UNIVERSITÉ DE NANTES
FACULTÉ DES SCIENCES ET DES TECHNIQUES

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET MATHÉMATIQUES

Année 2013

N° attribué par la bibliothèque

Acquisition de grammaire catégorielle de dépendances de grande envergure

---

THÈSE DE DOCTORAT
Discipline : Informatique
Spécialité : Informatique

Présentée
et soutenue publiquement par

Ramadan Alfared

le 18/12/2012 au LINA, Université de Nantes, devant le jury ci-dessous

Président	Isabelle Tellier, Professeur	Université Paris 3
Rapporteurs	Annie Foret, Maître de conférences HDR	Université de Rennes 1
	Christian Rétoré, Professeur	Université Bordeaux 1
Examinateurs	Denis Béchet, Maître de conférences	Université de Nantes
	Colin de La Higuera, Professeur	Université de Nantes
	Joseph Le Roux, Maître de conférences	Université Paris 13

Directeur de thèse : Alexandre Dikovsky
Co-encadrant de thèse : Denis Béchet

Acquisition de grammaire catégorielle de dépendances de grande envergure

Learning large-scale categorial dependency grammars

———

Ramadan Alfared

Ramadan.Alfared@etu.univ-nantes.fr
Équipe TALN
Laboratoire d'Informatique de Nantes Atlantique, UMR 6241

Remerciements

Vous vous apprêtez à lire un document retraçant plus de trois années de travaux de thèse. Une thèse, c'est à la fois une aventure scientifique et humaine. Dans les pages qui suivent, l'aspect scientifique de ce travail est présenté en détail. Si vous le permettez, j'aimerais prendre quelques lignes pour revenir sur son aspect humain.

Je remercie Annie Foret et Christian Rétoré, les rapporteurs de cette thése, ainsi que Colin de La Higuera, Isabelle Tellier, Denis Béchet et Joseph Le Roux d'avoir accepté de faire partie du jury.

Un grand merci également à mon directeur de thèse, Alexandre Dihovsky, et à mon encadrant, Denis Béchet, pour avoir cru en moi dès le début, puis pour la patience et le dévouement avec lesquels ils m'ont supporté (dans tous les sens du terme) tout au long de ces années.

D'un point de vue plus personnel, je veux remercier mes amis et ma famille qui ont été un grand soutien. Merci tout particulièrement à mes parents (mon père Alsayed et ma mère Aïsha) qui m'ont fait confiance dans mes choix, à mon frère Ali et à mes sœurs. Je remercie également mon cœur Samira et mes chers enfants Mohamed, Namareq, Rahma et Abdelmouhaimen.

En plus de certains collègues susmentionnés qui sont devenus des amis que j'ai eu plaisir à fréquenter hors du travail, Mohamed, Hafedh, Walid, Thomas, Prajol, Toufik, ..., et mes amis à Nancy Hassen, Matthieu, Thomas, ..., un grand merci aussi aux copains en Libye Salah, Hassain, Rajap, ..., qui se sont retrouvés un peu délaissés pendant ces années.

Table des matières

Table des matières **vi**

Liste des figures **vii**

1 Introduction **1**
- 1.1 Contexte . . . 1
  - 1.1.1 Présentation de la problématique . . . 2
  - 1.1.2 Résumé des contributions . . . 3
  - 1.1.3 Publications . . . 3
  - 1.1.4 Plan de la thèse . . . 4

I État de l'art **7**

2 L'analyse syntaxique et les grammaires de dépendances **9**
- 2.1 Introduction . . . 11
- 2.2 Les analyses en dépendances . . . 12
  - 2.2.1 Méthodes d'analyses en dépendances . . . 12
  - 2.2.2 Analyse en dépendances stochastiques pour le français . 23
- 2.3 Les formalismes de grammaires . . . 26
  - 2.3.1 Grammaires de liens . . . 26
  - 2.3.2 Les grammaires catégorielles (GC) . . . 28
  - 2.3.3 Grammaires algébriques . . . 30
  - 2.3.4 Les grammaires d'arbres adjoints (TAG) . . . 30
- 2.4 Conclusion . . . 37

3 Grammaires catégorielles de dépendances **39**
- 3.1 Introduction . . . 41
- 3.2 Les grammaires catégorielles de dépendances . . . 41
  - 3.2.1 Type de CDG . . . 41
  - 3.2.2 Calcul de types des dépendances . . . 43
  - 3.2.3 Extension des CDG . . . 44
  - 3.2.4 Expressivité des CDG . . . 47
- 3.3 L'analyseur des CDG . . . 48
- 3.4 Conclusion . . . 51

II Ressources lexicales et unités lexicales **53**

4 Le lexique morpho-syntaxique Le*fff* et la CDG du français **55**

4.1 Introduction . . . 57
4.2 Les ressources linguistiques et la CDG du français . . 57
4.2.1 Présentation de Le*fff* et de son architecture . . . 58
4.2.2 Construction du lexique de la base de données de la CDG du français . . . 59
4.2.3 Travaux après l'incorporation de Le*fff* . . . 63
4.3 Conclusion et perspectives . . . 67

5 Acquisition automatique des arguments de noms déverbaux **69**
5.1 Introduction . . . 71
5.2 La structure argumentale des déverbaux . . . 71
5.2.1 Recherche des déverbaux . . . 73
5.2.2 Cadres de sous-catégorisation des déverbaux . . . 78
5.2.3 Expériences et évaluation des résultats . . . 81
5.2.4 Construction du corpus de déverbaux . . . 84
5.3 Complétion de la liste des suffixes . . . 85
5.3.1 Conclusion et perspectives . . . 89

III Amélioration de l'analyse en dépendances **91**

6 Une approche pour améliorer l'analyse en dépendances **93**
6.1 Introduction . . . 95
6.2 Modèles d'analyses en dépendances . . . 95
6.2.1 Correspondances entre des étiquettes des étiqueteurs morpho-syntaxiques et les parties du discours de Le*fff* . . 97
6.2.2 Expériences et évaluation des résultats . . . 99
6.3 Discussions . . . 105
6.4 Conclusion . . . 107

7 Conclusion et perspectives **109**
7.1 Rappel des enjeux . . . 109
7.1.1 Contributions . . . 109
7.1.2 Travaux futurs . . . 110

A Annexes **111**
A.1 La liste des règles . . . 115

Bibliographie **119**

Liste des figures

2.1 Structure de constituant pour une phrase en anglais de Penn Treebank . . . 15

2.2 Structure de dépendance pour une phrase en anglais du Penn Treebank . . . 15
2.3 Structure de dépendance pour la phrase : *le garçon mange une pomme.* . . . 17
2.4 Application de l'algorithme de Chu-Liu-Edmonds pour la phrase : "John saw Mary." . . . 21
2.5 un exemple d'arbre de constituant du FTB-UC . . . 25
2.6 La correspondance de l'arbre de dependance . . . 25
2.7 Exemple d'entrées du dictionnaire des grammaires de liens 27
2.8 Chemin de liens obtenus pour la phrase : "A cat runs" . . . 27
2.9 Résultats de l'analyse de la phrase : *the dog chasses the cat* . . 28
2.10 Arbre de dérivation de la phrase : Jean qui aime Marie, déteste Paul . . . 29
2.11 Arbre de dérivation sans le nom des règles et FA-structure . 29
2.12 Les arbres élémentaires pour la phrase : Pierre aime beaucoup Marie . . . 31
2.13 Certains exemples sur les arbres d'auxiliaires . . . 32
2.14 L'opération de substitution sur la phrase : Jean marche . . . 32
2.15 L'opération d'adjonction dans la phrase : Jean marche vite . 33
2.16 Structures de traits et adjonctions d'arbres . . . 33
2.17 Structures de traits et substitution d'arbres . . . 34
2.18 Grammaire TAG engendrant le langage $a^n b^n e c^n d^n$ . . . 35
2.19 L'arbre dérivé et l'arbre de dérivation : Pierre aime beaucoup Marie . . . 36
2.20 *La localisation du nœud* . . . 36

3.1 Au commencement était le Verbe. . . . 45
3.2 Structure de dépendances non-projectives . . . 46
3.3 *Dépendances circonstancielles itérées* . . . 47
3.4 *La structure de dépendances pour "aaabbbccc"* . . . 47
3.5 *Preuve de correction de la structure de dépendances pour "aaabbbccc"* . . . 48
3.6 *Le menu principal du CDG lab* . . . 48
3.7 *Sélection d'une grammaire* . . . 49
3.8 *Sélection une ou plusieurs phrases.* . . . 50
3.9 *Analyse de la phrase : elle la lui a donnée.* . . . 50
3.10 *Statistiques globales* . . . 51
3.11 *Graphique virticle pour la phrase : au commencement était le Verbe.* 51

4.1 Le processus de création de l'état initial de la base de données 61
4.2 Modèle conceptuel de la base de données DB_LEFFF . . . 63
4.3 Connexion entre les tables . . . 64

5.1 Analyses des phrases de l'introduction . . . 72
5.2 Corpus des noms déverbaux . . . 85

6.1 Forme générale du modèle de base. . . . 96
6.2 Forme générale des modèles utilisant l'étiquetage des étiqueteurs morpho-syntaxiques . . . 96
6.3 Analyse normale de la phrase : *il parle en phrases courtes.* . . 103
6.4 *SD correcte pour la phrase : il la leur fait apprendre.* . . . 104

6.5 SD incorrecte pour la phrase : il la leur fait apprendre. . . . 104
6.6 *Phrase du corpus paris 7 : le deuxième problème est la nourriture.* 105
6.7 *structure de dépendances : Ève, vas-t'en !* 106
6.8 *Définir des oracles pour pré-sélection classe et type* 108

Introduction

1

L'objectif de ce chapitre introductif est de poser les bases nécessaires à la compréhension du travail de recherche réalisé dans le cadre de cette thèse. En particulier il permet de poser le contexte scientifique dans lequel se situent aujourd'hui les études sur "les *ressources lexicales et les grammaires de dépendances*". Puis, nous définissons la problématique de notre sujet de thèse. Enfin, après avoir résumé les principales contributions, nous présentons le plan du mémoire.

1.1 Contexte

Les ressources lexicales sont nécessaires pour obtenir des systèmes de traitement des langues performants. Ces ressources peuvent être soit construites à la main, soit acquises de manière automatique à partir de gros corpus. La disponibilité de telles ressources et leur utilisabilité dans les applications de TAL restent pourtant relatives.

Le présent travail est une étude qui s'inscrit dans le cadre de la complétion du lexique d'une grammaire catégorielle de dépendances du français et s'inscrit aussi dans le cadre de l'analyse mixte stochastique-déterministe des grammaires de dépendances de grande envergure. Les grammaires catégorielles et les grammaires de dépendances sont conçues pour modéliser les langues naturelles, avec en général la volonté de capturer avec précision les phénomènes linguistiques qui les caractérisent. Si l'on considère une définition très générale des grammaires de dépendances, incluant tout formalisme utilisant une relation de dépendances dans la représentation syntaxique, on peut y inclure les grammaires catégorielles (Adjukiewicz (1935)). Néanmoins on peut aussi restreindre la définition autour de la notion de dépendance, proposée par Tesnière (1959) et développée dans la théorie Sens-Texte élaborée principalement par Mel'čuk (1997), du fait de leur intérêt au niveau de l'analyse syntaxique et de l'extraction automatique de grammaires.

Dans le domaine du TAL, l'interprétation automatique des textes nécessite des analyseurs syntaxiques et des grammaires dont les dérivations facilitent l'extraction automatique des structures sémantiques. Dans ce cadre, les structures de dépendances sont très proches d'une forme logique des textes car elles représentent sous une forme explicite les relations entre les têtes nominales et verbales et leurs adjoints. La formalisation des grammaires de dépendances et leur utilisation dans des analyseurs syntaxiques qui soient à la fois robustes, rapides et à large couverture sont des objectifs importants dans le cadre de l'accès à l'information dans les textes écrits. Nous allons nous concentrer sur la notion de grammaires catégorielles de dépendances (Categorial Dependency Grammars CDG) et en particulier sur leur formalisation (Dikovsky (2001) ; Dikovsky (2004) ; Bechet et al.

(2005) ; Dekhtyar et Dikovsky (2008) ; Dikovsky (2009)). Cette classe de grammaires complètement lexicalisées distingue les dépendances locales dont la projection forme des constituants continus et les dépendances non-projectives plus spécifiques dont le principe simple de construction repose sur un mécanisme d'ancrage de valences polarisées et leur appariement par paire.

1.1.1 Présentation de la problématique

Dans le domaine des grammaires catégorielles, l'équipe TALN[1] du LINA dispose d'un analyseur[2] efficace pour les CDG (Dekhtyar et Dikovsky (2004)) qui continue d'être amélioré, en particulier, en ce qui concerne la robustesse et la précision. L'obtention d'une grammaire pour ce type d'analyseur s'effectue à partir de corpus de grande envergure et non plus à la main. Au cours des vingt dernières années, de nombreuses méthodes d'acquisition de grammaires à partir de corpus ont été développées (Collins (1999), Klein et Manning (2003)). En majorité, ces méthodes sont stochastiques et consistent à entraîner des analyseurs sur des corpus arborés à base de constituants. Ces méthodes utilisent des approches probabilistes qui reposent sur un modèle acquis à partir d'un corpus annoté manuellement. D'autres utilisent des approches symboliques qui utilisent une grammaire et/ou un lexique développés manuellement comme les CDG (Dikovsky (2009)).

Les grammaires catégorielles de dépendances traitent particulièrement les dépendances discontinues qui posent un défi pour la plupart des grammaires de dépendances. Par exemple les cas les plus courants du français comme les particules négatives, les pronoms comparatifs ou le déplacement des subordonnées de leur position canonique (e.g. conséquence de topicalisation, de clivage, etc.).

Le problème que nous allons aborder est celui de l'absence de solution proposée par l'analyseur pour certaines phrases. Dans les CDG, il y a de fortes chances que l'analyse échoue pour diverses raisons, par exemple :

- l'inexistence de la structure de dépendance, c'est-à-dire qu'aucune structure de dépendance ne représente la phrase. Dans ce cas, il y a quelque chose qui manque au niveau du lexique. Pour résoudre ce problème, nous devons trouver une méthode pour construire un grand lexique pour la CDG du français. Cette question est abordée dans le chapitre 4.
- aucune structure de dépendance linguistiquement correcte. Dans ce type de ce problème, il y a au moins une structure de dépendance pour une phrase, mais nous ne savons pas si elle est correcte linguistiquement ou pas. Dans ce cas, nous devons disposer d'un corpus arboré en dépendances annoté avec une grande précision dans divers domaines. Ce corpus d'entraînement doit être bien annoté pour chacun des domaines cibles. La qualité du corpus d'entraînement

[1] Traitement Automatique du Langage Naturel, LINA (Laboratoire d'Informatique de Nantes Atlantique) http ://www.lina.univ-nantes.fr/).

[2] L'analyseur *CDG* est une implémentation d'un analyseur syntaxique pour les grammaires catégorielles de dépendances qui est écrit en Common Lisp, PHP et bash. Développé dans le LINA.

est très importante et la validation linguistique est nécessaire. (cf. Alfared et al. (2011)).

Le second problème général de l'analyse syntaxique des grammaires à large couverture (en dépendances ou en constituants) est l'explosion combinatoire des analyses fallacieuses. L'analyseur des CDG donne actuellement toutes les solutions compatibles avec une CDG. Ce sujet est abordé dans le chapitre 6.

1.1.2 Résumé des contributions

L'objectif de cette thèse est de compléter le lexique de la CDG du français et aussi d'élaborer une méthode d'analyse mixte stochastique-symbolique (avec des dépendances discontinues). Dans ce cas, il faut d'abord développer un algorithme d'acquisition de classes en fonction des valeurs des traits des mots disponibles dans le corpus et conforme avec l'affectation de types acquise précédemment. Puis il faut construire le lexique typé des classes. De manière plus détaillée, nous avons une CDG à large couverture du français qui évolue au niveau des dépendances et au niveau du lexique. Dans la première partie de cette contribution, nous devons trouver des méthodes pour compléter automatiquement le lexique de la CDG du français ; par exemple, compléter le lexique en utilisant des cadres syntaxiques des noms déverbaux. La deuxième contribution consiste à développer un algorithme pour améliorer l'analyse en dépendances en utilisant des étiqueteurs morpho-syntaxiques qui peuvent réduire le taux d'ambiguïtés fallacieuses d'une grammaire catégorielle de dépendances. Il faut aussi comparer les résultats obtenus avec les autres méthodes.

1.1.3 Publications

Ce travail a débouché, pour l'instant, sur les publications suivantes :

- **Conférences internationales avec comité de lecture ;**
  - Ramadan Alfared, Denis Béchet and Alexander Dikovsky.
    CDG Lab : a toolbox for dependency grammars and dependency treebanks development. In Kim Gerdes, Eva Hajicova, and Leo Wanner, editors, Proceedings of the 1st International Conference on Dependency Linguistics (Depling 2011), Barcelona, Spain, September 5-7 2011, 2011. ISBN 978-84-615-1834-0, short paper.
    URL http ://depling.org/proceedingsDepling2011/.
  - Ramadan Alfared and Denis Béchet.
    On the adequacy of three POS taggers and a dependency parser.
    In Alexander Gelbukh, editor, Computational Linguistics and Intelligent Text Processing, 13th International Conference, CICLing 2012, New Delhi, India, March 11-17, 2012, Proceedings, Part I, volume 7181 of Lecture Notes in Computer Science (LNCS), pages 104-116. Springer-Verlag, 2012.
    ISBN 978-3-642-28603-2, URL http ://www.cicling.org/2012/.
- **Conférences francophones avec comité de lecture ;**
  - Ramadan Alfared, Denis Bechet et Alexander Dikovsky.
    Calcul des cadres de sous catégorisation des noms déverbaux fran-

çais (le cas du génitif). (on computing subcategorization frames of french deverbal nouns (case of genitive)) [in french]. Dans Actes de la conférence conjointe JEP-TALN-RECITAL 2012, volume 2 : TALN, pages 71-84, Grenoble, France, June 2012. ATALA/AFCP. URL http ://www.aclweb.org/anthology/F/F12/F12-2006.

1.1.4 Plan de la thèse

Afin de clarifier nos contributions et les travaux liés, notre manuscrit se divise en six chapitres principaux qui sont regroupés en trois parties.

Première partie : elle se consacre à l'état de l'art et aux données du problème. Elle comporte une description rapide de quelques-unes des méthodes d'analyses syntaxiques.

Le chapitre 2 (l'analyse syntaxique et les grammaires de dépendances) évoque l'analyse syntaxique, ainsi que les grammaires de dépendances. Nous décrivons quelques travaux qui utilisent des approches probabilistes pour faire avancer la recherche dans le domaine des grammaires de dépendances en résumant les méthodes d'analyses en dépendances.

Dans le chapitre 3 (grammaires catégorielles de dépendances), nous présentons un formalisme grammatical appelé grammaires catégorielles de dépendances. Nous montrons les principes généraux sous-jacents aux grammaires catégorielles de dépendances ainsi que les résultats d'analyses syntaxiques de ces grammaires. Nous montrons que ces grammaires traitent les dépendances discontinues (les particules négatives, les pronoms comparatifs, etc.) qui posent un défi pour la plupart des grammaires de dépendances.

Deuxième partie : elle s'attache à l'étude des ressources lexicales. Cette partie se compose de deux chapitres.

Dans le chapitre 4 (le lexique morpho-syntaxique Le*fff* et la CDG du français), nous présentons notre travail sur l'élaboration d'une ressource lexicale pour le français. Notre but est de compléter le lexique de la CDG du français à partir de cette ressource lexicale. En fait, le but ultime de ce travail est la construction d'un corpus en dépendance bien annoté.

Dans le chapitre 5 (acquisition automatique des arguments de noms déverbaux), nous présentons un algorithme de repérage des noms déverbaux ainsi que son évaluation. Nous discuterons sur des liens entre les cadres de sous-catégorisation des noms déverbaux et des verbes d'origine. Puis nous présentons nos expériences sur les règles de transformation des cadres (du verbe au déverbal). L'intérêt de ce travail est de compléter le lexique de la CDG du français. On termine par une conclusion et les perspectives de ce travail.

Troisième partie : elle traite le problème de l'analyse syntaxique. Elle se compose de deux chapitres.

Le chapitre 6 (une approche pour améliorer l'analyse en dépendances) montre qu'en utilisant un étiqueteur morpho-syntaxique pour choisir les classes grammaticales les plus probables des unités lexicales, nous pouvons sensiblement réduire le taux d'ambiguïtés fallacieuses d'une gram-

maire catégorielle de dépendances du français qui utilise Le*fff* comme base lexicale. Aussi, nous pouvons réduire l'espace de recherche de l'analyseur des CDG. On termine par une discussion et les perspectives de ce travail.

Le chapitre 7 (conclusion et perspectives) dresse un bilan du travail effectué sur la CDG du français, le corpus des noms déverbaux du français et l'analyse syntaxiques. Après avoir rappelé nos contributions principales, nous proposons quelques directions de recherche futures sur les points les plus intéressants ainsi que les perspectives qui pourraient étendre notre projet.

Première partie

État de l'art

L'analyse syntaxique et les grammaires de dépendances

2

Sommaire

2.1 Introduction 11
2.2 Les analyses en dépendances 12
2.2.1 Méthodes d'analyses en dépendances 12
2.2.2 Analyse en dépendances stochastiques pour le français . . 23
2.3 Les formalismes de grammaires 26
2.3.1 Grammaires de liens 26
2.3.2 Les grammaires catégorielles (GC) 28
2.3.3 Grammaires algébriques 30
2.3.4 Les grammaires d'arbres adjoints (TAG) 30
2.4 Conclusion 37

Dans ce chapitre nous présentons quelques méthodes existantes dans le domaine de l'analyse syntaxique qui est un domaine de la linguistique assez riche. Nous nous intéressons à deux techniques largement employées dans le cadre de l'analyse syntaxique. La première est *dirigée par les données*. Elle a récemment émergée car, non seulement c'est une des plus efficaces, mais aussi c'est l'une des méthodes les plus précises pour l'analyse en dépendances. La deuxième technique est *basée sur une grammaire*. Nous présentons certains formalismes grammaticaux. En fait, nous esquissons quelques-uns des principaux formalismes grammaticaux adaptés aux langues naturelles.

2.1 Introduction

L'analyse syntaxique de textes à l'aide de lexiques syntaxiques est au centre de projets de recherche récents sur le français et d'autres langues. Elle constitue un point clé dans un grand nombre de traitements automatiques, tels que la compréhension de textes, l'extraction des relations de dépendances entre des tokens[1], l'extraction d'information ou la traduction. Le but d'un analyseur syntaxique est de créer une ou plusieurs représentations de chacun des tokens qu'il reçoit en entrée. Autrement dit, sa tâche est de déterminer pour chaque token de la phrase sa fonction syntaxique, ainsi que les relations de dépendance syntaxique des éléments de la phrase, telles que *sujet-verbe* ou *verbe-objet*, l'objectif final étant de retirer toute forme d'ambiguïté à la phrase afin de ne générer qu'une seule représentation. Pour atteindre cet objectif, il faut notamment posséder un lexique où les formes lexicales sont les plus spécifiées possible afin d'augmenter les contraintes d'utilisation et ainsi diminuer l'ambiguïté.
Ces dernières années, de nombreux progrès ont été réalisés depuis l'analyseur syntaxique développé par l'équipe de *Z. Harris* dans les années 50 (Joshi et Hopely (1996)). De manière plus détaillée, nous pouvons diviser les différentes approches en deux catégories :

- les analyseurs symboliques qui utilisent une grammaire et/ou un lexique développés manuellement ;
- les analyseurs probabilistes qui reposent sur un modèle acquis à partir d'un corpus annoté manuellement.

L'analyse syntaxique probabiliste profonde obtient ses meilleurs résultats avec le formalisme des grammaires hors-contextes probabilistes (PCFG, Probabilistic Context-Free Grammar). Différentes stratégies d'apprentissage sont utilisées. Ces stratégies sont soit lexicalisées (par exemple, Collins (2003)), soit non lexicalisées (par exemple, Klein et Manning (2003)).
L'approche symbolique, bien que laborieuse puisque les ressources sont développées manuellement, permet de construire une base très riche d'informations linguistiques. Il s'agit notamment de décrire les caractéristiques syntaxiques des tokens, même si représenter toutes ces données est difficile. Nous présenterons dans le chapitre 4 un formalisme grammatical à large couverture de données lexicale (Dikovsky (2009)).

Définition 2.1 Un analyseur en dépendances reçoit une phrase en entrée $s = w_1, w_2,..., w_n$ et calcule le graphe de dépendances $G = (W, A)$. L'ensemble des nœuds $W = w_0, w_1,..., w_n$ correspondent aux tokens d'une phrase, et le nœud w_0 est la racine[2] de G. A est un ensemble d'arcs (w_i, w_j), dont chacun représente une relation de dépendance où w_i est la tête et w_j est le dépendant. On suppose que le résultat d'un graphe de dépendance pour une phrase est bien formé (Nivre (2008)). G est bien formé si et seulement s'il satisfait les quatre conditions suivantes : une seule tête pour une phrase, acyclique, connexe et enraciné.

[1] Les tokens représentent le découpage de la phrase en ponctuation, suite de lettres ou de chiffres.

[2] Cette idée ne se retrouve pas dans tous les analyseurs.

Dans les sections suivantes, nous nous intéressons aux travaux reliés à notre domaine de recherche. Nous allons débuter par mentionner quelques travaux qui utilisent des approches probabilistes et symboliques pour faire avancer la recherche dans le domaine des grammaires de dépendances.

2.2 Les analyses en dépendances

Les représentations en dépendance sont devenues de plus en plus populaires dans l'analyse syntaxique, en particulier pour les langues avec un ordre des mots flexible, comme le tchèque (Collins et al. (1999)), le bulgare (Marinov et Nivre (2005)), le turc (Eryiğit et Oflazer (2006)) et le russe (Boguslavsky et al. (2011)). De nombreuses implémentations pratiques de l'analyse en dépendances sont limitées aux structures projectives. Bien que cette contrainte garantit une bonne complexité, il est bien connu que certaines constructions syntaxiques ne peuvent être représentées de manière adéquate que par des structures de dépendances non-projectives, où la projection de la tête peut être discontinue. Ceci est particulièrement pertinent pour les langues avec ordre des mots libres ou flexibles.

Définition 2.2 *Notion de phrase* (Tesnière (1959)) : une phrase s est un ensemble organisé dont les éléments constituants sont les mots $w_1 w_2 ... w_n$. Tout mot qui fait partie d'une phrase cesse par lui-même d'être isolé comme dans le dictionnaire. Entre lui et ses voisins, l'esprit aperçoit des connexions, dont l'ensemble forme la charpente de la phrase. [...] Les connexions structurales établissent entre les mots des rapports de `dépendance`.

Mel'čuk donne la définition suivante de la dépendance linguistique (Mel'čuk (2003)) :

Définition 2.3 *La dépendance* est une relation non symétrique, de même type que l'implication : l'un des éléments présuppose en quelque sorte l'autre, mais l'inverse n'est (en général) pas vrai. La dépendance est notée par une flèche : $w_1 \longrightarrow w_2$ signifie que w_2 dépend de w_1, w_1 est appelé le `gouverneur` de w_2, on parle aussi de la `tête` de la dépendance.

Mel'cuk considère différents niveaux d'analyse avec la représentation de multiples composantes de la linguistique : la phonologie, la phonétique, la morphologie, la syntaxe et la sémantique qui équivalent aux différents niveaux de modélisation d'un énoncé.

2.2.1 Méthodes d'analyses en dépendances

Nous allons présenter certaines méthodes d'analyses en dépendances. Nous passons aussi au problème de l'analyse en dépendances qui consiste à trouver automatiquement la structure de dépendance d'une phrase donnée et de former une analyse ou les analyses les plus probables pour chaque phrase. Nous allons montrer un certain nombre de méthodes différentes pour résoudre ce problème (Nivre et al. (2009), (Dikovsky (2004), *chapitre 3*)), certaines sont basées sur l'apprentissage automatique inductif de grands ensembles de phrases qui sont annotées syntaxiquement,

d'autres basées sur des grammaires formelles qui définissent des structures de dépendance admissibles. D'une manière générale, ces approches peuvent être divisées en deux classes, que nous appelons *dirigée par les données* et *basée sur une grammaire*. Une approche est dirigée par les données si elle fait usage essentiellement de l'apprentissage automatique à partir des données linguistiques en vue d'analyser de nouvelles phrases. Une approche est basée sur la grammaire si elle s'appuie sur une grammaire formelle, la définition d'un langage formel[3], de sorte qu'il est logique de se demander si une phrase donnée en entrée dans la langue est définie par la grammaire ou pas.

La majeure partie des travaux sur l'analyse en dépendances sont consacrés aux méthodes *dirigées par les données* qui ont attiré le plus l'attention ces dernières années (Nivre et al. (2009)). Dans l'analyse en dépendances il y a deux problèmes :
– le premier est le problème de l'apprentissage qui est la tâche de l'apprentissage d'un modèle d'analyse à partir d'un échantillon représentatif de phrases et de leur structure de dépendance.
– le deuxième est le problème d'analyse qui est la tâche d'appliquer le modèle d'apprentissage à l'analyse d'une nouvelle phrase. Nous pouvons les représenter comme suit.
– **Acquisition** : étant donné un ensemble d'exemples D (annotées avec des graphes de dépendances), il faut induire un modèle d'analyse M qui peut être utilisé pour analyser de nouvelles phrases.
– **Analyse** : étant donné un modèle d'analyse M et une phrase S, on dérive le graphe de dépendances optimal G pour S correspondant à M. Plus précisément, le problème de l'apprentissage est d'induire un modèle de prédiction d'une transition vers l'état suivant (Section 2.2.1.2), étant donnée l'histoire des transitions, et le problème de l'analyse est de construire la séquence de transitions optimale pour une phrase en entrée. Nous pouvons résumer les méthodes d'analyses en dépendances comme suit :

1. l'analyse en dépendances dirigée par les données ;
   – analyse syntaxique basée sur les transitions ;
   – analyse syntaxique basée sur les graphes ;
2. analyse syntaxique basée sur les grammaires ;
   – l'analyse en dépendances hors-contexte ;
   – l'analyse en dépendances basée sur les contraintes.

2.2.1.1 *Types d'analyses syntaxiques et les algorithmes d'analyses*

Nous étudions les deux catégories d'analyses syntaxiques : l'analyse en dépendances dirigée par les données et l'analyse syntaxique basée sur les grammaires. Mais d'abord, nous donnons quelques définitions importantes pour la suite.

Définition 2.4 Supposons que $R = \{r_1, \cdots, r_m\}$ est un ensemble fini de noms de dépendances possibles (relation de dépendance) qui peuvent relier deux tokens

[3]Un langage formel est un ensemble de suites d'objets élémentaires qui, selon le langage défini et le niveau auquel on se place, peuvent être des lettres, des mots, des symboles, des caractères, etc. Tous ces ensembles d'objets élémentaires ont en commun le fait d'être finis, et sont appelés vocabulaire (ou alphabet). Un langage formel est un ensemble de suites finies (de chaînes) d'éléments de son vocabulaire.

dans une phrase. Un nom de dépendance $r \in R$ est aussi appelé un arc étiqueté. Par exemple une relation entre un verbe *"tête"* et son sujet peut être noté avec $r = SBJ$.

Définition 2.5 Un graphe de dépendances $G = (V, A)$ est un graphe étiqueté dans le sens standard de la théorie des graphes et consiste en un ensemble de nœuds V et en un ensemble d'arcs A, tel que pour une phrase $S = w_1 w_2 ... w_n$ et l'ensemble des noms de dépendances R :

1. $V \subseteq \{w_1 w_2 ... w_n\}$
2. $A \subseteq V \times R \times V$
3. $si\ (w_i, r, w_j) \in A$ alors $\left(w_i, r', w_j\right) \notin A$ pour tous $r' \neq r$

L'ensemble des arcs A représente les relations de dépendances étiquetées de l'analyse particulière de G. Plus précisément, un arc $(w_i, r, w_j) \in A$ représente une relation de dépendance de tête w_i vers w_j étiquetée avec le type de relation r. Un graphe de dépendances G est donc un ensemble de relations de dépendances entre les tokens dans S.

Nous présentons le graphe de dépendance qui se trouve dans la figure 2.2 par le graphe suivant :

- G=(V, A)
- V={Economic, news, had, little, effect, on, financial, markets, .}
- A={(news, NMOD, Economic), (had, SBJ, news), (had, P, .), (had, OBJ, effect), (effect, NMOD, little), (effect, NMOD, on), (on, PMOD, markets), (markets, NMOD, financial)}

La figure 2.1 présente la structure en constituants de la phrase "*Economic news had little effect on financial markets*" dans le corpus Penn Treebank de Marcus et al. (1993) et la figure 2.2 décrit la structure de dépendance de cette phrase.
La méthode basée sur les transitions porte essentiellement sur les points de la définition suivante :

Définition 2.6 Étant donné un ensemble de noms de dépendances R, une configuration pour une phrase $S = w_0 w_1 ... w_n$ est un triplet $c = (\sigma, \beta, A)$ où :

1. σ est une pile de tokens $w_i \in V$,
2. β est un tampon de tokens $w_i \in V$,
3. A est un ensemble d'arcs $(w_i, r, w_j) \in V \times R \times V$.

Les méthodes basées sur les graphes définissent un espace de graphes de dépendances candidats à une phrase. Le problème de l'apprentissage consiste à assigner des scores pour les graphes de dépendances candidats à une phrase, et le problème de l'analyse consiste à trouver le graphe de dépendance qui a le meilleur score pour une phrase en entrée. Ceci correspond parfois à rechercher *un arbre de recouvrement maximal*.

Définition 2.7 Un modèle d'analyse en dépendances consiste en un ensemble de contraintes Γ qui définit l'espace des structures de dépendances permises pour une phrase donnée, un ensemble de paramètres λ (possiblement vide) et un algorithme d'analyse fixé h. Un modèle est représenté par $M = (\Gamma, \lambda, h)$.

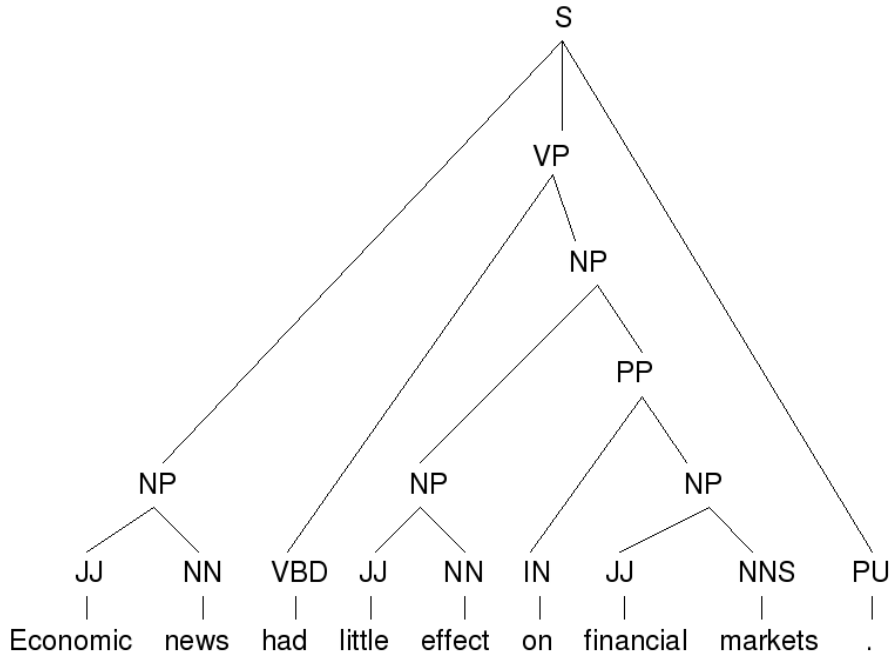


FIG. 2.1 – *Structure de constituant pour une phrase en anglais de Penn Treebank*

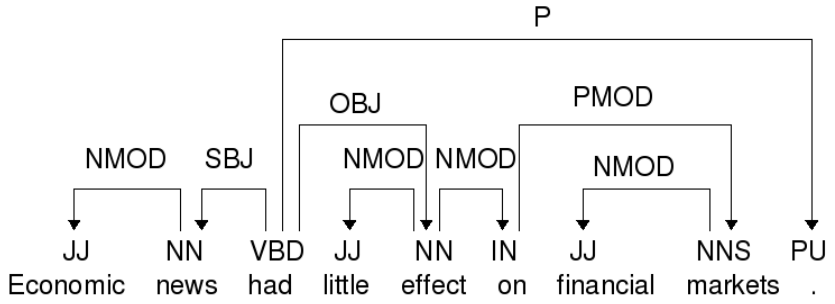


FIG. 2.2 – *Structure de dépendance pour une phrase en anglais du Penn Treebank*

Après qu'un modèle d'analyse ait été défini, un ensemble de contraintes formelles et des paramètres appropriés acquis, le modèle doit fixer un algorithme d'analyse pour résoudre le problème d'analyse. Compte tenu des contraintes, des paramètres et d'une nouvelle phrase S, cela consiste à savoir comment le système trouve le graphe de dépendances le plus probable pour cette phrase : $G = h(S, \Gamma, \lambda)$.
La fonction h fait une recherche sur l'ensemble des graphes de dépendances $\mathcal{G}_S$ bien-formés pour une phrase en entrée S. Elle produit un seul graphe ou rien si Γ définit une grammaire dans laquelle S n'est pas un membre de la langue définie. h peut prendre plusieurs formes algorithmiques, y compris des algorithmes *gloutons* et *récursifs*. En outre, h peut être exacte ou approximatif par rapport à une certaine fonction d'objectif.

