains absolus calculés par rapport aux scores de base de Malt (`Base`).

FIG. 6.33: Scores UAS obtenus par Malt sur le corpus d'évaluation du FTB-DEP en fonction des ensembles de propriétés utilisés.

[195]Des phénomènes identiques surviennent sur le corpus de développement.

# Conclusion et perspectives

Nous rappelons que, dans ce mémoire, notre objectif était de proposer des stratégies permettant d'exploiter des ressources lexicales et syntaxiques afin d'améliorer la reconnaissance des unités multi-mots (segmentation du texte), et de diminuer l'impact de la dispersion des données. Pour ces deux problématiques, nous proposons de conclure et d'énoncer quelques perspectives.

## 1 Segmentation du texte

Dans le chapitre 5, nous avons montré que la tâche de segmentation est intimement liée à la tâche d'étiquetage, et qu'il est naturel de traiter les deux simultanément. L'écart entre les performances de l'étiquetage avec ou sans segmentation est de 1 à 3 points suivant le type de mesure utilisé. Par ailleurs, nous avons montré l'intérêt certain de l'intégration de ressources lexicales dans un modèle discriminant CRF, et en particulier les ressources d'unités polylexicales pour la segmentation. Nous voyons aussi qu'à ce niveau de performance, il est extrêmement difficile de gagner quelques dixièmes de points, même en mettant en jeu des ressources riches et variées. Ce chapitre a aussi été l'occasion d'une réflexion méthodologique poussée sur les différents moyens d'intégrer une ressource linguistique externe dans une chaîne de traitements faisant appel à un CRF. Une bonne partie de cette réflexion est d'ailleurs transposable à l'utilisation d'autres techniques d'apprentissage automatique. Les CRF, en intégrant traits locaux et combinaison statistique globale, apparaissent comme un modèle particulièrement bien adapté à l'hybridation entre ressources symboliques et modèles statistiques. Grâce à cette intégration, il a été possible de produire en peu de temps des étiqueteurs-segmenteurs très performants.
Nous avons également évalué deux stratégies discriminantes pour intégrer la reconnaissance des mots composés dans un système d'analyse syntaxique probabiliste : pré-identification des mots composés et repérage final des mots composés après réordonnancement des $n$ meilleures analyses. Les différents modèles comprenaient des traits spécifiques aux unités polylexicales. Nous avons montré que le pré-repérage permettait d'améliorer grandement la reconnaissance des mots composés, ainsi que les performances générales. En revanche, la stratégie basée sur le réordonnancement déçoit en terme d'identification de mots composés par rapport à la première méthode. Par ailleurs, l'intégration des traits généraux de (Charniak & Johnson, 2005) rend caducs les traits consacrés aux unités polylexicales. Il semble qu'aucune des deux méthodes ne soit entièrement satisfaisante.
Mais ces expériences ouvrent de nouvelles perspectives intéressantes. En terme d'algorithme,

nous pourrions combiner efficacement ces deux stratégies en permettant au pre-segmenteur de générer l'automate pondéré des segmentations lexicales possibles et de combiner ce dernier avec le système d'analyse composé d'un analyseur et d'un réordonnanceur. Ce travail est en cours et les résultats préliminaires que nous obtenons sont pertinents. Nous pourrions également transposer ces différentes stratégies à l'analyse en dépendance. Il serait également intéressant de se pencher sur la prise en compte d'informations lexicales et syntaxiques d'unités multi-mots autres que les mots composés, comme les entités nommées ou encore les phrases figées, qui sont disponibles dans nos ressources du français. Et enfin, il faudrait étendre ces diverses expériences à d'autres langues d'étude comme l'anglais ou l'espagnol, qui disposent également de ressources lexicales liées aux unités multi-mots (Chrobot *et al.*, 1999; Blanco, 2000).

## 2 Dispersion des données

Dans le chapitre précédent, nous avons montré que la dispersion lexicale des données a un impact important sur les performances des analyseurs génératifs, et qu'il est possible d'atténuer cet impact en utilisant des regroupements pertinents des mots en classes. Ces classes sont générées automatiquement par un algorithme applicable sur tout lexique syntaxique. Cependant, chaque lexique ayant des spécifités de codage, il est nécessaire de définir au préalable les propriétés lexicales et syntaxiques des entrées à exploiter. Aussi, nous avons pu générer et évaluer différents ensembles de classes pour des mots appartenant aux catégories suivantes : verbes, noms prédicatifs et adjectifs. En ce qui concerne les verbes, les divers ensembles de classes ont conduit à obtenir des gains similaires, et ce, quel que soit le nombre de classes produites. Les scores de dépendances sont significativement améliorés, ainsi que les scores $F_1$ (uniquement pour Brown).
Les différences de gains (légères) constatées entre les deux analyseurs pourraient signifier que la méthode est plus adaptée aux analyseurs lexicalisés. Le contre-exemple vient du fait que nous obtenons, avec les anciennes versions de BKY, les mêmes gains que ceux de Brown, mais des gains supérieurs à la version actuelle. On peut donc en conclure que la force de l'impact des classes lexicales ne semble pas dépendre du paradigme utilisé (lexicalisé ou non lexicalisé), mais qu'en revanche, le type d'algorithme sous-jacent à l'analyseur soit le facteur déterminant. Cette remarque pourrait expliquer les faibles gains obtenus par les méthodes DFL et StatClust par rapport à ceux de la littérature[196]. Quant aux classes d'adjectifs et de noms, certains problèmes, comme l'étiquetage, la lemmatisation ou encore la couverture, sont plus accentués que pour les verbes, ce qui cause des gains négatifs ou nuls. Dans la même logique, les combinaisons de classes ne sont pas concluantes.
Lors de la discussion des résultats, nous avons également montré que les classes lexicales permettent de résoudre partiellement la problématique liée à l'adaptation des analyseurs à de petits corpus. De manière logique, moins la taille du corpus d'apprentissage est conséquente, plus la dispersion des données sera forte. Or, l'impact des classes est de plus en plus fort à mesure

[196]Les expériences décrites dans (Candito & Crabbé, 2009; Candito & Seddah, 2010) sont basées sur $BKY_{1.0}$ et $BKY_{1.1}$

que cette dispersion augmente (classes de verbes et d'adjectifs). Il s'agit donc d'une solution efficace pour les situations où les corpus disponibles ne sont pas de taille suffisante pour des traitements à base de probabilités. En ce qui concerne les systèmes d'analyse basés sur un modèle discriminant, l'hypothèse qu'ils souffriraient moins de la dispersion lexicale est vérifiée par nos expériences. En effet, que ce soit pour le réordonnanceur ou pour l'analyseur en dépendance Malt, les différentes stratégies d'intégration de données lexicales et syntaxiques se sont révélées inefficaces car les gains sont majoritairement nuls ou non-significatifs.

En terme de perspectives, il serait intéressant de réitérer nos expériences sur d'autres langues d'étude. L'anglais, par exemple, dispose de corpus annotés de grande taille mais également de nombreux lexiques syntaxiques : ComLex (Grishman *et al.*, 1994), NomLex (Macleod *et al.*, 1998), FrameNet (Baker *et al.*, 1998) ou encore VerbNet (Kipper *et al.*, 2000). Ce dernier est une ressource proche du Lexique-Grammaire car il regroupe les verbes en classes d'après leurs comportements syntaxiques et sémantiques. Plutôt que de nous focaliser uniquement sur les grammaires hors-contexte, nous pourrions également exploiter les grammaires d'unification comme HPSG, CCG ou encore LFG. En effet, elles disposent d'un formalisme théoriquement plus adapté à l'intégration de données lexicales et syntaxiques. (Carroll & Fang, 2004) ont, par exemple, réussi à réduire d'environ 5% le nombre d'erreurs d'un analyseur HPSG de l'anglais en utilisant des informations de sous-catégorisation de verbes contenues dans un lexique obtenu de manière automatique.

# Bibliographie

## 3 Bibliographie


ABEILLÉ A., CLÉMENT L. & TOUSSENEL F. (2003). Building a treebank for French. In A. ABEILLÉ, Ed., *Treebanks : building and using parsed corpora*, Kluwer, Dordrecht.

AGIRRE E., BALDWIN T. & MARTÍNEZ D. (2008). Improving Parsing and PP Attachment Performance with Sense Information. In K. MCKEOWN, J. D. MOORE, S. TEUFEL, J. ALLAN & S. FURUI, Eds., *ACL*, p. 317–325 : The Association for Computer Linguistics.

AGIRRE E., BENGOETXEA K., GOJENOLA K. & NIVRE J. (2011). Improving dependency parsing with semantic classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies : short papers - Volume 2*, HLT '11, p. 699–703, Stroudsburg, PA, USA : Association for Computational Linguistics.

ARUN A. & KELLER F. (2005). Lexicalization in crosslinguistic probabilistic parsing : The case of French. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, p. 306–313, Ann Arbor, MI.

ATTIA M., FOSTER J., HOGAN D., ROUX J. L., TOUNSI L. & VAN GENABITH J. (2010). Handling Unknown Words in Statistical Latent-Variable Parsing Models for Arabic, English and French. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, SPMRL '10, p. 67–75, Stroudsburg, PA, USA : Association for Computational Linguistics.

AW A., ZHANG M., XIAO J. & SU J. (2006). A phrase-based statistical model for SMS text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, p. 33–40, Stroudsburg, PA, USA : Association for Computational Linguistics.

BACCHIANI M., RILEY M., ROARK B. & SPROAT R. (2006). MAP adaptation of stochastic grammars. *Computer Speech and Language*, p. 41–68.

BAKER C. F., FILLMORE C. J. & LOWE J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, p. 86–90, Stroudsburg, PA, USA : Association for Computational Linguistics.

BAKER J. K. (1979). Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, p. 547–550.

BERGER A. L., PIETRA V. J. D. & PIETRA S. A. D. (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.*, **22**, 39–71.

BIKEL D. (2000). A statistical model for parsing and word-sense disambiguation. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong.

BIKEL D. M. (2004). Intricacies of Collins' Parsing Model. *Comput. Linguist.*, **30**, 479–511.

BLACK E., S.ABNEY, FLICKINGER D., GDANIEC C., GRISHMAN R., HARRISON P., HINDLE D., INGRIA R., JELINEK F., KLAVANS J., LIBERMAN M., MARCUS M., ROUKOS S., SANTORINI B. & STRZALKOWSKI T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA Speech and Naturale Language Workshop*, p. 306–311.

BLANC O., CONSTANT M. & WATRIN P. (2007). Segmentation in super-chunks with a finite-state approach. In *Proceedings of the 6th Workshop on Finite-State Methods and Natural Language Processing (FSMNLP'07)*, p. 62 – 73.

BLANCO X. (2000). Les dictionnaires électroniques de l'espagnol (DELASs et DELACs). In A. MANASTER-RAMER, Ed., *Lingvisticae Investigationes XXIII : 2*, Amsterdam/Philadelphia : John Benjamins Publishing Co.

BLUM A. & MITCHELL T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, p. 92–100, New York, NY, USA : ACM.

BÖHMOVÁ A., HAJIČ J., HAJIČOVÁ E. & HLADKÁ B. (2001). The Prague Dependency Treebank : Three-Level Annotation Scenario. In A. ABEILLÉ, Ed., *Treebanks : Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

BOOTH T. L. (1969). Probabilistic representation of formal languages. In *Proceedings of the 10th Annual Symposium on Switching and Automata Theory (swat 1969)*, p. 74–81, Washington, DC, USA : IEEE Computer Society.

BOSER B. E., GUYON I. M. & VAPNIK V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, p. 144–152, New York, NY, USA : ACM.

BOURIGAULT D., FABRE C., FRÉROT C., JACQUES M.-P. & OZDOWSKA S. (2005). Syntex, analyseur syntaxique de corpus. In *Actes des 12èmes journées sur le Traitement Automatique des Langues Naturelles*, Dourdan, France.

BOURIGAULT D. & FRÉROT C. (2006). Acquisition et évaluation sur corpus de propriétés de sous-catégorisation syntaxique. *Traitement Automatique des Langues*, **47**(3), 141–154.

BRANTS S., DIPPER S., HANSEN S., LEZIUS W. & SMITH G. (2002). The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

BRANTS T. (2000). TnT - A statistical part-of-speech tagger. In *ANLP 2000*, p. 224–231, Seattle, WA.

BRILL E. (1995). Transformation-based error-driven learning and natural language processing : A case study in part-of-speech tagging. In *ACL 1995*, p. 543–565, Cambridge, Massachusetts.

BRISCOE T. & CARROLL J. (1993). Generalized probabilistic LR parsing of natural language (Corpora) with unification-based grammars. *Comput. Linguist.*, **19**(1), 25–59.

BRISCOE T., CARROLL J., GRAHAM J. & COPESTAKE A. (2002). Relational Evaluation Schemes. In *Proceedings of the Beyond PARSEVAL Workshop at the 3rd International Conference on Language Resources and Evaluation*, p. 4–8, Las Palmas, Gran Canaria.

BRISCOE T. & WAEGNER N. (1992). Robust stochastic parsing using the inside-outside algorithm. In *AAAI-92 Workshop on Statistically-Based NLP Techniques*.

BROWN P. F., DELLA V. J., DESOUZA P. V., LAI J. C. & MERCER R. L. (1992). Class-based n-gram models of natural language. In *Computational linguistics, 18(4)*, p. 467–479.

BUCHHOLZ S. & MARSI E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, p. 149–164, Stroudsburg, PA, USA : Association for Computational Linguistics.

CAFFERKEY C., HOGAN D. & VAN GENABITH J. (2007). Multi-word units in treebank-based probabilistic parsing and generation. In *Proceedings of the 10th International Conference on Recent Advances in Natural Language Processing (RANLP-07)*.

CANDITO M. & CRABBÉ B. (2009). Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technology (IWPT'09)*, p. 138–141.

CANDITO M., CRABBÉ B. & DENIS P. (2010a). Statistical French dependency parsing : treebank conversion and first results. In *Proceedings of LREC10*.

CANDITO M., CRABBÉ B., DENIS P. & GUÉRIN F. (2009). Analyse syntaxique du français : des constituants aux dépendances. In *Proceedings of TALN'09*, Senlis, France.

CANDITO M., HENESTROZA ANGUIANO E. & SEDDAH D. (2011). A word clustering approach to domain adaptation : Effective parsing of biomedical texts. In *Proceedings of IWPT '11*.

CANDITO M., NIVRE J., DENIS P. & ANGUIANO E. H. (2010b). Benchmarking of statistical dependency parsers for French. In *Proceedings of the 23rd International Conference on Computational Linguistics : Posters*, COLING '10, p. 108–116, Stroudsburg, PA, USA : Association for Computational Linguistics.

CANDITO M. & SEDDAH D. (2010). Parsing word clusters. In *Proceedings of the first NAACL HLT Workshop on Morphologically-Rich Languages (SPRML2010)*, p. 76–84, Los Angeles, California.

CARRERAS X. (2007). Experiments with a higher-order projective dependency parser. In *In Proc. EMNLP-CoNLL*.

CARROLL J., BRISCOE T. & SANFILIPPO A. (1998). Parser Evaluation : a Survey and a New Proposal. In *Proc. LREC98*.

CARROLL J. & FANG A. C. (2004). The automatic acquisition of verb subcategorisations and their impact on the performance of an HPSG parser. In *Proceedings of the 1st International Conference onNatural Language Processing*, Sanya City, China.

CER D., DE MARNEFFE M.-C., JURAFSKY D. & MANNING C. D. (2010). Parsing to Stanford Dependencies : Trade-offs between speed and accuracy. In *7th International Conference on Language Resources and Evaluation (LREC 2010)*.

CHARNIAK E. (1997). Statistical Parsing with a Context-Free Grammar and Word Statistics. In *Proceedings of AAAI*, p. 598–603.

CHARNIAK E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, p. 132–139, San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

CHARNIAK E. & JOHNSON M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL05)*.

CHIEU H. L. & NG H. T. (2002). A maximum entropy approach to information extraction from semi-structured and free text. In *Eighteenth national conference on Artificial intelligence*, p. 786–791, Menlo Park, CA, USA : American Association for Artificial Intelligence.

CHROBOT A., COURTOIS B., HAMMANI M., GROSS M. & ZELLAGUI K. (1999). *Dictionnaire Electronique DELAC anglais : noms composés*. rapport technique n°59, LADL, Université Paris 7.

CHURCH K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, ANLC '88, p. 136–143, Stroudsburg, PA, USA : Association for Computational Linguistics.

CLARK S., CURRAN J. R. & OSBORNE M. (2003). Bootstrapping POS taggers using unlabelled data. In *Proceedings of CoNLL-2003*, p. 49–55.

COHN T., GOLDWATER S. & BLUNSOM P. (2009). Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies : The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, p. 548–556, Stroudsburg, PA, USA : Association for Computational Linguistics.

COLLINS M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the 17th ICML*, p. 175–182.