2.2.1.2 *Analyse syntaxique basée sur les transitions*

Le principe de l'analyse syntaxique basée sur les transitions est basée sur «l'analyse par décalages-réductions», qui est une technique d'analyse assez connue[4]. Cette technique a besoin de deux conteneurs d'information, une pile et un tampon. Au début, tous les tokens de la phrase sont dans le tampon et la pile est vide. Au cours de l'analyse, la pile contiendra les tokens traités.

Quatre sortes d'opérations sont possibles : transition arc-gauche, transition arc-droit, décalage et réduction. Plus précisément, la transition arc-gauche ajoute un arc de dépendance entre la tête du tampon et la tête de la pile, où le token sur la tête de la pile est le *mot-tête* de la dépendance. Ensuite, le token sur la tête de la pile est enlevé. La transition arc-droit ajoute un arc de dépendance entre la tête du tampon et la tête de la pile, où le token sur la tête du tampon est le *mot-tête* de la dépendance. Ensuite, le token sur la tête du tampon est ajouté sur le dessus de la pile.

L'opération se décalage permet de prendre le token au début du tampon et de le déplacer vers la pile.

La dernière est l'opération de réduction, qui permet d'enlever un token de la pile si celui-ci a déjà une tête.

Le tableau 2.1 et la figure 2.3 montrent la technique par décalages-réductions sur un exemple :

Opération	Pile	Tampon	Dépendance
	[ROOT]	[Le, ...]	-
Décalage	[ROOT, Le]	[garçon, ...]	-
Arc-gauche	[ROOT]	[garçon, ...]	(garçon, DET, Le)
Décalage	[ROOT, garçon]	mange, ...] -	
Arc-gauche	[ROOT]	[mange, ...]	(mange, SBJ, garçon)
Arc-droit	[ROOT, mange]	[une, ...]	(ROOT, ROOT, mange)
Décalage	[ROOT, mange, une]	[pomme, ...]	-
Arc-gauche	[ROOT, mange]	[pomme, ...]	(pomme, DET, une)
Arc-droit	[ROOT, mange,pomme]	[.]	(mange, OBJ, pomme)
Réduction	[ROOT, mange]	[.]	-
Arc-droit	[ROOT, mange,.]	[]	(mange, P, .)

Tableau 2.1 – *Exemple étape par étape de la technique "décalages-réductions" pour la phrase : le garçon mange une pomme.*

Pour réussir à déduire quelle opération effectuer dans un état donné, il faut avoir une sorte d'oracle auquel se référer. Il existe plusieurs techniques d'approximation d'oracles comme les grammaires formelles et les heuristiques de désambiguïsation. Par contre, la technique la plus utilisée est l'entraînement de classifieurs sur des corpus arborés.

Tout d'abord, pour pouvoir entraîner des classifieurs pour ce problème, il faut pouvoir représenter les données à traiter d'une façon traitable par la machine. Du coup, on identifie principalement chaque token avec son indice de position dans la phrase, son étiquette de catégorie syn-

[4]Nous allons détailler cette techniques plus précisément dans la méthode de Nivre.

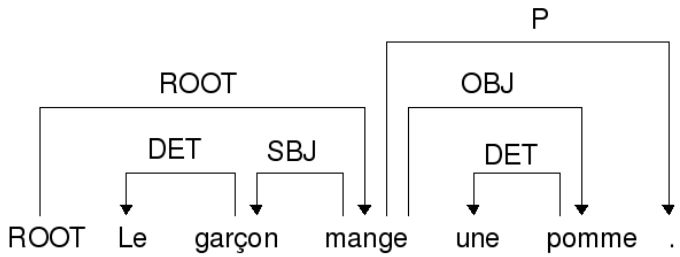


FIG. 2.3 – *Structure de dépendance pour la phrase :* le garçon mange une pomme.

taxique *POS* (*part-of-speech*, la partie du discours) et sa dépendance dans la phrase.

Le classifieur est entraîné sur des corpus arborés et les instances des règles découvertes sont répertoriées durant l'entraînement. Ensuite, on utilise notre approximation d'oracle pour définir l'analyse par décalages-réductions.

Méthode de Nivre : un algorithme d'analyse en dépendance incrémental a été proposé par Covington (2001). Certaines études prennent des approches dirigées par les données, comme Kudo et Matsumoto (2002), Yamada et Matsumoto (2003) et Nivre (2003). L'analyseur déterministe incrémental a été généralisé au système de transition d'états (Nivre (2008)).

	Transition	Précondition
Arc-gauche	$(\sigma\|w_i, w_j\|\beta, A) \Rightarrow (\sigma, w_j\|\beta, A \cup (w_j, w_i))$	$i \neq 0 \wedge \neg\exists w_k$ $(w_k, w_i) \in A$
Arc-droit	$(\sigma\|w_i, w_j\|\beta, A) \Rightarrow (\sigma\|w_i\|w_j, \beta, A \cup (w_i, w_j))$	
Réduction	$(\sigma\|w_i, \beta, A) \Rightarrow (\sigma, \beta, A)$	$\exists w_k(w_k, w_i) \in A$
Décalage	$(\sigma, w_j\|\beta, A) \Rightarrow (\sigma\|w_j, \beta, A)$	

Tableau 2.2 – *Les transitions de la méthode de Nivre*

La méthode de Nivre utilise une pile de tokens notée σ, un tampon β contenant initialement la phrase x. L'analyse est formulée comme un quadruplet (S, T_s, S_{init}, S_t), où chaque composant est défini comme suit :

- S est un ensemble d'états. Chacun est dénoté par $(\sigma, \beta, A) \in S$.
- T_s est un ensemble de transitions. Chaque élément de T_s est une fonction $t_s : S \longrightarrow S$.
- s_{init} =([w_0],[w_1, ..., w_n],ϕ) est l'état initial.
- S_t est l'ensemble des états terminaux.

L'analyse syntaxique génère une séquence de transitions optimales t_s fournie par un oracle $o : S \rightarrow T_s$, appliquée à une cible composée des éléments au-dessus de la pile w_i et du premier élément w_j dans le tampon. L'oracle est construit comme un classifieur entraîné sur des corpus arborés. Chaque transition est définie dans le tableau 2.2 et expliquée comme

suit :

- **Arc-droit** : w_j devient la tête de w_i et on dépile w_i où w_i est situé au dessus de la pile (noté $\sigma|w_i$), lorsque la tête du tampon est w_j (notée $w_j|\beta$).

- **Arc-gauche** : w_i devient la tête de w_j et on empile w_j.

- **Réduction** : dépiler w_i situé au dessus de la pile.

- **Décalage** : empiler le token w_j situé en tête du tampon, au dessus de la pile.

Kitagawa et Tanaka-Ishii (2010) ont proposé une autre approche basée sur la méthode de Nivre. Ils ont présenté un modèle "fondé sur les arbres" qui décide des relations de dépendances entre les têtes des arbres au lieu d'entre les tokens. Cela étend l'espace de recherche pour obtenir la meilleure tête d'un token dans un modèle déterministe.

L'idée de se fonder sur des arbres est potentiellement applicable à différentes méthodes précédentes d'analyses. Cette idée a amélioré la méthode de Nivre. Ce modèle fondé sur des arbres choisit d'abord la tête candidate qui est la plus probable parmi les arbres grâce à un tournoi et décide alors de l'action de l'analyse entre deux arbres.

Cette méthode a surpassé diverses méthodes d'analyses déterministes qui sont précédemment rapportées. Aussi cette méthode peut être située entre les méthodes d'analyses d'optimisations globales en élargissant l'espace de recherche.

Bien que la complexité dans le pire des cas de cette méthode est $O(n^2)$, la moyenne du temps d'analyse n'est pas beaucoup plus petit que $O(n)$.

Les algorithmes d'analyse en dépendances déterministes : l'analyse en dépendances déterministe a récemment émergé car, non seulement c'est une des plus efficaces, mais aussi c'est l'une des méthodes les plus précises pour l'analyse en dépendances, à condition que l'analyseur déterministe soit guidé par un classifieur entraîné sur des données de corpus arborés.

Définition 2.8 Un algorithme d'analyse en dépendances déterministe considère l'analyse comme une séquence d'actions d'analyse qui sont prises pas à pas sur une phrase en entrée. L'action d'analyse construit les relations de dépendances entre les tokens.

Cet algorithme a été introduit par Kudo et Matsumoto (2002) pour le Japonais et Yamada et Matsumoto (2003) pour l'Anglais et plus tard a été développé et appliqué à un large éventail de langues par Nivre et al. (2004), Cheng et al. (2004) et Attardi (2006), entre autres. L'approche déterministe basée sur un classifieur a été représentée par l'un des deux systèmes principaux les plus performants (Nilsson (2006)). Les algorithmes définis ci-après prennent une chaîne de tokens w_1, ..., w_n dans le vocabulaire W de la grammaire de dépendances G et construisent un graphe de dépendances avec des nœuds w_1, ..., w_n. Etant donné deux tokens w_i

et $w_j \in W$, un nom de dépendance r, on ajoute les règles $w_i \xleftarrow{r} w_j$ et/ou $w_i \xrightarrow{r} w_j$ qui sont compatibles avec les contraintes de G (Nivre et Nilsson (2003)). La grammaire G doit répondre aux questions suivantes :
Est ce que w_i peut être le dépendant (à gauche) de w_j avec le nom de dépendance r (est-ce que $w_i \xleftarrow{r} w_j$ est compatible avec G) ?
Est ce que w_j peut être le dépendant (à droite) de w_i avec le nom de dépendance r (est-ce que $w_i \xrightarrow{r} w_j$ est compatible avec G) ?
Si une des questions a été eu une réponse, donc on peut dire que G contient la règle $w_i \xleftarrow{r} w_j$ et/ou la règle $w_i \xrightarrow{r} w_j$.

L'algorithme de base : l'algorithme de base construit un graphe de dépendances en liant chaque token à son gouverneur le plus proche possible :

Algorithme 1 *L'algorithme de base*

1: **pour** $k = 1$ à $n - 1$ **faire**
2: **pour** *position* $i = 1$ à $n - k$ **faire**
3: Link(w_i, w_{i+k})
4: **fin pour**
5: **fin pour**

L'opération Link(w_i, w_j) est définie comme suit :
si w_i n'a pas de tête et $w_i \leftarrow w_j$ est compatible avec G **alors**
ajouter la règle (w_j, w_i)
sinon
si w_j n'a pas de tête et $w_i \rightarrow w_j$ est compatible avec G **alors**
ajouter la règle (w_i, w_j)
finisi
finisi

L'algorithme de base s'exécute $n - 1$ fois sur les tokens d'une phrase en entrée de la gauche à la droite (où n est le nombre de tokens en entrée), compte tenu des liens possibles de longueur k pendant l'itération k, ce qui donne un temps d'exécution qui est quadratique par rapport à la longueur de l'entrée.

L'algorithme incrémental : alors que l'algorithme de base relie chaque token à son gouverneur le plus proche possible, l'algorithme incrémental cherche plutôt à relier chaque nouveau token à un token précédent.

Algorithme 2 *L'algorithme incrmental*

1: **pour** position $j = 2$ à n **faire**
2: **pour** position $i = j - 1$ (vers le bas -1) **faire**
3: $Link(w_i, w_j)$
4: **fin pour**
5: **fin pour**

La complexité en temps est donc la même que pour l'algorithme de base.

L'algorithme projectif : sans contraintes supplémentaires, les deux premiers algorithmes ne sont pas garantis de produire des graphes de dépendances connexes et acycliques. L'algorithme projectif est une extension de l'algorithme de base qui élimine le problème de l'acyclicité. C'est-à-dire que le résultant d'un graphe est assuré d'être acyclique et projectif (mais pas nécessairement connexe). Nous définissons une notion d'accessibilité pour les nœuds d'un graphe : un nœud w_j est accessible si et seulement s'il n'y a pas de règle (w_i, w_k) tels que $i < j < k$.
Compte tenu de la notion d'*accessibilité*, nous pouvons définir l'algorithme projectif simplement en définissant une opération plus complexe Link(w_i, w_j) :

Algorithme 3 *L'algorithme projectif*

1: **si** w_i n'a pas de tête et $w_i \leftarrow w_j$ est compatible avec G et w_i et w_j sont accessibles et il n'y a pas d'arc de w_i à w_j **alors**
2: ajouter la règle (w_j, w_i)
3: **finsi**
4: **si** w_j n'a pas de tête et $w_i \rightarrow w_j$ est compatible avec G et w_i et w_j sont accessibles et il n'y a pas d'arc de w_j à w_i **alors**
5: ajouter la règle (w_i, w_j).
6: **finsi**

2.2.1.3 *Analyse syntaxique basée sur les graphes*

Nous présentons dans cette section l'nalyse syntaxique basée sur les graphes (Nivre et al. (2009)). Les graphes de dépendances sont des structures syntaxiques sur des phrases, comme nous l'avons mentionné dans la définition de la phrase (c'est-à-dire que la phrase est une séquence de tokens $S = w_1 w_2 ... w_n$).

L'un des principes fondamentaux des analyseurs basés sur les graphes est celui de la notion de scores. En effet, chaque graphe est évalué par un score qui indique la viabilité de ce graphe d'être la solution valide. En fait, ce score est la combinaison des scores des sous-graphes.

De la sorte, les scores sont attribués aux arcs plutôt qu'au sous-graphe contenant les deux nœuds aux extrémités de l'arc. Le terme arbre couvrant maximum est aussi utilisé pour référer au graphe avec le meilleur score. Des noms de dépendances sont aussi attachés aux arcs afin d'indiquer la relation de dépendance entre les deux nœuds.

D'abord, on doit passer d'analyse étiquetée LAS[5] à non étiquetée UAS[6]. La raison est qu'on utilise des algorithmes qui sont seulement compatibles avec une structure non étiquetée. Il faut donc retirer les noms de dépendances des arcs pour analyser la phrase. Il existe différents algorithmes pour l'analyse projective et non-projective. On se base sur l'algorithme de Chu-Liu-Edmonds (Chu et Liu (1965)). Ici seulement les grandes

[5]labelled attachment score LAS, qui calcule la proportion de tokens pour lesquels le gouverneur assigné est correct et le nom de dépendance est correct.

[6]Unlabelled attachment score UAS, qui calcule la proportion de tokens pour lesquels le gouverneur assigné est correct.

lignes de cet algorithme seront présentées. Nous présentons l'exemple de Nivre et al. (2009) comme référence pour faciliter la compréhension (cf. figure 2.4).

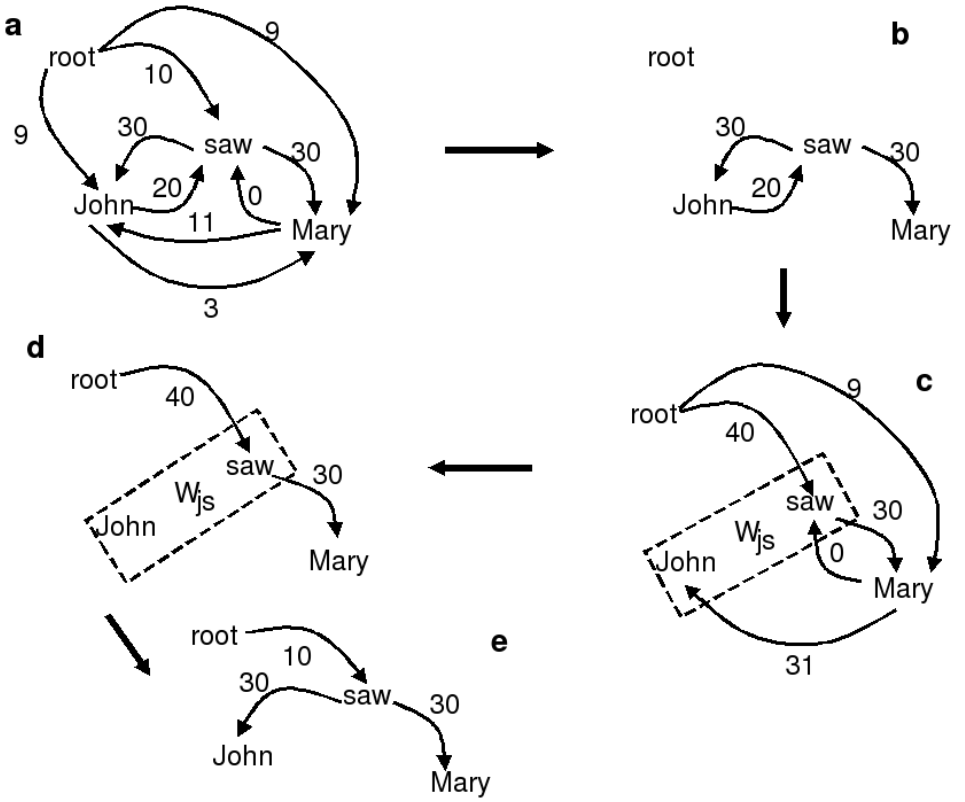


FIG. 2.4 – *Application de l'algorithme de Chu-Liu-Edmonds pour la phrase : "John saw Mary."*

Avant d'appliquer l'algorithme de *Chu-Liu-Edmonds*, il faut fixer les poids initiaux des arcs grâce à un modèle construit par apprentissage automatique. Une fois les poids initiaux fixés, on peut débuter l'analyse. Tout d'abord, on trouve l'arc arrivant avec la plus grande valeur pour chaque nœud. Par exemple, à l'étape **a**, il existe trois *arcs* arrivants sur le nœud *saw*, soit l'arc de poids 10 de *root* à *saw*, l'arc de poids 20 de *John* à *saw* et l'arc de poids 0 de *Mary* à *saw*. L'arc avec le plus grand poids est l'arc de poids 20 de *John* à *saw*. Il sera donc conservé. Ensuite, les autres arcs entrants pour ce nœud sont éliminés, soit l'arc de *root* à *saw* et l'arc de *Mary* à *saw* (étape **a** à **b** de la figure 2.4).

Par la suite, on trouve les cycles dans le graphe et on les regroupe de manière à ce qu'ils soient représentés par un seul nœud par cycle. Dans notre exemple, on a gardé l'arc de *John* à *saw* à l'étape précédente puisque c'était l'arc entrant avec le plus haut poids pour le nœud *saw*. Par contre, on a aussi gardé l'arc de *saw* vers *John* puisque c'est l'arc entrant avec le plus haut poids pour le nœud *John*. Ces deux arcs forment donc un cycle et on va le représenter par le nœud W_{js} (étape **b** à **c** de la figure 2.4).

On a donc un nouveau graphe réduit puisqu'un cycle a été remplacé par un seul nœud. Il est à noter que le poids des arcs qui entraient dans le cycle est remplacé par le poids du plus long chemin amenant dans le

cycle. Par exemple, l'arc de *root* vers *saw* de poids 10 est remplacé par un arc de poids 40. Ceci est dû au fait qu'il existe un chemin de *root* vers *John*, un autre membre du cycle, de poids 40. Ce chemin est constitué de l'arc de poids 10 de *root* vers *saw* suivi de l'arc de poids 30 de *saw* vers *John*. Ces deux arcs sont donc combinés pour former un seul arc entrant de poids 40 de *root* vers W_{js}. Sur ce nouveau graphe, on applique à nouveau la première étape, qui consistait à ne garder que l'arc entrant avec le plus haut poids pour chaque nœud (étape **c** à **d** de la figure 2.4). A l'étape **d**, on conserve donc le nouvel arc de poids 40 de *root* vers W_{js} ainsi que le même arc entrant de poids 30 pour le nœud *Mary*.

En se rappelant les points terminaux originaux, un arbre peut finalement être construit une fois que l'élimination des cycles ait été complétée (étape **d** à **e** de la figure 2.4).

Par la suite, l'arbre généré est traduit en graphe de dépendance en ajoutant les relations de dépendance associées à chaque branche.

Pour les analyses projectives, l'algorithme *Cocke-Younger-Kasami* est appliqué. Le principe de cet algorithme est de calculer toutes les séquences et sous-séquences de tokens de taille 1 à `n`, où `n` est le nombre de tokens dans la phrase. L'idée est de commencer par trouver le score de tous les sous-graphes de taille 1, ensuite prendre les sous-graphes adjacents et trouver le score de tous ces graphes de taille 2 et ainsi de suite. Toutefois, cet algorithme prend $O(n^3)$ pour être exécuté, ce qui est un problème significatif et rend l'algorithme beaucoup moins intéressant. Évidemment, des variantes de l'algorithme comme l'algorithme *d'Earley* ont été trouvées pour répondre à ce problème de complexité. Cet algorithme est toutefois assez complexe et sa compréhension approfondie n'est pas nécessaire pour la suite de nos travaux.

L'approche basée sur les graphes est aussi dirigée par les données. Tout d'abord, il faut représenter les données qu'on veut tirer de l'entraînement. On représente des caractéristiques linguistiques d'un arc comme l'étiquette *POS* du nœud de l'arc arrivant et sortant et des poids sont attribués à ces caractéristiques selon leur importance. Ensuite, durant l'entraînement sur un corpus arboré, on trouve le graphe avec le meilleur score.

2.2.1.4 *Analyse syntaxique basée sur les grammaires*

Certains techniques d'analyse basées sur les grammaires utilisent tous les principes fondamentaux d'une grammaire hors-contexte (notée CFG pour *context-free grammar*). Nous retrouvons donc les notions de symboles terminaux, non-terminaux et de règles de production. En fait, nous pouvons plutôt voir les grammaires de dépendance projective comme un cas spécial des CFG où les symboles terminaux sont des tokens. Plus formellement, nous définissons une grammaire hors-contexte comme un quadruplet (N, Σ, Π, S) où :

- N est un ensemble de symboles non-terminaux ;
- Σ est un ensemble de symboles terminaux ;
- Π est un ensemble de règles de production $X \rightarrow \alpha$, où X est un et un seul symbole appartenant à N et α une chaîne de symboles terminaux et non-terminaux ;
- S est le symbole de départ et appartient à l'ensemble N.

Ensuite, une notion statistique est ajoutée à ces concepts, transformant la *CFG* en grammaire hors-contexte probabiliste (noté PCFG pour *probabilistic context-free grammar*). Pour chaque règle de production, une probabilité d'application est ajoutée. Ces probabilités sont calculées par un apprentissage automatique sur un corpus. Par exemple, pour deux règles avec le même symbole non-terminal à gauche, des probabilités sont ajoutées aux deux règles pour un total de 1,00. Ainsi, quand nous voulons transformer un symbole non-terminal, la règle à appliquer est choisie de façon probabiliste. L'exemple qui suit a pour but de faciliter la compréhension :

- N={ *S, SV, SN, SA, N, V, Adj, Det* } (où SV est un syntagme verbal, SN un syntagme nominal, SA un syntagme adjectival, N est un nom, V est un verbe, Adj est un adjectif, Det est un déterminant et ϵ est un élément absent).
- Σ={ *le, petit, garçon, mange, une, pomme* }.
- S est le symbole de départ.

Le tableau 2.3 montre l'analyse syntaxique basée sur les grammaires pour la phrase : le petit garçon mange une pomme.

Règle appliquée	Résultat
	S
$S \rightarrow SN\ SV$	$SN\ SV$
$SN \rightarrow Det\ SA\ N$	$Det\ SA\ N\ SV$
$SA \rightarrow Adj\ SA$	$Det\ Adj\ SA\ N\ SV$
$SV \rightarrow V\ SN$	$Det\ Adj\ SA\ N\ V\ SN$
$SN \rightarrow Det\ SA\ N$	$Det\ Adj\ SA\ N\ V\ Det\ SA\ N$
$SA \rightarrow \epsilon$	$Det\ Adj\ SA\ N\ V\ Det\ \epsilon\ N$
$SA \rightarrow \epsilon$	$Det\ Adj\ \epsilon\ N\ V\ Det\ \epsilon\ N$
–	*Le petit garçon mange une pomme.*

Tableau 2.3 – *Analyse syntaxique basée sur les grammaires : construction par grammaire*

2.2.2 Analyse en dépendances stochastiques pour le français

Il devient également populaire d'utiliser les analyseurs de dépendances stochastiques. Dans cette section, nous présentons quelques analyseurs stochastiques qui produisent des étiquettes de dépendances pour le français. Ces analyseurs utilisent différents modèles d'analyses.

Candito et al. (2010b) présentent une comparaison entre trois architectures dont les sorties sont des arcs de dépendances en français. Cette comparaison concerne les performances de ces architectures d'analyses statistiques sur le problème de la dérivation de type des structures des dépendances pour le français. Ces trois systèmes sont comparés à la fois en termes de précision de l'analyse et de temps d'analyse.

Le premier est l'analyseur MSTparser (McDonald (2006)) qui effectue une recherche par *arbre de recouvrement maximal* dans un graphe complet en enlevant des arcs violant les contraintes d'un graphe de dépendance comme nous l'avons défini dans la section 2.2.1.3. Les analyseurs de ce type sont capables d'analyser des dépendances non-projectives.

La deuxième approche est une mise en œuvre de l'analyseur "décalage/réduction" (Yamada et Matsumoto (2003)), étendu par exemple dans (Nivre (2005), Nivre et al. (2006), Nivre (2008)). Dans ce modèle, le graphe de dépendance est construit de façon incrémentale en utilisant une pile pour stocker les tokens de la phrase et de quatre actions : *décalage, réduction, arc-gauche* et *arc-droit*. L'idée de ce modèle est de traverser les phrases de gauche à droite et à chaque étape, d'effectuer une des actions possibles, jusqu'à ce qu'un graphe complet soit construit. L'analyseur ne peut pas analyser les arcs non-projectifs (sauf par exemple la mise en oeuvre décrit dans Attardi (2006)), mais ils peuvent être analysés en utilisant la technique connue du pseudo-projectif de Nivre et Nilsson (2005).
Le troisième analyseur, l'analyseur de Berkeley (Petrov et al. (2006)) a une architecture "fondée sur les constituants ou analyse en constituants[7]".

Comparaisons : dans l'ensemble, MSTParser obtient la plus grande précision en étiqueté (LAS). Cependant, les résultats pour les trois systèmes sur l'ensemble des tests sont à peu près dans un point de pourcentage pour les deux précisions en étiquetés et en non étiquetés, ce qui signifie qu'on ne trouve pas le décalage entre les analyseurs fondés sur les constituants et basés sur les dépendances qui a été signalé pour l'anglais par Cer et al. (2010).

	Le score d'attachement en étiqueté LAS	Le score en non-étiqueté UAS	Temps d'analyses
Berkeley	86.8	91.0	12m46s
MaltParser	87.3	89.7	1m25s
MSTParser	88.8	90.9	14m39s

Tableau 2.4 – *Comparaisons de Candito et al. (2010b) sur trois analyseurs*

MaltParser s'exécute environ 9 fois plus rapidement que le système de Berkeley et 10 fois plus rapidement que MSTParser. La différence d'efficacité est principalement due au fait que MaltParser utilise un algorithme d'analyse en temps linéaire, tandis que les deux autres analyseurs ont une complexité en temps cubique.

Compte tenu de la différence plutôt petite en précision d'étiquettes, MaltParser semble être un bon choix pour le traitement des corpus de grande envergure.

Ressources importantes pour les analyses en dépendances : pour construire des analyseurs stochastiques avec une grande précision dans divers domaines, nous avons besoin de préparer un corpus d'entraînement bien annoté pour chacun des domaines cibles. Il existe des corpus en dépendances pour certaines langues comme :

[7]L'analyse en constituants ou les grammaires de constituants organisent les groupes de tokens en syntagmes alors que les grammaires de dépendances mettent la dépendance entre tokens au centre de la structure syntaxique.

– le Prague Arabic Dependency Treebank (PADT) (Hajič et Zemánek (2004)) ;
– le BulTreeBank (BTB) Simov et al. (2005) ;
– le Turin University Treebank (TUT) (Bosco et Lombardo (2004)) ;
– le Slovene Dependency Treebank (SDT) (Džeroski et al. (2006)).

Leurs schémas d'annotations sont différents (avec PADT et SDT les schémas d'annotation sont assez similaires). PADT, TUT et SDT sont des corpus arborés en dépendances originaux tandis que BTB a été converti du format *Head-driven Phrase Structure Grammar* (HPSG) en graphes de dépendance dans Chanev et al. (2006). Pour le français, nous décrivons des corpus en dépendances qui sont automatiquement convertis à partir d'un corpus arboré du français (Abeillé et Barrier (2004)), un corpus arboré basé sur les constituants et qui se compose de 12.531 phrases du journal *Le Monde*. Chaque phrase est annotée avec une structure de constituant et les tokens portent les traits suivantes : genre, nombre, mode, temps, personne.

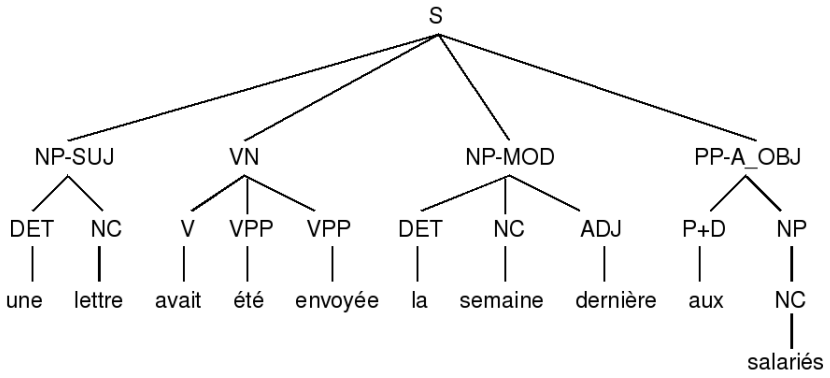


FIG. 2.5 – *un exemple d'arbre de constituant du FTB-UC*

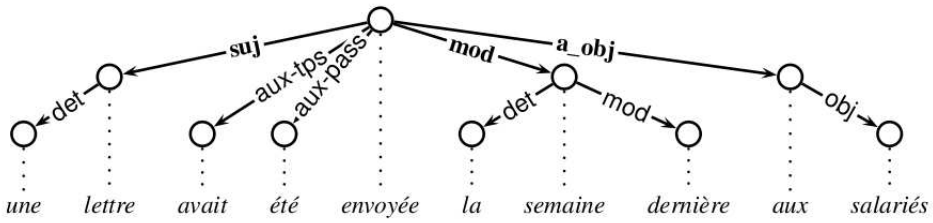


FIG. 2.6 – *La correspondance de l'arbre de dependance*

Tous ces corpus sont décrits dans Candito et al. (2010a). FTB-UC est une version modifiée du corpus FTB basé sur les constituants, où l'annotation morphologique riche a été transformée en un ensemble de 28 étiquettes de parties du discours. Le corpus FTB-UC-DEP est un corpus arboré en dépendance (présenté dans Candito et al. (2009)) qui est dérivé de FTB en utilisant la technique classique des règles de propagation des têtes, proposée pour l'anglais par Magerman (1995).

Avec cette technique de conversion, les graphe de dépendances sont nécessairement projectifs, et les dépendances extraites sont nécessairement locales pour une phrase.

Ce qui signifie que les arbres convertis automatiquement peuvent être considérés comme des approximations de pseudo-projectifs pour les arbres de dépendance corrects (Kahane et al. (1998)). Candito et al. (2010a) ont évalué les arbres convertis pour 120 phrases et obtiennent un score de 98% en étiquetés (LAS) lorsqu'on compare les arbres de dépendances automatiquement convertis aux arbres corrigés manuellement.

D'autre travaux sur le corpus arboré du français sont présentés dans Poupard et al. (2006a,b). Les auteurs présentent leur solution qui reprend les deux phases présentées dans leurs articles : transformation du corpus d'arbres en structures *FA* annotées puis application d'un algorithme d'apprentissage sur ces structures pour obtenir le lexique.

2.3 Les formalismes de grammaires

2.3.1 Grammaires de liens

Il existe différents types de formalismes capables de structurer un langage donné en terme de relations syntaxiques. Une de ces applications grammaticales s'appelle les grammaires de liens *Link-Grammar* (Sleator et Temperley (1991)), qui consiste à représenter une phrase comme un ensemble de tokens reliés entre eux par des liens syntaxiques.

Le principe général des grammaires de liens est qu'une phrase est correcte s'il est possible d'établir des liens entre les tokens, selon les contraintes imposées par les définitions des tokens dans le lexique. Ces liens représentent les relations syntaxiques entre les tokens.

Définition 2.9 Une grammaire de liens est définie par un vocabulaire (dictionnaire), un ensemble de connecteurs et une relation qui met en correspondance chaque token avec un ensemble d'étiquettes. Une étiquette est un couple de listes ordonnées de connecteurs. Une phrase donnée appartient au langage décrit par la grammaire de liens s'il est possible de dessiner tous ses liens au dessus des tokens.

De manière plus précise, la figure 2.7 montre les entrées du dictionnaire des grammaires de liens. On peut dire que le token "cat" requiert un connecteur *D* (Déterminant) par son contexte gauche et qu'il peut se connecter soit en tant qu'objet *O* par contexte gauche soit en tant que sujet *S* par contexte droit.

Pour bien comprendre, les exemples donnés ci-dessous sont représentés graphiquement. En fait, ils sont codés dans le dictionnaire à l'aide de formules logiques : par exemple, les doublons[8] de la figure 2.7 seront représentés par la formule logique suivante :
(cat, dog, ...) $-$ > D- & (S+ or O-).
Les signes - et + correspondent au contexte auquel le token se raccroche,

[8]Ces doublons se définissent par un ensemble de tokens associé à une formule logique représentant les différents liens possibles auxquels ces tokens peuvent prétendre.

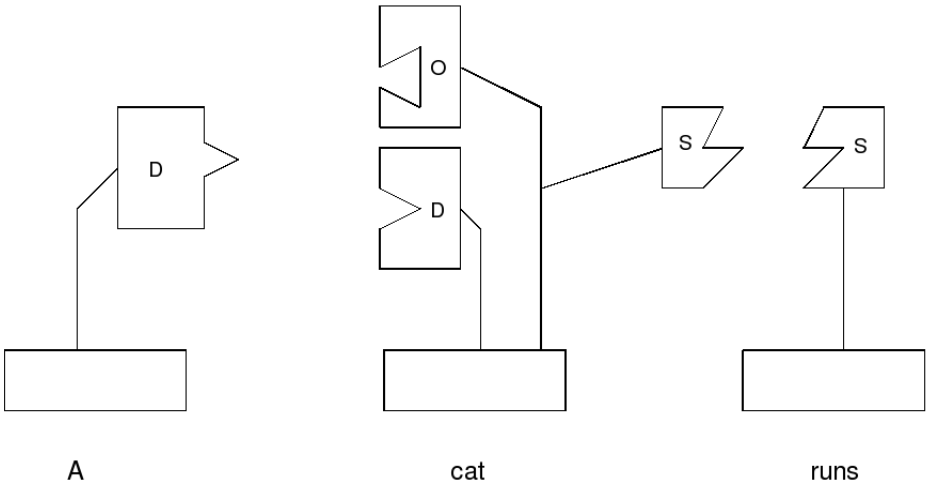


FIG. 2.7 – *Exemple d'entrées du dictionnaire des grammaires de liens*

respectivement gauche et droit et les catégories D, S, O correspondent respectivement aux fonctions : Déterminant, Sujet, Objet direct.

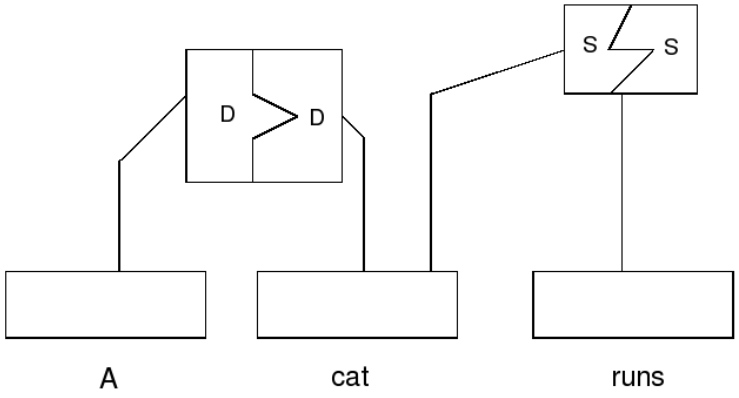


FIG. 2.8 – *Chemin de liens obtenus pour la phrase : "A cat runs"*

L'analyseur de Sleator et Temperley est puissant, rapide et robuste. Il a aussi des avantages. L'un des avantages des grammaires de liens est que le dictionnaire utilisé est facilement modulable, en effet la structure du dictionnaire a été réalisée pour être facilement simplifiée automatiquement. Par exemple, si l'on décide que les informations sur les propositions relatives ne sont pas pertinentes pour notre application, il suffit de supprimer certaines occurrences du dictionnaire qui traitent ce phénomène linguistique.

On donne un exemple de l'implémentation de l'analyseur des grammaires de liens dans la fiqure 2.9.

Le résultat de l'analyse est une suite de relations grammaticales entre les tokens et qui se visualise sous la forme d'un graphe d'arcs planaires[9].