COLLINS M. (2002). Discriminative training methods for hidden Markov models : theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, p. 1–8, Stroudsburg, PA, USA : Association for Computational Linguistics.

COLLINS M. (2003). Head-driven statistical models for natural language parsing. In *Computational Linguistics*.

COLLINS M. & KOO T. (2005). Discriminative Reranking for Natural Language Parsing. *Comput. Linguist.*, **31**(1), 25–70.

COLLINS M. & ROARK B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA : Association for Computational Linguistics.

COLLINS M. & SINGER Y. (1999). Unsupervised Models for Named Entity Classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, p. 100–110.

COLLINS M. J. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, p. 184–191, Stroudsburg, PA, USA : Association for Computational Linguistics.

COLLINS M. J. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI9926110.

CONSTANT M. & SIGOGNE A. (2011). MWU-aware Part-of-Speech Tagging with a CRF model and lexical resources. In *ACL Workshop on Multiword Expressions : from Parsing and Generation to the Real World (MWE'11)*, France.

CONSTANT M., SIGOGNE A. & WATRIN P. (2012a). Discriminative Strategies to Integrate Multiword Expression Recognition and Parsing. In *ACL 2012. To appear.*

CONSTANT M., SIGOGNE A. & WATRIN P. (2012b). La reconnaissance des mots composés à l'épreuve de l'analyse syntaxique et vice-versa : évaluation de deux stratégies discriminantes. In *TALN 2012. To appear.*

CONSTANT M., TELLIER I., DUCHIER D., DUPONT Y., SIGOGNE A. & BILLOT S. (2011). Intégrer des connaissances linguistiques dans un CRF : application à l'apprentissage d'un segmenteur-étiqueteur du français. In *Conférence sur le traitement automatique des langues naturelles (TALN'11)*, France.

CONSTANT M. & TOLONE E. (2010). A generic tool to generate a lexicon for NLP from Lexicon-Grammar tables. In M. D. GIOIA, Ed., *Actes du 27e Colloque international sur le lexique et la grammaire (L'Aquila, 10-13 septembre 2008). Seconde partie*, volume 1 of *Lingue d'Europa e del Mediterraneo, Grammatica comparata*, p. 79–93. Aracne. ISBN 978-88-548-3166-7.

CONSTANT M. & WATRIN P. (2008). Networking Multiword Units. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL'08)*, number 5221 in Lecture Notes in Artificial Intelligence, p. 120 – 125 : Springer-Verlag.

COURTOIS B., GARRIGUES M., GROSS G., GROSS M., JUNG R., MATHIEU-COLAS M., MONCEAUX A., PONCET-MONTANGE A., SILBERZTEIN M. & VIVÉS R. (1997). *Dictionnaire électronique DELAC : les mots composés binaires*. Rapport interne 56, University Paris 7, LADL.

COURTOIS B. & SILBERZTEIN M. (1990). Dictionnaires électroniques du français. Présentation. In *Larousse, editor, Langue Française*.

COWELL R. G., DAWID A. P., LAURITZEN S. L. & SPIEGELHALTER D. J. (1999). *Probabilistic networks and expert systems*. Statistics for Engineering and Information Science. New York : Springer-Verlag.

COWELL R. G., DAWID A. P., LAURITZEN S. L. & SPIEGELHALTER D. J. (2007). *Probabilistic Networks and Expert Systems : Exact Computational Methods for Bayesian Networks*. Springer Publishing Company, Incorporated, 1st edition.

CRABBÉ B. & CANDITO M. (2008). Expériences d'analyse syntaxique statistique du français. In *Actes de la 15ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN'08)*, p. 45–54, Avignon, France.

CRAMMER K., DEKEL O., KESHET J., SHALEV-SHWARTZ S. & SINGER Y. (2006). Online Passive-Aggressive Algorithms. *J. Mach. Learn. Res.*, **7**, 551–585.

CRAMMER K. & SINGER Y. (2003). Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, **3**, 951–991.

CURRY H. B. & FEYS R. (1958). *Combinatory Logic, Vol. 1.* North-Holland.

DAILLE B. (1995). Repérage et extraction de terminologie par une approche mixte statistique et linguistique. *traitement Automatique des Langues (TAL)*, **36**(1-2), 101–118.

DANLOS L. (1985). *Génération automatique de textes en langues naturelles*. ERI, études et recherches en informatiques. Paris : Masson.

DANLOS L. & SAGOT B. (2007). Comparaison du Lexique-Grammaire et de Dicovalence : vers une intégration dans le Lefff. In *Actes de TALN 07*.

DARROCH J. N. & RATCLIFF D. (1972). Generalized iterative scaling for log-linear models. In *The Annals of Mathematical Statistics*, volume 43, p. 1470–1480.

DE MARNEFFE M.-C., MACCARTNEY B. & MANNING C. D. (2006). Generating typed dependency parses from phrase structure parses. In *In LREC 2006*.

DELLA PIETRA S., DELLA PIETRA V. & LAFFERTY J. (1997). Inducing Features of Random Fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, **19**, 380–393.

DENIS P. & SAGOT B. (2009). Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *PACLIC 2009*, Hong Kong.

DENIS P. & SAGOT B. (2010). Exploitation d'une ressource lexicale pour la construction d'un étiqueteur morphosyntaxique état-de-l'art du français. In *Traitement Automatique des Langues Naturelles : TALN 2010*, Montréal, Canada.

DEOSKAR T. (2008). Re-estimation of lexical parameters for treebank PCFGs. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, p. 193–200, Manchester, Great Britain.

DEOSKAR T., ROOTH M. & SIMA'AN K. (2009). Smoothing fine-grained PCFG lexicons. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, p. 214–217, Stroudsburg, PA, USA : Association for Computational Linguistics.

DIAS G. (2003). Multiword Unit Hybrid Extraction. In *Proceedings of the Workshop on Multiword Expressions of the 41st Annual Meeting of the Association of Computational Linguistics (ACL 2003)*, p. 41–49.

DYBRO-JOHANSENN A. (2004). Extraction automatique de grammaires à partir d'un corpus français. In *MASTER'S THESIS, Université Paris 7*.

EARLEY J. (1970). An efficient context-free parsing algorithm. In *Communication of the ACM, 13(2)*.

EISNER J. M. (1996). Three new probabilistic models for dependency parsing : an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, COLING '96, p. 340–345, Stroudsburg, PA, USA : Association for Computational Linguistics.

ERYIGIT G., ILBAY T. & ARKAN CAN O. (2011). Multiword Expressions in Statistical Dependency Parsing. In *Proceedings of the IWPT Workshop on Statistical Parsing of Morphologically-Rich Languages (SPRML'11)*.

EYNDE K. & BLANCHE-BENVENISTE C. (1978). Syntaxe et mécanismes descriptifs : présentation de l'approche pronominale. *Cahiers de Lexicologie 32, 3-27*.

EYNDE K. & PIET M. (2003). La valence : l'approche pronominale et son application au lexique verbal. *Journal of French Language studies*, p. 63–104.

FAN R.-E., CHANG K.-W., HSIEH C.-J., WANG X.-R. & LIN C.-J. (2008). LIBLINEAR : A Library for Large Linear Classification. *Journal of Machine Learning Research*, **9**.

C. FELLBAUM, Ed. (1998). *WordNet : An Electronic Lexical Database*. MIT Press.

FINKEL J. R., GRENAGER T. & MANNING C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, p. 363–370, Stroudsburg, PA, USA : Association for Computational Linguistics.

FINKEL J. R., KLEEMAN A. & MANNING C. D. (2008). Efficient, feature-based, conditional random field parsing. In *In Proc. ACL/HLT*.

FINKEL J. R. & MANNING C. D. (2009). Joint Parsing and Named Entity Recognition. In *Proceedings of NAACL*.

FOSTER J. (2010). "cba to check the spelling" investigating parser performance on discussion forum posts. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, p. 381–384, Stroudsburg, PA, USA : Association for Computational Linguistics.

FOSTER J., WAGNER J., SEDDAH D. & VAN GENABITH J. (2007). Adapting WSJ-trained parsers to the British National Corpus using in-domain self-training. In *Proceedings of the 10th International Conference on Parsing Technologies*, IWPT '07, p. 33–35, Stroudsburg, PA, USA : Association for Computational Linguistics.

FREUND Y., IYER R., SCHAPIRE R. E. & SINGER Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, **4**, 933–969.

GARDENT C., GUILLAUME B., PERRIER G. & FALK I. (2006). Extraction d'information de sous-catégorisation à partir des tables du LADL. In *Traitement Automatique de la Langue Naturelle - TALN 2006*, Leuven/Belgique.

GILDEA D. (2001). Corpus Variation and Parser Performance. In L. LEE & D. HARMAN, Eds., *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, p. 167–202.

GILLICK L. & COX S. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of ICASSP'89*.

GIMÉNEZ J. & MÀRQUEZ L. (2004). SVMTool : A general pos tagger generator based on support vector machines. In *Language Resources and Evaluation*.

GRAFF D. (1995). North American News Text Corpus. In *Linguistic Data Consortium*, Philadelphia, USA.

GREEN S., DE MARNEFFE M.-C., BAUER J. & MANNING C. D. (2011). Multiword expression identification with tree substitution grammars : a parsing tour de force with French. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, p. 725–735, Stroudsburg, PA, USA : Association for Computational Linguistics.

GRISHMAN R., MACLEOD C. & MEYERS A. (1994). Comlex syntax : building a computational lexicon. In *COLING-94*.

GROSS M. (1975). ,. In HERMANN, Ed., *Méthodes en syntaxe*, Paris, France.

GROSS M. (1986). Lexicon Grammar. The Representation of Compound Words. In *Proceedings of Computational Linguistics (COLING'86)*.

GROSS M. (1994). Constructing Lexicon-grammars. In ATKINS & ZAMPOLLI, Eds., *Computational Approaches to the Lexicon*, p. 213–263.

GROSS M. (1997). The construction of local grammars. In D. J. LIPCOLL, D. H. LAWRIE & A. H. SAMEH, Eds., *Finite-State Language Processing*, p. 329–352. Cambridge, Mass. : The MIT Press.

HAJIČ J. (1998). Building a Syntactically Annotated Corpus : The Prague Dependency Treebank. In E. HAJIČOVÁ, Ed., *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, p. 12–19. Prague Karolinum, Charles University Press.

HAJIČ J. (2000). Morphological tagging : data vs. dictionaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, p. 94–101, Stroudsburg, PA, USA : Association for Computational Linguistics.

HALL K. (2007). K-best spanning tree parsing. In *Proceedings of ACL, Prague, Czech Republic, June*.

HATHOUT N. & NAMER F. (1998). Automatic Construction and Validation of French Large Lexical Resources : Reuse of Verb Theoretical Linguistic Descriptions. In *Proceedings of the Language Resources and Evaluation Conference (LREC'98)*.

HECKERMAN D., CHICKERING D. M., MEEK C., ROUNTHWAITE R. & KADIE C. (2001). Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, **1**, 49–75.

HOGAN D., FOSTER J. & VAN GENABITH J. (2011). Decreasing lexical data sparsity in statistical syntactic parsing - experiments with named entities. In *Proceedings of ACL Workshop Multiword Expressions : from Parsing and Generation to the Real World (MWE'2011)*.

HOPKINS M. & LANGMEAD G. (2009). Cube pruning as heuristic search. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 1 - Volume 1*, EMNLP '09, p. 62–71, Stroudsburg, PA, USA : Association for Computational Linguistics.

HOVY E., MARCUS M., PALMER M., RAMSHAW L. & WEISCHEDEL R. (2006). OntoNotes : the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume : Short Papers*, NAACL-Short '06, p. 57–60, Stroudsburg, PA, USA : Association for Computational Linguistics.

HUANG L. (2008). Discriminative Parsing with Non-Local Features. In *Proceedings of ACL 08*.

HUANG L. & CHIANG D. (2005). Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, p. 53–64, Stroudsburg, PA, USA : Association for Computational Linguistics.

HUANG Z. (2009). Selftraining PCFG grammars with latent annotations across languages. In *In EMNLP*.

HYE HAN C., LAVOIE B., PALMER M., RAMBOW O., KITTREDGE R., KORELSKY T., KIM N. & KIM M. (2000). Handling Structural Divergences and Recovering Dropped Arguments in a Korean/English Machine Translation System. *Envisioning Machine Translation in the Information Future*, p. 168–176.

JAYNES E. (1957). Information Theory and Statistical Mechanics. II. *Physical Review*, **106**(2), 171–190.

JIMÉNEZ V. M. & MARZAL A. (2000). Computation of the N Best Parse Trees for Weighted and Stochastic Context-Free Grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, p. 183–192, London, UK : Springer-Verlag.

JOHANSSON R. & NUGUES P. (2007). Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia.

JOHNSON M. (1998). PCFG models of linguistic tree representations. *Comput. Linguist.*, **24**, 613–632.

JOHNSON M. & URAL A. E. (2010). Reranking the Berkeley and brown parsers. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, p. 665–668, Stroudsburg, PA, USA : Association for Computational Linguistics.

JUDGE J., CAHILL A. & VAN GENABITH J. (2006). QuestionBank : creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, p. 497–504, Stroudsburg, PA, USA : Association for Computational Linguistics.

JURAFSKY D. & MARTIN J. H. (2000). *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 1 edition. neue Auflage kommt im Frühjahr 2008.

KAHANE S. (2001). Grammaires de dépendance formelles et théorie Sens-Texte. In *Traitement Automatique des Langues Naturelles*.

KAPLAN R. M. & BRESNAN J. (1995). Lexical-Functional Grammar : A Formal System for Grammatical Representation.

KARLSSON F., VOUTILAINEN A., HEIKKILÄ J. & ANTILLA A. (1995). Constraint Grammar : A language-independent system for parsing unrestricted text. In M. DE GRUYTER, Ed., *Natural Language Processing, No 4*, Berlin.

KASAMI T. (1965). An efficient recognition and syntax-analysis algorithm for context-free languages. In *Scientific report AFCRL-65-758, Air Force Cambridge Research Lab*.

KIPPER K., DANG H. T. & PALMER M. (2000). Class-Based Construction of a Verb Lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, p. 691–696 : AAAI Press.

KLEIN D. & MANNING C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the Annual Meeting of the ACL*.

KOO T., CARRERAS X. & COLLINS M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of ACL-08*.

KORHONEN A., GORRELL G. & MCCARTHY D. (2000). Statistical filtering and subcategorization frame acquisition. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora : held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, p. 199–206, Stroudsburg, PA, USA : Association for Computational Linguistics.

KUBLER S. (2008). The PaGe 2008 shared task on parsing german. In *Proceedings of ACL-08*.

KÜBLER S., HINRICHS E. W. & MAIER W. (2006). Is it really that difficult to parse German ? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, p. 111–119, Stroudsburg, PA, USA : Association for Computational Linguistics.

KUPSC A. (2007). Extraction automatique de cadres de sous-catégorisation verbale pour le français à partir d'un corpus arboré. In *TALN*, Toulouse, France.

KÜBLER S. & TÜBINGEN U. (2005). How do treebank annotation schemes influence parsing results ? or how not to compare apples and oranges. In *In RANLP*, p. 293–300.

LAFFERTY J., MCCALLUM A. & PEREIRA F. (2001). Conditional Random Fields : Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, p. 282–289, Williamstown, USA.

LANDES S., LEACOCK C. & TENGI R. I. (1998). Building Semantic Concordances. In *WordNet : an Electronic Lexical Database* : MIT MIT Press.

LAPORTE E. & MONCEAUX A. (1999). Elimination of lexical ambiguities by grammars. The ELAG system. In *Lingvisticae Investigationes XXII*, p. 341–367.

LARI K. & YOUNG S. (1990). The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, **4**(1), 35 – 56.

LAVERGNE T., CAPPÉ O. & YVON F. (2010). Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, p. 504–513, Stroudsburg, PA, USA : Association for Computational Linguistics.

LEASE M. & CHARNIAK E. (2005). Parsing Biomedical Literature. In R. DALE, K.-F. WONG, J. SU & O. KWONG, Eds., *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP'05)*, volume 3651 of *Lecture Notes in Computer Science*, p. 58 – 69, Jeju Island, Korea : Springer-Verlag.

LEROUX J., FAVRE B., MIRROSHANDEL S. & NASR A. (2011). Modèles génératif et discriminant en analyse syntaxique : expériences sur le corpus arboré de Paris 7. In *TALN 2011*, Montpellier, France.

LEVENSHTEIN (1966). *Binary code capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady 10.

LIANG P. (2005). Semi-supervised learning for natural language. In *MASTER'S THESIS, MIT*.

LIN D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, p. 768–774, Stroudsburg, PA, USA : Association for Computational Linguistics.

LIN X., FAN Y., ZHANG M., WU X. & CHI H. (2009). Refining Grammars for Parsing with Hierarchical Semantic Knowledge. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, p. 1298–1307, Singapore : Association for Computational Linguistics.

LOWERRE B. T. (1976). *The harpy speech recognition system.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA. AAI7619331.