[9]Ce résultat est pris du site Web *http ://www.link.cs.cmu.edu/link/submit-sentence-4.html*

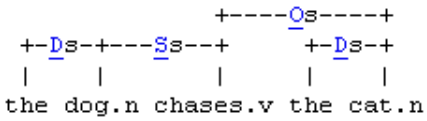


FIG. 2.9 – *Résultats de l'analyse de la phrase : the dog chasses the cat*

2.3.2 Les grammaires catégorielles (GC)

Contrairement aux grammaires de dépendances, les grammaires catégorielles proposent un formalisme strict de représentation des phrases. Ce type de grammaires obéit à des règles de calcul logique, ce qui constitue un avantage très important pour le traitement informatique, et pour l'étude des propriétés mathématiques de ces grammaires. Les grammaires catégorielles que nous présentons ici sont les grammaires catégorielles classiques, aussi appelées grammaires AB.

2.3.2.1 *Grammaires AB*

Les grammaires catégorielles classiques, nommées aussi grammaires AB, ont été introduites dans Bar-Hillel (1953). Ces grammaires sont complètement lexicalisées : cela signifie qu'une grammaire est décrite uniquement par son lexique, le lexique étant l'association d'une ou plusieurs catégories à chaque token du vocabulaire. Les règles utilisées dans les dérivations sont donc *universelles*. Ces règles sont :

$$A/B, B \longmapsto A \quad \textbf{FA}\ (Forward\ Application)$$
$$B, B \setminus A \longmapsto A \quad \textbf{BA}\ (Backward\ Application)$$

Les catégories (aussi nommées types) sont des termes utilisant les opérateurs binaires / et \. Intuitivement, une expression est de type A/B (resp. $B\backslash A$) si cette expression est de type A lorsqu'elle est suivie (resp. précédée) par une expression de type B. Une phrase est correcte s'il est possible d'associer à chaque token l'une de ses catégories, de telle sorte que les règles universelles permettent de transformer cette séquence de catégories en la catégorie spéciale S.

Exemple 2.1 Soit G la grammaire constituée du lexique ci-dessous :

Lexique	
token	type
Jean, Marie, Paul	SN
aime, déteste	(SN\S)/SN
qui	(SN\SN)/(SN\S)

Tableau 2.5 – *Lexique de la phrase : Jean qui aime Marie, déteste Paul*

La phrase «Jean qui aime Marie, déteste Paul» appartient au langage de cette grammaire. On peut le montrer avec la dérivation suivante :
⇒ SN, (SN\SN)/(SN\S), (SN\S)/SN, SN, (SN\S)/SN, SN

⇒ SN, (SN\SN)/(SN\S), (SN\S)/SN, SN, SN\S
⇒ SN, (SN\SN)/(SN\S), SN\S, SN\S
⇒ SN, SN\SN, SN\S
⇒ SN, SN\S
⇒ S

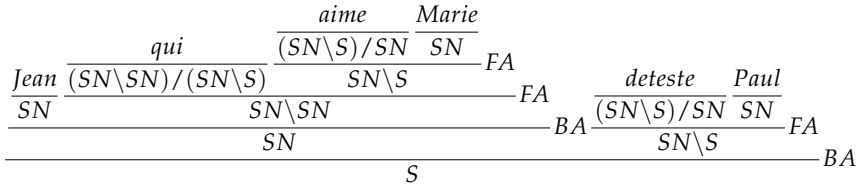


FIG. 2.10 – *Arbre de dérivation de la phrase : Jean qui aime Marie, déteste Paul*

Une grammaire est rigide si à chaque token du vocabulaire n'est associée qu'une seule catégorie syntaxique. De même, une grammaire est dite k-valuée si chaque token est défini par au plus k catégories. On peut décrire une dérivation à l'aide d'un arbre de manière classique, en étiquetant les nœuds par la catégorie du constituant qu'ils représentent. De plus, la forme des règles permet aussi de représenter un arbre de dérivation en étiquetant les nœuds seulement par l'identifiant de la règle utilisée (FA ou BA). Une telle structure, dans laquelle les feuilles ne sont étiquetées que par un token, est appelée *FA-structure* (pour Functor-Argument structure). Cette représentation est unique pour un arbre de dérivation donné (voir figure 2.10). En revanche une FA-structure donnée peut représenter un nombre infini d'arbres de dérivations.

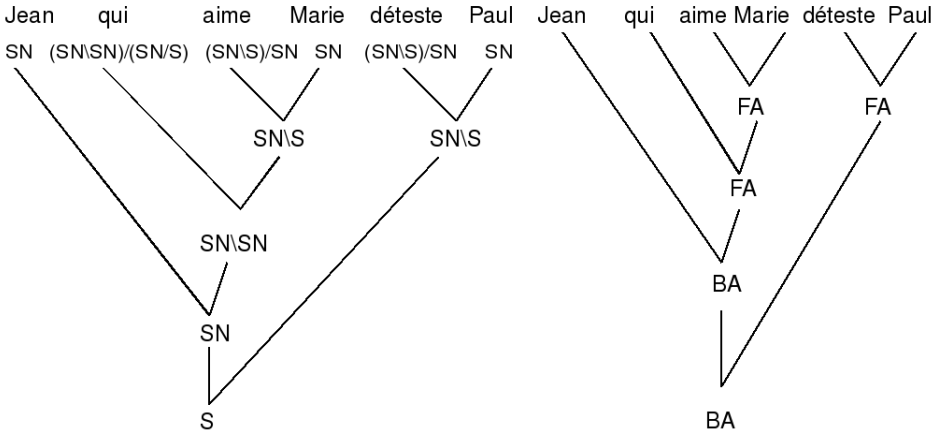


FIG. 2.11 – *Arbre de dérivation sans le nom des règles et FA-structure*

Les grammaires catégorielles classiques de Bar-Hillel et al. (1964) ou Lambek (1958) peuvent rendre compte de la structure des phrases simples en liant des notions qui parfois doivent être séparées : l'ordre des constituants et la structure prédicative.

2.3.3 Grammaires algébriques

Définition 2.10 (Grammaires algébriques) : une grammaire algébrique est un quadruplet $\langle N, \Sigma, \Pi, S\rangle$ où :

- N est un ensemble de symboles non-terminaux ;
- Σ est un ensemble de symboles terminaux ;
- Π est un ensemble de règles de production $X \rightarrow \alpha$, où X est un et un seul symbole appartenant à N et α une chaîne de symboles terminaux et non-terminaux ;
- S est le symbole de départ et appartient à l'ensemble N.

Les grammaires algébriques, aussi appelées grammaires hors-contexte, sont connues pour être analysables en temps polynomial (voir Sikkel et Nijholt (1997), l'algorithme d'Earley (Earley (1970)) et l'algorithme CKY (Cocke et Schwartz (1970), Kasami (1965) et Younger (1967))).

Les grammaires algébriques sont assez centrales dans la hiérarchie de Chomsky, du fait de leur expressivité souvent suffisante et de leurs propriétés qui les rendent facilement utilisables en pratique. Elles ont été très étudiées et très utilisées. Il y a en particulier deux formes normales pour ces grammaires :
– une grammaire algébrique $G=\langle N, \Sigma, \Pi, S\rangle$ est en `forme normale` de Chomsky si toutes ses règles sont de la forme $A \rightarrow BC$ ou $A \rightarrow a$, avec $A, B, C \in N$ et $a \in \Sigma$.
– une grammaire algébrique $G=\langle N, \Sigma, \Pi, S\rangle$ est en `forme normale` de Greibach si toutes ses règles sont de la forme $A \rightarrow aA_1A_2...A_n$, avec $A_1, A_2, ..., A_n \in N$ et $a \in \Sigma$. Si n $\leq$ 2, la grammaire est dite *en forme normale forte de* Greibach.
Il est démontré que toute grammaire algébrique est (faiblemente) équivalente à une grammaire sous *forme normale de* Chomsky et aussi sous *forme normale forte de* Greibach.

2.3.4 Les grammaires d'arbres adjoints (TAG)

Dans cette section, nous décrivons le formalisme syntaxique des TAG, notamment ses propriétés linguistiques et informatiques. Le formalisme des grammaires d'arbres adjoints ou "Tree Adjoining Grammars (TAG)" a permis des analyses linguistiques originales et efficaces de plusieurs phénomènes pour un certain nombre de langues naturelles, comme l'anglais, le français ou l'allemand.

Les TAG sont un formalisme initialement développé dans le cadre de la théorie des langages formels, dont la pertinence pour la représentation de phénomènes linguistiques a été argumentée après coup (Kroch et Joshi (1985)). Le formalisme des TAG permet une analyse dans le pire des cas en temps polynomial (Vijay-Shanker (1987) ; Schabes (1990)) et permet également de modéliser de façon psycholinguistiquement pertinente le langage humain. Tout cela a rendu possible le développement de grammaires à large couverture aussi bien pour le français (Abeillé (1991) et Crabbé (2005)) que pour l'anglais[10] (XTAG-Research-Group (1995)) ainsi

[10]Le format XTAG permet de factoriser la description des grammaires et de donner suffisamment d'information pour lexicaliser à la demande des parties de la grammaire.

que la mise au point d'outils de génération semi-automatique et de maintenance de ces grammaires (Candito (1996)).

Grammaires d'arbres adjoints lexicalisées (LTAG)
Une grammaire d'arbres adjoints lexicalisée ou Lexicalized Tree Adjoining Grammar (LTAG) est une grammaire d'arbres adjoints dans laquelle tout arbre élémentaire est muni d'au moins un symbole terminal appelé nœud ancre. Une grammaire d'arbres adjoints lexicalisée est un formalisme fortement lexicalisé au même titre que les grammaires catégorielles de Bar-Hillel (1953). Dans la suite de ce chapitre, nous nous intéressons exclusivement aux LTAG.

Composants d'une grammaire TAG
Une TAG se compose d'un ensemble fini d'arbres. L'analyse avec une grammaire TAG fait intervenir les définitions suivantes :

Définition 2.11 Les arbres élémentaires : dans une grammaire TAG, chaque entrée lexicale `L` est associée à une famille d'arbres syntagmatiques, appelés arbres élémentaires, qui décrivent les différentes configurations possibles pour `L`. La lexie `L` ancre chaque arbre élémentaire de sa famille, c'est-à-dire figure comme feuille de l'arbre.

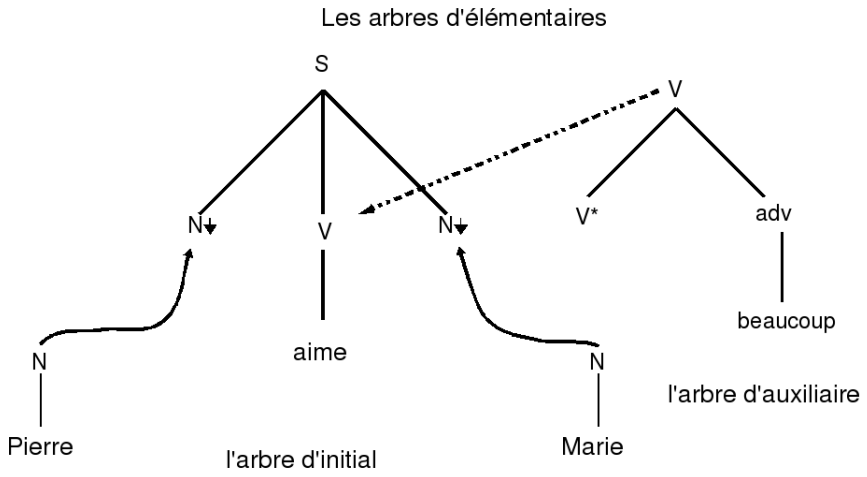


FIG. 2.12 – *Les arbres élémentaires pour la phrase : Pierre aime beaucoup Marie*

On distingue deux types d'arbres élémentaires : `arbres initiaux` et `arbres auxiliaires`. Ces deux types d'arbres se combinent avec d'autres arbres suivant des opérations différentes : les `arbres initiaux` par substitution (l'arbre vient se substituer par sa racine à une feuille d'un autre arbre) et les `arbres auxiliaires` par adjonction (l'arbre vient s'insérer à la place d'un nœud non-feuille d'un autre arbre).

Définition 2.12 Les arbres initiaux : les arbres initiaux sont caractérisés par des nœuds internes non terminaux, et des nœuds feuilles terminaux et non terminaux. Les nœuds feuilles non terminaux sont appelés nœuds à substitution, et sont notés par convention $(\downarrow)$.

Définition 2.13 Les arbres auxiliaires : les arbres auxiliaires sont caractérisés par des nœuds internes non terminaux, et des nœuds feuilles terminaux et non terminaux. À la différence des arbres initiaux, il y a au moins un nœud feuille non terminal. Parmi les nœuds feuilles non terminaux, nous en distinguons un appelé nœud pied, noté par convention avec une étoile $(*)$. Le nœud pied est de la même catégorie que le nœud racine. La figure 2.13 montre plusieurs exemples d'arbres d'auxiliaires.

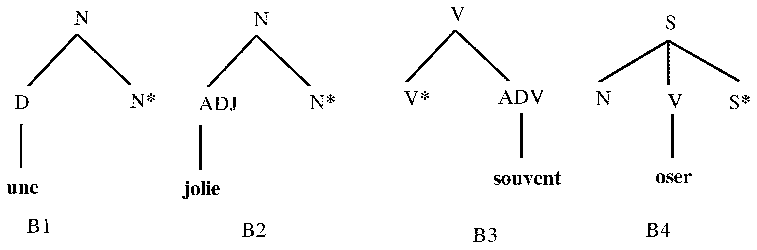


FIG. 2.13 – *Certains exemples sur les arbres d'auxiliaires*

Chaque nœud d'un arbre élémentaire est renseigné par deux structures de traits *top* et *bottom* qu'on présentera par la suite.

Définition 2.14 L'opération de substitution : l'opération de substitution insère un arbre de racine X à un nœud feuille de même catégorie dans un arbre élémentaire. La figure 2.14 illustre cette opération.

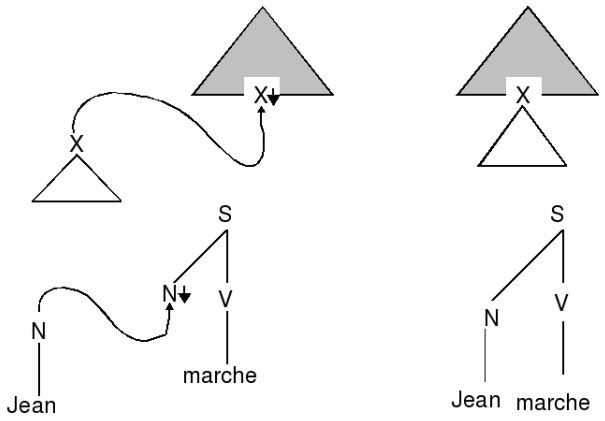


FIG. 2.14 – *L'opération de substitution sur la phrase : Jean marche*

Définition 2.15 L'opération d'adjonction insère un arbre auxiliaire au sein d'un arbre élémentaire. Le nœud X où a lieu l'adjonction est remplacé par l'arbre auxiliaire, dont la racine et le nœud pied sont de catégorie X. Le sous-arbre initialement dominé par X de l'arbre auxiliaire, est inséré en dessous du nœud pied. La figure 2.15 illustre cette opération.

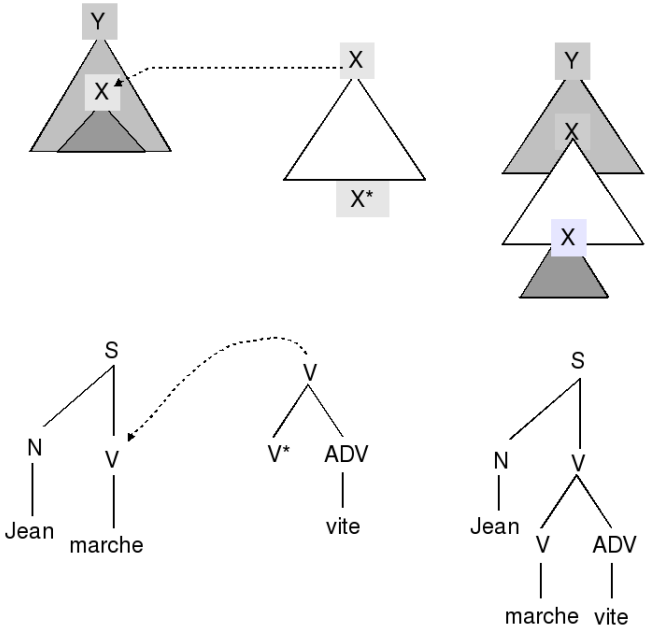


FIG. 2.15 – *L'opération d'adjonction dans la phrase : Jean marche vite*

Grammaires TAG et structures de traits
Les structures de traits[11] (Shieber (1985)) ont été introduites initialement par le formalisme des grammaires fonctionnelles d'unification. À l'origine, elles proviennent du traitement des phénomènes d'accord.
En TAG, ces traits sont associés aux nœuds, et sont tous à valeur atomique. Ils peuvent donner une information morphologique, syntaxique ou encore sémantique.

Unification de deux structures de traits
Les structures de traits des nœuds impactés par l'opération de combinaison doivent pouvoir s'unifier. Si l'unification échoue, la combinaison est interdite. On associe aux nœuds d'un arbre TAG deux structures de traits : une structure *top* et une structure *bottom*. Lors de l'adjonction à un nœud X la structure de traits *top* de ce nœud doit s'unifier avec la structure *top* de la racine de l'arbre auxiliaire comme l'illustre la figure 2.16. Pour une

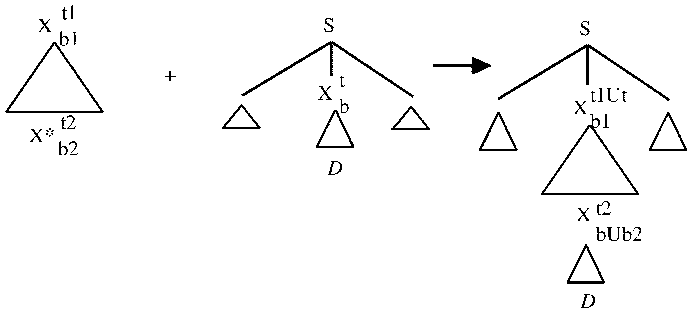


FIG. 2.16 – *Structures de traits et adjonctions d'arbres*

[11] Feature structure ou structure de trait : un trait est un couple attribut-valeur.

substitution, seules les structures *top* de la feuille concernée et du nœud racine de l'arbre intial doivent s'unifier comme illustré sur la figure 2.17.

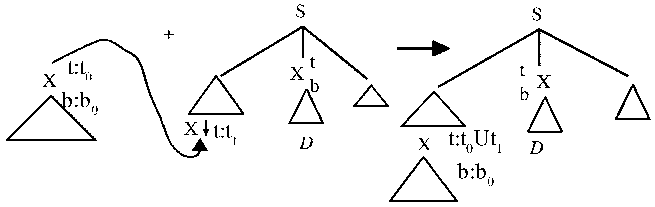


FIG. 2.17 – *Structures de traits et substitution d'arbres*

Les traits *top* et *bottom* doivent être en mesure de s'unifier à chaque nœud. Nous trouvons en partie *top*, les traits indiquant les relations du nœud avec les nœuds qui le dominent, ou du même niveau en partie *bottom*, les traits indiquant les relations du nœud avec ceux qu'il domine.
Le résultat de l'unification de structures de traits est défini par : la structure de traits *top* du nouveau nœud est le résultat de l'unification des deux structures de traits *top* des deux nœuds d'origine, la structure de traits *bottom* recevant alors simplement la structure de traits *bottom* du nœud de la racine de l'arbre de substitution.

Exemple 2.2 La structure de traits du token "*Jean*" est de la forme suivante :

$$\begin{bmatrix} cat & = & n \\ pers & = & 3 \\ num & = & sg \\ gen & = & m \end{bmatrix}$$

Dans des structures de traits différentes d'un même arbre, il peut exister des traits de même nom qui partagent une même valeur. En effet, l'information se propage d'un nœud vers un autre pour satisfaire des contraintes morpho-syntaxiques (accord, temps,...). Par exemple, dans la figure 2.15 le verbe "marche" demande un sujet qui soit de la première ou la troisième personne du singulier. Donc les valeurs attribuées aux traits *num* et *pers* du nœud du sujet doivent être les mêmes que ceux du verbe.

Propriétés formelles
Formellement, une grammaire d'arbres adjoints G est définie par le quintuplet $G = (\Sigma, \mathcal{N}, S, \mathcal{I}, \mathcal{A})$, où :
- Σ est un ensemble fini de symboles terminaux ;
- $\mathcal{N}$ est un ensemble fini de symboles non-terminaux, $\mathcal{N} \cap \Sigma = \emptyset$. En pratique, ce sont les catégories syntaxiques ;
- $S \in \mathcal{N}$ est le symbole distingué ou l'axiome ;
- $\mathcal{I}$ est un ensemble fini d'arbres initiaux ;
- $\mathcal{A}$ est un ensemble fini d'arbres auxiliaires.

Une grammaire TAG lexicalisée (Lexicalized TAG-LTAG) est une grammaire TAG dont chaque arbre élémentaire de $\mathcal{I} \cup \mathcal{A}$ contient au moins un nœud feuille étiqueté par un symbole terminal. Une grammaire lexicalisée peut être vue comme une fonction associant à chaque token du lexique un

ensemble de structures syntaxiques (arbres) représentant l'usage de ce token dans les différentes phrases de la langue (Parmentier (2007)). Notons que les grammaires lexicalisées présentent un avantage pratique : lors de l'analyse syntaxique, l'analyseur peut sélectionner une sous-grammaire suivant les tokens de la chaîne à analyser, ce qui facilite la complexité en temps de l'analyse.

A partir de ces définitions d'une grammaire TAG et des opérations permises, il est possible de définir le langage généré par une grammaire TAG. Soit G une grammaire TAG. On note T_G l'ensemble des arbres dérivés complets engendrés par G. On appelle langage généré par G, l'ensemble des chaînes de symboles terminaux situées sur les feuilles des éléments de T_G.

Expressivité des grammaires TAG

En considérant la classification des langages de Chomsky (1956), les grammaires TAG appartiennent à la famille des grammaires permettant d'engendrer tous les langages hors-contextes, ainsi que certains langages contextuels, comme par exemple le langage $a^n b^n e c^n d^n$ (cf. la figure 2.18[12]).

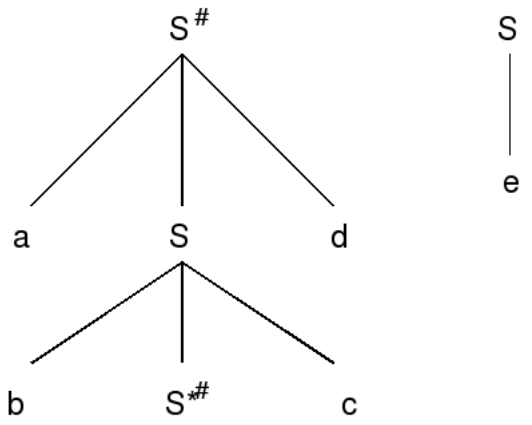


FIG. 2.18 – *Grammaire TAG engendrant le langage $a^n b^n e c^n d^n$*

Cependant, il existe des langages contextuels qui ne peuvent pas être engendrés par une grammaire TAG, comme par exemple le langage contextuel $a^n b^n e^n c^n d^n$. De ce fait, les grammaires TAG sont considérées comme appartenant à la classe des grammaires légèrement sensibles au contexte (Villemonte de La Clergerie et Miguel Alonso Pardo (1998)).

[12]Le symbole # indique une adjonction interdite.

Les résultats de l'analyse TAG

Définition 2.16 L'arbre dérivé : un arbre dérivé est obtenu par une suite d'opérations sur des arbres élémentaires, initiaux ou auxiliaires comme illustré sur la figure 2.19.

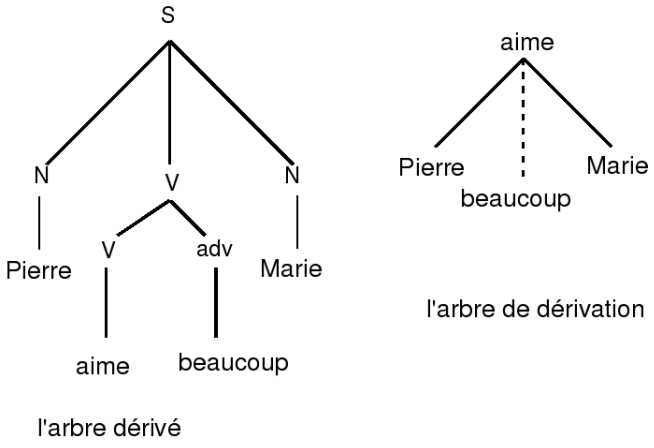


FIG. 2.19 – *L'arbre dérivé et l'arbre de dérivation : Pierre aime beaucoup Marie*

Définition 2.17 L'arbre de dérivation : l'arbre de dérivation est parfois appelé arbre de syntaxe concrète pour le distinguer de l'arbre de syntaxe abstraite construit généralement par le compilateur d'un langage de programmation. Cet arbre de syntaxe abstraite est plus compact que le précédent et contient des informations sur la suite des actions effectuées par un programme. Chaque nœud interne de cet arbre possède une étiquette qui désigne une opération à exécuter. Il s'obtient par des transformations simples à partir de l'arbre de dérivation. Nous donnons un exemple dans la figure 2.19.

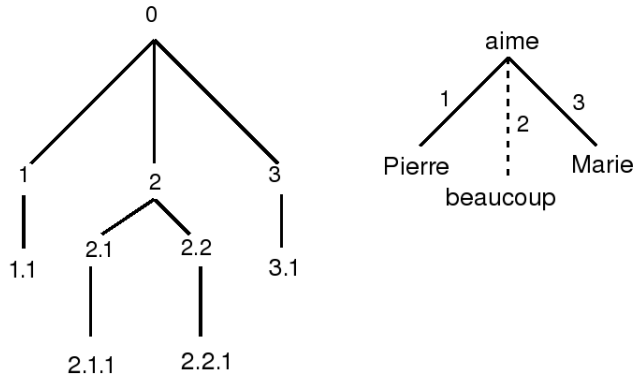


FIG. 2.20 – *La localisation du nœud*

Dans l'arbre de dérivation, *les opérations de substitution* sont notées avec une ligne continue, celles d'*adjonction* avec une ligne en pointillés. Il faut noter que chaque lien de l'arbre de dérivation spécifie un numéro de liens

dans la paire d'arbres élémentaires. Ces liens donnent l'adresse des opérations dans l'arbre syntaxique. La figure 2.20 représente "la localisation" de chaque nœud de l'arbre dérivé et de l'arbre de dérivation.

Ressources linguistiques TAG
Un certain nombre de ressources linguistiques de taille importante ont été développées pour le formalisme TAG. Il existe une grammaire TAG pour certaines langues comme par exemple pour l'anglais (XTAG-Research-Group (1995)), l'allemand (Kallmeyer et al. (2008)) et le coréen (Yoon (2004)). Pour le français, l'Université Paris 7 a développé une grammaire de couverture importante (Abeillé (2002)). Il existe également une grammaire TAG développée au laboratoire LORIA (Crabbé (2005)).
On présente un analyseur syntaxique pour une grammaire TAG pour le français appelé *LLP2*.
LLP2 (Roussanaly et al. (2005)) est un analyseur syntaxique profond qui s'appuie sur la grammaire d'arbres adjoints lexicalisés *LTAG* (Joshi et al. (1975)). L'algorithme implémenté est celui de l'analyse par connexité. L'intégration d'un module de traitement de structures de traits et d'unification, permet de prendre en compte les traits *top* et *bottom* aux nœud des LTAG. En d'autres termes, LLP2 a la capacité de traiter des grammaires d'arbres adjoints avec unification de structures de traits (FB-TAG, *Featured-based* TAG (Vijay-Shanker (1987))). *LLP2* a été développé en Java et est disponible sous licence GPL. Du point de vue des ressources, LLP2 s'inspire de l'architecture XTAG qui distingue le lexique morphologique (permettant d'étiqueter les segments et d'identifier les lemmes correspondants), le lexique syntaxique (qui permet la sélection des arbres par filtrage et leur ancrage) et la grammaire (qui contient les arbres TAG). Le lexique morphologique est majoritairement construit à partir de *MULTEXT* (Ide et Véronis (1994)). Le lexique syntaxique est extrait du lexique fourni par L. Clément et utilisés par l'analyseur XLFG (Clément (2001)). Un mécanisme par défaut de sélection des arbres élémentaires sur la base de règles reposant sur les traits morphologiques a été mise en place pour pallier les insuffisances du lexique syntaxique. La grammaire a été engendrée à l'aide d'une méta-grammaire conçue par Crabbé (2005) et compilée avec l'outil XMG développé au LORIA (Duchier et al. (2005)).

Pour résumer cette section (2.3.4), nous avons présenté la grammaire TAG et ses combinaisons, et les opérations qui par combinaison, constituent le processus d'analyse.

2.4 Conclusion

Ce chapitre porte essentiellement sur quelques méthodes existantes dans le domaine de l'analyse syntaxique qui est un domaine de la linguistique assez riche. À partir d'un besoin de clarté en matière d'analyseurs syntaxiques et morphosyntaxiques, l'idée est venue de faire ce point, très certainement non-exhaustif, sur la terminologie du domaine théorique et applicatif (TAL) de la syntaxe.

Nous nous sommes intéressés à deux techniques largement employées

dans le cadre de l'analyse syntaxique. La première est *dirigée par les données* qui a récemment émergée car, non seulement c'est une des plus efficaces, mais aussi c'est l'une des méthodes les plus précises pour l'analyse en dépendances. La deuxième technique est *basée sur une grammaire*. Nous avons présenté certains formalismes grammaticaux. En fait, nous avons esquissé quelques-uns des principaux formalismes grammaticaux adaptés aux langues naturelles.

Dans le prochain chapitre 3, nous décrivons un formalisme pour les grammaires de dépendances, appelé CDG, grammaire catégorielle de dépendance. Cette grammaire nous intéressera pour la suite de notre travail.

Grammaires catégorielles de dépendances

3

Sommaire

3.1 Introduction 41
3.2 Les grammaires catégorielles de dépendances 41
3.2.1 Type de CDG 41
3.2.2 Calcul de types des dépendances 43
3.2.3 Extension des CDG 44
3.2.4 Expressivité des CDG 47
3.3 L'analyseur des CDG 48
3.4 Conclusion 51

Ce chapitre donne une solution à un problème important dans l'analyse en dépendances. Ce problème est les dépendances discontinues.
Ce chapitre est organisé comme suit :
La section 3.1 évoque l'introduction de ce chapitre. La section 3.2 illustre la particularité d'une grammaire catégorielle de dépendances qui est la gestion de dépendances discontinues (non projectif). La section 3.2.2 explique les règles qui permettent de formaliser le calcul de types des dépendances d'une grammaire catégorielle de dépendances. La section 3.2.3 montre la proposition de Dikovsky (2009) de faire des CDG à large couverture avec des classes lexicales et des expressions régulières pour les types. La section 3.3 évoque l'analyseur des CDG Lab. La section 3.4 conclut ce chapitre.

3.1 Introduction

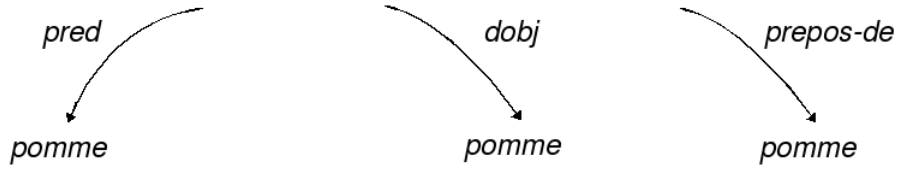
Nous décrivons dans ce chapitre un formalisme grammatical appelé grammaires catégorielles de dépendances (noté CDG pour *Categorial Dependency Grammars*) ainsi que les résultats de l'analyse syntaxique avec ces grammaires. Nous montrons les principes généraux sous-jacents aux grammaires catégorielles de dépendances. Elles traitent les dépendances discontinues (les particules négatives, les pronoms comparatifs, etc.) qui posent un défi pour la plupart des grammaires de dépendances.

3.2 Les grammaires catégorielles de dépendances

Les grammaires catégorielles de dépendances (CDG) sont des grammaires catégorielles dont la particularité est d'expliciter les relations de dépendances entre les mots d'une phrase plutôt que les relations de dominances entre ses constituants. Toutes les unités lexicales sont regroupées en classes lexicales. Toutes les unités d'une classe partagent les mêmes propriétés syntaxiques attribués à la classe.
Elles sont très bien adaptées aux définitions de la syntaxe du langage naturel en termes de dépendances syntaxiques entre les mots dans la phrase. Les CDG proposées par Dikovsky (2004) sont un modèle de grammaires de dépendances basé sur le même fonctionnement que les grammaires catégorielles. Leur particularité réside dans la gestion des dépendances discontinues.

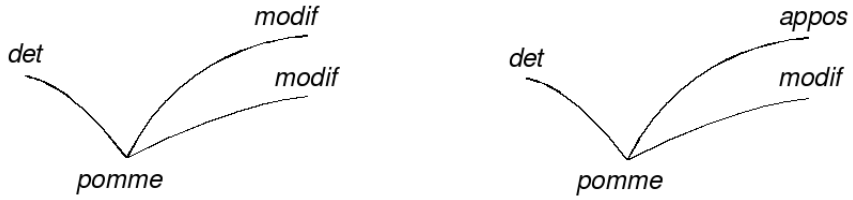
3.2.1 Type de CDG

Nous avons défini les dépendances comme des relations binaires de surface entre les mots. $w_1 \xrightarrow{d} w_2$ veut dire w_1 autorise ou réclame w_2 en fonction de la dépendance d (w_1 est le gouverneur, w_2 est le subordonné). Supposons qu'on a la phrase "Jean aime Marie.". Nous pouvons trouver deux relations comme suit : $Jean \xleftarrow{subj} aime \xrightarrow{dobj} Marie$. La première est la dépendance *subj* entre le verbe *aime* et son sujet *Jean*. La deuxième est la dépendance *dobj* entre le verbe *aime* et son objet direct *Marie*. Les types de dépendances d'un mot w sont définis en tenant compte des deux statuts de w : en tant que gouverneur *début des flèches* et en tant que subordonné *fin des flèches*. Nous prenons le mot français *pomme* qui en tant que subordonné peut avoir les dépendances :

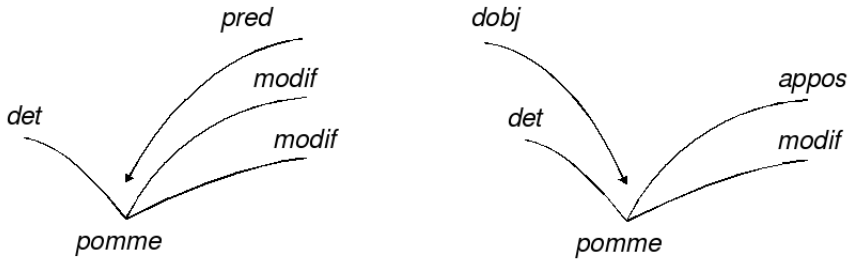


Le même mot *pomme* en tant que gouverneur peut avoir les dépen-

dances :



Les types de dépendances sont les combinaisons compatibles des deux statuts :



La première combinaison est compatible avec le type $[det\backslash pred/modif/modif]$. La deuxième avec le type $[det\backslash dobj/appos/modif]$. Par exemple, le deuxième type réclame à gauche un subordonné avec la dépendance *det* (déterminant) et à droite il admet zéro ou plusieurs subordonnés via la dépendance itérée *modif* (modifieur) et réclame un subordonné avec la dépendance *appos* (appositif), dans cet ordre, pour devenir un objet direct subordonné via la dépendance *dobj*. Ces types définissent les dépendances *locales continues* des mots.

Polarités des dépendances distantes (discontinues)
Selon les propositions dans Dikovsky (2001) et Dikovsky (2004) les dépendances discontinues sont définies par des types polarisés de dépendance appelées des valences. Une valence positive indique le nom et la direction d'une dépendance discontinue sortante. La valence négative correspondante avec le même nom a une direction opposée et indique la fin de cette dépendance entrante (les deux valences sont duales). Ainsi les dépendances discontinues sont indiquées par des paires de valences duales. Nous représentons la définition de Béchet et al. (2011).

Définition 3.1 Soit **C** un ensemble de noms locaux de dépendances et **V** un ensemble de noms de valences.

Les valences polarisées sont définies par les expressions $\swarrow v$, $\nwarrow v$, $\searrow v$ et $\nearrow v$ où $v \in \mathbf{V}$, $\nwarrow v$ et $\nearrow v$ sont positives, $\swarrow v$ et $\searrow v$ sont négatives ; $\nwarrow v$ et $\swarrow v$ sont à gauche, $\nearrow v$ et $\searrow v$ sont à droite. Deux valences polarisées avec le même nom et la même orientation mais avec des signes opposés

sont duales. Une expression de la forme #($\swarrow v$), #($\searrow v$), $v \in \mathbf{V}$ est appelée type ancre ou plus simplement ancre. Une expression de la forme d^* et $d^?$ où $d \in \mathbf{C}$ représente respectivement un type itéré et optionnel de dépendances. Les noms locaux de dépendances, les types itérés de dépendances et les types ancres forment les types primitifs.
Valence négative ancrées (à un type t) : elles réclament les fins des dépendances discontinues définies par les valences négatives voisines à t (à gauche ou à droite). Les valences ancrées servent à placer les subordonnées distantes et définir l'ordre des clitiques et leur adjacence avec le verbe. Par exemple, en français le nom de dépendance #($\swarrow clit-3d-iobj$) d'un objet indirect pronominalisé (un clitique) s'ancre *à gauche* à un type (au type d'un verbe auxiliaire ou principal), comme décrit dans la figure 3.2.
Les valences polarisées positives peuvent être vues comme des dépendances discontinues sortant du gouverneur, et les valences polarisées négatives comme celle des dépendances discontinues entrant dans les mots subordonnés. Elles correspondent donc respectivement aux débuts et aux fins des dépendances discontinues (distantes).