MACLEOD C., GRISHMAN R., MEYERS A., BARRETT L. & REEVES R. (1998). NOMLEX : A Lexicon of Nominalizations. In *In Proceedings of Euralex98*, p. 187–193.

MAGERMAN D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, p. 276–283, Stroudsburg, PA, USA : Association for Computational Linguistics.

MAIER W. (2006). Annotation schemes and their influence on parsing results. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics : Student Research Workshop*, COLING ACL '06, p. 19–24, Stroudsburg, PA, USA : Association for Computational Linguistics.

MALOUF R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *proceedings of the 6th conference on Natural language learning - Volume 20*, COLING-02, p. 1–7, Stroudsburg, PA, USA : Association for Computational Linguistics.

MANNING C. D. (2011). Part-of-speech tagging from 97% to 100% : is it time for some linguistics ? In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I*, CICLing'11, p. 171–189, Berlin, Heidelberg : Springer-Verlag.

MARCUS M., SANTORINI H. & MARCINKIEWICZ M. (1994). Building a large annotated corpus of English : The Penn Treebank. In *Computational Linguistics*, p. 313–330.

MARTINEAU C., NAKAMURA T., VARGA L. & VOYATZI S. (2009). Annotation et normalisation des entités nommées. *Arena Romanistica*, **4**, 234–243.

MATSUZAKI T., MIYAO Y. & TSUJII J. (2005). Probabilistic CFG with latent annotations. In *Proceedings of ACL-05*, p. 75–82, Ann Arbor, USA.

MCCALLUM A. (2003). Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, UAI'03, p. 403–410, San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

MCCALLUM A., FREITAG D. & PEREIRA F. C. N. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, p. 591–598, San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

MCCARTHY D., KOELING R., WEEDS J. & CARROLL J. (2004). Finding Predominant Word Senses in Untagged Text. In *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, p. 280–287.

MCCLOSKY D. & CHARNIAK E. (2008). Self-Training for Biomedical Parsing. In *Proceedings of ACL-08 : HLT, Short Papers*, p. 101–104, Columbus, Ohio : Association for Computational Linguistics.

MCCLOSKY D., CHARNIAK E. & JOHNSON M. (2006a). Effective Self-Training for Parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, p. 152–159, New York City, USA : Association for Computational Linguistics.

MCCLOSKY D., CHARNIAK E. & JOHNSON M. (2006b). Reranking and Self-Training for Parser Adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*, p. 337–344, Sydney, Australia : Association for Computational Linguistics.

MCCLOSKY D., CHARNIAK E. & JOHNSON M. (2008). When is Self-training Effective for Parsing ? In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*, Manchester, UK.

MCCLOSKY D., CHARNIAK E. & JOHNSON M. (2010). Automatic domain adaptation for parsing. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, p. 28–36, Stroudsburg, PA, USA : Association for Computational Linguistics.

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

MCDONALD R., CRAMMER K. & PEREIRA F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, p. 91–98, Stroudsburg, PA, USA : Association for Computational Linguistics.

MCDONALD R., LERMAN K. & PEREIRA F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, p. 216–220, Stroudsburg, PA, USA : Association for Computational Linguistics.

MERTENS P. (2010). Restrictions de sélection et réalisations syntagmatiques dans DICOVALENCE. Conversion vers un format utilisable en TAL. In *TALN 2010*.

MESSIANT C. (2008). A subcategorization acquisition system for French verbs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies : Student Research Workshop*, HLT-SRWS '08, p. 55–60, Stroudsburg, PA, USA : Association for Computational Linguistics.

MESSIANT C., KORHONEN A. & POIBEAU T. (2008). LexSchem : A Large Subcategorization Lexicon for French Verbs. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Marrakech.

MILLER S., GUINNESS J. & ZAMANIAN A. (2004). Name tagging with word clusters and discriminative training. In *Proceedings of HLT*, p. 337–342.

NASR A., BÉCHET F. & REY J. F. (2010). MACAON : Une chaîne linguistique pour le traitement de graphes de mots. In *Traitement Automatique des Langues Naturelles - session de démonstrations*, Montréal.

NEY H. (1991). Dynamic programming parsing for context-free grammars in continuous speech recognition. In *IEEE Transactions on Signal Processing*.

NIANWEN XUE, FEI XIA F.-D. C. & PALMER M. (2005). The Penn Chinese Treebank : Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering, 11(2)*, p. 207–238.

NIGAM K. (1999). Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, p. 61–67.

NIVRE J. (2006). *Inductive Dependency Parsing (Text, Speech and Language Technology)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc.

NIVRE J. (2008). Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, **34**(4), 513–553.

NIVRE J., HALL J., KÜBLER S., MCDONALD R., NILSSON J., RIEDEL S. & YURET D. (2007). The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, p. 915–932, Prague, Czech Republic : Association for Computational Linguistics.

NIVRE J., HALL J. & NILSSON J. (2006). MaltParser : A data-driven parser-generator for dependency parsing. In *In Proc. of LREC-2006*, p. 2216–2219.

NIVRE J. & NILSSON J. (2004). Multiword units in syntactic parsing. In *Proceedings of Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*.

OHTA T., TATEISI Y. & TSUJII J. (2005). Syntax Annotation for the GENIA corpus. In *In the Proceedings of the IJCNLP 2005*, volume Companion, p. 222–227.

OSBORNE M. (2002a). Shallow parsing using noisy and non-stationary training material. *J. Mach. Learn. Res.*, **2**, 695–719.

OSBORNE M. (2002b). Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, AS '02, p. 1–8, Stroudsburg, PA, USA : Association for Computational Linguistics.

PAL C., SUTTON C. & MCCALLUM A. (2006). Sparse Forward-Backward using Minimum Divergence Beams for Fast Training of Conditional Random Fields. In *ICASSP*.

PAROUBEK P., POUILLOT L.-G. R. I. . V. A. (2005). EASy : campagne d'évaluation des analyseurs syntaxiques. In *Actes de l'atelier EASy de TALN'05*.

PAUMIER S. (2011). Unitex 2.1 - User Manual. http ://igm.univ-mlv.fr/˜unitex.

PAUMIER S., GROSS M. & LAPORTE E. (2003). De la reconnaissance de formes linguistiques à l'analyse syntaxique. *Université de Marne-la-Vallée*, p. 324.

PETROV S. (2010). Products of random latent variable grammars. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, p. 19–27, Stroudsburg, PA, USA : Association for Computational Linguistics.

PETROV S., BARRETT L., THIBAUX R. & KLEIN D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia.

PETROV S., CHANG P.-C., RINGGAARD M. & ALSHAWI H. (2010). Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, p. 705–713, Stroudsburg, PA, USA : Association for Computational Linguistics.

PETROV S. & KLEIN D. (2007). Improved inference for unlexicalized parsing. In *In HLT-NAACL '07*.

PIERCE D. & CARDIE C. (2001). Limitations of Co-Training for Natural Language Learning from Large Datasets. In *In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, p. 1–9.

PITON O., MAUREL D. & BELLEIL C. (1999). The Prolex Data Base : Toponyms and gentiles for NLP. In *Proceedings of the Third International Workshop on Applications of Natural Language to Data Bases (NLDB'99)*, p. 233–237.

POLLARD C. & SAG I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

PRESCHER D. (2005). Head-driven PCFGs with latent-head statistics. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, p. 115–124, Stroudsburg, PA, USA : Association for Computational Linguistics.

QUINLAN J. R. (1986). Induction of Decision Trees. *Mach. Learn.*, **1**(1), 81–106.

RABINER L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, p. 257–286.

RAMISCH C., VILLAVICENCIO A. & BOITET C. (2010). mwe-toolkit : a framework for multiword expression identification. In *Proceedings of LREC*.

RAMSHAW L. A. & MARCUS M. P. (1995). Text Chunking using transformation-based learning. In *Proceedings of the Third Annual Workshop on Very Large Corpora*, p. 82–94 : ACL.

RATNAPARKHI A. (1996). A Maximum Entropy Model for Part-Of-Speech Tagging. In E. BRILL & K. CHURCH, Eds., *Proceedings of the Empirical Methods in Natural Language Processing*, p. 133–142.

RATNAPARKHI A. (1997). A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. *CoRR*, **cmp-lg/9706014**.

RATNAPARKHI A. (1999). Learning to Parse Natural Language with Maximum Entropy Models. *Mach. Learn.*, **34**, 151–175.

RATNAPARKHI A., REYNAR J. & ROUKOS S. (1994). A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, HLT '94, p. 250–255, Stroudsburg, PA, USA : Association for Computational Linguistics.

REHBEIN I. & VAN GENABITH J. (2007). Evaluating Evaluation Measures. In *NODALIDA*, p. 372–379.

REICHART R. & RAPPOPORT A. (2007). Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. In *ACL 2007*.

RIEZLER S. & VASSERMAN A. (2004). Incremental Feature Selection and L1 Regularization for Relaxed Maximum-Entropy Modeling. In *EMNLP'04*.

ROARK B. (2001). Probabilistic top-down parsing and language modeling. *Comput. Linguist.*, **27**, 249–276.

ROSENBLATT F. (1958). The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, **65**(6), 386–408.

SAG I. A., BALDWIN T., BOND F., COPESTAKE A. A. & FLICKINGER D. (2002). Multiword Expressions : A Pain in the Neck for NLP. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing (CICLing '02)*, p. 1–15, London, UK : Springer-Verlag.

SAGAE K. (2010). Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. In *Proceedings of the ACL 2010 Workshop on Domain Adaptation for Natural Language Processing*, p. 37–44.

SAGAE K. & GORDON A. S. (2009). Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, p. 192–201, Stroudsburg, PA, USA : Association for Computational Linguistics.

SAGAE K. & LAVIE A. (2005). A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, p. 125–132, Stroudsburg, PA, USA : Association for Computational Linguistics.

SAGAE K. & LAVIE A. (2006). A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, p. 691–698, Stroudsburg, PA, USA : Association for Computational Linguistics.

SAGAE K. & TSUJII J. (2008). Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, p. 753–760, Stroudsburg, PA, USA : Association for Computational Linguistics.

SAGOT B. (2006). *Analyse automatique du français : lexiques, formalismes, analyseurs*. PhD thesis, Université Paris VII.

SAGOT B. (2010). The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French. In *7th international conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malte.

SAGOT B. & BOULLIER P. (2008). SxPipe 2 : architecture pour le traitement pré-syntaxique de corpus bruts. *Traitement Automatique des Langues*, **49**(2), 155–188.

SAGOT B. & FORT K. (2007). Améliorer un lexique syntaxique á l'aide des tables du lexique-grammaire : adverbes en -ment. In *Proceedings of the Lexis and Grammar Conference*, Bonifacio, France.

SAMPSON G. & BABARCZY A. (2003). A test of the leaf-ancestor metric for parsing accuracy. In *Natural Language Engineering, 9 (4)*, p. 365–380.

SAMUELSSON C. (1993). Morphological tagging based entirely on Bayesian inference. In *Proceedings of the 9th Nordic Conference on Computational Linguistics NODALIDA-93*.

SARKAR A. (2001). Applying co-training methods to statistical parsing. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, p. 1–8, Stroudsburg, PA, USA : Association for Computational Linguistics.

SCHAPIRE R. E. & SINGER Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Mach. Learn.*, **37**(3), 297–336.

SCHLUTER N. & GENABITH J. V. (2007). Preparing, restructuring, and augmenting a French treebank : Lexicalised parsers or coherent treebanks ? In *Proceedings of PACLING 07*.

SCHMID H. (1995). Probabilistic part-of-speech tagging using decision trees. In *ICNMLP1995*, Manchester, UK.

SEDDAH D., CANDITO M. & CRABBÉ B. (2009a). Cross parser evaluation and tagset variation : a French treebank study. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, p. 150–161, Stroudsburg, PA, USA : Association for Computational Linguistics.

SEDDAH D., CANDITO M. & CRABBÉ B. (2009b). Adaptation de parsers statistiques lexicalisés pour le français : Une évaluation complète sur corpus arborés. In *Actes de la 15ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN'09)*, Senlis, France.

SEDDAH D., CHRUPALA G., CETINOGLU O., VAN GENABITH J. & CANDITO M. (2010). Lemmatization and lexicalized statistical parsing of morphologically rich languages : the case of French. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, SPMRL '10, p. 85–93, Stroudsburg, PA, USA : Association for Computational Linguistics.

SEKINE S. (1997). The domain dependence of parsing. In *Proceedings of the fifth conference on Applied natural language processing*, ANLC '97, p. 96–102, Stroudsburg, PA, USA : Association for Computational Linguistics.

SERETAN V., NERIMA L. & WEHRLI E. (2003). Extraction of Multi-Word Collocations Using Syntactic Bigram Composition. In *Proceedings of the Fourth International Conference on Recent Advances in NLP (RANLP-2003)*, p. 424–431, Borovets, Bulgaria.

SHEN L., SATTA G. & JOSHI A. (2007). Guided learning for bidirectional sequence classification. In *ACL 2007*, p. 760–767, Czech Republic, Prague.

SIGOGNE A. (2010). HybridTagger : un étiqueteur hybride pour le Français. In *8ème MAnifestation des JEunes Chercheurs en Sciences et Technologies de l'Information et de la Communication (MajecSTIC'10)*, Bordeaux. Prix "Best paper" de la conférence electronic version (8 pp.).

SIGOGNE A., CONSTANT M. & LAPORTE E. (2011). Integration of Data from a Syntactic Lexicon into Generative and Discriminative Probabilistic Parsers. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, p. 363–370, Hissar, Bulgaria : RANLP 2011 Organising Committee.

SILBERZTEIN M. (2000). INTEX : an FST toolbox. *Theoretical Computer Science*, **231**(1), 33–46.

SKUT W., KRENN B., BRANTS T. & USZKOREIT H. (1997). An annotation scheme for free word order languages. In *Proceedings of the fifth conference on Applied natural language processing*, ANLC '97, p. 88–95, Stroudsburg, PA, USA : Association for Computational Linguistics.

SPOUSTOVÁ D. J., HAJIČ J., RAAB J. & SPOUSTA M. (2009). Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, p. 763–771, Stroudsburg, PA, USA : Association for Computational Linguistics.

STEEDMAN M., OSBORNE M., SARKAR A., CLARK S., HWA R., HOCKENMAIER J., RUHLEN P., BAKER S. & CRIM J. (2003). Bootstrapping statistical parsers from small datasets. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, p. 331–338, Stroudsburg, PA, USA : Association for Computational Linguistics.

STEINBACH M., KARYPIS G. & KUMAR V. (2000). A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*.

SURDEANU M., HARABAGIU S., WILLIAMS J. & AARSETH P. (2003). Using Predicate-Argument Structures for Information Extraction. In *IN PROCEEDINGS OF ACL 2003*, p. 8–15.

SUTTON C. & MCCALLUM A. (2007). An Introduction to Conditional Random Fields for Relational Learning. *In : GETOOR, Lise, TASKAR Benjamin (Editors.), MIT Press.*

SUZUKI J., ISOZAKI H., CARRERAS X. & COLLINS M. (2009). An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 2 - Volume 2*, EMNLP '09, p. 551–560, Stroudsburg, PA, USA : Association for Computational Linguistics.

TAN Y., YAO T., CHEN Q. & ZHU J. (2005). Applying Conditional Random Fields to Chinese Shallow Parsing. In *CICLing'05*, p. 167–176.

TELLIER I. & TOMMASI M. (2011). Champs Markoviens Conditionnels pour l'extraction d'information. In ERIC GAUSSIER & FRANÇOIS YVON, Eds., *Modèles probabilistes pour l'accès à l'information textuelle*. Hermès.

TELLJOHANN H., HINRICHS E., KÜBLER S., KÜBLER R. & TÜBINGEN U. (2004). The Tüba-D/Z Treebank : Annotating German with a Context-Free Backbone. In *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004*, p. 2229–2235.

THEDE S. & HARPER M. (1999). A second-order hidden Markov model for part-of-speech tagging. In *ACL 1999*, p. 175–182, College Park, Maryland.

THOMASSET F. & DE LA CLERGERIE E. (2005). Comment obtenir plus des Méta-Grammaires. In *Actes de la 12ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN'05)*, Dourdan, France.

TOLONE E. (2011a). *Analyse syntaxique à l'aide des tables du Lexique-Grammaire du français*. PhD thesis, LIGM, Université Paris-Est, France, Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est Marne-la-Vallée, France. (340 pp.).

TOLONE E. (2011b). *Analyse syntaxique à l'aide des tables du Lexique-Grammaire du français*. PhD thesis, Université Paris-Est Marne-la-Vallée.

TOLONE E. & SAGOT B. (2011). Using Lexicon-Grammar tables for French verbs in a large-coverage parser. In Z. VETULANI, Ed., *Human Language Technology. Forth Language and Technology Conference, LTC 2009, Poznań, Poland, November 2009. Revised Selected Papers*, Lecture Notes in Artificial Intelligence (LNAI). Springer Verlag.