Une expression de la forme t=$[l_m \backslash \cdots \backslash l_1 \backslash H / \cdots / r_1 \cdots / r_n]$ dans laquelle m, $n \geq 0$, l_1, $\cdots$, l_m, r_1, $\cdots$, r_n sont des types primitifs et H est soit un nom local de dépendance, soit un type ancre est appelé un type de dépendances de base. l_1, $\cdots$, l_m et r_1, $\cdots$, r_n sont respectivement les sous types gauches et droits de t. H est appelé le sous type tête de t (ou type tête). La forme générale d'un type CDG est une expression B^P dans laquelle B est un type de dépendances de base B^P=$[l_m \backslash \cdots \backslash l_1 \backslash H / \cdots / r_1 \cdots / r_n]^P$ et P est un potentiel, une chaîne de valences polarisées (éventuellement vide) qui définit les dépendances distances.
CAT(**C**, **V**) dénote l'ensemble de tous les types de dépendances sur **C** et **V**.

3.2.2 Calcul de types des dépendances

Les règles suivantes permettent de formaliser le calcul sur les types de dépendances[1] (avec C $\in$ **C**, α un type de base et β une partie d'un type de base) :
L^l. $C^{P_1}[C \backslash \beta]^{P_2} \vdash [\beta]^{P_1 P_2}$
I^l. $C^{P_1}[C^* \backslash \beta]^{P_2} \vdash [C^* \backslash \beta]^{P_1 P_2}$
Ω^l. $[C^* \backslash \beta]^P \vdash [\beta]^P$
D^l. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1 P P_2}$ si le potentiel $(\swarrow C)P(\nwarrow C)$ satisfait la règle d'appariement **FA** (First Available).
L^l est la règle classique d'élimination. En éliminant l'argument de type C, elle construit la dépendance projective entre le mot gouverneur et son subordonné et conserve les potentiels par concaténation.
I^l et Ω^l concernent les dépendances itérées, c'est-à-dire les dépendances pouvant se répéter un nombre quelconque de fois. I^l est proche de L^l car elle construit aussi des dépendances projectives en concaténant les potentiels. I^l dérive $k > 0$ instances de C. Ω^l s'applique dans le cas $k = 0$ c'est pourquoi elle fait disparaître C^* de la partie gauche sans consommer

[1] Les règles gauches et droites étant symétriques, seules les règles gauches sont exposées.

de C. La dernière règle D^l gère les dépendances distantes en appariant et éliminant les valences polarisées duales ($\swarrow$ et $\nwarrow$ ou $\nearrow$ et $\searrow$) et en créant une dépendance distante entre les deux mots dont les types portaient ces valences.

3.2.3 Extension des CDG

Dikovsky (2009) propose de faire des CDG à large couverture avec des classes lexicales et des expressions régulières pour les types. L'affectation de type λ est étendue aux classes : $\lambda : M \longrightarrow 2^{Types}$ dans la façon dont :
– tous les mots dans une classe h partagent les types qui sont assignés à h ;
– chaque mot possède tous les types d'une classe à laquelle il appartient $\lambda(w) =_{df} \cup_{w \in h} \lambda(h)$.
L'extension des CDG utilise les expressions régulières suivantes pour définir des ensembles de types.

- Choix. $(t_1|...|t_k)$ une des options.
- Choix optionnel. $(t_1|...|t_k)?$ une des options ou absence de dépendance.
- Itération. $(t_1|...|t_k)*$ itération.
- L'ordre flexible. $[\{t_1, ...t_k\} \backslash \alpha \backslash H / \beta]^P$ (les subordonnés $t_1, ..., t_k$, sont quelque part à gauche de la tête H. Symétriquement à droite : $[\alpha \backslash H / \beta / \{t_1, ...t_k\}]^P$.
- $\{t_1, ...t_k\}\ [\alpha \backslash H / \beta]^P$: les subordonnées sont à gauche ou à droite du verbe.

Règles pour les choix :
- $LA^l_{gov}.\ \ t^{P_1}[(t|\alpha)\backslash\beta]^{P_2} \vdash [\beta]^{P_1 P_2}$
- $LA^l_{sub}.\ \ (t|\alpha)^{p_1}[(t\backslash\beta)]^{P_2} \vdash [\beta]^{P_1 P_2}$

les règles des sous-types flexibles à gauche :
- $LB^l_i.\ \ t^{P_1}\ [\{t, \alpha\}\ \backslash\beta]^{P_2} \vdash [\{\alpha\}\ \backslash\beta]^{P_1 P_2}$
- $LB^l_o.\ \ t^{P_1}\ [\{\alpha\}\ \backslash t \backslash\beta]^{P_2} \vdash [\{\alpha\}\ \backslash\beta]^{P_1 P_2}$

La règle LA^l_{gov} dit qu'afin d'éliminer le choix, il suffit d'éliminer l'un de ses éléments.
La règle LA^l_{sub} dit qu'un choix peut éliminer un de ses éléments.
La règle LB^l_i dit que les sous-types flexibles peuvent être éliminés indépendamment de ceux qui sont fixés.
La règle LB^l_o dit que les sous-types qui sont fixés sont éliminés dans l'ordre indiqué.

Définition 3.2 *Projectivité* : une structure projective (non croisée) induit que les relations de dépendances restent à un niveau local, ce qui permet une analyse simple et efficace. Cette notion initialement définie pour les arbres peut se transposer sur les graphes : une structure de dépendances SD est dite projective si pour tout mot donné, l'ensemble des nœuds atteignables depuis ce mot dans la structure de dépendances correspond à un segment continu de l'énoncé (on note que les dépendances projectives sont notées

avec des lignes continues). La figure 3.1 décrit une structure projective pour la phrase : *au commencement était le Verbe.*

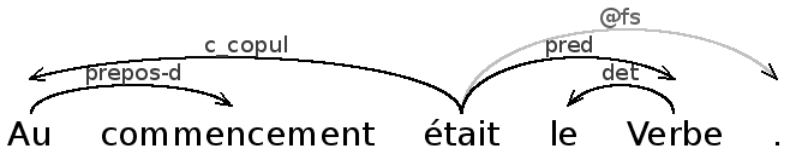


FIG. 3.1 – *Au commencement était le Verbe.*

L'analyse de cette phrase est un arbre de dépendances projectif dont la tête (la racine) est le mot *était*. Elle est définie par les types neutres suivants :

$$
\begin{array}{lcl}
au^{PP(F=compl,C=d)} & \mapsto & [c\_copul/prepos-d] \\
commencement^{N(Lex=common)} & \mapsto & [prepos-d] \\
était^{Vcopul(F=fin)} & \mapsto & [c\_copul\backslash S/@fs/pred] \\
le^{Det(Lex=art|pn)} & \mapsto & [det] \\
Verbe^{N(Lex=common)} & \mapsto & [det\backslash pred] \\
.^{FullStop(Lex=".")} & \mapsto & [@fs]
\end{array}
$$

Le mot *au* est classé dans la classe des prépositions (PP) avec complément "datif" et les mots *commencement* et *Verbe* sont classés dans la classe des nominatifs (N) tandis que le mot *était* est classé dans la classe des copules (Vcopul). Enfin le mot *le* est classé dans la classe des déterminants (Det) comme article.

Dépendances non-projectives (discontinues) : un des problèmes importants des grammaires de dépendances porte sur les dépendances discontinues. Ces dépendances posent un problème important pour l'analyse syntaxique non seulement du point de vue théorique, mais également concernant les mécanismes nécessaires pour leur implantation.

Définition 3.3 *Non projectivité* : les dépendances discontinues représentent les dépendances entre le mot-tête et le mot subordonné séparés par des mots ne dépendant pas de la tête. Ces dépendances sont dues aux constructions discontinues.

Nous pouvons noter que les dépendances non projectives (ou croisées) apparaissent dans plusieurs cas en français, par exemple :

- la négation : ne ... pas (constituants discontinus) comme dans la phrase : "Marie n'aime pas Jean".
- les comparatifs : "plus de ... que, plus grand que".
- la topicalisation des groupes nominaux qui se manifeste par leur déplacement au début de la phrase.
- les clitiques : "Jean en mange un morceau.". Ici *en* est le complément de l'objet *morceau*, mais *en* est séparé par le verbe *mange* lui-même lié au sujet *Jean*.

La figure 3.2 illustre sur un exemple les clitiques en langue française dans la phrase "*elle la lui a donnée.*". Dans cette phrase nous avons deux dépen-

dances discontinues (non-projectives – les flèches en pointillés) associées à deux relations d'ancrage (au-dessous de la phrase).

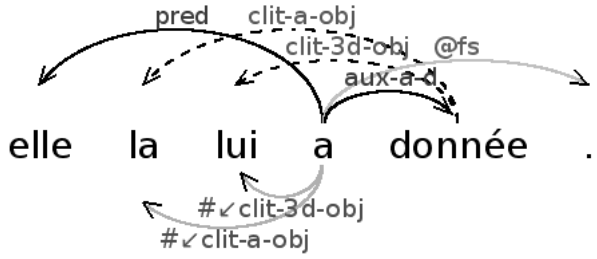


FIG. 3.2 – *Structure de dépendances non-projectives*

La grammaire catégorielle de dépendances qui définit cet arbre de dépendances non projectif affecte les types qui ancrent les clitiques *la* et *lui* sur l'auxiliaire *a*. Aussi cet arbre comporte deux dépendances discontinues étiquetées $clit-a-obj$ et $clit-3d-obj$.

$$
\begin{array}{lcl}
elle^{PN(Lex=pers,C=n)} & \mapsto & [pred] \\
la^{PN(Lex=pn,F=clit,C=a)} & \mapsto & [\ \#(\swarrow\ clit-a-obj)]^{\swarrow clit-a-obj} \\
lui^{PN(Lex=pn,F=clit,P=3,C=d)} & \mapsto & [\ \#(\swarrow\ clit-3d-obj)]^{\swarrow clit-3d-obj} \\
a^{Vaux(Lex=avoir,F=fin)} & \mapsto & [\ \#(\swarrow\ clit-3d-obj)\backslash\#(\swarrow\ clit-a-obj) \\
 & & \backslash pred\backslash S/@fs/aux-a-d] \\
donnée^{V2t(F=pz,C1=a,C2=d|g|l,T=past)} & \mapsto & [aux-a-d]^{\nwarrow clit-3d-obj\nwarrow clit-a-obj} \\
.^{FullStop(Lex=".")} & \mapsto & [@fs]
\end{array}
$$

Le mot *elle* est classé dans la classe des pronoms personnels au cas nominatif (le sujet). Le mot *la* est classé dans une classe qui correspond aux pronoms clitiques au cas accusatif. Le mot *lui* est classé dans la classe des pronoms clitiques à la 3^{me} personne et au cas datif.

Le mot *a* est classé comme un verbe auxiliaire sous une forme finie "F = fin" (à l'opposé des formes infinitives), alors que le mot *donnée* est classé comme un verbe di-transitif (deux compléments). Le premier complément est un complément direct (à l'accusatif) et le deuxième complément est datif, génitif ou locatif. *pz* indique que la forme est au "participe passé".

Définition 3.4 Une grammaire catégorielle de dépendances (CDG) est un système *G=(W,* ***C, V****, S,* λ*)* où *W* est un ensemble fini de mots, **C** un ensemble fini de noms locaux de dépendances contenant *S* (l'axiome), **V** un ensemble de noms de valences et λ appelé le lexique associe à chaque mot de *W* un ensemble fini de types λ(a) $\subset$ ***CAT(C, V)***. Pour une *SD D* et une phrase x, nous notons *G(D, x)* la relation :

"$D = SDx$ (ρ) où ρ est une dérivation de $\lambda \vdash$ S pour $\lambda \in \lambda(x)$". Le langage engendré par *G* est l'ensemble $L(G) =_{df} \{w|\ \exists\ D\ G(D,w)\}$ et le langage de *SD* engendré par G est l'ensemble $\Delta(G) =_{df} \{\ D \mid \exists\ w\ G(D, w)\ \}$. D(CDG) et L(CDG) dénotent les familles de langages de *SD* et de langages engendrés par ces grammaires.

Définition 3.5 *Types itérés* : les noms de dépendances itérés expriment naturellement des dépendances optionnelles répétables. Par exemple, les compléments circonstanciels répétables *circ* de la figure 3.3 sont obtenus par le type *[pred\circ*\S/a-obj]* assigné au verbe `fallait`.

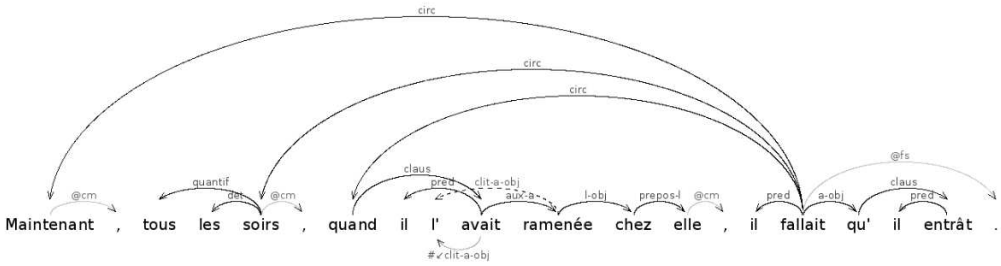


FIG. 3.3 – *Dépendances circonstancielles itérées*

3.2.4 Expressivité des CDG

Les *CDG* sont plus expressives que les grammaires hors-contextes. Cette supériorité des *CDG* vient du fait qu'elles offrent la possibilité de décrire très facilement des dépendances discontinues. Elles peuvent en effet générer certains langages contextuels. Par exemple, elles peuvent générer le langage $\{a^n b^n c^n | n > 0\}$, ce dont les CFG sont incapables. Pour ce langage on peut utiliser la CDG suivante :
$a \mapsto [A^{\swarrow A}], [A \backslash A]^{\swarrow A}$,
$b \mapsto [B/C]^{\nwarrow A}, [A \backslash S/C]^{\nwarrow A}$,
$c \mapsto C, [B \backslash C]$.

La chaîne "aaabbbccc" possède la structure de dépendances (*SD*) dans la figure 3.4.

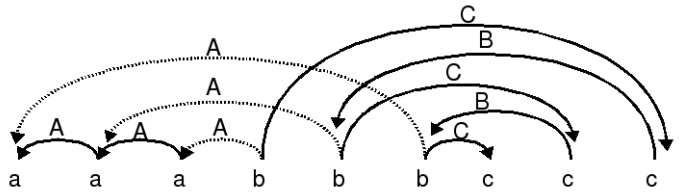


FIG. 3.4 – *La structure de dépendances pour "aaabbbccc"*

Nous pouvons le prouver par l'arbre de dérivation de la figure 3.5. Chaque trait horizontal symbolise l'utilisation de la règle écrite entre parenthèses. Cette règle est appliquée aux éléments se situant au-dessus du trait et génère l'élément se situant sous le trait.

A l'image des hors-contextes, les dépendances projectives ne permettent de générer que des mots de la forme $a^n b^n$. Ce sont donc bien les dépendances polarisées qui rendent possible l'expression de langages plus complexes.

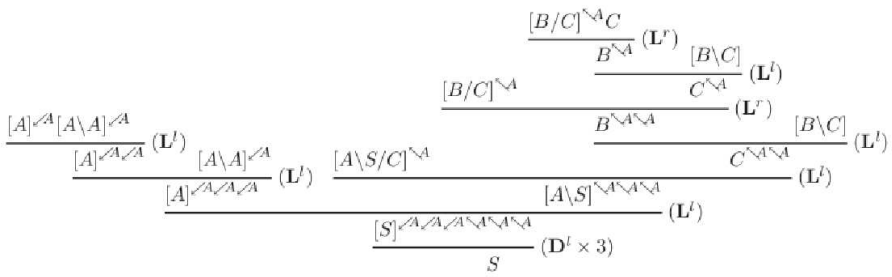


FIG. 3.5 – *Preuve de correction de la structure de dépendances pour "aaabbbccc"*

3.3 L'analyseur des CDG

L'équipe TALN du LINA dispose d'un analyseur syntaxique pour la grammaire catégorielle de dépendances qui est écrit en Common Lisp, PHP et bash. Cet analyseur a été développé au LINA au sein de l'équipe TALN et continue d'être amélioré, en particulier, en ce qui concerne la robustesse et la précision. Récemment a été publiée sa version 3.1 (ci-dessous nous appelons cet analyseur Parser-3.1). Toutes les données d'entrée et de sortie de l'analyseur 3.1 sont des structures *XML*.

L'interface de l'analyseur est un site web. Voici quelques captures d'écrans de l'interface graphique, montrant les différents choix possibles pour l'exécution de l'analyseur.

FIG. 3.6 – *Le menu principal du CDG lab*

La figure 3.6 montre la page d'accueil qui propose d'accéder aux différents choix de l'analyseur.

Après avoir choisi la version, nous pouvons choisir d'utiliser l'analyseur avec différentes grammaires parmi lesquelles plusieurs versions pour le français, le russe, une de l'allemand et une pour le hollandais (cf. figure 3.7).

Ces grammaires ne représentent pas la totalité du langage en question, mais seulement une plus ou moins grande partie. Une fois la grammaire

FIG. 3.7 – *Sélection d'une grammaire*

choisie, nous arrivons sur une page sur laquelle nous pouvons choisir une phrase parmi des exemples, ou en écrire une ou plusieurs dans un champ de texte. On peut aussi choisir un fichier qui contient plusieurs phrases. Nous pouvons aussi choisir un fichier de type XML[2] qui contient les phrases annotées avec des CDG du français. Un certain nombre d'options sont disponibles (cf figure 3.8) : le nombre maximum d'analyses calculées, le temps maximum accordé à l'analyse de chaque phrase, la présentation des résultats.

En validant la phrase, l'analyseur s'exécute et nous fournit un rapport des résultats obtenus. La figure 3.9 représente le résultat de l'analyse de la phrase "elle la lui a donnée." et la figure 3.10 montre le rapport des résultats obtenus pour analyser un ensemble de phrases.

On voit bien dans la figure 3.10 que l'analyseur génère un rapport *HTML* qui comprend diverses statistiques utiles. Il produit aussi une représentation *XML* pour chaque SD avec toutes les informations nécessaires.

L'analyse d'une phrase se déroule en plusieurs étapes, la première est le chargement de la grammaire. Il s'agit d'un chargement paresseux, c'est-à-dire que seuls les mots présents dans le texte sont chargés. Si un mot n'est pas présent dans le lexique, le programme s'arrête en expliquant quel mot manque. Cette partie du programme est écrite en plusieurs langages : du *HTML* pour l'interface, *PHP* pour la recherche dans la grammaire, la grammaire étant soit sous forme *XML*, soit stockée dans une base de données.

[2]Ce mode particulièr prend des phrases avec les classes grammaticales les plus probables pour les unités lexicales en utilisant un étiqueteur morphologique POS.

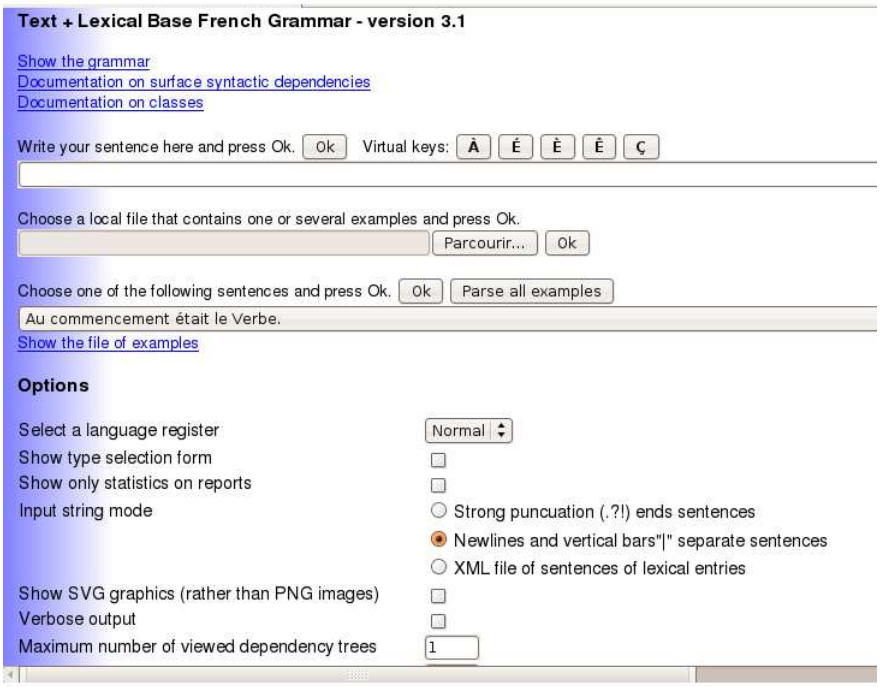


FIG. 3.8 – *Sélection une ou plusieurs phrases.*

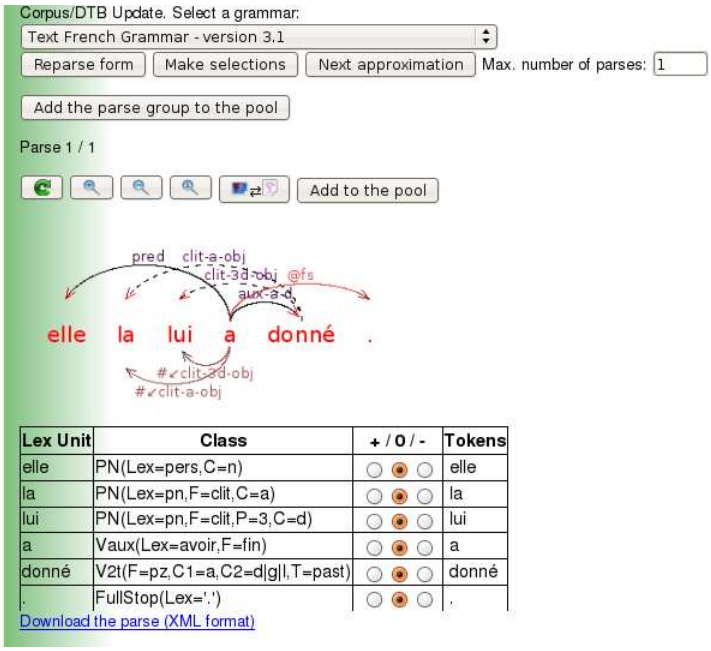
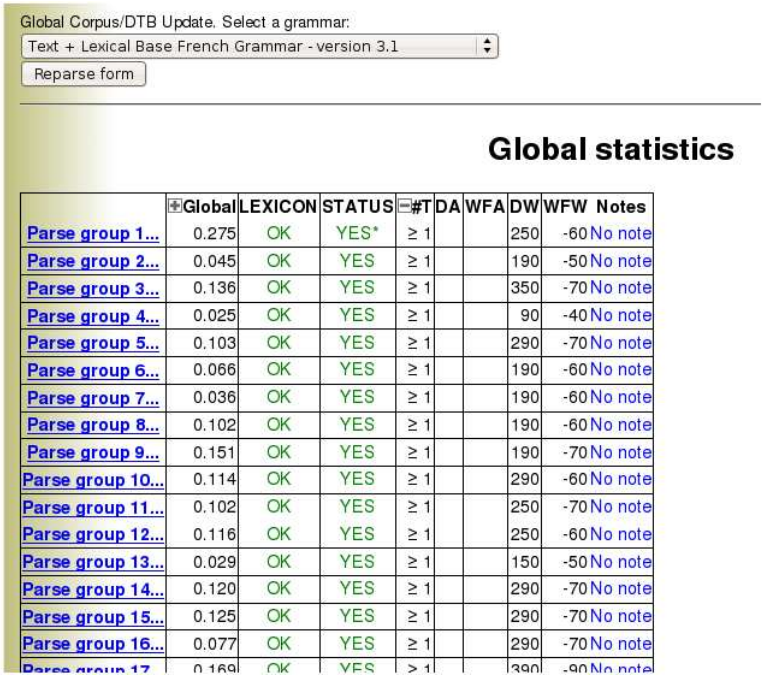


FIG. 3.9 – *Analyse de la phrase : elle la lui a donnée.*

La partie qui s'exécute ensuite est implémentée en Lisp. Elle se charge de découper la phrase en "tokens" : ponctuation ou suite de lettres ou de chiffres. Puis, l'analyseur construit une matrice carrée ayant pour taille le nombre de tokens de la phrase. La diagonale de la matrice est remplie par les règles associées au mot correspondant : la case de coordonnées (i, i) comporte les règles du i-ème mot de la phrase. La partie de la matrice au-


Global Corpus/DTB Update. Select a grammar:
Text + Lexical Base French Grammar - version 3.1
Reparse form

Global statistics

	Global	LEXICON	STATUS	#T	DA	WFA	DW	WFW	Notes
Parse group 1...	0.275	OK	YES*	≥ 1			250	-60	No note
Parse group 2...	0.045	OK	YES	≥ 1			190	-50	No note
Parse group 3...	0.136	OK	YES	≥ 1			350	-70	No note
Parse group 4...	0.025	OK	YES	≥ 1			90	-40	No note
Parse group 5...	0.103	OK	YES	≥ 1			290	-70	No note
Parse group 6...	0.066	OK	YES	≥ 1			190	-60	No note
Parse group 7...	0.036	OK	YES	≥ 1			190	-60	No note
Parse group 8...	0.102	OK	YES	≥ 1			190	-60	No note
Parse group 9...	0.151	OK	YES	≥ 1			190	-70	No note
Parse group 10...	0.114	OK	YES	≥ 1			290	-60	No note
Parse group 11...	0.102	OK	YES	≥ 1			250	-70	No note
Parse group 12...	0.116	OK	YES	≥ 1			250	-60	No note
Parse group 13...	0.029	OK	YES	≥ 1			150	-50	No note
Parse group 14...	0.120	OK	YES	≥ 1			290	-70	No note
Parse group 15...	0.125	OK	YES	≥ 1			290	-70	No note
Parse group 16...	0.077	OK	YES	≥ 1			290	-70	No note
Parse group 17...	0.169	OK	YES	≥ 1			390	-90	No note

FIG. 3.10 – *Statistiques globales*

dessus de la diagonale est ensuite remplie en composant les règles suivant l'algorithme de Cooke-Kasami-Younger, avec les compositions de règles expliquées précédemment. À partir de cette matrice, l'analyseur calcule les dépendances distantes, aussi appelées valences. Pour cette étape, il existe deux modes : `First available` qui lie les valences les plus proches et `First cross` qui les lie de manière à les croiser. Toujours à partir de la matrice remplie, un rapport d'analyse est généré en *HTML*, ainsi que des graphiques représentant les dépendances. Deux types de graphiques sont proposés : un graphique horizontal avec des flèches entre les mots et un graphique vertical sous forme d'arbre (cf. la figure 3.11).

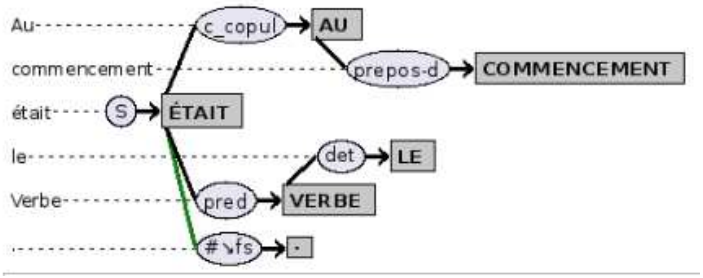


FIG. 3.11 – *Graphique virticle pour la phrase : au commencement était le Verbe.*

3.4 Conclusion

Dans ce chapitre nous avons décrit un formalisme pour les grammaires de dépendances appelé grammaire catégorielle de dépendance (CDG).

Leur particularité réside dans la gestion de dépendances discontinues (non projectives). Elles traitent les dépendances discontinues (les particules négatives, les pronoms comparatifs, etc.) qui posent un problème pour la plupart des grammaires de dépendances. Elles utilisent un principe simple *de valences polarisées* pour les traiter.

Deuxième partie

Ressources lexicales et unités lexicales

4 Le lexique morpho-syntaxique Le*fff* et la CDG du français

Sommaire

4.1 Introduction 57
4.2 Les ressources linguistiques et la CDG du français . . 57
4.2.1 Présentation de Le*fff* et de son architecture 58
4.2.2 Construction du lexique de la base de données de la CDG du français 59
4.2.3 Travaux après l'incorporation de Le*fff* 63
4.3 Conclusion et perspectives 67

Ce chapitre est organisé comme suit. Nous commençons par présenter la motivation de ce travail dans la section 4.1. Ensuite, nous expliquons le processus de la construction du lexique de la CDG du français dans la section 4.2.2. La section 4.2.3 évoque les travaux que nous avons faits après l'incorporation de Le*fff*. Enfin, nous terminons par une conclusion et quelques perspectives dans la section 4.3.

4.1 Introduction

Les besoins spécifiques du TAL ont contribué au développement de ressources lexicales. Ces ressources sont de plus en plus utilisées pour toutes les applications du TAL. Dans notre domaine, nous en avons besoin pour développer une grammaire à large couverture (Dikovsky (2009)).

Ces dernières années, un certain nombre de ressources lexicales syntaxiques pour le français ont été développées depuis le début du traitement automatique des langues en France. Les objectifs de ces lexiques sont de définir le cadre de sous-catégorisation pour chaque lemme[1] verbal donné qui spécifie le nombre et le type de ses arguments et les informations complémentaires qui s'y rapportent. Certaines de ces ressources lexicales ont également été acquises semi-automatiquement à partir de lexiques telles que le *Lexique-Grammaire* (Gross (1975) ; Leclère (2002)), *LGLex* (Tolone (2011)), *LxValf* (Salkoff et Valli (2005)), *DicoValence* (van den Eynde et Mertens (2003)), *SynLex* (Gardent et al. (2008)), et *Lefff* (Sagot et al. (2006)). D'autres ressources ont été obtenues automatiquement à partir de corpus telles que *LexSchem* (Messiant et al. (2008)), *EasyLex* (Gardent (2009)) et *TreeLex* (Kupsc et Abeillé (2008)).

Les ressources acquises à partir de corpus ne sont pas complètes et aussi précises que les ressources construites à la main, mais elles apportent des informations qui ne sont souvent pas disponibles dans les travaux manuels comme par exemple la fréquence des schémas de sous-catégorisation en corpus.

C'est le cas notamment de L*efff* qui n'est pas complet : c'est un lexique des formes fléchies du français constitué en partie par des moyens automatiques (analyse de corpus, fusion de données provenant de différentes ressources) et en partie manuellement, notamment pour la validation des entrées. Il comprend dans sa version actuelle plus de 7000 lemmes verbaux. Le lexique est disponible sous une forme compacte (niveau intensionnel) ou sous une forme éclatée (niveau extensionnel, où chaque entrée est une forme fléchie), disponible sous licence libre LGPL-LR[2].

Ce chapitre a pour objectif de décrire notre travail de complétion du lexique de la CDG du français en utilisant L*efff* et aussi de décrire les processus de construction de grammaires de grande envergure à partir de corpus. En fait, le but ultime de l'équipe est de construire un corpus en dépendances bien annoté.

4.2 Les ressources linguistiques et la CDG du français

Dans les sections suivantes nous envisageons l'utilisation de L*efff* et son incorporation dans la CDG du français. Nous présentons les processus de la construction du lexique de la base de données de la CDG du français en utilisant ce lexique morpho-syntaxique du français. Il s'agit d'une ressource lexicale pour le français L*efff* (LExique des Formes Fléchies du Français, Sagot (2010)). Au niveau morphologique, ce lexique

[1] Le `lemme` (*ou lexie, ou item lexical*) est la forme canonique du mot, le plus souvent au masculin singulier ou à l'infinitif. Il représente un ensemble de mots avec la même racine, la même catégorie lexicale et le même sens.

[2] http ://alpage.inria.fr/~sagot/lefff.html

contient 536375 entrées correspondant à 110477 lemmes distincts couvrant toutes les catégories morpho-syntaxiques. Au niveau syntaxique, 10273 de ces entrées possèdent un cadre de sous-catégorisation. Au niveau extensionnel, la version actuelle du Lefff est Lefff 3.0.3. Nous avons travaillé sur la version Lefff 3.0.1 qui comporte un nombre très important de lemmes et de formes pour toutes les catégories lexicales. Par exemple, on y trouve 361817 formes pour les verbes, 78338 formes pour les noms communs, 34096 formes pour les adjectifs.

La construction de Lefff a été le résultat de diverses techniques d'acquisition suivies par des étapes de validation manuelle. Ces techniques ont été appliquées sur des corpus bruts (cf. Clément et al. (2004) ; Sagot (2005)). L'acquisition automatique (d'entrées morphologiques d'informations syntaxiques) suivie par les étapes de validation manuelle sont décrites dans Sagot (2006).

4.2.1 Présentation de Lefff et de son architecture

Le Lefff repose sur une architecture à deux niveaux : un niveau intensionnel et un niveau extensionnel. Le lexique intensionnel factorise l'information lexicale en associant à chaque lemme une classe morphologique. Cette classe morphologique permet de fléchir un lemme. Le lexique extensionnel, produit automatiquement par compilation du lexique intensionnel, associe à chaque forme fléchie une structure détaillée (morphologiques et syntaxiques).

Chaque entrée dans le lexique intensionnel est généralement définie par un lemme et une catégorie lexicale. Il est possible de trouver une forme fléchie qui correspond à plusieurs lemmes. Par exemple, *suis* correspond aux lemmes *suivre* et *être*. Il est aussi possible de trouver plusieurs entrées avec le même lemme et la même partie du discours. Dans ce cas il y a quelques informations morphologiques et syntaxiques différentes sur ces lemmes. Cela permet de diviser un lemme en différents sens sémantiques impliquant des constructions syntaxiques ou sémantiques différentes. Cette distinction est conservée une fois le lexique compilé grâce à une numérotation (___1, ___2 ou ___3, *etc.*) attachée au lemme qui est différente pour chaque entrée lexicale ayant un sens différent.
On peut dire qu'une entrée intensionnelle contient les informations suivantes :
– une classe morphologique qui permet de fléchir les lemmes ;
– un poids (calculé selon des heuristiques ou renseigné manuellement), le poids standard est de 100. Quelques entrées ont le poids 200, 300, 400 ou 600, comme par exemple l'auxiliaire avoir ;
– une catégorie syntaxique associée à chaque forme fléchie, ces catégories syntaxiques sont divisées en deux types : ouvertes[3] (aussi appelées productives) et fermées[4] (aussi appelées grammaticales) ;
– une structure syntaxique complète, y compris un cadre de sous-catégorisation qui montre explicitement comment le lemme peut être uti-

[3]Une catégorie *ouverte* signifie qu'elle permet d'ajouter de nouvelles formes. Cette classe a la capacité d'être étendue : adverbes, adjectifs, noms, etc.

[4]Une catégorie fermée signifie qu'il n'est pas possible d'ajouter de nouvelles formes : prépositions, pronoms, conjonctions, etc.

lisé dans une construction syntaxique particulière. Voici un exemple d'une entrée simple du L*efff* :
aime 100 *v* [*pred='aimer___1<Suj :cln|sn, Obj:cla|sn, Att:sa|sn>', @AttObj, @pers, cat=v, @PS1*3*s*] %*actif*

Cette information syntaxique sur le cadre de sous-catégorisation sera très utile pour notre travail de construction du lexique de la base de données de la CDG du français. Les fonctions syntaxiques sont définies dans le L*efff* par des critères proches de ceux de *DicoValence*[5] (Poibeau et Messiant (2006)).
La liste des fonctions syntaxiques ci-dessous, ainsi que leurs critères définitoires décrits dans Sagot et Fort (2007) et Sagot (2010) sont les suivants :
Suj : fonction *sujet*. La forme clitique[6] est celle d'un clitique nominatif personnel ;
Obj : fonction *objet direct*. La forme clitique est celle d'un clitique accusatif, substituable par *ceci/cela*, translaté par passivation lorsque c'est possible ;
Objà : fonction *objet indirect* introduit canoniquement par la préposition *à*, ou fonction à-objet : substituable par un syntagme prépositionnel de la forme *à +pronom non-clitique* mais pas par *ici* ou *là-bas*. La cliticisation est possible à l'aide du clitique datif ;
Objde : fonction *objet indirect* introduit canoniquement par la préposition *de*. La cliticisation est possible à l'aide du clitique génitif *en*. Elle se distingue de la fonction délocative Dloc par la non-substituabilité avec les locutions pronominales *de là*, *d'ici*.
Loc : fonction *argument locatif*, les pronoms *ici* ou *là-bas* sont substituables. Il est possible de cliticiser à l'aide du clitique locatif *y* (Adam va `à Paris`);
Dloc : fonction *argument délocatif*, les locutions pronominales *d'ici* ou *de là* sont substituable. Il est possible de cliticiser à l'aide du clitique génitif *en* (Adam vient `de Paris`);
Att : fonction *attribut* (du sujet, de l'objet ou de l'à-object) et pseudo-objet (J'ai vendu ceci `3 euros`);
Obl et **Obl2** pour les autres arguments obliques (non-cliticisables) ; Obl2 ne s'utilise que lorsque les verbes ont deux compléments obliques, tels que *plaider auprès de quelqu'un en faveur de quelqu'un d'autre*.