TOUTANOVA K., KLEIN D., MANNING C. D. & SINGER Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, p. 173–180, Stroudsburg, PA, USA : Association for Computational Linguistics.

TOUTANOVA K. & MANNING C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora : held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, p. 63–70, Stroudsburg, PA, USA : Association for Computational Linguistics.

TSARFATY R. (2006). Integrated morphological and syntactic disambiguation for Modern Hebrew. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics : Student Research Workshop*, COLING ACL '06, p. 49–54, Stroudsburg, PA, USA : Association for Computational Linguistics.

TSURUOKA Y., TSUJII J. & ANANIADOU S. (2009). Fast full parsing by linear-chain conditional random fields. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, p. 790–798, Stroudsburg, PA, USA : Association for Computational Linguistics.

VAN DER PLAS L., HENDERSON J. & MERLO P. (2009). Domain adaptation with artificial data for semantic parsing of speech. In *Proceedings of Human Language Technologies : The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume : Short Papers*, NAACL-Short '09, p. 125–128, Stroudsburg, PA, USA : Association for Computational Linguistics.

VERSLEY Y. & REHBEIN I. (2009). Scalable discriminative parsing for German. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, p. 134–137, Stroudsburg, PA, USA : Association for Computational Linguistics.

VILLEMONTE DE LA CLERGERIE É., HAMON O., MOSTEFA D., AYACHE C., PAROUBEK P. & VILNAT A. (2008). PASSAGE : from French Parser Evaluation to Large Sized Treebank. In E. L. R. A. (ELRA), Ed., *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Maroc.

VITERBI A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, **13**(2), 260–269.

VOUTILAINEN A. (1995). A syntax-based part-of-speech analyser. In *EACL 1995*, p. 157–164, University College Dublin, Belfield.

WATRIN P. & FRANÇOIS T. (2011). N-gram frequency database reference to handle MWE extraction in NLP applications. In *Proceedings of the Workshop on Multiword Expressions : from Parsing and Generation to the Real World (MWE'11)*.

WEHRLI E., SERETAN V. & NERIMA L. (2010). Sentence analysis and collocation identification. In *Proceedings of the Workshop on Multiword Expression : From Theory to Applications (MWE'10)*.

WEISCHEDEL R., SCHWARTZ R., PALMUCCI J., METEER M. & RAMSHAW L. (1993). Coping with ambiguity and unknown words through probabilistic models. *Comput. Linguist.*, **19**(2), 361–382.

YAMADA H. & MATSUMOTO Y. (2003). Statistical Dependency Analysis with Support Vector Machines. In *In Proceedings of IWPT*, p. 195–206.

YAROWSKY D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *In proceedings of the 33rd annual meeting of the Association for Computational Linguistics*, p. 189–196.

ZHOU Y., WU L., WENG F. & SCHMIDT H. (2003). A fast algorithm for feature selection in conditional maximum entropy modeling. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, p. 153–159, Stroudsburg, PA, USA : Association for Computational Linguistics.

ZHU S., JI X., XU W. & GONG Y. (2005). Multi-labelled classification using maximum entropy method. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, p. 274–281, New York, NY, USA : ACM.

# Annexes

# 1 Hiérarchie des classes du `Lexique-Grammaire`

Pour les verbes, nous disposons d'une hiérarchie des classes du Lexique-Grammaire [LG] sur plusieurs niveaux (4 au total) créée manuellement, décrite pour la première fois dans (Sigogne *et al.*, 2011). Chaque niveau contient plusieurs classes qui regroupent des classes du LG qui ne partagent pas forcément toutes leurs propriétés mais dont les entrées ont un comportement syntaxique relativement similaire. Une description plus complète est effectuée dans la section 4.4. Les classes de verbes du LG étant au nombre de 76 et pour des raisons de clarté, la hiérarchie est montrée dans plusieurs tableaux regroupés en fonction du dernier niveau (le premier niveau étant les classes du LG).
Le tableau 34 montre la hiérarchie des verbes transitifs directs à trois arguments : `QTD3`, les arguments peuvent être sous la forme de complétives ; `TD3NLN`, absence d'arguments locatifs et aucun argument ne peut être sous la forme d'une complétive ; `TD3NL`, absence d'arguments locatifs.

| 9 | 10 | 11 | 13 | 16 | 2T | 39 | 36S | 36DT | 38PL | 32CV | 38R | 36R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QTD3 | | | | | TD3NLN | | | | | | | |
| QTD3 | | | | | TD3NLN | | | | | | | |
| TD3NL | | | | | | | | | | | | |

TAB. 34: Verbes transitifs directs à trois arguments.

Les tableaux de la figure 34 montrent la hiérarchie des verbes transitifs directs à trois ou quatre arguments : `TD4`, quatre arguments ; `TD3C`, trois arguments et l'argument locatif est direct ; `TD3L`, trois arguments et l'argument locatif est indirect ; `TD34L`, présence d'au moins un argument locatif.

| 38L | 38LH | 38RR | 37E | 37M1 | 37M2 | 37M3 | 37M4 | 37M5 | 37M6 |
|---|---|---|---|---|---|---|---|---|---|
| TD4 | | | TD3C | | | | | | |
| TD4 | | | TD3C | | | | | | |
| TD34L | | | | | | | | | |

(a)

| 3 | 36SL | 38LD | 38LS | 38LR | 38LHD | 38LHS | 38LHR |
|---|---|---|---|---|---|---|---|
| TD3L | | | | | | | |
| TD3L | | | | | | | |
| TD34L | | | | | | | |

(b)

FIG. 34: Verbes transitifs directs à trois ou quatre arguments.

Les tableaux de la figure 35 montrent la hiérarchie des verbes transitifs directs à deux arguments : `QTD2`, les arguments peuvent être sous la forme de complétives ; `TD2NL`, les arguments indirects ne sont pas introduits par des prépositions locatives ; `TD2`, deux arguments.

| 4 6 12 | 32RA 38L0 38L1 32R3 32A 32D |
|---|---|
| QTD2 | TD2NL |
| QTD2 | TD2NL |
| TD2 | |

(a)

| 32PL 32NM 32H 32CL 32R1 32R2 32C |
|---|
| TD2NL |
| TD2NL |
| TD2 |

(b)

FIG. 35: Verbes transitifs directs à deux arguments.

Les tableaux de la figure 36 montrent la hiérarchie des verbes intransitifs : `QTI3`, verbe transitif indirect à trois arguments dont au moins un peut être sous la forme d'une complétive ; `TI3N`, aucun argument ne peut être sous la forme d'une complétive ; `INT, INT1`, verbe intransitif ; `QTI2`, verbe transitif indirect à deux arguments dont au moins un peut être sous la forme d'une complétive ; `TI3`, verbe transitif indirect à trois arguments ; `TI2`, verbe transitif indirect à deux arguments ; `TI2NL`, absence d'argument locatif ; `TI2L`, présence d'un argument locatif introduit par une préposition locative.

| 14 15 18 | 35L 35RR | 31R 31H | 5 7 8 |
|---|---|---|---|
| QTI3 | TI3N | INT1 | QTI2 |
| TI3 | | INT1 | TI2 |
| INT | | | |

(a)

| 35S 35R 33 1 2 | 35LD 34L0 35LS 35ST 35LR |
|---|---|
| TI2NL | TI2L |
| TI2 | |
| INT | |

(b)

FIG. 36: Verbes intransitifs.

# 2 README de l'étiqueteur LGTagger

LGTagger is an open-source Part-of-speech tagger that also recognizes Multiword units. It is based on Conditional Random Fields (CRF) and large-coverage lexical resources. The lexical resources can be composed of morphosyntactic dictionaries (including simple and compound words) and strongly lexicalized local grammars.

LGTagger is written in Java and uses two different open-source toolkits :
* Wapiti version 1.3 (http://wapiti.limsi.fr/, (Lavergne *et al.*, 2010) : to learn and use the CRF model
* Unitex version 1.2 (http://www-igm.univ-mlv.fr/~unitex/) : to apply lexical resources on the texts

Note that the historical beta version used CRF++ (http://crfpp.sourceforge.net/).

LGTagger is self-content, i.e. the useful parts of Wapiti and Unitex are automatically installed during LGTagger installation. It has only been tested on linux.

LGTagger presently only works for French, but it has been designed to be easily adapted to other languages. LGTagger model is configurable by the users (cf. section "How to configure the training of your model") by specifying its lexical resources and its feature templates in a configuration file.

The current default model was trained with the training section of the FTB-UC version of the French Treebank, optimized for parsing (http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php). The lexical resources used are :
* the public version of the general language dictionary DELA (found in Unitex with tagset adapted to French Treebank)
* the Lefff (extracted from the POS tagger MElt (Denis & Sagot, 2009, 2010), https://gforge.inria.fr/frs/download.php/27240/melt-0.6.tar.gz)
* the Prolex dictionary containing toponyms (found in Unitex with tagset adapted to French Treebank)
* a local grammar of different MWU expressions (constructed for LGtagger purpose)

**Licenses**

LGTagger is available under LGPL license (http://www.gnu.org/licenses/lgpl.html). The used resources are available under the LGPL-LR licence (http://www-igm.univ-mlv.fr/~unitex/index.php?page=7).

**LGTagger directory**
* directory 'lib' contains JSAP-2.1.jar (http://sourceforge.net/projects/jsap, LGPL license) and jxl.jar (http://sourceforge.net/projects/jexcelapi/, LGPL license)
* directory 'unitex' containing Unitex sources and data useful to apply lexical resources
* directory 'wapiti' containing Wapiti sources useful to train and use a CRF model

* directory 'French' containing the models (directory 'French/models'), the configuration files (directory 'French/config') and the lexical resources (directory 'French/lex')
* the scripts :
  - 'lgtagger' for tagging
  - 'train-lgtagger' for training of a new model
  - 'resource-compiler' for compiling lexical resources in Unitex format
  - 'extract-properties' for extracting properties of tokens of a text

**How to install LGTagger**

1) unzip the archive
2) set an environment variable LGTAGGER_PATH indicating the path of the lgtagger directory
3) go into directory 'lgtagger-1.0'
4) launch the command make
5) to ease the call to lgtagger, either put the lgtagger directory in your PATH environment variable or place the files 'lgtagger' and 'train-lgtagger' in a directory belonging to your PATH

**Tagging with LGTagger**

#./lgtagger [-t] [-s] [-e] [-l language] [-m <model name>] <text>
Options :
-t : tagger takes a raw text as input, that will be tokenized and segmented in sentences
-s : tagger also segments in multiword units
-e : evaluation is performed, i.e. the input text should be a text already tagged (a word per line with format => word tag) ; Option -t has no effect
-l language : the language (default : French)
-m <model name> : the name of the crf model (default : 'default' or 'default-mwe' if option -s is activated) => LGTAGGER_PATH/<language>/models/<model name>.crf
<text> : the path of the text to be tagged

**Training a new model with LGTagger**

#./train-lgtagger [-s] [-n l1] [-a algo] [-d devCorpusPath] [-l language] [-c configname] <model name> <input corpus>
Options :
-s : tagger also segments in multiword units
-d devCorpusPath : the development corpus path for training with Wapiti
-l language : the language (default : French)
-n l1 : the l1 penalty when training with Wapiti, generally comprised between 0.2 and 0.8 (Default : 0.5)
-a algo : the algo that Wapiti uses for training : 'rprop' or 'l-bfgs' (default : rprop)
-c configname : the configuration filename LGTAGGER_PATH/<language>/config/<configname>.cfg (default : default)

<model name> : the name of the crf model generated in directory LGTAGGER_PATH/<language>/models
<input corpus> : the path of the reference corpus

**How to configure the training of your model**
The model training can be configured in a configuration file. The configuration files have to be located in the directory LGTAGGER_PATH/<language>/config/

1) Specifying the lexical resources

Put your lexical resources in the directory LGTAGGER_PATH/<language>/lex
Note that your lexical resources have to be compiled in the Unitex format (.bin/.inf for dictionaries and .fst2 for local grammars)
In the section RESOURCES of the configuration file, you just need to list the filename of the resources you want to use (cf. the default configuration file 'default.cfg' for French).

2) Specifying the feature templates

You can configure the unigram feature templates used by the CRF training. They have to be defined in the section TEMPLATES (cf. the default configuration file 'default.cfg' for French).

Each feature template is a combination of token properties located at different relative positions. For instance, w(0)/w(1) represents token bigrams composed of the token lexical value at the current position (0) and the token lexical value at the next position (1)

The following token properties are available :
– w : lexical form of the token (textual)
– lowercase : lowercased lexical form of the token (textual)
– prefix(<n>) : prefix of length n of the token (textual)
– suffix(<n>) : suffix of length n of the token (textual)
– hasHyphen : token contains an hyphen [character '-'] (binary)
– isMultiword : token contains character '_' (token separator in multiword units) (binary)
– hasDigit : token contains a digit (binary)
– allUppercase : token is all in uppercase (binary)
– isCapitalized : token is capitalized (binary)
– BOS : token is at the beginning of the sentence (binary)
– AC : concatenation of part-of-speech tags found in the lexical resources for the token by taking into account multi-token words (textual)
– ACZ : concatenation of part-of-speech tags found in the lexical resources for the token by NOT into account multi-token words (textual)
– BPOS : set of binary properties : for each part-of-speech tag, indicates whether it is a possible analysis for the token according to the lexical resources
– by taking into account multi-token words (binary)
– MWE(<type>) : category of type <type> of the multiword unit containing the token (ex. in our default configuration for MWU-aware tagging, the types are POS,
– STRUCT and SEM : the types are defined in the tagset specified in the file given in section TAGSET of the configuration file)

– MWE_IOB : IOB scheme (O : the token is outside a mutiword unit of the resources ; B : it is at the beginning ; I : is inside a multiword except at the first position)

**How to cite LGTagger**

Matthieu Constant and Anthony Sigogne. MWU-aware Part-of-Speech Tagging with a CRF model and lexical resources. ACL Workshop on Multiword Expressions : from Parsing and Generation to the Real World (MWE'11).

# 3 README du réordonnanceur discriminatif

This implementation of the reranker of Charniak and Johnson is an open-source tool that reranks n-best parses generated by a n-best parser. It is based on Maximum Entropy and is written in Python. It uses two different open-source toolkits (written in C++), that you have to install yourself :
* Petsc version 2.3 (`http://www.mcs.anl.gov/petsc/`
* Tao version 1.9 (`http://www.mcs.anl.gov/research/projects/tao/`)
The reranker is configurable by the users (cf. section "How to configure the training of your model") by specifying its feature templates in a configuration file, and perhaps some external data to use in templates.

The reranker package contains the following elements :
* train_reranker.py -> main module for training
* rerank.py -> main module for reranking
* DEFAULT_CONFIG_FILE -> default configuration file
* algorithm/ -> folder that contains training and reranking algorithms
* features/ -> folder that contains the description of feature templates
* estimator/ -> folder that contains the MaxEnt estimator
* utility/ -> folder that contains utility functions

**Thanks**
We thank Eugene Charniak and Mark Johnson for having made available their version of the reranker (written in C++). It can be downloaded at the following address : `http://www.cog.brown.edu/~mj/software.htm`.

**How to install the reranker**
1) unzip the archive
2) download and install PETSC and TAO libraries (see Charniak and Johnson's README)
3) open a terminal and go into directory 'src/wlle/'
4) launch the command make
5) go into directory '../src/'
6) launch the command "sh unittest.sh" to verify that the program works correctly. If all goes well, no error message is displayed

**Parsing with the reranker**
#./python rerank.py [-c config] [-m model] [-n trees] [-o trees]
Options :
-c <config> : the configuration file (default : DEFAULT_CONFIG_FILE)
-m <model> : the reranker model
-n <trees> : nbest trees to rerank, generated by a nbest parser
-o <trees> : the output file that contains reranked trees

**Training a new model with the reranker**
#./python train_reranker.py [-c config] [-m model] [-n trees] [-g trees]
Options :
-c <config> : the configuration file (default : DEFAULT_CONFIG_FILE)
-m <model> : the output filename of the reranker model
-n <trees> : nbest trees to rerank, generated by a nbest parser
-g <trees> : gold trees

**How to configure the reranker**
The model training and the reranking step can be configured in a configuration file.

1) Specifying the feature template list

In the section "features" of the configuration file, you just need to list the name of the templates you want to use. Templates are located in the filename indicated by the section "feature_file". To create new templates, see the generic template in "reranker/features/features.py".

2) Specifying the percolation table

You need a percolation table to determine syntactic heads of constituents (for an example, see the percolation table for French located in "reranker/percolation.py"). The section "percolation" indicates the location of this table.

3) Using external data for features

By default, no external data is used when extracting features. However, one might want to use external data, such as syntactic lexicons. To do this, use the function called "init_data" contained in the module "features/features_data.py".

This function contains a dictionary called "data" that works as follows :

"data[feature_schema_name] = your_data"

"feature_schema_name" is the name of the template and "your_data" is the external data to use. Then, the data dictionary is given as a parameter of the specified template.