4.2.2 Construction du lexique de la base de données de la CDG du français

Nous avons une CDG du français qui évolue au niveau des classes et des types associés. On l'appelle la grammaire *texte* : c'est un dictionnaire (lexique de base de la CDG du français) qui contient actuellement 176 classes lexicales et environ 1500 formes. Elle ne contient donc que très peu

[5]Le dictionnaire de valence *DicoValence* est une ressource informatique qui répertorie les cadres de valence de plus de 3700 verbes simples du français. Par cadre de valence on entend traditionnellement le nombre et la nature des compléments valenciels du verbe, y compris le sujet, avec mention de leur fonction syntaxique.

[6]Un pronom est dit «clitique» lorsqu'il ne peut pas être séparé du verbe auquel il se rattache. Il est à la fois libre morphologiquement et dépendant du point de vue syntaxique. C'est le cas des pronoms personnels sujets et compléments dits faibles ou atones : je, tu , il, elle, on, nous, vous, ils, elles, me, te, se, le, la, les, lui, leur, en, y. (Ces clitiques s'opposent aux formes disjointes dites toniques : moi, toi, lui, elle(s), eux, soi).

de formes et ces formes ne portent pas de traits. Nous devons trouver des méthodes pour compléter automatiquement cette grammaire. Comme nous l'avons mentionné dans l'introduction de ce chapitre, nous avons utilisé le lexique Lefff pour compléter le lexique de la CDG du français.

Bien que le Lefff contienne une information très riche et importante, son format actuel ne permet pas un usage facile dans les applications du TAL. La méthode de conversion de ce lexique consiste à l'enregistrer dans une base de données *postgresql*[7] sans introduire de changement à Lefff. Le nom de cette base de données dans le CDG lab du LINA est `DB_LEFFF`.

Dans cette base de données, nous avons construit cinq tables différentes qui sont reliées entre elles. Il y a 590654 entités lexicales dans la table principale `lexicon`. En fait, cette table comporte 18 colonnes sur des traits morphologiques comme *form, cat, pred, gender, number, person, mood, tense, lemma, etc*. Chaque *form* a une partie du discours *cat* qui est la colonne la plus importante pour connaître la catégorie lexicale d'une unité lexicale. La colonne *pred* est aussi importante, parce qu'elle contient l'information sur les cadres de sous-catégorisation du lemme.

Dans Lefff, lorsque plusieurs codes de même nature se suivent, cela signifie que la forme est commune aux valeurs en question.
L'exemple suivant montre comment nous traitons ces codes ou plus précisément cette ligne.

Exemple 4.1 *aime* 100 *v* [*pred='aimer___1<Suj :cln|sn, Obj:cla|sn, Att:sa|sn>', @AttObj, @pers, cat=v, @PS13s*] *%actif*

Le code *PS13s* représente le présent de l'indicatif ou du subjonctif, à la 1^{re} ou à la 3^{e} personne du singulier. Dans ce cas, nous avons séparé cette ligne en quatre lignes (Ps1, Ps3, Ss1 et Ss3) dans la base de données. Cela explique que la taille de la table `lexicon` est plus grande que le lexique Lefff. Le tableau 4.1 représente le code indiquant le mode et le temps dans Lefff.

Code	Mode	Temps
P	indicatif	présent
F	indicatif	futur
I	indicatif	imparfait
J	indicatif	passé-simple
C	conditionnel	présent
Y	impératif	présent
S	subjonctif	présent
T	subjonctif	imparfait
K	participe	passé
G	participe	présent
W	infinitif	présent

Tableau 4.1 – *Représentation du code indiquant le mode et le temps dans Lefff.*

[7]PostgreSQL est un système de gestion de bases de données relationnelles des plus intéressants, bien qu'il ne soit pas le plus populaire. Dans sa version 8.x il vient avec un client graphique, pgAdmin III, qui permet d'effectuer les opérations d'administration les plus courantes.

Le tableau 4.2 représente *la personne, le nombre* et *le genre* d'une forme fléchie dans L*efff*.

Code	Signification
1	1^{re} personne
2	2^{me} personne
3	3^{me} personne
m	genre masculin
f	genre féminin
s	singulier
p	pluriel

Tableau 4.2 – *Représentation le code indiquant gendre nombre*

La deuxième table est `class_type` qui contient les classes et les types au format *texte* et les types au format *xml* de la grammaire. Cette table donne, pour chaque classe, la liste des types associés. Chaque *form* de la première table `lexicon` peut correspondre à une ou plusieurs classes. Nous avons rempli cette table à partir de la grammaire *texte* CDG du français. La troisième table est `lexicon_oid_class` qui contient trois colonnes *reference*, *class*, *origin*[8]. Nous l'avons remplie à partir de la table `lexicon`. Cette table est reliée avec la table `lexicon` en utilisant la colonne *oid*[9] de la table `lexicon` qui est créée de manière automatique par *Postgresql* (cf. Figure 4.3). Nous avons relié cette colonne *oid* avec la colonne *reference* de la table `lexicon_oid_class`. La figure 4.1 illustre le processus de création de l'état initial de la base de données.

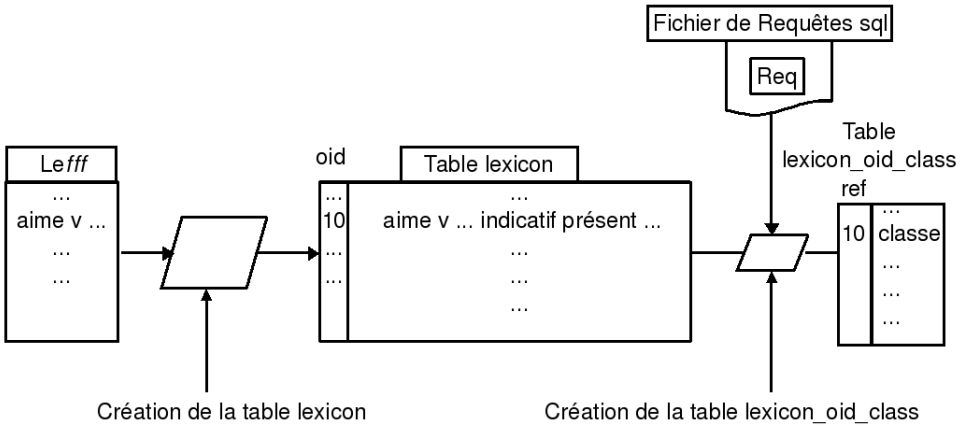


FIG. 4.1 – *Le processus de création de l'état initial de la base de données*

La méthode que nous avons utilisée pour faciliter l'extraction de l'information et pour remplir la table `lexicon_oid_class` est d'utiliser des expressions rationnelles. Soit $\mathcal{R}$ une requête. Chaque requête

[8]Cette colonne a pour objectif de décrire d'où provient l'unité lexicale. Nous avons deux possibilités : elle provient de L*efff* "ref-lefff" ou de la grammaire *texte* "ref-txt".

[9]Les identifiants d'objets (OID) sont utilisés en interne par *PostgreSQL* comme clés primaires de différentes tables système. De plus, une colonne système OID est ajoutée aux tables créées par les utilisateurs (sauf si WITHOUT OIDS est indiqué à la création de la table).

se compose de certaines expressions rationnelles $\alpha_1, \cdots, \alpha_n$. Une requête ($\mathcal{R} \equiv \alpha_1, \cdots, \alpha_n$) pour chaque classe permet d'associer toutes les unités lexicales à la classe lexicale correspondante. Le tableau 4.3 présente certains opérateurs de correspondance des expressions rationnelles que nous avons utilisés pour les requêtes.

Opérateur	Description
~	correspond à l'expression régulière
∧	correspond au début de la chaîne
$	correspond à la fin de la chaîne
[...]	correspond à un caractère dans la liste entre crochets
[^...]	correspond à tout caractère non compris dans la liste entre crochets
*	une séquence de 0,1 ou plusieurs occurrences

Tableau 4.3 – *Certains opérateurs des expressions rationnelles qui sont utilisés.*

Par exemple pour trouver toutes les formes qui correspondent à la classe di-transitif *V2t(F=inf, C1=a|p, C2=o)*, nous avons écrit la requête suivante :

```
select oid from lexicon where
((Pred ~ '<((Suj:|Obj:|Objà:)([^:]*(Att:)?[^:]*)){3}>'
or Pred ~ '<((Suj:|Obj:|Objde:)([^:]*(Att:)?[^:]*)){3}>'
or Pred ~ '<((Suj:|Obj:|Objde:|Objà:)([^:]*(Att:)?[^:]*
)){4}>' or Pred ~ 'Dloc:'
or Pred ~ '<((Suj:|Obj:|Objà:|Obl:)[^:]*){4}>'
or (obj = 'rester_____2' and mood ~ 'infinitif')
or(Pred ~ '<((Suj:|Obj:|Objde:)[^:]*){3}>')
or Pred ~ '<((Suj:|Obj:|Objà:)[^:]*){3}>'
 or Pred ~ '<((Obj:|Objà:)([^:]*(Att:)?[^:]*)){2}>'
or Pred ~ '<((Obj:|Objde:|Objà:)([^:]*(Att:)?[^:]*)){3}>'
or Pred ~ '<((Obj:|Objà:|Obl:)[^:]*){3}>')
and mood ~ 'infinitif')
```

Cette requête permet d'ajouter à la classe *V2t(F=inf, C1=a|p, C2=o)* toutes les formes infinitives correspondant à cette requête. Cette classe portent deux compléments. Le premier complément est un complément direct (à l'accusatif ou au partitif). Le deuxième complément est oblique.

Pour trouver toutes les formes qui sont des noms communs (nc) que nous voulons ajouter à la classe *N(Lex=common)*, nous nous appuyons sur la colonne *cat* avec le test *cat='nc'*. L'algorithme 4 illustre l'opération de remplissage de cette table à partir de la table `lexicon`.

Les figures 4.2 et 4.3 illustrent la connexion entre les tables dans la base de données.

Les trois tables (`lexicon`, `lexicon_oid_class` et `class_type`) sont reliées. C'est-à-dire que chaque ligne de la table `lexicon` correspond à une ou plusieurs lignes de la table `lexicon_oid_class`. Chaque ligne de la table `lexicon_oid_class` correspond à une ou plusieurs lignes de la table `class_type`.

Algorithme 4 *L'algorithme de remplissage de la table lexicon_oid_class*

Entrées : Fichier contenant toutes les classes CDG avec les requêtes correspondantes.

But : Remplissage de la table *lexicon_oid_class*.

```
/* Req ≡ expression rationnelle R ≡ α1, ··· , αn */
/* no-classes représentent le nombre total des classes de la CDG */
Fonction LoadOid(Req)
    List_oid ← résultats de l'execution de la requête Req
    Retourne List_oid
Fin
/* Début de l'algorithme */
pour i = 1 à no-classes faire
    Classe ← nom_classe(i), Req ← Requête_classe(i)
    pour chaque Résultat ∈ LoadOid(Req) faire
        OID ← Résultat.oid
        Insérer la ligne OID, Classe, "ref-lefff" sur les colonnes reference, class, origin de la table lexicon_oid_class
    fin pour
fin pour
```

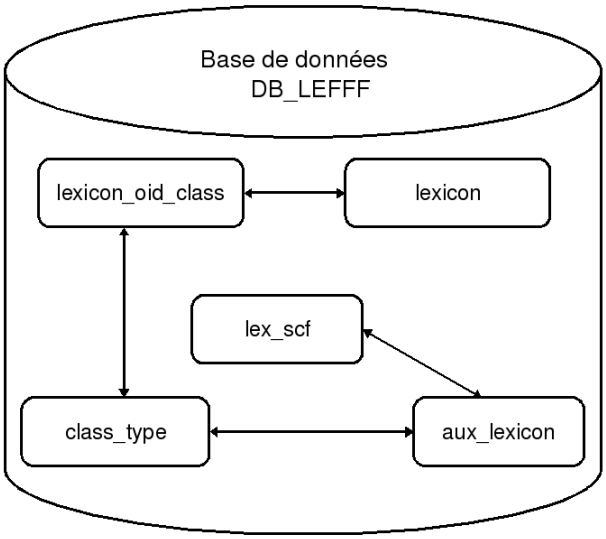


FIG. 4.2 – *Modèle conceptuel de la base de données DB_LEFFF*

4.2.3 Travaux après l'incorporation de Le*fff*

Nous avons évalué la qualité de la classification que nous avons déjà réalisée et avons développé des algorithmes pour son optimisation.
Nous pouvons dire que cette base de données a servi comme outil pour développer le lexique de la base de données de la CDG du français et aussi pour développer un corpus arboré en dépendances (Dikovsky (2011)).
Au cours de ce travail nous avons élaboré des algorithmes pour les grammaires catégorielles de dépendances. Nous avons résolu plusieurs problèmes pour l'analyse de ces grammaires tels que certains échecs de l'ana-

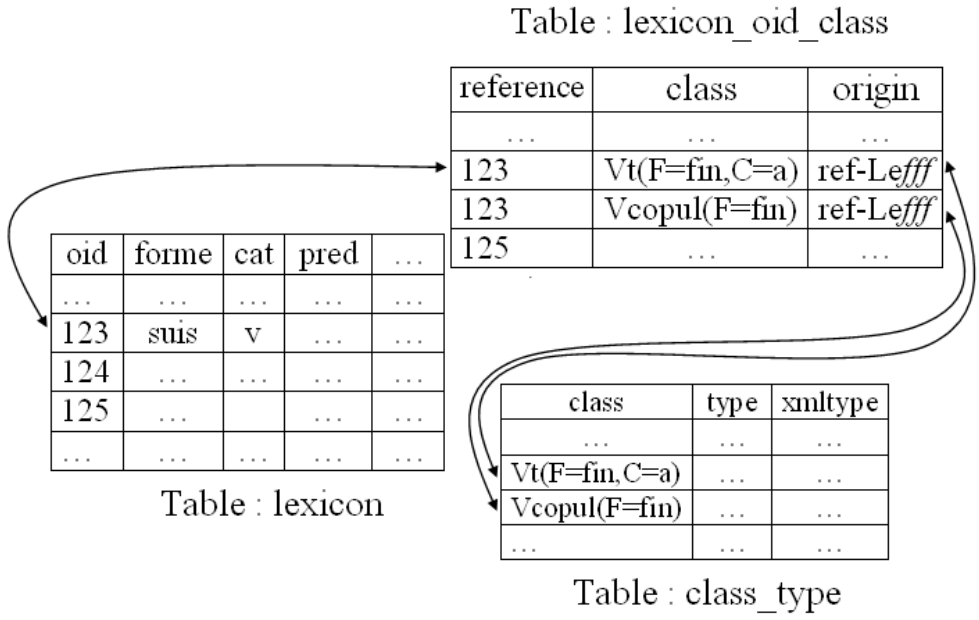


FIG. 4.3 – *Connexion entre les tables*

lyse. Avec la grammaire de la base de données, il y a de fortes chances que l'analyseur produise au moins un résultat.

En revanche, nous sommes tombés sur un problème important : le Lefff n'est pas complet et contient des erreurs. Nous avons été obligé de définir d'autres tables pour essayer de compléter le lexique par certaines unités lexicales. La table `aux_lexicon` contient quelques unités lexicales qui n'existent pas dans Lefff. Cette table a été remplie à partir de différentes méthodes *manuelles*, *semi-automatiques* ou *automatiques* comme c'est expliqué ci-dessous :

- à partir d'informations entrées manuellement, nous avons développé une interface graphique qui permet d'ajouter, de copier, de modifier ou de supprimer les informations.
- à partir de la grammaire *texte* (méthodes semi-automatiques), nous avons utilisé un algorithme complexe.
- de manière automatique, nous avons ajouté à cette table certaines formes complexes avec leurs traits morphologiques (*personne*, *genre* et *nombre*). Ces formes concernent par exemple les verbes suivant : *s'en aller, s'en relever* et *s'en falloir* (*les formes sont : m'en vais, t'en vas, s'en va, ..., m'en relève, t'en relèves, s'en relève, ..., s'en falloir, s'en fallût, s'en faudrait, ..., etc*).

Nous avons aussi écrit un script *PHP* qui permet d'ajouter quelques unités lexicales qui ne se trouvent pas dans Lefff. Par exemple, nous avons ajouté le mode impératif du temps présent à la deuxième personne du singulier (6084 lemmes). Nous avons ajouté ces formes à la table principale `lexicon`.

La dernière table utilisée pour la génération du lexique est *lex_scf*. Cette table se compose de trois colonnes *lemma*, *cat* et *scf*. Nous l'avons définie à partir de la table *aux_lexicon*. Nous ne nous intéressons ici qu'aux verbes. Cette table a pour objectif de compléter la table *lexicon_oid_class* en uti-

lisant les cadres de sous-catégorisation supplémentaires pour les verbes. De manière plus précise, nous prenons un exemple simple dans le tableau 4.4 :

Forme	Cadres de sous-catégorisation	Plus précise
adresser	V2t(C1=a, C2=d) V2t(C1=a, C2=d|g|l)	V2t(C1=a, C2=d)

Tableau 4.4 – *Sélection le cadre de sous-catégorisation le plus précis pour une forme*

Algorithme 5 *L'algorithme remplit la table lexicon_oid_class par* les cadres de sous-catégorisation (SCF)

Fonction LoadInfo()
  Retourne toutes les lignes de la table *lex_scf*
Fin
Fonction sub_cat_fram_more_general(SUB_c_f, $SUB\_c\_f\_bis$)
  Retourner vrai si SUB_c_f moins précis que SUB_c_f_bis
  Retourner faux dans le cas contraire
Fin
Fonction LoadOid(*Lemme*)
  Retourner oid, mood, tense, lemma, active de la table *lexicon* qui correspondent au *Lemme*
Fin
/* Début de l'algorithme */
pour ***chaque*** *Résultat* ∈ LoadInfo() **faire**
  Lemme ← *Résultat.lemme*, $SUB\_c\_f \leftarrow$ *Résultat.SCF*
  pour ***chaque*** *Résultat_bis* ∈ LoadInfo() **faire**
    Lemme_bis ← *Résultat_bis.lemme*, $SUB\_c\_f\_bis \leftarrow$ *Résultat_bis.SCF*
    si (*Lemme*==*Lemme_bis* et $sub\_cat\_fram\_more\_general(SUB\_c\_f, SUB\_c\_f\_bis)$) **alors**
      continuer avec *Résultat* suivant
    finsi
  fin pour
  pour ***chaque*** *Ligne* ∈ LoadOid(*Lemme*) **faire**
    $OID \leftarrow Ligne.oid$
    Classe ← Selon forme, mode, temps, nombre on trouve les classes qui correspondent au SCF.
    $Origin \leftarrow$ "$ref\text{-}txt$"
    Insérer la ligne *OID*, *Classe*, *Origin* sur les colonnes $reference$, *class*, *origin* de la table *lexicon_oid_class*
  fin pour
fin pour

Dans le tableau 4.4, il existe deux cadres de sous catégorisations pour la forme *adresser*. Nous avons utilisé une fonction qui cherche les cadres de sous-catégorisation les plus précis[10] parmi les cadres proposés pour cette forme. Cette fonction retourne *vrai* si le premier cadre de sous-

[10]Il est possible qu'une forme puisse avoir plusieurs cadres de sous-catégorisations.

catégorisation est plus général que le deuxième. Une fois on obtient le cadre de sous-catégorisation le plus précis pour cette forme, on retrouve toutes les classes qui correspondent aux cadres de sous-catégorisation pour cette forme par rapport au tableau 4.5. L'algorithme 5 montre comment nous avons fait pour compléter la table `lexicon_oid_class` par les cadres de sous-catégorisation.

SCF	classes correspondantes (F=fin)
Vt(C=a)	Vt(F=fin, C=a)
Vt(C=d)	Vt(F=fin, C=d)
Vt(C=g)	Vt(F=fin, C=g)
Vt(C=l)	Vt(F=fin, C=l)
Vt(C=o)	Vt(F=fin, C=o)
V2t(C1=a, C2=d)	V2t(F=fin, C1=a, C2=d)
V2t(C1=a, C2=g)	V2t(F=fin, C1=a, C2=g)
V2t(C1=a, C2=inf)	V2t(F=fin, C1=a, C2=inf)
V2t(C1=a, C2=o)	V2t(F=fin, C1=a, C2=o)
V2t(C1=d, C2=g)	V2t(F=fin, C1=d, C2=g)
V2t(C1=d, C2=inf)	V2t(F=fin, C1=d, C2=inf)
SCF	classes correspondantes (F=inf)
Vt(C=a)	Vt(F=inf, C=a)
Vt(C=d)	Vt(F=inf, C !=a)
Vt(C=g)	Vt(F=inf, C !=a)
Vt(C=l)	Vt(F=inf, C !=a)
Vt(C=o)	Vt(F=inf, C !=a)
V2t(C1=a, C2=d)	V2t(F=inf, C1=a|p, C2=d|g|l)
V2t(C1=a, C2=g)	V2t(F=inf, C1=a|p, C2=d|g|l)
V2t(C1=a, C2=inf)	Vt(F=inf, C=a)
V2t(C1=a, C2=o)	V2t(F=inf, C1=a|p, C2=o)
V2t(C1=d, C2=g)	V2t(F=inf, C1=d, C2=g|inf)
V2t(C1=d, C2=inf)	V2t(F=inf, C1=d, C2=g|inf)
SCF	classes correspondantes (F=pres)
Vt(C=a)	Vt(F=pz, C=a, T=pres)
Vt(C=d)	Vt(F=pz, C !=a, T=pres)
Vt(C=g)	Vt(F=pz, C !=a, T=pres)
Vt(C=l)	Vt(F=pz, C !=a, T=pres)
Vt(C=o)	Vt(F=pz, C !=a, T=pres)
V2t(C1=a, C2=d)	V2t(F=pz, C1=-, C2=d|g|l, T=pres)
V2t(C1=a, C2=g)	V2t(F=pz, C1=-, C2=d|g|l, T=pres)
V2t(C1=a, C2=inf)	Vt(F=pz, C1=a, T=pres)
V2t(C1=a, C2=o)	V2t(F=pz, C1=-, C2=o, T=pres)
V2t(C1=d, C2=g)	V2t(F=pz, C1=-, C2=d|g|, T=pres)
V2t(C1=d, C2=inf)	Vt(F=pz, C=a, T=pres)
SCF	classes correspondantes (F=past)
Vt(C=a)	Vt(F=pz, C=a, T=past)
Vt(C=d)	Vt(F=pz, C !=a, T=past)
Vt(C=g)	Vt(F=pz, C !=a, T=past)
Vt(C=l)	Vt(F=pz, C !=a, T=past)
Vt(C=o)	Vt(F=pz, C !=a, T=past)
V2t(C1=a, C2=d)	V2t(F=pz, C1=a, C2=d|g|l, T=past)
V2t(C1=a, C2=g)	V2t(F=pz, C1=a, C2=d|g|l, T=past)
V2t(C1=a, C2=inf)	Vt(F=pz, C1=a, T=past)
V2t(C1=a, C2=o)	V2t(F=pz, C1=a, C2=o, T=past)
V2t(C1=d, C2=g)	V2t(F=pz, C1=d, C2=g|inf, T=past)
V2t(C1=d, C2=inf)	V2t(F=pz, C1=d, C2=g|inf, T=past)

Tableau 4.5 – *Les correspondances entre les classes*

4.3 Conclusion et perspectives

Nous avons décrit dans ce chapitre l'architecture du lexique L*efff* et son incorporation dans la CDG du français. Cette incorporation nous a permis de construire le lexique de base de la CDG du français. Nous avons décrit le processus de cette construction. Nous avons développé certains algorithmes qui nous ont permis d'améliorer le lexique de la base et de corriger les erreurs qui sont fournis par L*efff*. Cette incorporation nous a permis ultérieurement de construire un corpus arboré en dépendances pour le français (Dikovsky (2011)). Nous avons déjà dit que L*efff* n'est pas complet en particulier pour les arguments des noms. Dans le prochain chapitre notre intérêt principal est de calculer les cadres de sous-catégorisation des noms déverbaux à partir de ceux des verbes d'origine pour le français. Nous décrivons un algorithme pour compléter le lexique de la CDG du français en plaçant les déverbaux dans les classes lexicales qui correspondent à leurs cadres de sous-catégorisation.

5 Acquisition automatique des arguments de noms déverbaux

Sommaire

5.1 Introduction 71
5.2 La structure argumentale des déverbaux 71
5.2.1 Recherche des déverbaux 73
5.2.2 Cadres de sous-catégorisation des déverbaux 78
5.2.3 Expériences et évaluation des résultats 81
5.2.4 Construction du corpus de déverbaux 84
5.3 Complétion de la liste des suffixes 85
5.3.1 Conclusion et perspectives 89

Ce chapitre a pour objectif de compléter le lexique de la grammaire du français. La section 5.1 décrit la motivation de ce travail. La section 5.2.2 évoque les cadres de sous-catégorisation des déverbaux. La section 5.2.3 présente les expériences et les évaluations des résultats obtenus. On termine par une conclusion et quelques perspectives dans la section 5.3.1.

5.1 Introduction

Les noms déverbaux sont un domaine de recherche particulièrement intéressant et bien étudié en linguistique. Nous présentons une expérience d'extraction automatique des cadres de sous-catégorisation pour les noms déverbaux à partir de ceux des verbes français. Pour ce faire, nous présentons un algorithme **proto-déverb** qui permet de constituer et d'enrichir semi-automatiquement des arguments des noms déverbaux en utilisant L*fff* comme une ressource référentielle pour filtrer les noms. Cet algorithme nous permet de compléter le lexique de la CDG du français en plaçant les noms déverbaux dans les classes lexicales qui correspondent à leurs cadres de sous-catégorisation. Dans un premier temps nous étudions les cas les plus fréquents du génitif, c'est-à-dire ceux des groupes prépositionnels introduits par *de*, *des*, etc. Nous analysons la provenance des groupes prépositionnels subordonnés des déverbaux dans des contextes différents en tenant compte du verbe d'origine.

Ce chapitre est divisé en deux parties. La première partie (la section 5.2) est un résultat qui a été validé par des experts. Nous allons étudier plus en détails le cas génitif. Puis nous développons un corpus de noms déverbaux. La deuxième partie (la section 5.3) est un résultat quantitatif, c'est-à-dire sans mesure de la précision et de l'exactitude de ces résultats.

5.2 La structure argumentale des déverbaux

Comparons les quatre phrases suivantes :
(1) *Le service de livraison est fermé.*
(2) *Le besoin de reconnaissance est un besoin fondamental de tous les êtres humains.*
(3) *La livraison des commandes est assurée directement par le service de livraison du producteur.*
(4) *On attend l'abrogation définitive de cette loi.*
Nous voyons que dans chaque phrase il y a au moins un groupe prépositionnel (GP) subordonné à un nom : e.g. ***de*** *livraison* est subordonné à *service*, ***des*** *commandes* est subordonné à *livraison*, ***du*** *producteur* est subordonné à une autre occurrence de *livraison*. Parmi ces dépendances entre les noms gouverneurs et les prépositions têtes des GP, lesquelles sont *attributives*, lesquelles sont *dépendances d'objet* ? On ne peut pas effectuer une analyse fine en dépendances sans savoir répondre à cette question. En effet, dans la grammaire de dépendances il faut au moins faire la distinction entre les dépendances obligatoires d'objet et les dépendances facultatives de modifieurs en tout genre. Cette différence entre les deux types de dépendances correspond aussi à une différence fondamentale sémantique : dans le cas de dépendances attributives la sémantique du GP définit la valeur d'un attribut de la sémantique du nom gouverneur, tandis que dans le cas de dépendances d'objet la sémantique du nom gouverneur est une fonction appliquée à la sémantique du GP. Mais comment peut-on savoir dans quel cas de figure on se trouve ?

On peut distinguer deux catégories de substantifs :

(c_1) ceux dont au moins un GP subordonné est un complément d'objet (c'est-à-dire, correspond à un argument sémantique)[1] ,
(c_2) et ceux qui n'en ont aucun.

Les compléments des substantifs de la catégorie c_1 ont le même statut (par rapport au substantif) que celui des éléments du cadre de sous-catégorisation des verbes (par rapport aux verbes). Aussi ces substantifs peuvent avoir des compléments d'objet dans tous les contextes (ils peuvent aussi avoir des GP attributifs). Parmi ces substantifs on trouve les nombres indéterminés exprimant une quantité limitée et indéterminée d'objets, tels *dizaine*, *douzaine*, *trentaine*, etc. On y trouve aussi les noms communs tels *besoin*, *peur*, *marre*, etc. qui avec un verbe léger (e.g. *avoir*) forment un phrasème et qui en héritent les compléments d'objet (au génitif). Mais surtout on y trouve de très nombreux noms déverbaux. On s'intéresse dans ce chapitre surtout aux noms déverbaux parce qu'il existe un certain nombre de règles qui définissent leurs arguments à partir des arguments des verbes d'origine.

Les déverbaux héritent du cadre de sous-catégorisation du verbe d'origine et ils le transforment. E.g. dans les phrases (1)-(4) nous trouvons les déverbaux *reconnaissance*, *livraison*, *service*, *producteur*, *abrogation*. Comme on peut le voir dans les analyses[2] ci-dessous, *service* (catégorie **(c_2)**) a un GP attributif *de livraison*, tandis que *besoin* et *livraison* (catégorie **(c_1)**) ont un GP d'objet *de reconnaissance* dans la phrase (2), *des commandes* dans la phrase (3). Le déverbal *abrogation* a un GP d'objet *de cette loi* dans la phrase (4). En même temps, *besoin* peut aussi avoir un GP attributif, e.g. *de tous les êtres humains* dans la phrase (2).

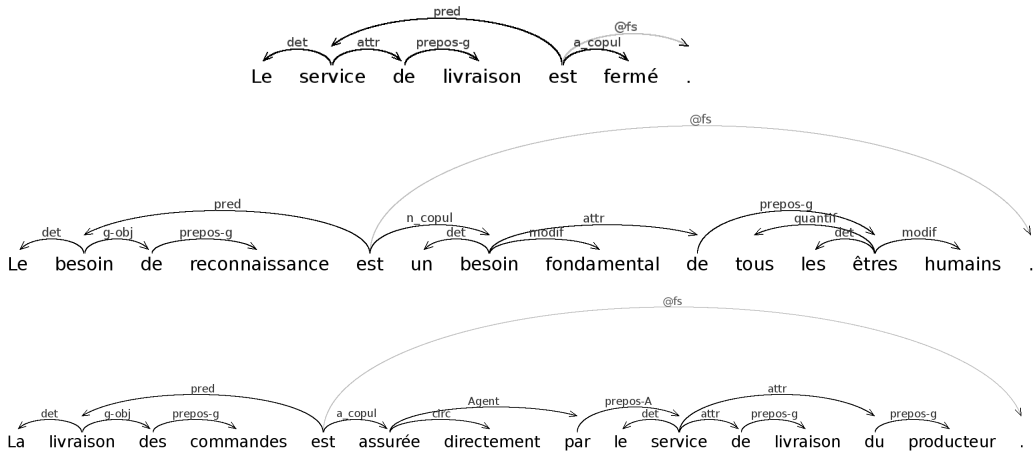


FIG. 5.1 – *Analyses des phrases de l'introduction*

Les nombres indéterminés et les noms composant les phrasèmes ne posent pas de problème parce qu'ils sont peu nombreux. Par contre, la recherche des déverbaux n'est pas une tâche simple (surtout pour les verbes

[1] Comme tous les arguments sémantiques, ces arguments peuvent être liés à des éléments du contexte, ce qui correspond à l'ellipse de surface mais ne les prive pas du statut d'arguments obligatoires.

[2] Ces analyses sont obtenues avec l'analyseur syntaxique et en utilisant la grammaire catégorielle de dépendance du français qui fait partie du système CDG Lab Alfared et al. (2011).

des deuxièmes et troisièmes groupes). Et les difficultés ne s'arrêtent pas là : le vrai problème est de déterminer les cadres de sous-catégorisation des déverbaux à partir des cadres de sous-catégorisation des verbes. Ce sont les deux problèmes abordés dans ce chapitre. Par rapport au système CDG Lab mentionné ci-dessus, ces problèmes se traduisent par des algorithmes de complétion du lexique de la grammaire du français plaçant les déverbaux dans les classes lexicales qui correspondent à leurs cadres de sous-catégorisation. Cela explique notre intérêt pour cette problématique.

5.2.1 Recherche des déverbaux

En français le nom déverbal est un nom formé à partir d'une racine verbale selon le schéma :

$$\text{BASE+TERMINAISON} \Rightarrow \text{var(BASE)+SUFFIXE},$$

où var(BASE) est une variation de la base et SUFFIXE est un suffixe formateur (parfois vide : -0). Le plus simple est d'obtenir la liste plus ou moins complète des suffixes formateurs cf. Namer (2009). En ce qui concerne les variations de la base, il s'avère difficile de trouver une source où elles sont toutes présentées systématiquement. Dans le premier temps, nous utilisons une liste de suffixes pour les verbes du premier groupe, mais elle est limitée à certaines règles. Nous avons appliqué un algorithme simpliste (appelé ci-dessous **proto-déverb**) qui effectue la transformation :

$$\text{BASE+TERMINAISON} \Rightarrow \text{BASE+SUFFIXE}$$

selon 17 règles citées dans le tableau 5.3. Nous avons appliqué **proto-déverb** à deux corpus : un d'arbres syntaxiques : le corpus Paris 7 Abeillé et al. (2003)[3], un autre textuel : le corpus GEOPO (du CLLE-ERSS Ho-Dac (2007)). **Proto-déverb** pose plusieurs problèmes :

Problème de faux déverbaux. L'algorithme **proto-déverb** peut construire des combinaisons BASE+SUFFIXE qui ne sont pas des déverbaux. Par exemple, pour le suffixe *-ence* : *scier* $\Rightarrow$ *science*, pour le suffixe *-ation* : *roter* $\Rightarrow$ *rotation*.

Problème de fausse origine. La combinaison BASE+SUFFIXE construit par **proto-déverb** peut être un déverbal, mais provenir d'une autre base. Par exemple, pour le suffixe *-ant* : *garer* $\Rightarrow$ *garant*, tandis que *garant* est formé du verbe *garantir*.

Problème de fausse direction. Parfois le nom construit par **proto-déverb** n'est pas un déverbal, mais c'est le verbe d'origine qui est dérivé du nom. Par exemple, pour le suffixe *-euse* : *mitrailler* $\Rightarrow$ *mitrailleuse*.

Présentation de l'algorithme proto-déverb

Nous décrivons l'algorithme 6 qui se compose de deux étapes : la première étape consiste à chercher tous les verbes du premier groupe qui se trouvent dans la base de données (dans la table `lexicon`). Ensuite on enlève la terminaison "er" en la remplaçant par un suffixe de la liste des suffixes. Puis on met le résultat dans la variable *deverbal*. Ensuite on cherche si *deverbal* se trouve dans la base de données ou pas. Une fois cette variable trouvée, nous cherchons tous les traits morphologiques correspondants.

[3]Parmi les 17 règles seulement 12 du tableau 5.2 ont été appliquées au moins une fois dans ce corpus.

La deuxième étape consiste à trouver l'occurrence de *deverbal* dans le corpus Paris 7 en utilisant le patron suivant : `//NP/PP/w[@cat='P' and @lemma='de']/parent::PP/preceding-sibling::w[@cat='N' and texte()='$deverbal']`. Cela veut dire qu'on cherche les séquences constituées d'un nom commun suivi d'une des formes de la préposition du lemme *de*.

Algorithme 6 *L'algorithme* **protodéverbe**

Entrées : Extraction les noms déverbaux en utilisant l'algorithme **protodéverb**.

But : Filtrage de ces noms en utilisant le corpus Paris 7

Fonction LoadVerbe(*verbe*)
  Retourner tous les verbes qui se terminent par "er" dans la base de données ;
Fin
Fonction LoadNoun(*deverbal*)
  Retourner les informations correspondantes (traits morphologiques) au *deverbal*
Fin
Fonction LoadDeverbal(*deverbal*)
/* //NP/PP/w[@cat='P' and @lemma='de']/parent : :PP/preceding-sibling : :w[@cat='N' and texte()='*deverbal*'] */
  $Info \leftarrow$ chercher *deverbal*, *phrase*, *no-phrase*, $nom\text{-}fichier$ dans le fichier de Paris 7
  Retourner $Info$
Fin
/* Début de l'algorithme **protodéverbe** */
/* *n* représente la liste des suffixes du tableau 5.3*/
pour ***chaque*** *Lignes* $\in$ LoadVerbe("er") **faire**
  Verbe $\leftarrow$ *Lignes.verbe*
  pour *i*=1 à *n* **faire**
    Enlever la terminaison "er" de la variable *Verbe* et la remplaçant par le suffixe(*i*) de la liste et mettre le résultat dans la variable *deverbal*
    pour ***chaque*** *résultat* $\in$ LoadNoun(*deverbal*) **faire**
      Insérer *résultat* dans la table *derived_noun*
      /*Nous mettons à jour la table *derived_noun*/
      /*$no\text{-}fichier$ représente le nombre de fichiers utilisés (Paris 7)*/
      pour *i*=1 **à** $no\text{-}fichier$ **faire**
        pour ***chaque*** $Info$ $\in$ LoadDeverbal(*deverbal*) **faire**
          Insérer $Info$ dans la table *derived_deverbal*
          /*Nous mettons à jour la table *derived_deverbal*
        fin pour
      fin pour
    fin pour
  fin pour
fin pour

5.2.1.1 *Évaluation de l'algorithme* proto-déverb

On voit que l'application de l'algorithme **proto-déverb** nécessite une expertise du résultat de son application. Notre démarche d'évaluation consiste ici à réaliser deux expérimentations indépendantes ainsi qu'une comparaison directe avec la liste Verbaction de déverbaux dénotant l'action ou l'activité exprimée par le verbe. Nous avons stocké la liste Verbaction sur la base de données pour faciliter les comparaisons, mais sans rien changer à cette liste. La première expérimentation compare la liste des déverbaux produit par **proto-déverb** avec une liste produite par un expert depuis une partie des textes du corpus GEOPO, Ho-Dac (2007). La seconde expérimentation, effectuée sur le corpus arboré de Paris 7 Abeillé et al. (2003) (cf. l'algorithme 6), qui sera longuement présentée et commentée dans la section 5.2.3 consacré aux GP des déverbaux, permet d'obtenir, en la restreignant à la notion d'être ou non un déverbal, une évaluation des déverbaux proposés par l'algorithme **proto-déverb**. Finalement, nous avons utilisé la liste des déverbaux de Verbaction Hathout et al. (2002); Tanguy et Hathout (2002) développée à l'INaLF/ATILF et à l'ERSS pour la comparer avec la liste produite par **proto-déverb**.

5.2.1.2 *Évaluation sur des textes du corpus GEOPO*

La première expérience vérifie la pertinence de l'algorithme **proto-déverb** pour la construction des couples déverbal/verbe sur des textes du corpus GEOPO. Ce corpus est constitué d'environ 270 000 mots. Il regroupe 32 textes longs qui sont des articles expositifs (informatifs et argumentatifs) proposant des réflexions relatives à notre monde d'aujourd'hui (la crise pétrolière, la guerre contre "l'axe du mal", le terrorisme, l'explosion chinoise, la paix au moyen-orient, etc). La méthodologie utilisée pour l'évaluation consiste à repérer automatiquement les séquences constituées d'un nom commun[4] suivi d'une des formes de la préposition *de* (*de*, *d'*, *du* ou *des*). Ensuite, les séquences repérées sont annotées à la fois par l'algorithme **proto-déverb** et par un expert. L'annotation de **proto-déverb** indique pour un nom commun s'il est un déverbal. Dans ce cas l'algorithme fournit le verbe [5] dont il est dérivé. L'expert indique de même si les séquences repérées correspondent à un déverbal (en particulier, le mot doit être un nom commun dans la phrase). Dans ce cas, le verbe d'origine est fourni par l'expert. Dans le cas contraire, l'annotateur indique la raison pour laquelle le mot n'est pas un déverbal. Cela peut être :

1. le mot n'est pas un nom commun dans le contexte, comme par exemple *parler* dans *plutôt que de parler de centralisation, [...]*,
2. le nom fait partie d'une locution comme *point* dans *ainsi, d'un point de vue législatif [...]*,
3. le nom ne correspond pas (dans le sens de la phrase) à un déverbal comme *priorité* dans *Il s'agit au contraire d'une priorité des pouvoirs publics*,

[4]En fait, il s'agit des mots qui sont des noms communs dans le lexique Le*fff*, Sagot (2010).

[5]Exceptionnellement, l'algorithme peut fournir plusieurs verbes correspondant au mot. Par exemple, *parade* correspond à *parer* et à *parader*.

4. l'expert annotateur n'est pas sûr que le nom soit un déverbal comme le nom *poids* pour le verbe *peser*.

L'algorithme **proto-déverb** ne pouvant pas détecter les situations 1 et 2, nous avons éliminé les séquences correspondantes des statistiques. Les cas 4 étant incertains, nous avons aussi préféré les supprimer. Nous n'avons donc gardé que les séquences du point 3 et celles qui ont été annotées par les experts comme des déverbaux.

Sur les 286 séquences de phrases traitées, 220 ont été retenues par les experts (14 de statut incertain (point 4) et 52 correspondant aux points 1 ou 2).

	Déverbal (proto-déverb)	Non déverbal (proto-déverb)
Déverbal (expert)	28 (12,7%)	63 (28,6%)
Non déverbal (expert)	4 (1,8%)	125 (56,8%)

Tableau 5.1 – *Comparaison sur GEOPO entre* **proto-déverb** *et les experts annotateurs*

Ces résultats suggèrent que les règles utilisées par l'algorithme **proto-déverb** donnent pour la grande majorité des occurrences de vrais déverbaux (28 contre 4) dans ce type de corpus. Par contre, la couverture des règles pourrait être améliorée. En fait, l'algorithme simpliste **proto-déverb** se contente de traiter les verbes du premier groupe. De plus, certaines règles trop peu précises n'ont pas été incluses dans cette version de l'algorithme comme la règle qui supprime le *r* final BASE+er $\Rightarrow$ BASE+e et qui permet par exemple de générer le couple *contrôle/contrôler*. L'ajout de ce type de règles aurait conduit à une production d'un trop grand nombre de faux couples déverbaux/verbes qui aurait complexifié la tâche des experts sur le corpus Paris 7.

5.2.1.3 *Évaluation sur le corpus arboré Paris 7*

Pour compléter ces statistiques, nous pouvons aussi utiliser une partie des résultats de la seconde expérience pour juger de la qualité de l'algorithme **proto-déverb**. Lors de cette seconde expérience nous avons repéré, dans le corpus d'arbres syntaxiques de Paris 7, les constructions où un nom commun provenant d'une règle de l'algorithme **proto-déverb** possède un groupe prépositionnel (GP) introduit par la préposition *de*. Sur ces éléments, nous avons demandé à des experts annotateurs de repérer les séquences qui contiennent effectivement le déverbal du verbe donné par l'algorithme **proto-déverb**. Nous verrons dans la première partie du chapitre sur les compléments au génitif des déverbaux que, dans le cas où l'expert pense que le nom est un déverbal, il doit fournir l'éventuel lien entre le GP et un des arguments du verbe d'origine. Par exemple dans la phrase *Il ne faut pas secondariser l'enseignement supérieur ni inciter à un démembrement des universités*, le GP *des universités* du déverbal *démembrement* correspond au complément à l'accusatif (complément d'objet direct) du verbe *démembrer*. En faisant abstraction de ce résultat sur le GP pour l'instant, nous pouvons savoir si la proposition de l'algorithme **proto-déverb**

correspond bien à un déverbal dans le contexte d'une phrase. Les résultats en fonction des règles de l'algorithme **proto-déverb** sont les suivants :

Règle	Déverbal (expert)	Non déverbal (expert)	Nombre de couples
er ⇒ ade	2	1	3
er ⇒ age	63	10	73
er ⇒ aison	5	0	5
er ⇒ ant	26	5	31
er ⇒ ation	274	7	281
er ⇒ ement	132	6	138
er ⇒ ence	24	7	31
er ⇒ eur	45	17	62
er ⇒ euse	1	0	1
er ⇒ oir	1	4	5
er ⇒ rice	1	1	2
er ⇒ ure	10	4	14
Total	584 (90,4%)	62 (9,6%)	646

Tableau 5.2 – *Évaluation de* **proto-déverb** *sur le corpus Paris 7 en fonction des règles*

Ce tableau indique pour chaque règle utilisée pour l'analyse du corpus le nombre de couples nom/verbe qui correspondent ou non à un couple déverbal/verbe pour l'expert ainsi que le total. Globalement, le résultat est assez bon mais n'est pas uniforme en fonction de la règle de passage du verbe au nom. La précision globale est de 90,4%.

5.2.1.4 *Comparaison avec la liste de déverbaux Verbaction*

Une troisième expérience consiste à comparer la liste de 5537 couples déverbal/verbe produits par l'algorithme **proto-déverb** à partir du lexique Le*fff* avec les 8432 couples du lexique Verbaction qui comprend une liste portant sur des noms morphologiquement apparentés au verbe et qui peuvent être utilisés pour dénoter l'action ou l'activité exprimée par le verbe. Pour cette comparaison nous avons trois possibilités suivant qu'un couple nom/verbe correspond à une ou aux deux listes. Le tableau 5.3 présente les résultats pour les verbes du premier groupe qui sont produits par l'algorithme **proto-déverb**.

Le nombre de couples avec un verbe du premier groupe du lexique Verbaction qui ne sont pas produits par l'algorithme est de 5623 (sur 8432). En fait, comme nous l'avons déjà vu précédemment, l'algorithme **proto-déverb** ne couvre pas la totalité des verbes du premier groupe. Par contre, il fournit des couples qui ne sont pas dans Verbaction comme les noms de suffixe *ant*, *ante*, *ette*, *eur*[6], etc qui ne désignent pas directement l'action ou l'activité exprimée par le verbe d'où l'intérêt de cet algorithme dans cette étude.

[6]Les nom déverbaux en -eur/-euse sont habituellement décrits comme agentifs. Plus précisément, la totalité de ces suffixés désignent des agents, en particulier des êtres animés humains qui font l'action décrite par la base verbale.

Règle	Déverbal (Verbaction)	Non déverbal (Verbaction)	Nombre de couples
er ⇒ ade	36	33	69
er ⇒ age	994	60	1054
er ⇒ aison	22	15	37
er ⇒ ant		312	312
er ⇒ ante	1	127	128
er ⇒ ation	918	85	1003
er ⇒ ement	778	61	839
er ⇒ ence	19	18	37
er ⇒ ette	6	187	193
er ⇒ eur	1	938	939
er ⇒ euse		538	538
er ⇒ oir		145	145
er ⇒ oire		20	20
er ⇒ rice		17	17
er ⇒ trice		5	5
er ⇒ ure	33	166	199
er ⇒ xion	1	1	2
Total	2809 (50,7%)	2728 (49,2%)	5537

Tableau 5.3 – *Comparaisons entre* **proto-déverb** *et le lexique Verbaction en fonction des règles*

Pour être complet, nous avons aussi comparé les annotations de nos experts sur le corpus GEOPO avec la liste de Verbaction ce qui a donné les résultats du tableau 5.4. L'adéquation entre nos experts et le lexique Verbaction est assez bonne (85,5%) mais il reste quelques différences comme les couples *structure/structurer*, *position/positionner*, *craintes/craindre* ou *résistant/résister* qui s'explique par le choix de Verbaction de ne s'intéresser qu'aux déverbaux dénotant l'action ou l'activité exprimée par le verbe.

En conclusion, l'algorithme **proto-déverb** est simple et assez précis. Il fournit un grand nombre de déverbaux ayant des liens variés avec le verbe (contrairement à Verbaction qui est spécialisé).

	Déverbal (Verbaction)	Non déverbal (Verbaction)
Déverbal (expert)	69 (31,4%)	22 (10%)
Non déverbal (expert)	10 (4,5%)	119 (54,1%)

Tableau 5.4 – *Comparaison sur GEOPO entre le lexique Verbaction et les experts annotateurs*

5.2.2 Cadres de sous-catégorisation des déverbaux

Passons maintenant au problème principal, celui de déterminer les cadres de sous-catégorisation des déverbaux à partir des cadres de sous-catégorisation des verbes d'origine. Avant tout il faut expliquer comment

nous représentons ces cadres. Notre représentation est basée sur une affectation de cas aux compléments d'objet proposée dans Dikovsky (2011). Cette affectation est faite en fonction des clitiques selon la règle suivante (simplifiée ici) :
Soit π un complément d'objet d'un verbe V dans une phrase. Alors π est au cas C si dans la phrase il peut être pronominalisé et transformé en un clitique P au cas C ancré sur V. Enfin, si π ne peut pas être pronominalisé, alors il est au cas *oblique* (o)[7].

En français il n'y a que quatre cas de clitiques : *accusatif* (a), e.g. *me, la, en*, *génitif* (g) : *en*, *datif* (d), e.g. *te, lui, y*, et *locatif* (l) : *y*. Respectivement, selon cette règle chaque complément d'un verbe a un des cinq cas : a, g, d, l ou o[8]. Aussi le cadre de sous-catégorisation du verbe est défini par les fonctions syntaxiques de ses arguments obligatoires (*actants*) : *sujet* (*subj*), *objet direct* (*od*), *objet indirect* (*oi*)), réalisés par les cas qui leur correspondent, e.g. *chanter*($subj/n$) (intransitif), ou *chanter*($subj/n, od/a$), *donner*($subj/n, od/a, oi/d$), *émerger*($subj/n, oi/g$), *parler*($subj/n, od/a$) ou *parler*($subj/n, oi1/g, oi2/d$), *placer*($subj/n, od/a, oi/l$), *échanger*($subj/n, od/a, oi/o$), etc. Parfois, pour préciser la réalisation du complément direct par une phrase dont la tête est un verbe à l'infinitif, nous utilisons l'étiquette inf : *faire*($subj/n, od/a, oi/inf$). Le cadre s'étend sur les GP en position d'argument circonstanciel : $circ/C$, quand C est le cas du GP.

L'idée principale de notre calcul des cadres de sous-catégorisation des noms déverbaux est que dans le cas où on sait de quel verbe est formé le nom déverbal, on peut déduire le cadre du second à partir de celui du premier. Il s'agit donc de définir les règles de transformation des cadres[9] des verbes en les cadres des noms dérivés. Ce calcul des cadres par réduction rend possible de contourner le problème difficile de définition des cadres de sous-catégorisation des noms (cf. Fillmore (1968, 1977); Grimshaw (1990); Dowty (1991); Van Valin (1997); Mel'čuk (2004)). Les cadres de sous-catégorisation des verbes sont alors donnés a priori et sont représentés par les listes des réalisations des actants : fonction syntaxique/cas définies ci-dessus : $subj/n$, od/a, oi/d, oi/g,oi/l,oi/o, oi/inf. Les cadres des noms déverbaux sont représentés par les réalisations des objets (sauf od/a), mais aussi des attributs : $attr/C$. Quand un argument circonstanciel d'un verbe ou un argument attributif d'un nom n'est pas réalisé par un cas nous allons noter cette réalisation par $circ/\perp$ ($attr/\perp$). Sur cette signature étendue on peut définir les transformations diathétiques de nominalisation des verbes par les séquences de règles du genre ($subj/n \rightarrow 0$)[10] ou (($od/a \rightarrow oi/g$), ($subj/n \rightarrow attr/g$)) ou ($oi/l \rightarrow oi/l$), etc.

Notre hypothèse initiale était que les transformations diathétiques de nominalisation en français sont définies par quatre règles simples :
- le sujet du verbe disparaît et est converti en un attribut du nom dérivé : ($subj/n \rightarrow attr/C$),

[7] Certes, il faut définir plus précisement cette règle pour les clitiques ambigus : *y* au datif vs. *y* au locatif, *en* au génitif vs. *en* à l'accusatif, etc. et la compléter par le cas n (*nominatif*) du sujet.

[8] Cf. (van den Eynde et Mertens (2003)) où on tient compte de tous les pronoms.

[9] Diathetic shift rules (ang.).

[10] $A \rightarrow 0$ veut dire que A est supprimé.

- son objet direct ($C = a$) devient l'objet indirect au génitif ($od/a \rightarrow oi/g$),
- tous les autres objets indirects sont tout simplement hérités sous réserve de leur présence dans le cadre (e.g ($oi/d \rightarrow oi/d$) ou ($oi/o \rightarrow oi/o$),
- les GP circonstanciels deviennent GP attributifs ($circ/C_1 \rightarrow attr/C_2$).

Il s'avère que le tableau réel est bien plus complexe. Pour commencer, certains arguments des déverbaux hérités des cadres des verbes d'origine ont tendance à perdre leur statut obligatoire. Il s'agit surtout des compléments locatifs et obliques. Par exemple, le cadre du verbe pronominal *se déplacer* est soit (oi/l) soit (oi/o) (un des deux est obligatoire : cf. *Il se déplace à Bordeaux* ; *Il se déplace en voiture* mais **Il se déplace*). Encore pire est la situation où le verbe d'origine est di-transitif avec le cadre ($subj/n, od/a, oi/g$), (e.g. *saupoudrer*($subj/n, od/a, oi/g$)). Dans ce cas un des compléments est supprimé. En fonction du contexte le complément d'objet au génitif du déverbal *saupoudrage*(oi/g) peut provenir soit du complément d'objet direct du verbe (cf. *saupoudrage des champs* ($subj/n \rightarrow 0, od/a \rightarrow oi/g, oi/g \rightarrow 0$)), soit de son complément d'objet indirect au génitif (cf. *saupoudrage de réformettes* ($subj/n \rightarrow 0, od/a \rightarrow 0, oi/g \rightarrow oi/g$). Et enfin, très souvent le GP subordonné à un nom déverbal peut provenir du sujet et même d'un des arguments circonstanciels du verbe d'origine. Par exemple, dans la phrase *L'argumentation du clan pro-inflation consiste à ...* le GP *du clan pro-inflation* est un attribut du nom déverbal *argumentation*(0). Cet attribut provient du sujet du verbe *argumenter*($subj/n, od/a$). Ainsi dans ce contexte le cadre de sous-catégorisation du verbe est transformé selon la règle ($subj/n \rightarrow attr/g, od/a \rightarrow 0$). D'un autre côté, dans la phrase *La débandade de 1982-1983 qui vit le pays...* le verbe *débander*($subj/n$) est intransitif. Aussi l'attribut *de 1982-1983* du déverbal *débandade* provient d'un argument circonstanciel du verbe (*débander en 1982-1983*) : ($circ/o \rightarrow attr/g$).

Sans argument supplémentaire ces exemples peuvent faire croire qu'il faut faire face à un nombre important de transformations de cadres de sous-catégorisation. Heureusement, les actants des noms déverbaux obéissent au postulat suivant (visiblement remontant à la présomption d'invariabilité des actants des mots de Mel'čuk et Holodovič (1970))[11] :
postulat de provenance : les actants des noms déverbaux proviennent d'un des actants du verbe d'origine.

Une conséquence directe de ce postulat est que l'actant du déverbal ne peut pas provenir d'un argument circonstanciel du verbe d'origine mais seulement de son sujet ou d'un des objets présent dans son cadre. Dû à ce postulat notre objectif devient plus précis : on peut établir la provenance des actants des noms déverbaux réalisés par des GN non attributifs seulement à partir des GN non circonstanciels du verbe d'origine. Quoique très restrictif, le postulat de provenance laisse beaucoup de variantes : différents suffixes formateurs, différents rôles syntaxiques des noms déverbaux dans différents contextes et différents actants correspondants du verbe d'origine. Pour simplifier la tâche nous nous limitons dans ce chapitre aux GN au génitif qui réalisent les actants des noms déverbaux. Dans les expériences présentées dans la section suivante nous étudions leur provenance et les règles de conversion qui les concernent.

[11] Nos expériences avec le corpus Paris 7 confirment ce postulat (au moins pour les noms déverbaux repérés dans le corpus).

5.2.3 Expériences et évaluation des résultats

Ici, nous nous intéressons à découvrir le rôle des GP des déverbaux pour le verbe qui leur correspond. Nous avons identifié dans un premier temps plusieurs possibilités puis nous avons cherché à comprendre sur des exemples d'un corpus les liens entre les compléments du verbe et les GP introduits par la préposition *de* du déverbal. Notre étude a consisté sur le corpus d'arbres syntaxiques de phrases de Paris 7 Abeillé et al. (2003), à repérer toutes les occurrences de noms communs qui correspondent à l'algorithme **proto-déverb** et qui possèdent un (ou plusieurs) GP introduits par la préposition *de*. Sur ces occurrences, nous avons demandé à des experts d'indiquer pour ces occurrences si les couples nom/verbe produits par l'algorithme **proto-déverb** correspondent effectivement (dans le contexte des occurrences) à un couple déverbal/verbe. Dans un deuxième temps, pour les occurrences validées comme des déverbaux, les experts devaient indiquer la fonction que pouvaient prendre les GP par rapport au verbe. Voici quelques exemples qui illustrent les différents cas de figure rencontrés par les experts annotateurs :

1. *Le* langage de la formation*, trop souvent fait d'approximations, [...]* avec le couple *langage/langer* : en fait, *langage* n'a rien à voir avec *langer*.
2. *L'alliance des vrais-faux contraires suscite la* parade des supposés amis avec le couple *parade/parer* : le nom *parade* peut être associé à deux verbes suivant son sens. Soit *parer*, soit *parader*. Dans cette phrase, le sens de *parade* correspond à *parader*. Ces deux premiers exemples sont étiquetés **non D** (non déverbal).
3. *Avec une croissance de 3,3% à la même date, les départements d'outre-mer bénéficieront d'un* rattrapage d'un point supplémentaire avec le couple *rattrapage/rattraper* : ici nous avons clairement un déverbal. Le GP *d'un point supplémentaire* joue le rôle du complément à l'accusatif (complément d'objet direct) du verbe *rattraper* car nous pourrions dire *ceci rattrape un point supplémentaire*. Cet exemple est étiqueté **C=a** par les experts.
4. *Reports ou renvois qui n'empêchent pas M. Bérégovoy de juger "injuste" l'*accusation d'immobilisme avec le couple *accusation/accuser* : le GP du déverbal correspond à un complément au génitif du verbe *accuser* (complément d'objet indirect introduit par *de*) car on peut dire *on l'accuse d'immobilisme*. Cet exemple est étiqueté **C=g** par les experts.
5. *Dans l'intervention humanitaire, il n'y a pas d'*obligation de résultats avec le couple *obligation/obliger* : cette fois le GP correspond à un complément au datif (complément d'objet indirect introduit par *à*) car nous pouvons dire *cela nous oblige à des résultats*. Cet exemple est étiqueté **C=d** par les experts.
6. *De plus, la Russie, mastodonte pétrolier empêtré dans des problèmes infinis, paraît incapable de stopper la* dégringolade de sa production*, [...]* avec le couple *dégringolade/dégringoler* : ici, *la production* pourrait être le sujet de dégringoler car nous pouvons dire *la production dégringole*. Cet exemple est étiqueté **subj** par les experts.
7. *Au plan agricole, la délégation de Gdansk a pris contact avec les professionnels de la race bovine limousine, dont les performances zootechniques,*

notamment en élevage *extensif* de plein air*, leur ont semblé particulièrement adaptées aux conditions polonaises* avec le couple *élevage/élever* : cette fois, le GP correspond à un argument circonstanciel (optionnel) du verbe *élever* puisque nous pourrions dire *on les élève en plein air*. Cet exemple est étiqueté **circ** par les experts.

8. *Début 1991, autour de Joachim Trautwein, les six "élus" de l'ancien encadrement suivent trois séminaires de deux jours, en présence d'un sociologue est-allemand et d'un* chercheur *français* du CNRS avec le couple *chercheur/chercher* : dans cette phrase, il est difficile de rattacher *le CNRS* au verbe *chercher* car le GP désigne l'appartenance à une organisation de la personne qui cherche. Nous pouvons dire dans ce cas que bien que nous ayons un déverbal, le GP de ce déverbal ne provient pas de la structure des compléments du verbe d'origine. Cet exemple est étiqueté **V/S** (un déverbal dont le GP ne peut pas être subordonné au verbe : (0→*attr/g*)).

Les annotations ont été classées par la règle de l'algorithme **proto-déverb** utilisée pour faire la correspondance entre le déverbal et le verbe. Les séquences ont été extraites d'un sous ensemble du corpus Abeillé et al. (2003) constitué de 31 fichiers sur 45 (les autres fichiers nous permettant de vérifier nos hypothèses) et comportant 15106 phrases (sur 21776). Le programme d'extraction a identifié 647 séquences *nom + GP* avec *nom*, un nom commun produit par l'algorithme **proto-déverb** (le nom peut être séparé de son GP par d'autres éléments du groupe nominal). Sur ces 647 séquences les experts ont supprimé une phrase pour laquelle le lien entre le GP et le verbe n'était pas clair[12]. Les experts ayant divergé sur l'interprétation de la fonction de certains GP entre **C=a** et les autres cas, nous avons dans un premier temps conservé l'annotation qui n'était pas **C=a**. Toutefois, nous indiquons, dans la colonne **C=a**, avec le symbole + suivi d'un nombre, le nombre de cas **C=a** divergents. En fait, sur les 646 séquences, 18 annotations sont restées différentes même après concertation des annotateurs. Comme nous l'avons vu précédemment, les experts ont repéré 62 séquences ne correspondant pas à un déverbal et 584 correspondant à un déverbal. Voici le tableau des résultats classés par règle :

Nous voyons que très majoritairement (413 sur 584 soit 70,7%), le GP correspond au complément d'objet direct du verbe. Les deux autres types de compléments d'objet indirect (C=d et C=g) sont bien moins nombreux car les verbes du premier groupe avec ce type de complément sont peu nombreux. Les annotateurs en ont repéré 19 soit 3,3%.

À la lecture des exemples annotés C=d, on s'aperçoit que les déverbaux autorisent de manière générale qu'un GP normalement introduit par la préposition *à* soit introduit par la préposition *de* avec le déverbal. Ainsi, on peut dire *Dans l'intervention humanitaire, il n'y a pas d'obligation* **de** *résultats* à la place de *Dans l'intervention humanitaire, il n'y a pas d'obligation* **à** *des résultats*.

Par contre, les compléments du verbe au génitif (C=g) se traduisent naturellement en un GP du déverbal introduit par *de*. Ainsi, dans la phrase

[12] "*Le* déplacement *en milieu psychiatrique* du traitement de l'acte *- qui ne serait parlé que là, médicalement - exclut l'inculpé de la société : non seulement il est dispensé, mais il est empêché de répondre de son acte devant les parties directement concernées alors que les deux démarches devraient être complémentaires*" avec le couple *déplacement/déplacer*.

Règle	non D	C=a	C=d	C=g	subj	circ	V/S	Total D	Total
er ⇒ ade	1				1	1		2	3
er ⇒ age	10	44 + 3		4	11	4		63	73
er ⇒ aison		5						5	5
er ⇒ ant	5	17 + 2	1		2	5	1	26	31
er ⇒ ation	7	210 + 8	3	3	39	17	2	274	281
er ⇒ ement	6	101 + 4	3	2	11	12	3	132	138
er ⇒ ence	7	5	1		14	4		24	31
er ⇒ eur	17	24 + 1		1	2	10	8	45	62
er ⇒ euse						1		1	1
er ⇒ oir	4					1		1	5
er ⇒ rice	1	1						1	2
er ⇒ ure	4	6	1			2	1	10	14
Total	62	413 + 18	9	10	80	57	15	584	646

Tableau 5.5 – *Analyse sur le corpus Paris 7 de la fonction des GP des déverbaux*

reports ou renvois qui n'empêchent pas M. Bérégovoy de juger injuste l'accusation d'immobilisme, la séquence *accusation d'immobilisme* correspond à *accuser d'immobilisme*.

Le cas des arguments circonstanciels est aussi assez simple puisque les 57 exemples montrent que ces compléments du verbe introduits par des prépositions comme *en*, *pendant*, *vers*, *pour*, *dans*, etc, peuvent se transformer en GP introduits par *de* d'un déverbal. Dans les 57 cas rencontrés par les annotateurs, le GP était toujours optionnel et représentait un attribut plutôt qu'un actant du déverbal.

Le cas de la transformation du sujet du verbe en un GP introduit par *de* du déverbal est plus problématique. Les annotateurs en ont repéré un grand nombre (80 sur 584 soit 13,7%). Toutefois, ils ne sont pas toujours tombés d'accord sur ces séquences car une partie (18 sur 80) a été annotée comme **C=a** par au moins un annotateur. En fait, cela s'explique assez bien car la majorité de ces cas problématiques correspondent soit à des verbes pouvant être pronominaux, soit à des verbes dont le complément d'objet direct peut devenir sujet. Par exemple, nous pouvons dire que *les achats de biens durables redémarrent* mais aussi qu'*on redémarre les achats de biens durables*. Dans ce cas, le GP du déverbal *redémarrage* peut effectivement correspondre au sujet ou à l'objet de *redémarrer*. Dans le même ordre d'idée, si l'on parle de *la précarisation du marché du travail*, on peut considérer soit que *le marché du travail précarise les travailleurs*, soit que *la société précarise le marché du travail*.

Mis à part ces problèmes d'ambiguïté, une analyse fine des exemples annotés **subj** rejoint et complète l'analyse que nous avions faite à propos du cas **C=d**. Alors que la forme normale de la transformation du sujet dans le déverbal devrait être un GP introduit par *par*, il semble courant que cette préposition *par* soit remplacée par la préposition *de*. Cela est particulièrement vrai lorsque le verbe est intransitif comme *la dégringolade de sa production* dans la phrase *De plus, la Russie [...] paraît incapable de stopper la dégringolade de sa production [...]*. Dans ce cas très précis, il est même impossible d'utiliser un GP introduit par *par*. Pour les verbes transitifs dont un des compléments est introduit par *de*, les exemples du corpus montrent que le nom déverbal provenant du verbe perd son actant-sujet.

La conclusion de cette étude semble bien indiquer que notre hypothèse d'origine sur les transformations des cadres de sous-catégorisation des verbes vers ceux des déverbaux était globalement fondée. Nous avons toutefois dû revoir le cas du sujet des verbes qui peuvent sous certaines conditions devenir de vrais arguments du déverbal sous la forme d'un GP introduit par *de* et ajouter le cas de la transformation des compléments au datif en un GP au génitif :
- soit le sujet du verbe disparaît et est converti en un attribut optionnel du nom dérivé (en général un GP introduit par la préposition *par*) : $(subj/n \rightarrow attr/\perp)$, soit il est le complément unique du déverbal (en général un GP introduit par la préposition *de*), en particulier si le verbe est intransitif ou bien lorsque le verbe comporte normalement un complément qui est alors élidé : $(subj/n \rightarrow oi/g)$,
- l'objet direct ($C = a$), s'il n'est pas élidé, devient un GP introduit par *de* $(od/a \rightarrow oi/g)$,
- tout autre objet indirect, s'il n'est pas élidé, est tout simplement hérité sous réserve de sa présence dans le cadre (e.g $(oi/d \rightarrow oi/d)$ ou $(oi/o \rightarrow oi/o)$) avec la possibilité pour le datif de se retrouver transformé en un GP introduit par la préposition *de* à la place de la préposition *à* : $(oi/d \rightarrow oi/g)$,
- les GP circonstanciels au génitif sont transformés en GP attributifs au génitif.
Cette analyse complète celle de Benveniste (1966) qui ne prévoit que deux alternatives : génitif subjectif / génitif objectif.

Pour terminer ce tableau, nous nous sommes aperçus qu'il y a très peu de cas de déverbaux comportant deux GP introduits par *de* qui correspondent à deux compléments du verbe (contrairement au cas où l'un des deux GP introduit par *de* correspond à un argument circonstanciel du verbe). Un exemple intéressant provient du déverbal *radiation* et de son verbe d'origine *radier* qui comporte deux compléments, un complément d'objet direct $C = a$ et un complément d'objet indirect introduit par *de* $C = g$. *Et semble pencher, en privé, pour une radiation de son groupe du second marché boursier* : les deux GP *de son groupe* et *du second marché boursier* correspondent à *on a radié le groupe du second marché*.

Par contre, nous n'avons trouvé aucun cas de déverbal comportant un GP au génitif provenant du sujet du verbe et un autre GP du déverbal (pas forcément au génitif) provenant d'un complément de ce verbe. Il semble donc que les cadres de sous-catégorisation des déverbaux avec deux actants dont un provient du sujet du verbe transitif, s'ils existent, sont très restreints.

5.2.4 Construction du corpus de déverbaux

Dans le but de construire un corpus des noms déverbaux du français à large couverture et utilisable dans les applications du traitement automatique des langues. Nous présentons le corpus des déverbaux *VerbSCF-Deverbal*. Ce corpus est formé pour l'instant par l'ensemble des déverbaux obtenus à partir des annotations des experts pour les 17 règles. Ce corpus des déverbaux contient des noms déverbaux avec leurs traits morphologiques, des phrases où ils apparaissent et leur fonction syntaxique par

rapport au verbe d'origine. La figure 5.2 montre une exemple de la format de ce lexique.

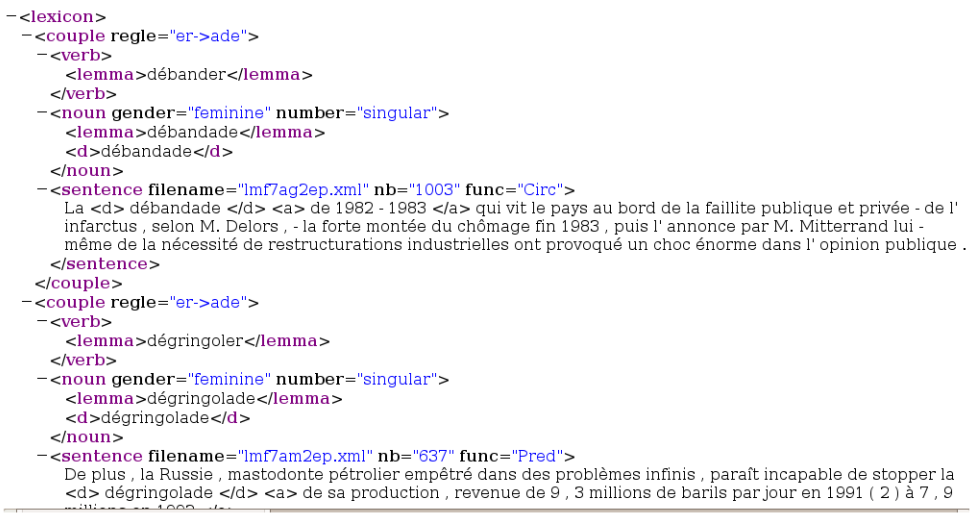


Fig. 5.2 – *Corpus des noms déverbaux*

5.3 Complétion de la liste des suffixes

Comme nous l'avons déjà vu précédemment, l'algorithme **proto-déverb** ne couvre pas la totalité des verbes du premier groupe. Dans cette partie nous complétons la liste des suffixes pour le premier groupe et nous traitons aussi les terminaisons des deuxième et troisième groupes *ir, re*. Les tableau 5.8, 5.3 et 5.3 donnent plus en détails la liste des règles de la terminaison "er", "ir' et '"re", nous avons une liste qui contient plus de 120 règles. Pour simplifier nous n'avons fait les statistiques que sur les noms singuliers pour éviter les répétitions de noms. Les statistiques sont sur une liste de 75 règles. Le nombre de couples du lexique Verbaction qui ne sont pas produits par l'algorithme est de 5404 (sur 9305). En fait, la raison principale est qu'il existe 4000 noms dans Verbaction mais qui n'existent pas dans Le*fff* comme nous avons déjà mentionné que Le*fff* n'est pas complet. (e.g. (abalourdissement/abalourdir), (abandonnement/abandonner), (avoyage/avoyer), (basage/baser), bonnetade/bonneter), (montrance/montrer), (rossade/rosser), etc). Par contre, on voit bien dans le tableau 5.6 que l'algorithme **proto-déverb** produit 7113 couples nom/verbe qui n'existent pas dans Verbaction. Ce résultat est calculé selon les nouvelle règles qui produises 11014 couples nom/verbe.

Déverbal (Verbaction)	Non déverbal (Verbaction)	Total produit par l'algo **proto-déverb**
3901 (35,41%)	7113 (64,58%)	11014

Tableau 5.6 – *Comparaison entre* **proto-déverb** *et le lexique Verbaction en fonction des règles (75 règles)*

Suffixes	Déverbal (Verbaction)	Non déverbal (Verbaction)	Total **proto-déverb**
action	1		1
ade	3		3
age	52	8	60
ain		1	1
aine		1	1
aire		11	11
aison	2	1	3
ance	13	7	20
ant		14	14
ante		5	5
at	1	6	7
ation	199	16	215
c	4	3	7
ction	9		9
e	104	202	306
é	4	21	25
ée	17	4	21
ement	104	8	112
vide	45	119	164
ence	10	9	19
erie		7	7
ette		3	3
eur	1	27	28
ie	4	3	7
iment	1	1	2
in		2	2
ing	1		1
ion	2	2	4
isation		3	3
isme		1	1
issage	1		1
issement	27		27
isseur		3	3
issure		1	1
ité		14	14
ition	10	2	12
l	2	4	6
m		3	3
me		14	14
n	20	33	53
oir		1	1
on		12	12
ption	1		1
rice		2	2
sion		1	1
t	1	7	8
te	11	3	14
tion	14	4	18
ure	2	7	9
Total	666 (52,77%)	596 (47,22%)	1262

Tableau 5.7 – *Évaluation de* **proto-déverb** *sur le corpus Paris 7 en fonction des règles*

Le tableau 5.7 indique pour chaque règle utilisée pour l'analyse du corpus Paris 7, le nombre de couples nom/verbe qui correspondent ou non à un couple nom/verbe de Verbaction pour toutes les terminaisons (*er, ir et re*).

Liste de la terminaison "er"					
terminaison	suffixe	total	terminaison	suffixe	total
er	vide	715	er	euse(s)	1574
er	ade(s)	164	er	in	144
er	eur(s)	2237	er	ing	42
er	age(s)	2475	er	is	202
er	ance(s)	106	er	isation(s)	60
er	aies	12	er	ise	36
er	aille(s)	120	er	isme	150
er	aison(s)	126	er	ité	98
er	ain(e)(s)	104	er	ition(s)	192
er	é(s)	2804	er	me	366
er	aire	172	er	oir(e)(s)	408
er	amini	4	er	oison(s)	4
er	ssion(s)	8	er	on	380
er	anderie(s)	4	er	xion(s)	8
er	at	100	er	rice(s)	38
er	ant(e)(s)	1151	er	sion	2
er	sions	2	er	sse	14
er	ation(s)	2189	nner	n	225
er	cace	2	pper	p	3
er	e(s)	6862	quer	c	50
er	ée(s)	1353	ner	vide	38
er	ement(s)	1904	ser	e	42
er	eur(s)	2237	sser	s	78
er	ence(s)	132	ter	vide	99
er	ette(s)	900	tter	t	28
er	erie	454	cer	vide	13
er	eries	454	ffer	f	5
er	tion	64	ier	e	42
er	trice(s)	10	ller	l	32
er	ure(s)	466	mmer	m	4

Tableau 5.8 – *Liste des terminaisons "er"*

Liste de la terminaison “ir”		
términaision	suffixe	total
ir	vide	48
ir	ie	53
ir	iment	12
ir	ion	8
ir	issage(s)	96
ir	issance(s)	147
ir	issant(e)(s)	16
ir	issement(s)	320
ir	isseur(s)	112
ir	isseuse(s)	54
ir	issoir(e)(s)	24
ir	issure(s)	28

Tableau 5.9 – *Liste des terminaisons “ir”*

Liste de la terminaisons “re”		
terminaison	suffixe	total
re	vide	14
re	age(s)	33
re	aison(s)	12
re	ance(s)	22
re	ant(e)(s)	105
re	ation(s)	6
re	e(s)	39
re	ement(s)	34
re	erie(s)	28
re	eur(s)	108
re	ition	1
re	oir(s)	32
re	ption	11
re	rice(s)	4
re	sade(s)	6
re	sage	5
re	sance	16
re	seur	18
re	sson	4
re	t	42
re	ure(s)	18
aire	action	17
dre	te	46
tre	vide	11
ire	ction	23

Tableau 5.10 – *Liste des terminaisons “re”*

5.3.1 Conclusion et perspectives

Dans la première partie de ce chapitre nous avons présenté un algorithme **proto-déverb** qui nous a permis de compléter le lexique de la CDG du français en plaçant les déverbaux dans les classes lexicales qui correspondent à leurs cadres de sous-catégorisation. Nous avons étudié le cas le plus fréquent du génitif, c'est-à-dire des groupes prépositionnels introduits par *de*, *des*, etc. Dans un premier temps nous avons présenté une liste de déverbaux construite à l'aide de 17 règles. Une évaluation est effectuée sur le corpus Paris 7. Les expertises ont montré que 584 séquences sur 646 soit 90,4% correspondent à un déverbal. Nous avons aussi comparé la liste obtenue avec la liste Verbaction des déverbaux exprimant l'action ou l'activité du verbe.

Nous avons développé le corpus des déverbaux *VerbSCFDeverbal*. Ce corpus est formé pour l'instant par l'ensemble des déverbaux obtenus à partir des annotations des experts pour les 17 règles. Ce corpus contient des noms déverbaux avec leurs traits morphologiques, des phrases où ils apparaissent et leurs fonctions syntaxiques par rapport au verbe d'origine.

Dans la deuxième partie, nous avons complété la liste de déverbaux pour le premier groupe et nous avons aussi traité les terminaisons du deuxième et du troisième groupe *ir, re*. L'algorithme **proto-déverb** produit 7113 couples nom/verbe qui n'existent pas dans Verbaction. Ce résultat est calculé avec des nouvelles règles (75 règles) qui produisent 11014 couples nom/verbe.

Pour conclure, notre calcul de la provenance des compléments au génitif des noms déverbaux ne donne pas encore une solution complète et définitive au problème du calcul des cadres de sous-catégorisation des déverbaux même s'il fournit une approximation tout-à-fait satisfaisante. Cependant :
1) dans les contextes où le nom déverbal a deux compléments d'objet, nous n'avons pas de règle contextuelle définissant la provenance de chacun (nos règles sont individuelles) ;
2) et même dans le cas le plus fréquent où le nom déverbal n'a qu'un seul actant, nos règles ne résolvent pas le choix précis de l'actant du verbe d'origine dont il provient.

Comme perspective, nous souhaitons compléter le corpus de déverbaux *VerbSCFDeverbal* en traitant toutes les règles que nous avons trouvées.

Troisième partie

Amélioration de l'analyse en dépendances

6 Une approche pour améliorer l'analyse en dépendances

Sommaire

6.1 Introduction 95
6.2 Modèles d'analyses en dépendances 95
6.2.1 Correspondances entre des étiquettes des étiqueteurs morpho-syntaxiques et les parties du discours de L*efff* . . 97
6.2.2 Expériences et évaluation des résultats 99
6.3 Discussions 105
6.4 Conclusion 107

Ce chapitre évoque le problème principal de l'analyse syntaxique des grammaires à large couverture (en dépendances) qui est l'explosion combinatoire des analyses fallacieuses.
Ce chapitre se compose des sections suivantes : La section 6.1 décrit nos motivations. Dans la section 6.2, nous présentons nos modèles d'analyses. La section 6.2.2 évoque les expériences et l'évaluation des résultats obtenus. On termine par une discussion et conclusion dans les sections 6.3 et 6.4.

6.1 Introduction

L'analyse syntaxique constitue aujourd'hui une étape essentielle du traitement automatique des langues dès lors qu'elle recherche une connaissance relativement fine des relations grammaticales présentes dans une phrase. L'analyse syntaxique des grammaires à large couverture fait face au problème de l'explosion du nombre des combinaisons des sous arbres et du nombre d'arbres de dépendances théoriquement possibles mais qui, en majorité, donnent des analyses fallacieuses. En effet l'analyseur des CDG donne actuellement toutes les solutions compatibles avec une CDG. Avec le modèle que nous proposons, nous montrons qu'en utilisant des étiqueteurs morpho-syntaxiques pour choisir les classes grammaticales les plus probables des unités lexicales, nous pouvons sensiblement réduire le taux d'ambiguïtés fallacieuses d'une grammaire catégorielle de dépendances du français qui utilise Le*fff* comme base lexicale. De cette manière, nous pouvons réduire l'espace de recherche de l'analyseur des CDG. Nous comparons aussi les résultats de l'association de trois étiqueteurs morpho-syntaxiques à l'analyseur de la CDG du français. Notre étude conclue que l'adéquation de ces solutions est basée principalement sur la compatibilité entre les unités lexicales qui sont définies par les étiqueteurs syntaxiques et la grammaire de dépendance. De plus, les résultats expérimentaux montrent que nos modèles sont plus performants que le modèle qui n'utilise pas d'étiqueteur syntaxique en termes de précision d'analyses.

6.2 Modèles d'analyses en dépendances

On peut dire en général que la tâche de désambiguïsation d'un analyseur de dépendance consiste à dériver un seul arbre correct des dépendances pour une phrase donnée en entrée. Certains analyseurs peuvent résoudre ce problème en dérivant une analyse unique pour chaque phrase ou parfois seulement les analyses les plus probables pour une phrase. Notre tâche est différente : nous devrions plutôt réduire l'ambiguïté de la CDG du français en utilisant des étiqueteurs morpho-syntaxiques "POS-taggers[1]" et nous évaluons l'effet obtenu par notre méthode. Nos modèles fondés sur des étiqueteurs morpho-syntaxiques choisissent d'abord les classes de CDG les plus probables à l'aide des étiquettes des étiqueteurs morpho-syntaxiques des mots dans une phrase.

En appliquant notre méthode, nous résolvons un problème technique qui découle de la nature de la base de données lexicales de la CDG du français.

Avant de décrire notre approche, nous devons expliquer que l'analyseur de CDG utilise les deux modes suivants pour le lexique (appelés ci-dessous les modèles) :

– Le modèle de base donne accès à des formes contenues dans les classes de la CDG du français (la grammaire texte, environ 1500 formes), et donne également accès à des définitions originales de Le*fff* liées aux

[1]Un POS-Tagger est un module de traitement de textes qui donne une partie du discours (Part of Speech) à chaque mot d'une langue dans une phrase concrète.

classes des CDG dans la base de données (lexique de la base de la CDG du français). La figure 6.1 illustre la forme générale du modèle de base.

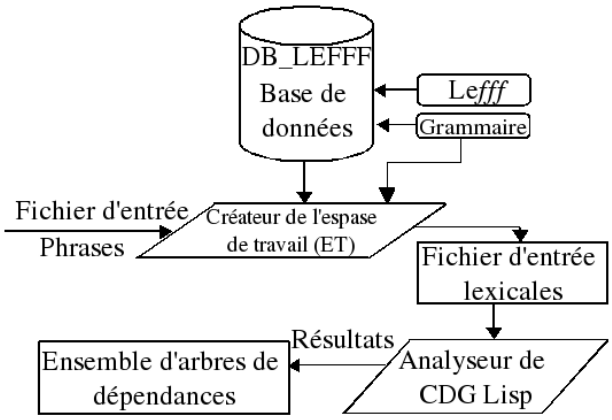


FIG. 6.1 – *Forme générale du modèle de base.*

Les trois autres modèles utilisent Le*fff* et implicitement la CDG du français. Dans ces modèles nous utilisons trois étiqueteurs morpho-syntaxiques.

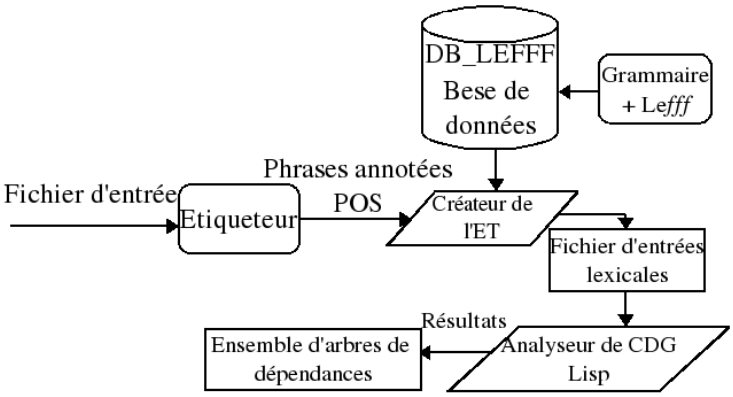


FIG. 6.2 – *Forme générale des modèles utilisant l'étiquetage des étiqueteurs morpho-syntaxiques*

Le tagging : Le tagging, plus communément appelé analyse morpho-syntaxique, gère après une étape de segmentation d'un texte (découpage de la chaîne de caractères en unités pertinentes de type paragraphe, mot, verbatim, phrase, etc.), l'attribution d'étiquettes aux unités lexicales (c'est-à-dire aux unités des phrases). Ces unités peuvent être simples ou complexes. Ce module se scinde en quatre phases principales : segmentation, détection, désambiguïsation et étiquetage. Par ailleurs, il appose deux types d'étiquettes aux unités du texte :

- étiquette syntaxique : pour chaque mot, une étiquette grammaticale (nom, adverbe, adjectif, etc.) pour réduire les ambiguïtés et permettre à d'autres traitements sur le texte d'aller plus loin dans l'analyse.
- étiquette morphologique : traits morphologiques (masculin, féminin, singulier, pluriel, etc.).

Dans notre approche, un étiqueteur est appliqué à la phrase en entrée (Tree-Tagger Schmid (1994) dans le modèle T.T, MElt-Tagger Denis et Sagot (2009) dans le modèle M.T et Brill-Tagger Brill (1994) dans le modèle B.T). Ensuite, selon le calcul (composite en général) des UL et leur partie du discours, une définition lexicale compatible pour chaque paire (UL, partie du discours) avec les classes de CDG correspondantes sont trouvées dans la base de données. Si elles existent, elles sont intégrées au fichier en entrée qui est envoyé à l'analyseur. Le résultat est un ensemble d'arbres de dépendances. La figure 6.2 présente cette stratégie. Plus précisément, étant donnée une phrases $x = w_1, w_2, ..., w_n$, on note sa séquence d'étiquettes de l'étiqueteur morpho-syntaxique sélectionné par $\mathbf{t} = t_1, t_2, ...t_n$, où $t_i \in \mathcal{T}, 1 \leq i \leq n$ où $\mathcal{T}$ est l'ensemble des étiquettes de l'étiqueteur morpho-syntaxique et n représente le nombre de tokens dans la phrase. Nous comparons (w_i, t_i) avec (UL_i, c_i), où UL_i est une forme fléchie et c_i indique l'une des parties du discours attribuées par la base de données stockant L*efff*. Nous donnons plus de détails dans l'algorithme 7.

6.2.1 Correspondances entre des étiquettes des étiqueteurs morpho-syntaxiques et les parties du discours de L*efff*

Les correspondances entre les classes de la CDG (basée sur les parties du discours de L*efff*) et des étiquettes des étiqueteurs morpho-syntaxiques sont établies par le créateur de l'espace de travail qu'on montre dans la figure 6.2. Nous avons essayé de trouver les correspondances entre les deux ensembles des parties du discours. Ces correspondances sont approximatives, car les modèles lexicaux des étiqueteurs morpho-syntaxiques et de la CDG du français avec L*efff* sont différents. Le tableau 6.1 montre les correspondances entre les étiquettes de Tree Tagger et les parties du discours de L*efff*.

Tree Tagger	L*efff*
NAM	np
KON	coo
DET :ART,DET :POS,PRP :det	det
NOM, NUM	nc
ADV	adv
ADJ	adj
PRO	pro
SENT	poncts
PUN	ponctw
VER :pres, VER :impf, VER :futu, VER :infi, VER :simp VER :cond, VER :pper, VER :ppre, VER :subp, VER :impe	v

Tableau 6.1 – *Les correspondances entre les étiquettes de Tree Tagger et les parties du discours de Lefff*

Le tableau 6.2 donne des détails sur le mode et le temps pour un verbe.

Comme nous le voyons dans les tables 6.1 et 6.2, Tree Tagger nous donne quelques informations importantes sur les étiquetages. Par exemple, *VER :futu* est très utile pour déterminer à la fois le mode et le

Etiquetage	Mode	Temps
VER :cond	conditionnel	
VER :futu	indicatif	futur
VER :impe	impératif	
VER :impf	imparfait	
VER :infi	infinitif	
VER :pper	participe	passé
VER :ppre	participe	présent
VER :pres	indicatif	présent
VER :simp	indicatif	passé simple
VER :subi	subjonctif	imparfait
VER :subp	subjunctif	présent

Tableau 6.2 – *Informations supplémentaires fournir par Tree Tagger*

temps d'un verbe. Dans ce cas, nous comparons aussi les champs *mood*[2] et *tense* de la base lexicale. *VER :futu* signifie que le mode est l'indicatif et le temps est le futur.

Le tableau 6.3 représente les correspondances entre les étiquetages de MElt Tagger et les parties du discours de L*efff*. Cette table nous donne aussi quelques détails supplémentaires sur les verbes.

MElt tagger	L*efff*
DET, DETWH	det
CLO, CLR, CLR, CLS, PRO	pro
VINF, VPP, VS, VIMP, VPR, V	v
P	prep
P+D	prep
NPP	np
ADV	adv
ADJ	adj
PRO, PROREL, PROWH, PREL	pro
NC	nc

Tableau 6.3 – *Les correspondances entre les étiquettes de MElt Tagger et les parties du discours de Lefff*

Le tableau 6.4 représente les correspondances entre les étiquettes de Brill Tagger et les parties du discours de L*efff*.

Le créateur de l'espace de travail pour les différents modèles prend un fichier en entrée qui contient les phrases avec leurs parties du discours (phrases annotées). La sortie est un fichier avec les entrées lexicales, les classes de CDG et l'ensemble des traits des mots stockée dans la base de données lexicale. L'algorithme choisit les classes de CDG les plus probables pour une UL en utilisant les étiquettes des étiqueteurs morpho-syntaxiques et les parties du discours de L*efff* d'une UL qui se trouve dans la base de données. Nous utilisons le même algorithme pour les

[2]Les noms en italique comme mood, tense, person, gender, number et lemma sont des champs de la base de données de la CDG du français qui sont fournies par L*efff*.

Brill tagger	L*efff*
PREP	prep
COO	coo
PREP :det	det
DTC	det
SBP	np
CAR	nc
ADV	adv
ADJ	adj
REL, PRO, SUB, PRN	pro
PUN	ponctw
VCJ, ACJ, ECJ, VNCFF VPAR,APAR, EPAR	v

Tableau 6.4 – *Les correspondances entre les étiquettes de Brill Tagger et les parties du discours de Lefff*

trois modèles Tree Tagger, MElt Tagger et Brill Tagger, mais les différences portent sur l'ensemble des étiquettes. Cet algorithme se compose de trois alternatives :

- nous recherchons d'abord le mot de la phrase et son étiquette en les comparant avec les formes et les parties du discours de L*efff* correspondant à l'étiquette du POS dans la base de données. En cas d'égalité, on cherche les classes de CDG qui correspondent à l'UL et aux traits morphologiques tels que le mode, le temps, la personne, le genre, le nombre et le lemme. Puis nous enregistrons toutes ces informations dans le fichier de sortie du créateur.
- s'il n'y a pas de résultat pour la première étape, nous regardons seulement le mot de la phrase et nous le comparons avec les formes de la base de données. Toutes les informations morphologiques qui correspondent sont ajoutées au fichier de sortie sous indication de classe. Dans ce cas, c'est l'analyseur qui déterminera les classes de l'UL.
- si le mot ne correspond à rien, on le classe dans la classe "`UT(Lex=V|N|Adj|Adv)`". Cette classe de CDG est attribuée aux UL inconnues (unknown terms).

6.2.2 Expériences et évaluation des résultats

Dans nos expériences, nous utilisons un corpus de phrases divisé en deux sous-ensembles. Le premier sous-ensemble, agissant comme un ensemble de test, se compose des 1443 phrases en français qui ont été analysées par A. Dikovsky pour construire le corpus arboré en dépendances du français (French Gold Standard dependency corpus). C'est un corpus avec des phrases de sources variées. Ces phrases comportent 14974 dépendances projectives et non projectives (discontinues).

Le deuxième sous-ensemble de ce corpus comporte 185 phrases du corpus arboré de Paris 7 du français (Abeillé et Barrier (2004)).

Pour l'expérience avec le premier sous-ensemble, nous exécutons

Algorithme 7 Modèles d'analyses en dépendance basés sur des étiqueteurs morpho-syntaxiques

Entrées : fichier contenant des phrases avec leurs parties du discours (phrases annotées).

But : fichier xml (entrées lexicales) avec leurs classes de CDG.

1: /* *word* correspond au mot dans une phrase, *POS* correspond à son étiquette de l'étiqueteur morpho-syntaxique. $form$ correspond à la forme fléchie du lemme dans la base de données, *cat* correspond à sa partie du discours dans la base de données.*/
2: **Fonction** LoadClass($word, POS$)
3: **Retourner** les classes et les traits où ($word$==$form$ et POS==cat). Le mode et le temps sont aussi testés, si *cat* est un verbe. Le nom de groupe de la classe est aussi testé. Par exemple si *cat* est "det", les classes retournées devrait commencer par "Det".
4: **Fin**
5: **Fonction** LoadInfoWithoutPOS(*word*)
6: **Retourner** les classes et les traits morphologiques qui correspondent aux *word* dans la base de données.
7: **Fin**
8: /* Début de l'algorithme, n représente le nombre de phrases, *tokens* sont les tokens et leurs étiquettes ;*/
9: **pour** i=1 **à** n **faire**
10: **pour** j=1 **à** *tokens* **faire**
11: $flag \leftarrow faux$
12: **pour** ***chaque*** $résultat \in$ LoadClass($word(j), POS(j)$) **faire**
13: Insérer $résultat$ dans le fichier xml (entrées lexicales) de sortie, $flag \leftarrow vrai$
14: **fin pour**
15: **si** $flag$ n'est pas $vrai$ **alors**
16: $noclass \leftarrow faux$
17: **pour** ***chaque*** $ligne \in$ LoadInfoWithoutPOS($word(j)$) **faire**
18: Insérer $ligne$ dans le fichier xml (entrées lexicales), $noclass \leftarrow vrai$
19: **fin pour**
20: **finsi**
21: /* Si *word* n'existe pas dans la base de données, alors une classe spéciale est choisie (unknown lexical units).*/
22: **si** $flag$ n'est pas $vrai$ et $noclass$ n'est pas $vrai$ **alors**
23: $class \leftarrow UT(Lex = V|N|Adj|Adv)$, insérer $word, POS, class$ dans le fichier xml (entrés lexicales)
24: **finsi**
25: **fin pour**
26: **fin pour**

d'abord l'analyseur avec le paramètre du nombre maximum d'arbres de dépendances fixé à 2000. Nous ne pouvons pas demander tous les arbres de dépendances possibles par phrase. En effet, la CDG du français génère des centaines de structures fallacieuses par phrase. Donc pour les phrases longues et complexes, il est pratiquement impossible de savoir combien

de structures de dépendances (SD) sont produites. Jusqu'à l'étape finale de l'analyseur où les SD sont extraites, l'algorithme d'analyse est polynomial. Etant donné que le nombre de ces SD peut être exponentiel par rapport à la taille de la phrase, la dernière étape est exponentielle dans le pire des cas. Dans cette étape, les SD sont générées dans un certain ordre en se limitant à un nombre maximum fixé au départ. L'analyseur génère aussi un rapport *HTML* qui comprend diverses statistiques utiles. Il produit une représentation *XML* pour chaque SD avec toutes les informations nécessaires.

Pour nos modèles d'analyses fondés sur les étiqueteurs morpho-syntaxiques, nous comptons le nombre d'arbres de dépendances en utilisant la formule $X^j = \sum_{i=1}^{N} A_i^j$, où A_i^j est le nombre d'arbres de dépendances qui est trouvé pour le modèle j, où j est le modèle de base, le modèle T.T, le modèle M.T ou le modèle B.T, et où i=1, ..., N avec N représentant le nombre de phrases qui ont une analyse correcte à 100% dans chaque modèle. Pour nos expériences, N=325. La réduction du nombre d'arbres de dépendances du modèle j est $\frac{X^{Base}-X^j}{X^{Base}} \times 100$ où j est différent du modèle de base. Nous ne comptons pas les phrases qui ont plus 2000 analyses pour chaque modèle. Le tableau 6.5 montre la réduction sur les arbres de dépendances pour chaque modèle.

	Base	T.T	M.T	B.T
# SD (325 phrases)	153938	42572	44056	46718
Réduction # SD	p/r au modèle j	72.34%	71.38%	69.65%
Moyenne géométrique $\sum_{1}^{325} \#\,\mathrm{SD}_j / \#\,\mathrm{SD}_{Base}$	-	0,24	0,23	0,26

Tableau 6.5 – *Comparaisons des quatre modèles d'analyses (pour # SD).*

Pour la deuxième expérience avec le premier sous-ensemble, nous exécutons l'analyseur avec le paramètre du nombre maximum d'arbres de dépendances fixé sur 1, afin d'obtenir un nombre maximum de phrases analysées et aussi pour savoir combien il y a de dépendances correctes dans une phrases. Nous comptons le nombre total des arbres de compositions[3] "composition trees CT" en utilisant la formule $Y^j = \sum_{i=1}^{M} B_i^j$, où B_i^j est le nombre d'arbres de compositions pour les phrases trouvées (analysées) pour le modèle j, où j est le modèle de base, le modèle T.T, le modèle M.T ou le modèle B.T, et où i=1,...M. M représente le nombre des phrases qui ont au moins une analyse dans chaque modèle. Pour notre expérience M=780. La réduction du nombre d'arbres de combinaison pour le modèle

[3]Pour chaque arbre de dépendances ou structure des dépendances SD, il y a plusieurs arbres de compositions (composition trees CT) parce que chaque arbre de composition spécifie aussi un ensemble des traits, une classe et un type. Nous utilisons le nombre d'arbres de composition au lieu du nombre d'arbres de dépendances parce qu'il n'est généralement pas possible d'évaluer le nombre total des arbres de dépendances.

j est $\frac{Y^{Base}-Y^{j}}{Y^{Base}} \times 100$, où j est différent du modèle de base. Le tableau 6.6 montre la réduction sur les arbres de compositions pour chaque modèle.

	Base	T.T	M.T	B.T
# CT (780 phrases)	16330×10^8	27×10^8	34×10^8	28×10^8
Réduction de # CT	p/r au modèle j	99.83%	99.79%	99.82%
Moyenne géométrique $\sum_{1}^{780} \#\ \mathrm{CT}_j / \#\ \mathrm{CT}_{Base}$	-	0,035	0,037	0,033

Tableau 6.6 – *Comparaisons des quatre modèles d'analyses (pour # CT).*

L'évaluation de l'analyseur utilise des mesures classiques. Il utilise l'AS_L "labeled attachment score", les scores d'attachement avec étiquette pour le mode de la figure 6.2 qui calcule la proportion de mots pour lesquels le gouverneur assigné est correct et le nom de dépendance est correct.

	Base	T.T	M.T	B.T
# Phrases qui ont au moins une analyse	1089	1125	1005	949
# Phrases qui ont 100% de dépendances correctes	1089	874	892	667
Rapel	75.46%	60.65%	61.81%	46.22%
Précision	100%	77.68%	88.75%	70.28%
# des dépendances	8255	9571	7730	7603
# des dépendances correctes	8255	8465	7380	6838
Rappel des dépendances correctes	55.12%	56.53%	49.28%	45.66%
Précision	100%	88.44%	95.47%	89.93%
Moyenne de la précision en étiquetés (pour 1443 phrases)	100%	82.27%	85%	69%

Tableau 6.7 – *Comparaisons des quatre modèles d'analyses (précision des analyses)*

Il y a des phrases qui ont une précision de plus de 90% de dépendances correctes, mais nous ne comptons que les phrases qui ont 100% des dépendances correctes.

Le tableau 6.7 résume les résultats expérimentaux pour chaque modèle d'analyses. Il est aussi montré que nos modèles atteignent entre 88% et 95% de précision pour l'étiquetage correcte (Labeled accuracy) de la relation de dépendances.

Nous n'avons pas besoin d'utiliser AS_U (score d'attachment sans étiquette), parce que nous ne comparons pas le résultat de plusieurs analyseurs, AS_U est utilisée par Candito et al. (2010a) qui comparent deux analyseurs syntaxiques par rapport à la précision sur les mots inconnus. En effet Bky+Flabeler Petrov et al. (2006) ne réalise qu'une précision de marquage de 82,56% pour les mots inconnus dans l'ensemble de dévelop-

pement (5,96% "tokens") alors que (MElt+MST McDonald (2006)) réalise 90,01%.

Le résultat dans le tableau 6.6 montre que le nombre d'arbres de compositions des trois modèles d'analyses basés sur les étiqueteurs morpho-syntaxiques est inférieur au modèle de base. Nos modèles atteignent une forte réduction à la fois des arbres de compositions et des arbres de dépendances (plus de 99% et plus de 70% respectivement).

Si on compare les trois modèles d'analyse basés sur les étiqueteurs morpho-syntaxiques, nous notons que le modèle M.T est plus performant que les modèles T.T et B.T en termes de précision d'analyse. Toutefois le modèle T.T est meilleur que les autres modèles en termes de réduction de l'ambiguïté et de temps d'analyse. Le tableau 6.8 donne une comparaison du temps d'analyse pour chaque modèle.

	Base	T.T	M.T	B.T
# Phrases qui ont au moins une analyse	1089	1125	1005	949
# Phrases qui sont analysées incorrectes	0	141	127	314
Totale phrases analysées	1089	1266	1132	1263
Temps d'analyses	03h 37mn	01h 32mn	02h 31mn	02h 8mn
# phrases qui ne sont pas analysées	354	177	311	180
Temps d'analyses	05h 09mn	03h 35mn	05h 18mn	03h 00mn
Temps total d'analyses	8h 46m	5h 07m	7h 49m	5h 08m

Tableau 6.8 – *Comparaisons des temps d'analyses des quatre modèles pour 1443 phrases*

Le tableau 6.9 montre les résultats d'une expérience sur la réduction du nombre d'arbres de dépendances et d'arbres de compositions pour les quatre modèles d'analyses pour la phrase : "il parle en phrases courtes.".

	Base	T.T	M.T	B.T
Réduction (# SD)	268	54	211	54
Réduction (# CT)	28732	3336	8559	3336

Tableau 6.9 – *Réduction (# SD et # CT) pour la phrase : il parle en phrases courtes.*

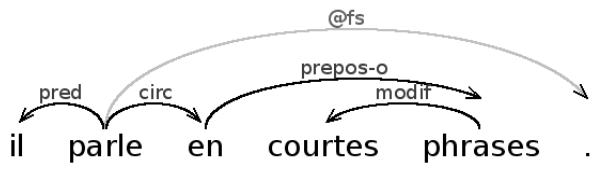


FIG. 6.3 – *Analyse normale de la phrase : il parle en phrases courtes.*

$$
\begin{aligned}
il &\longmapsto PN(Lex = pers, C = n)\\
parle &\longmapsto Vt(F = fin, C = g)\\
en &\longmapsto PP(F = compl - obl, C = o)\\
courtes &\longmapsto Adj(F = modifier)\\
phrases &\longmapsto N(Lex = common)
\end{aligned}
$$

La figure 6.4 et la figure 6.5 montrent différentes SD pour la phrase : "Il la leur fait apprendre".

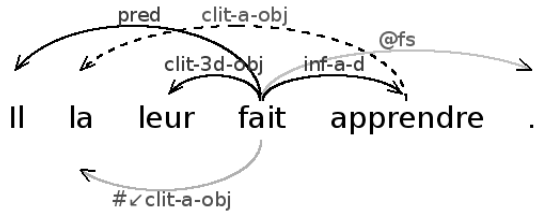


FIG. 6.4 – *SD correcte pour la phrase : il la leur fait apprendre.*

$$
\begin{aligned}
Il &\longmapsto PN(Lex = pers, C = n)\\
la &\longmapsto PN(Lex = pn, F = clit, C = a)\\
leur &\longmapsto PN(Lex = pn, F = clit, P = 3, C = d)\\
fait &\longmapsto Vlight(F = fin)\\
apprendre &\longmapsto V2t(F = inf, C1 = a|p, C2 = d|g|l)
\end{aligned}
$$

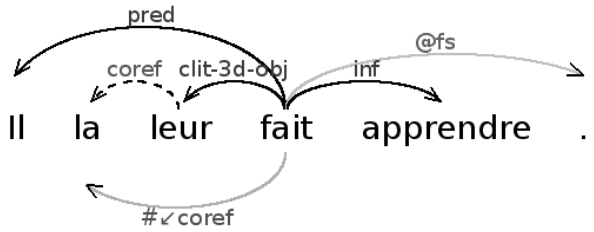


FIG. 6.5 – *SD incorrecte pour la phrase : il la leur fait apprendre.*

Pour le deuxième sous-ensemble de 184 phrases en français, nous utilisons seulement le modèle de base et le modèle T.T. Nous ne comptons que le nombre d'arbres de compositions en utilisant la même formule que pour le premier sous-ensemble.

Le résultat donné en tableau 6.10 montre que la pré-sélection des classes réduit l'ambiguïté en termes d'arbres de composition de plus de 99%.

	Base	T.T
#CT	1097325498316350	7048627222816
#CT	10973254×10^8	70486×10^8
Total reduction #CT	p/r au modèle de base	99,9%
Moyenne géométrique de $\sum_{1}^{184} \#\ CT_{T.T} / \#\ CT_{Base}$	-	0.002
# CT de la phrase de la figure 6.6	17284	241
# SD de la phrase de la figure 6.6	1295	68

Tableau 6.10 – *Effet de la pré-sélection des classes (corpus Paris 7)*

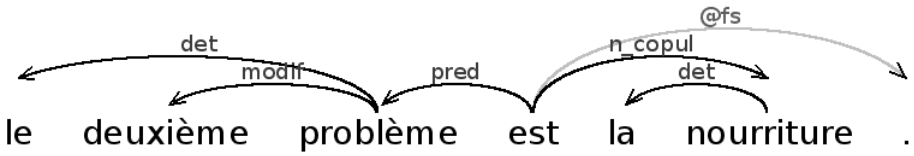


Fig. 6.6 – *Phrase du corpus paris 7 : le deuxième problème est la nourriture.*

6.3 Discussions

Nous fournissons une brève analyse des erreurs commises par les étiqueteurs morpho-syntaxiques pour le premier sous-ensemble.

	Base	T.T	M.T	B.T
Mémoire insuffisante	12	17	1	0
Pas d'analyse trouvée	0	141	127	314
Temps de calcul insuffisant	342	160	310	180

Tableau 6.11 – *Erreurs faites par l'analyseur pour les quatre modèles d'analyses*

Dans le modèle T.T, il y a 318 phrases qui n'ont aucune structure de dépendances, 177 de ces phrases ne sont pas analysées par l'analyseur (temps dépassé), ce qui signifie qu'il n'y a pas assez de temps pour les analyser comme nous l'avons indiqué ci-dessus à propos des ambiguïtés de la CDG (le nombre maximum de secondes par phrase est fixé à 60 secondes). Il y a 141 phrases qui sont analysés comme des phrases incorrectes. Une première raison de ce fait est qu'il y a au moins un des mots composés suivants dans les phrases : *à peu près*, *Hé bien*, *dès lors*, *de loin*, *au dessous*, *la-bas*, *des EU*, *de l'*, *etc*. Dans ce cas, Tree-Tagger étiquette ces mots composés comme des mots séparés : *à* comme *prep*, *peu* comme *adv*, *près* comme *adv*, etc. Mais la base de données a seulement des entrées pour le mot composé complet. Ce type de problème se retrouve avec tous les étiqueteurs. Nous donnons un exemple valable pour tous les étiqueteurs morpho-syntaxiques que nous avons utilisés. Cet exemple montre que la

compatibilité entre les étiqueteurs et l'analyseur est nécessaire dans certains cas tels que les mots composés ou les expressions complexes (comme pomme de terre), etc. L'analyse de la phrase "Ève, vas-t'en !" est représenté dans la figure 6.7.

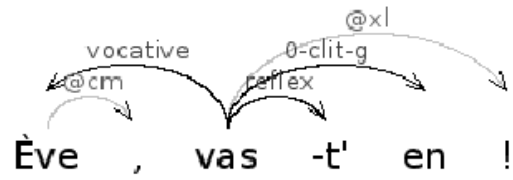


FIG. 6.7 – *structure de dépendances : Ève, vas-t'en !*

Chaque étiqueteur étiquette cette phrase de manière différente comme on peut le voir dans le tableau 6.12.

Tree Tagger	MElt Tagger	Bril Tagger
vas-t'en/NOM	vas-t'en/ADV	vas-t/VCJ en/PREP

Tableau 6.12 – *Étiquetages des différents étiqueteurs morpho-syntaxiques*

Dans la grammaire de CDG, ces unités lexicales ont différentes classes grammaticales. En conséquence cela donne différentes classes lexicales pour chaque unité lexicale. Le tableau 6.13 illustre les classes lexicales qui correspondent à la phrase "Ève, vas-t'en !".

CDG du français	
Unité lexicale	Classe
Éve	N(Lex=proper)
vas	Vt(F=fin,C=l), Vt(F=fin,C=d), Vt(F=fin,C=a)
t'	PN(Lex=pn,F=refl)
en	PN(Lex=pn,F=clit,C=g|p), PN(Lex=attach-npers,C=g|p)
!	EmphatMark(Lex=' !')

Tableau 6.13 – *Les classes assignées aux unités lexicales*

La deuxième raison principale est que chaque étiqueteur fait des erreurs d'étiquetage pour certaines UL. En conséquence le créateur d'espace de travail ne trouve pas de bonne classe de CDG pour ces UL. Nous avons vu que les résultats du modèle B.T sont moins bons que ceux des modèles T.T et M.T parce que Brill-Tagger fait aussi de nombreuses erreurs d'étiquetages. Par exemple, la phrase "`Adam ne donne à Ève pas que les pommes.`" est annotée comme `Adam/SBC :sg ne/ADV donne/SBC :sg à/PREP Ève/SBC :sg pas/ADV que/SUB les/DTN :pl pommes/SBC :pl ./..`

Le verbe «*donne*» est étiqueté comme *SBC*. Des erreurs comme celles-ci conduisent aux 314 phrases qui ont été analysées comme des phrases incorrectes. L'exemple de la figure 6.6 montre la raison pour laquelle nous avons obtenu plusieurs analyses pour cette phrase. Nous notons que le mot "la" est seulement classé par les modèles T.T, M.T et B.T en tant que

déterminant. Ainsi, il n'y a qu'une seule classe de CDG correspondant à cette UL : «`Det(Lex=art|pn)`», tandis que le modèle de base laisse toutes les classes de CDG pour cette UL. Plus précisément, le mot "la" a dans la grammaire trois classes de CDG différentes, parce que cette UL a différentes parties du discours dans L*efff* tels que *det*, *nc* et *cla* (voir le tableau 6.14).

Forme	Cat	Classe
la	cla	PN(Lex=pers,C=a)
la	cla	PN(Lex=pn,F=clit,C=a)
la	det	Det(Lex=art | pn)
la	nc	N(Lex=common)

Tableau 6.14 – *Les classes et les parties du discours de l'UL "la" dans la CDG*

Cette ambiguïté lexicale dans le modèle de base conduit à plusieurs analyses pour cette phrase. Cet exemple montre l'importance de désigner des étiquettes d'un étiqueteur morpho-syntaxique correctes à chaque mot dans la phrase qui doit être analysée.

Lorsque nous avons comparé les trois étiqueteurs, nous avons vu que le modèle M.T a une meilleure précision que le modèle T.T et le modèle B.T parce que les étiquettes du modèle M.T sont similaires aux parties du discours de L*efff*.

Le modèle T.T est meilleur que les autres modèles en termes de réduction d'ambiguïté, de temps d'analyse et du nombre d'étiquettes de dépendance correctes, principalement parce que l'analyseur parvient plus souvent à trouver au moins une analyse (pas toujours à 100% correcte).

D'une part, l'étiquetage des étiqueteurs morpho-syntaxiques réduit et limite l'espace de recherche pour l'analyseur et réduit aussi l'ambiguïté d'analyses. Les phrases sont également plus souvent complètement analysées par l'analyseur.

D'autre part, en utilisant l'étiquetage d'un étiqueteur nous avons perdu certaines analyses à cause des erreurs d'étiquetages des étiqueteurs et de la compatibilité entre les étiqueteurs et l'analyseur. En conséquence ces phrases ont été considérées comme des phrases incorrectes par l'analyseur.

6.4 Conclusion

Ce chapitre évalue le taux d'amélioration de l'analyse en dépendance à l'aide de trois modèles différents d'étiqueteurs morpho-syntaxiques. Ces modèles choisissent les classes grammaticales les plus probables pour une UL dans une phrase en fonction des étiquettes des étiqueteurs morpho-syntaxiques, bien que le coût est la perte de certaines analyses correctes. Nous avons réalisé deux expériences.
La première expérience effectuée sur un ensemble de test, se compose de 1443 phrases en français. Nos résultats expérimentaux ont démontré l'utilité des modèles d'analyse basés sur des étiqueteurs morpho-syntaxiques. Ces modèles ont obtenu des réductions appréciables du nombre d'arbres de dépendances et d'arbres de compositions par phrase.

Nous avons ensuite effectué une deuxième expérience pour le deuxième sous-ensemble de 184 phrases en français (corpus Paris 7). Notre modèle a aussi obtenu des réductions substantielles du nombre d'arbres de compositions par phrase.

Les résultats pourraient être meilleurs si nous pouvions trouver un étiqueteur syntaxique qui est complètement compatible avec L*efff* et la CDG du français. Par exemple, les prépositions complexes comme "au dessous" ne sont pas correctement reconnues par les étiqueteurs morpho-syntaxiques mais sont nécessaires pour la CDG du français.

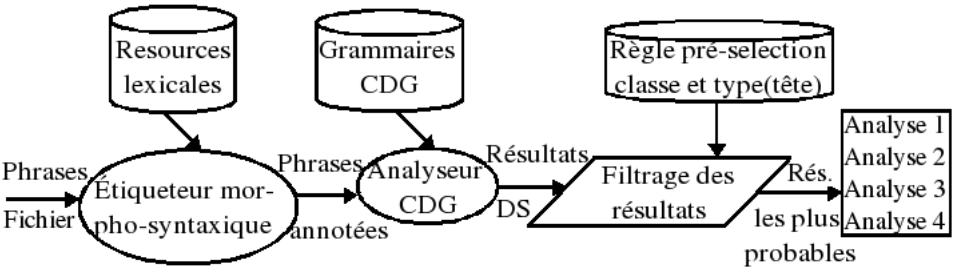


FIG. 6.8 – *Définir des oracles pour pré-sélection classe et type*

Adoptant une telle perspective, notre travail sur ce sujet vise à définir et réaliser des oracles stochastiques pour l'analyseur mixte pour pré-sélectionner la classe la plus probable de l'unité lexicale et pour pré-sélectionner la tête de la dépendance.
Le but sera de produire une représentation plus courte de l'ensemble des arbres d'analyses possibles pour la phrase analysée. Cette étape peut par exemple consister à présenter l'ensemble de la forêt d'analyses les plus probables. La figure 6.8 montre le modèle proposé.

Conclusion et perspectives

7

7.1 Rappel des enjeux

Nous dressons un bilan du travail effectué sur le lexique de la grammaire catégorielle de dépendance du français (CDG) et son utilisation pour l'analyse syntaxique en dépendances. Nous avons identifié deux enjeux importants. Le premier concerne le lexique de la CDG du français. Le deuxième évoque l'analyse en dépendances de la CDG. Nous terminons par quelques pistes afin d'améliorer le lexique de la CDG du français.

7.1.1 Contributions

Au cours de cette thèse, nous avons réalisé un travail qui concerne les grammaires catégorielles de dépendances du français. Nous pouvons résumer nos contributions en deux points décrits ci-après :
(1) *Construction du lexique de la base de données de la grammaire catégorielle de dépendance du français ;*
(2) *Proposition d'une approche permettant d'améliorer l'analyse en dépendances.*
Dans la deuxième partie et plus précisément dans le chapitre 4, nous avons présenté le processus de construction du lexique de la base de données de la CDG du français. Nous avons utilisé le lexique morpho-syntaxique du français Le*fff* (Sagot (2010)) pour constuire le lexique de la base de données. Notre objectif était de compléter le lexique de base de la CDG (*grammaire texte*). Pour ce faire, la première étape était de stocker Le*fff* dans la table `lexicon` de la base de données en le reliant avec les classes de CDG. Au cours de ce travail nous avons élaboré des algorithmes pour les grammaires catégorielles de dépendances. Nous avons résolu plusieurs problèmes pour l'analyse de ces grammaires à savoir l'absence de solution proposée par l'analyseur pour certaines phrases. Avec la grammaire de la base de données, il y a maintenant de fortes chances que l'analyseur produise au moins un résultat pour une phrase.

En revanche, nous sommes tombés sur un problème important : le Le*fff* n'est pas complet et contient des erreurs. Nous avons été obligé de définir d'autres tables pour essayer de compléter le lexique par certaines unités lexicales. Pour cela, nous avons utilisé différentes méthodes *manuelles, semi-automatiques* ou *automatiques*. Par exemple dans le chapitre 5, nous avons développé un algorithme pour ajouter au lexique de la base de données du français les noms déverbaux dans les classes lexicales qui correspondent aux cadres de sous-catégorisation déduits de ceux des verbes correspondants.

Enfin, le troisième volet traité dans la thèse porte sur l'amélioration de l'analyse syntaxique en dépendances. En effet, l'analyseur donne toutes les solutions compatibles avec une CDG. Donc la CDG du français génère des centaines de structures fallacieuses par phrase. Dans ce travail, nous avons présenté une approche qui réduit le taux d'ambiguïtés fallacieuses de la CDG du français malgré la perte de certaines analyses correctes. Pour cette approche, nous avons comparé l'utilisation de trois étiqueteurs morpho-syntaxiques différents. L'idée de cette approche est de choisir les classes grammaticales les plus probables pour les unités lexicales. Cet approche permet de filtrer le résultat des analyses pour chaque phrase. Notre étude conclut que la qualité de cette solution est basée principalement sur la compatibilité entre les unités lexicales qui sont définies par les étiqueteurs morpho-syntaxiques et la grammaire de dépendance. Malgré tout, les résultats expérimentaux ont montré que nos modèles sont plus performants, en termes de précision d'analyses que le modèle qui n'utilise pas d'étiqueteur morpho-syntaxique.

7.1.2 Travaux futurs

Les résultats présentés dans cette thèse pourront évidemment être améliorés. Nous envisageons les études suivantes.
Tout d'abord, nous voulons terminer le corpus de déverbaux en traitant toutes les règles que nous avons trouvées.

Il serait intéressant d'étudier le cas du *datif*, c'est-à-dire des groupes prépositionnels introduits par la préposition *à*. Cette étude nous permettra de préciser les classes nominatives des déverbaux.

Il serait aussi intéressant de réaliser des oracles stochastiques pour l'analyseur mixte stochastique-déterministe pour pré-sélectionner une et une seule classe (l'algorithme actuel peut sélectionner plusieurs classes pour une unité lexicale) et la tête de la dépendance la plus probable pour une unité lexicale. Le but sera de produire une représentation compacte de l'ensemble des arbres d'analyses possibles pour la phrase analysée. Cette étape peut par exemple consister à présenter l'ensemble des analyses les plus probables par une forêt d'arbres.

Pour conclure, les résultats pourraient être meilleurs si nous pouvions trouver un étiqueteur morpho-syntaxique qui soit complètement compatible avec L*efff* et la CDG du français. Par exemple, les prépositions complexes comme "au dessous" ne sont pas correctement reconnues par les étiqueteurs morphologiques que nous avons mais sont nécessaires pour la CDG du français.

Enfin, la dernière perspective concerne différentes grammaires disponibles actuellement dans le CDG lab : le français, le russe, l'allemand et le hollandais. Nous souhaitons développer une grammaire catégorielle de dépendance pour l'arabe.

Annexes

A

Les listes des règles des déverbaux

Ce Annexe se compose de la section A.1 qui illustre les listes des règles complètes pour les déverbaux pour toutes les terminaisons `er, ir` et `re`.

A.1 La liste des règles

Le tableau A.1 donne plus en détails sur la liste des règles de la terminaison "ir". Le tableau A.1 donne plus en détails sur la liste des règles de la terminaison "ir". La tableau A.1 donne plus en détails sur la liste des règles de la terminaison "re".

Liste de la terminaison "er"					
terminaison	suffixe	totale	terminaison	suffixe	totale
er	Empty	715	er	euses	786
er	ade	82	er	in	144
er	ades	82	er	ing	42
er	age	1156	er	is	202
er	ages	1155	er	isation	30
er	aie		er	isations	30
er	aies	6	er	ise	36
er	aille	60	er	isme	150
er	ailles	66	er	ité	98
er	ain	30	er	ition	96
er	aine	22	er	itions	96
er	aines	22	er	me	366
er	ains	30	er	oir	175
er	aire	172	er	oire	28
er	aison	42	er	oires	28
er	aisons	84	er	oirs	175
er	amini	4	er	oison	2
er	ance	103	er	oisons	2
er	ances	103	er	on	380
er	anderie	2	er	rice	19
er	anderies	2	er	rices	19
er	ant	403	er	sion	2
er	ante	174	er	sions	2
er	antes	174	er	sse	14
er	ants	402	er	ssion	4
er	at	100	er	ssions	4
er	ation	1093	er	tion	64
er	ations	1096	er	trice	5
er	cace	2	er	trices	5
er	e	3400	er	ure	233
er	é	1402	er	ures	233
er	ée	677	er	xion	4
er	ées	676	er	xions	4
er	ement	952	ser	e	42
er	ements	952	sser	s	78
er	ence	44	ter	Empty	99
er	ences	88	tter	t	28
er	erie	454	cer	Empty	13
er	eries	454	ffer	f	5
er	es	3462	ier	e	42
er	és	1402	ller	l	32
er	ette	450	mmer	m	4
er	ettes	450	ner	Empty	38
er	eur	1119	nner	n	225
er	eurs	1118	pper	p	3
er	euse	788	quer	c	50

Tableau A.1 – *Liste des terminaisons "er"*

Liste de la terminaison “ir”		
termination	suffixe	totale
ir	Empty	48
ir	ie	53
ir	iment	12
ir	ion	8
ir	issage	48
ir	issages	48
ir	issance	7
ir	issances	7
ir	issant	10
ir	issante	5
ir	issantes	5
ir	issants	11
ir	issement	160
ir	issements	160
ir	isseur	56
ir	isseurs	56
ir	isseuse	27
ir	isseuses	27
ir	issoir	8
ir	issoire	4
ir	issoires	4
ir	issoirs	8
ir	issure	14
ir	issures	14

Tableau A.2 – *Liste des terminaisons “ir”*

Liste de la terminaisons “re”		
terminaison	suffixe	totale
re	Empty	14
re	age	17
re	ages	16
re	aison	6
re	aisons	6
re	ance	11
re	ances	11
re	ant	34
re	ante	18
re	antes	18
re	ants	35
re	ation	3
re	ations	3
re	e	19
re	ement	17
re	ements	17
re	erie	14
re	eries	14
re	es	20
re	eur	54
re	eurs	54
re	euse	38
re	euses	38
re	ition	1
re	oir	16
re	oirs	16
re	ption	11
re	rice	2
re	rices	2
re	sade	3
re	sades	3
re	sage	5
re	sance	16
re	seur	18
re	sson	4
re	t	42
re	ure	9
re	ures	9
aire	action	17
dre	te	46
tre	Empty	11
ire	ction	23

Tableau A.3 – *Liste des terminaisons “re”*

Bibliographie

Anne Abeillé. *Une grammaire lexicalisée d'arbres adjoints pour le français : application à l'analyse automatique*. A.N.R.T, 1991. URL `http://books.google.com/books?id=1GSHZwEACAAJ`. (Cité page 30.)

Anne Abeillé. *Une grammaire électronique du français*. CNRS, Paris, 2002. (Cité page 37.)

Anne Abeillé et Nicolas Barrier. Enriching a french treebank. Dans *Proc. of LREC'04*, Lisbon, Portugal, 2004. (Cité pages 25 et 99.)

Anne Abeillé, Lionel Clément, et François Toussenel. Building a treebank for French. Dans A. Abeillé, éditeur, *Treebanks : Building and Using Parsed Corpora*, pages 165–188. Kluwer, Dordrecht, 2003. (Cité pages 73, 75, 81 et 82.)

Kazimierz Adjukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1 :1–27, 1935. (Cité page 1.)

Ramadan Alfared, Denis Béchet, et Alexander Dikovsky. CDG Lab : a toolbox for dependency grammars and dependency treebanks development. Dans K. Gerdes, E. Hajicova, et L. Wanner, éditeurs, *Proc. of the 1st Intern. Conf. on Dependency Linguistics (Depling 2011)*, Barcelona, Spain, 2011. http ://depling.org/proceedingsDepling2011/. (Cité pages 3 et 72.)

Giuseppe Attardi. Experiments with a Multilanguage Non-Projective Dependency Parser. Dans *CoNLLX*, 2006. (Cité pages 18 et 24.)

Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29 :47–58, 1953. (Cité pages 28 et 31.)

Yehoshua Bar-Hillel, Chaim Gaifman, et Eli Shamir. On categorial and phrase structure grammars. Dans Y. Bar-Hillel, éditeur, *Language and Information. Selected Essays on their Theory and Application*, pages 99–115. Addison-Wesley, Reading, MA, 1964. (Cité page 29.)

Denis Bechet, Alexander Dikovsky, et Annie Foret. Dependency Structure Grammars. Dans *Proceedings of the LACL 2005 Conference : Logical Aspects of Computational Linguistics, LNCS(LNAI) 3492*, pages 18–34. springer, 2005. (Cité page 1.)

Denis Béchet, Alexander Dikovsky, et Annie Foret. Sur les itérations dispersées et les choix itérés pour l'apprentissage incrémental des types dans les grammaires de dépendances. Dans *Conférence Francophone d'Apprentissage 2011 (CAP), Chambéry, France*, 2011. (Cité page 42.)

Emile Benveniste. *Problèmes de linguistique générale*. Gallimard, Paris, 1966. (Cité page 84.)

Igor Boguslavsky, Leonid Iomdin, Victor Sizov, Leonid Tsinman, et Vadim Petrochenkov. Rule-Based Dependency Parser Refined by Empirical and Corpus Statistics. Dans *Proc. of the 1st Intern. Conf. on Dependency Linguistics (Depling 2011)*, Barcelona, Spain, September 2011. (Cité page 12.)

Cristina Bosco et Vincenzo Lombardo. Dependency and relational structure in treebank annotation. Dans Geert-Jan M. Kruijff et Denys Duchier, éditeurs, *COLING 2004 Recent Advances in Dependency Grammar*, pages 1–8, Geneva, Switzerland, August 28 2004. COLING. (Cité page 25.)

Eric Brill. Some Advances in Transformation-Based Part of Speech Tagging. Dans *Proceedings of the twelfth national conference on artificial intelligence*, pages 722–727, 1994. (Cité page 97.)

Marie Candito, Benoît Crabbé, et Pascal Denis. Statistical French Dependency Parsing : Treebank Conversion and First Results. Dans Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, et Daniel Tapias, éditeurs, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010a. European Language Resources Association (ELRA). ISBN 2-9517408-6-7. (Cité pages 25, 26 et 102.)

Marie Candito, Benoît Crabbé, Pascal Denis, et François Guérin. Analyse syntaxique du français : des constituants aux dépendances. Dans *Actes de la 16e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2009)*, Senlis, France, Juin 2009. URL `http://hal.archives-ouvertes.fr/hal-00495287`. 10 pages. (Cité page 25.)

Marie Candito, Joakim Nivre, Pascal Denis, et Enrique Henestroza Anguiano. Benchmarking of Statistical Dependency Parsers for French, 2010b. (Cité pages 23 et 24.)

Marie-Hélène Candito. A Principle-Based Hierarchical Representation of LTAGs, 1996. (Cité page 31.)

Daniel Cer, Marie catherine De Marneffe, Daniel Jurafsky, et Christopher D. Manning. Parsing to stanford dependencies : Trade-offs between speed and accuracy. Dans *In LREC 2010*, 2010. (Cité page 24.)

Atanas Chanev, Kiril Simov, Petya Osenova, et Svetoslav Marinov. Dependency conversion and parsing of the BulTreeBank. Dans *Proc. of the LREC-Workshop Merging and Layering Linguistic Information*, 2006. (Cité page 25.)

Yuchang Cheng, Masayuki Asahara, et Yuji Matsumoto. Deterministic Dependency Structure Analyzer for Chinese. Dans *IJCNLP*, pages 500–508, 2004. (Cité page 18.)

Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2 :113–124, 1956. (Cité page 35.)

Y. J. Chu et T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14 :1396–1400, 1965. (Cité page 20.)

Lionel Clément, Bernard Lang, et Benoît Sagot. Morphology based automatic acquisition of large-coverage lexica. Dans *LREC 04*, pages 1841–1844, Lisbonne, Portugal, 2004. (Cité page 58.)

Lionnel Clément. XLFG : a parser to learn the LFG framework. Dans *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'01)*, Pittsburgh, États-Unis, 2001. (Cité page 37.)

John Cocke et Jacob T. Schwartz. *Programming Languages and their Compilers*. Courant Institute, New York University, 1970. (Cité page 30.)

Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4) :589–637, 2003. ISSN 0891-2017. (Cité page 11.)

Michael Collins, Lance Ramshaw, Jan Hajic, et Christoph Tillmann. A Statistical Parser for Czech, 1999. (Cité page 12.)

Michael John Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, 1999. (Cité page 2.)

Michael A. Covington. A fundamental algorithm for dependency parsing. Dans *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, 2001. (Cité page 17.)

Benoît Crabbé. *Représentation informatique de grammaires d'arbres fortement lexicalisées : le cas de la grammaire d'arbres adjoints*. PhD thesis, Université Nancy 2, Nancy, France, 2005. (Cité pages 30 et 37.)

Michael Dekhtyar et Alexander Dikovsky. Categorial Dependency Grammars. Dans M. Moortgat, éditeur, *Proceedings of Categorial Grammars 2004*, pages 76–91, 2004. (Cité page 2.)

Michael Dekhtyar et Alexander Dikovsky. Generalized categorial dependency grammars. Dans *Pillars of Computer Science'08*, pages 230–255, 2008. (Cité page 2.)

Pascal Denis et Benoît Sagot. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. Dans *Pacific Asia Conference on Language, Information and Computation*, Hong Kong, Chine, 2009. (Cité page 97.)

Alexander Dikovsky. Grammars for Local and Long Dependencies. Dans *ACL'01*, pages 156–163, 2001. (Cité pages 1 et 42.)

Alexander Dikovsky. Dependencies as Categories. Dans *"Recent Advances in Dependency Grammars". COLING'04 Workshop*, pages 90–97, 2004. (Cité pages 1, 12, 41 et 42.)

Alexander Dikovsky. Towards wide coverage categorical dependency grammars. Dans *Proceedings of the ESSLLI'2009 Workshop Parsing with Categorial Grammars - Parsing with Categorial Grammars Workshop ESSLLI 2009 Book of Abstracts*, pages 230–255, 2009. (Cité pages 2, 11, 39, 44 et 57.)

Alexander Dikovsky. Categorial Dependency Grammars : from Theory to Large Scale Grammars. Dans K. Gerdes, E. Hajicova, et L. Wanner, éditeurs, *Proc. of the 1st Intern. Conf. on Dependency Linguistics (Depling 2011)*, Barcelona, Spain, 2011. http ://depling.org/proceedingsDepling2011/. (Cité pages 63, 67 et 79.)

David Dowty. Thematic proto-roles and argument selection. *Language*, 67 (3) :547–619, 1991. (Cité page 79.)

Denys Duchier, Joseph Le Roux, et Yannick Parmentier. XMG : Un Compilateur de Méta-Grammaires Extensible. Dans *12e Conférence Annuelle sur leTraitement Automatiques des Langues Naturelles - TALN 2005*, pages 13–22, Dourdan, France, Jun 2005. URL `http://hal.inria.fr/inria-00000199/en/`. (Cité page 37.)

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, et Andreja Žele. Towards a Slovene dependency treebank. Dans *LREC2006*, 2006. (Cité page 25.)

Jay Earley. An Efficient Context-Free Parsing Algorithm. *Commun. ACM*, pages 94–102, 1970. (Cité page 30.)

Gülşen Eryiğit et Kemal Oflazer. Statistical Dependency Parsing for Turkish. Dans *Proceedings of the 11th eacl'06*, pages 89–96, Trento, 3-7 April 2006. (Cité page 12.)

Charles J. Fillmore. The case for case. Dans E. Bach et R. T. Harms, éditeurs, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Winston, 1968. (Cité page 79.)

Charles J. Fillmore. The case for case reopened. Dans P. Cole et J. M. Sadock, éditeurs, *Syntax and Semantics 8 : Grammatical Relations*, pages 59–81. New York : Academic Press, 1977. (Cité page 79.)

Claire Gardent. Evaluating an automatically extracted lexicon. Dans *Proceedings of 4nd Language and Technology Conference*. Poznan, Poland, 2009. (Cité page 57.)

Claire Gardent, Bruno Guillaume, Guy Perrier, et Ingrid Falk. Extraction d'information de sous-catégorisation à partir des tables du ladl. Dans *Actes de la conférence TALN 2012*, Nancy, France, avr 2008. (Cité page 57.)

Jane Grimshaw. *Argument Structure*. MIT Press, Cambridge, Mass., 1990. (Cité page 79.)

Maurice Gross. Methodes en syntaxe. Dans *Hermann*, page 142, Paris, France, 1975. (Cité page 57.)

Jan Hajič et Petr Zemánek. Prague Arabic dependency treebank : Development in data and tools. Dans *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117, 2004. (Cité page 25.)

Nabil Hathout, Fiammetta Namer, et Georgette Dal. An Experimental Constructional Database : The MorTAL Project. Dans Paul Boucher, éditeur, *Many Morphologies*, pages 178–209. Cascadilla, Somerville, Mass., 2002. (Cité page 75.)

Lydia-Mai Ho-Dac. *La position initiale dans l'organisation du discours, une exploration en corpus*. PhD thesis, Université Toulouse-le Mirail, France, 2007. (Cité pages 73 et 75.)

Nancy Ide et Jean Véronis. MULTEXT : Multilingual Text Tools and Corpora. Dans *COLING'94*, pages 588–592, 1994. (Cité page 37.)

Aravind K. Joshi et Phil Hopely. A parser from antiquity. *Nat. Lang. Eng.*, 2 :291–294, December 1996. ISSN 1351-3249. URL `http://dx.doi.org/10.1017/S1351324997001538`. (Cité page 11.)

Aravind K. Joshi, Leon S. Levy, et Masako Takahashi. Tree Adjunct Grammars. *J. Comput. Syst. Sci.*, 10(1) :136–163, 1975. (Cité page 37.)

Sylvain Kahane, Alexis Nasr T, et Owen Rambow. Pseudo-projectivity : a polynomially parsable non-projective dependency grammar. Dans *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL '98*, pages 646–652, 1998. (Cité page 26.)

Laura Kallmeyer, Timm Lichte, Wolfgang Maier, Yannick Parmentier, et Johannes Dellert. Developing a TT-MCTAG for German with an RCG-based Parser. Dans Nicoletta Calzolari (Conference Chair) Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, et Daniel Tapias, éditeurs, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). ISBN 2-9517408-4-0. http ://www.lrec-conf.org/proceedings/lrec2008/. (Cité page 37.)

Tadao Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. Rapport Technique AFCRL-65-758, Air Force Cambridge Research Laboratory, 1965. (Cité page 30.)

Kotaro Kitagawa et Kumiko Tanaka-Ishii. Tree-Based Deterministic Dependency Parsing — An Application to Nivre's Method —. Dans *Proceedings of the ACL 2010 Conference Short Papers*, pages 189–193, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P10-2035`. (Cité page 18.)

Dan Klein et Christopher D. Manning. Accurate Unlexicalized Parsing. Dans *Proceedings OF THE 41ST ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, pages 423–430, 2003. (Cité pages 2 et 11.)

Anthony Kroch et Aravind Joshi. The linguistic relevance of tree adjoining grammars. Technical report MS-CIS-85-16, Department of Computer and Information Science, University of Pennsylvania, 1985. (Cité page 30.)

Taku Kudo et Yuji Matsumoto. Japanese Dependency Analysis using Cascaded Chunking, 2002. (Cité pages 17 et 18.)

Anna Kupsc et Anne Abeillé. Growing TreeLex. Dans Alexander F. Gelbukh, éditeur, *CICLing*, volume 4919 de *Lecture Notes in Computer Science*, pages 28–39. Springer, 2008. ISBN 978-3-540-78134-9. (Cité page 57.)

Joachim Lambek. The Mathematics of Sentence Structure. *American Mathematical Monthly*, 65(3) :154–170, 1958. (Cité page 29.)

Christian Leclère. Organization of the lexicon-grammar of French verbs. volume 25, pages 29–48, Paris, France, 2002. (Cité page 57.)

David M. Magerman. Statistical Decision-Tree Models for Parsing. Dans *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, 1995. (Cité page 25.)

Mitchell P. Marcus, Beatrice Santorini, et Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English : The Penn Treebank. *COMPUTATIONAL LINGUISTICS*, 19(2) :313–330, 1993. (Cité page 14.)

Svetoslav Marinov et Joakim Nivre. A data-driven dependency parser for Bulgarian. Dans *Proceedings of TLT*, pages 89–100, 2005. (Cité page 12.)

Ryan McDonald. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis, University of Pennsylvania, jul 2006. (Cité pages 23 et 103.)

Igor Mel'čuk. Vers une linguistique Sens-Texte. Paris- Collège de France, 1997. (Cité page 1.)

Igor Mel'čuk. Dependency in Linguistic Description, 2003. (Cité page 12.)

Igor Mel'čuk. Actants in semantics and syntax 1,2. *Linguistics*, 42(1,2), 2004. (Cité page 79.)

Igor Mel'čuk et Alexander Holodovič. To the theory of grammatical voice : (definition, calculus). Dans *Problemy lingvisticheskoj tipologii i struktury jazyka*. Nauka, Leningrad, 1970. (Rus.). (Cité page 80.)

Cédric Messiant, Anna Korhonen, et Thierry Poibeau. LexSchem : A Large Subcategorization Lexicon for French Verbs. Dans *Proceedings of the Language Resources and Evaluation Conference (LREC'2008)*, page 142, Marrakech, Maroc, 2008. (Cité page 57.)

Fiametta Namer. *Morphologie, lexique et traitement automatique des langues*. Hermes Science, Lavoisier, 2009. (Cité page 73.)

Jens Nilsson. Graph transformations in data-driven dependency parsing. Dans *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, 2006. (Cité page 18.)

Joakim Nivre. An Efficient Algorithm for Projective Dependency Parsing. Dans *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160, 2003. (Cité page 17.)

Joakim Nivre. Inductive Dependency Parsing of Natural Language Text. Rapport technique, 2005. (Cité page 24.)

Joakim Nivre. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4) :513–553, 2008. (Cité pages 11, 17 et 24.)

Joakim Nivre, Johan Hall, et Jens Nilsson. Memory-Based Dependency Parsing. Dans Hwee Tou Ng et Ellen Riloff, éditeurs, *HLT-NAACL 2004 Workshop : Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 49–56, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics. (Cité page 18.)

Joakim Nivre, Johan Hall, et Jens Nilsson. MaltParser : A Data-Driven Parser-Generator for Dependency Parsing. Dans *Language Resources and Evaluation LREC'2006*, pages 2216–2219, 2006. (Cité page 24.)

Joakim Nivre, Sandra Kübler, et Ryan McDonald. Dependency parsing. Chapitre Graph-based Parsing. Morgan & Claypool, 2009. (Cité pages 12, 13, 20 et 21.)

Joakim Nivre et Jens Nilsson. Three Algorithms for Deterministic Dependency Parsing, 2003. (Cité page 19.)

Joakim Nivre et Jens Nilsson. Pseudo-Projective Dependency Parsing. Dans *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P05-1013`. (Cité page 24.)

Yannick Parmentier. *SemTAG : Une plate-forme pour le calcul sémantique à partir de grammaires d'arbres adjoints*. PhD thesis, Université Henri Poincaré, Nancy I, 2007. (Cité page 35.)

Slav Petrov, Leon Barrett, Romain Thibaux, et Dan Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. Dans *ACL '06*, pages 433–440, 2006. (Cité pages 24 et 102.)

Thierry Poibeau et Cédric Messiant. Le dictionnaire de valence Dicovalence : manuel d'utilisation, 2006. (Cité page 59.)

Eric Poupard, Denis Béchet, et Annie Foret. Acquisition d'une grammaire catégorielle depuis un corpus d'arbre en français. Dans *Actes de la Conférence d'Apprentissage 2006 (CAp'2006)*, 2006a. (Cité pages 26 et 125.)

Eric Poupard, Denis Béchet, et Annie Foret. Categorial Grammar Acquisition from a French Treebank, 2006b. (Long version in English of Poupard et al. (2006a)). (Cité page 26.)

Azim Roussanaly, Benoît Crabbé, et Jérôme Perrin. Premier bilan de la participation du loria à la campagne d'évaluation easy. Dans *Proceedings of the TALN'05*, pages 49–52, Dourdan, France, 2005. (Cité page 37.)

Benoît Sagot. Automatic Acquisition of a Slovak Lexicon from a Raw Corpus, September 2005. (Cité page 58.)

Benoît Sagot. *Analyse automatique du français : lexiques, formalismes, analyseurs*. PhD thesis, Université Paris 7, apr 2006. PhD thesis, supervized by Laurence Danlos (Lattice, Université Paris 7) and co-supervized by Éric de La Clergerie (Atoll, INRIA Rocquencourt). (Cité page 58.)

Benoît Sagot. The Lefff, a Freely Available and Large-coverage Morphological and Syntactic Lexicon for French. Dans Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, et Daniel Tapias, éditeurs, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7. (Cité pages 57, 59, 75 et 109.)

Benoît Sagot, Lionel Clément, Éric Villemonte de La Clergerie, et Pierre Boullier. The Lefff 2 syntactic lexicon for French : architecture, acquisition, use. Dans *Proc. of LREC'06*, 2006. URL `http://atoll.inria.fr/~sagot/pub/LREC06b.pdf`. (Cité page 57.)

Benoît Sagot et Karen Fort. Améliorer un lexique syntaxique à l'aide des tables du Lexique-Grammaire : Adverbes en -ment. Dans *26e Colloque International sur le Lexique et la grammaire 2007*, Bonifacio, France, 2007. URL `http://hal.inria.fr/inria-00186779/en/`. (Cité page 59.)

Morris Salkoff et André Valli. A dictionary of french verbal complementation. Dans *Proceedings of 2nd Language and Technology Conference*, Poznan, Poland, 2005. Human Language and Technologies as a Challenge for Computer Science and Linguistics. In memory of M. Gross and A. Zampolli. (Cité page 57.)

Yves Schabes. *Mathematical and computational aspects of lexicalized grammars*. PhD thesis, University of Pennsylvany, Philadelphie, 1990. (Cité page 30.)

Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. Dans *Proceedings of International Conference on New Methods in Language Processing, Manchester, UK*, 1994. (Cité page 97.)

Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8 :333–343, 1985. (Cité page 33.)

Klaas Sikkel et Anton Nijholt. Parsing of Context-Free Languages. Dans *Rozenberg G., Salomaa A. (Eds), Handbook of Formal Languages*, pages 61–100. Springer-Verlag, 1997. (Cité page 30.)

Kiril Simov, Petya Osenova, Alexander Simov, et Milen Kouylekov. Design and implementation of the Bulgarian HPSG-based treebank. Dans *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers, 2005. (Cité page 25.)

Daniel D. Sleator et Davy Temperley. Parsing English with a Link Grammar, 1991. (Cité page 26.)

Ludovic Tanguy et Nabil Hathout. Webaffix : un outil d'acquisition morphologique dérivationnelle à partir du web. Dans Jean-Marie Pierrel, éditeur, *Actes de la 9e Conférence Annuelle sur le Traitement Automatique des Langues Naturelles (TALN-2002)*, pages 245–254, Nancy, 2002. ATALA. (Cité page 75.)

Lucien Tesnière. *Éleménts de syntaxe structurale*. Klincksieck, Paris, 1959. (Cité pages 1 et 12.)

Elsa Tolone. *Analyse syntaxique à l'aide des tables du Lexique-Grammaire du français*. PhD thesis, Université Paris-Est, mar 2011. (Cité page 57.)

Karel van den Eynde et Piet Mertens. La valence : l'approche pronominale et son application au lexique verbal. *Journal of French Language Studies*, 13 :63–104, 2003. (Cité pages 57 et 79.)

Robert D. Van Valin, Jr. Generalized semantic roles and the syntax-semantics interface. Dans F. Corblin, C. Dobrovie-Sorin, et J.-M. Marandin, éditeurs, *Empirical Issues in Formal Syntax and Semantics 2 : Selected papers from the Colloque de Syntaxe et Semantique à Paris*, pages 373–388. Peter Lang, 1997. (Cité page 79.)

K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania, 1987. (Cité pages 30 et 37.)

Éric Villemonte de La Clergerie et Miguel Alonso Pardo. A tabular interpretation of a class of 2-stack automata. Dans *Proc. of ACL/COLING'98*, aug 1998. (Cité page 35.)

XTAG-Research-Group. A Lexicalized Tree Adjoining Grammar for English, 1995. (Cité pages 30 et 37.)

Hiroyasu Yamada et Yuji Matsumoto. Statistical Dependency Analysis with Support Vector Machines. Dans *Proceedings of IWPT*, pages 195–206, 2003. (Cité pages 17, 18 et 24.)

SinWon Yoon. Using a meta-grammar for LTAG Korean grammar. Dans *Proceedings of the seventh International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+7)*, Vancouver, BC, Canada, 2004. (Cité page 37.)

Daniel H. Younger. Recognition and Parsing of Context-Free Languages in Time n3. *Information and Control*, pages 189–208, 1967. (Cité page 30.)

Ce document a été pérparé à l'aide de l'éditeur Kile et le modèle de thèse
élaboré par Julien Chiquet
(http ://stat.genopole.cnrs.fr/members/jchiquet/theselatex).

Acquisition de grammaire catégorielle de dépendances de grande envergure

Ce travail est une étude qui s'inscrit dans le cadre de la création d'un lexique complet d'une grammaire catégorielle de dépendance du français (CDG) et s'inscrit aussi dans le cadre de l'analyse mixte stochastique-déterministe des grammaires de dépendances de grande envergure. En particulier, nous élaborons des algorithmes pour améliorer le lexique de base de la CDG du français. Nous résolvons plusieurs problèmes pour l'analyse avec cette grammaire à savoir l'absence de solution proposée par l'analyseur pour certaines phrases. Nous présentons un algorithme **proto-déverb** qui permet de compléter le lexique de la CDG du français en plaçant les déverbaux dans les classes lexicales qui correspondent à leurs cadres de sous-catégorisation déduits de ceux des verbes correspondants. Le second problème auquel nous nous intéressons provient du fait que l'analyseur de CDG donne actuellement toutes les solutions compatibles avec une CDG. Nous proposons une approche de filtrage qui consiste à utiliser un étiqueteur morpho-syntaxique pour choisir les classes grammaticales les plus probables des unités lexicales. Notre approche permet de réduire de manière significative le taux d'ambiguïtés fallacieuses de la CDG. Notre étude conclue que la qualité de cette solution est basée principalement sur la compatibilité entre les unités lexicales qui sont définies par les étiqueteurs morpho-syntaxiques et la grammaire de dépendance.

Mots-clés : analyses en dépendances, grammaires de dépendances, unités lexicales, étiqueteur morpho-syntaxique, déverbaux, cadres de sous-catégorisation.

Learning large-scale categorial dependency grammars

This work is a study that is part of the creation of the lexicon of a categorial dependency grammars (CDG) for French and also part of a mixed stochastic-deterministic analysis for large-scale dependency grammars. In particular we develop algorithms for CDG to improve the existing lexicon of the French CDG. We solve several problems for the analysis of these grammars for example, the absence of analysis proposed by the parser for some sentences. We present an algorithm **proto-déverb** which allows to complete the lexicon of the French CDG by using the sub-categorisation frame for deverbals. The second problem we consider is the fact that the CDG parser currently provides all the compatible solutions for a CDG. We propose a filtering approach to improve dependency parsing. We show that using a morpho-syntactic tagger that chooses the most probable grammatical classes for each lexical unit, we can significantly reduce the rate of ambiguities of the French CDG. Our study concluded that the adequacy of these solutions is mainly based on the compatibility between the lexical units defined by the taggers and the dependency grammar.

Keywords: Dependency parsing, Dependency grammars, Lexical units, POS taggers, Deverbal, Sub-categorisation frame