Méthodes d'analyse supervisée pour l'interface syntaxe-sémantique

Corentin Ribeyre

▶ **To cite this version:**

Corentin Ribeyre. Méthodes d'analyse supervisée pour l'interface syntaxe-sémantique. Informatique et langage [cs.CL]. Université Paris Diderot, 2016. Français. NNT : . tel-01323245

HAL Id: tel-01323245
https://hal.science/tel-01323245v1

Submitted on 30 May 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

UNIVERSITÉ SORBONNE PARIS CITÉ

UNIVERSITÉ PARIS DIDEROT

université PARIS DIDEROT PARIS 7

École doctorale de Sciences du Langage

Laboratoire ALPAGE

DOCTORAT

Sciences du Langage – Linguistique Informatique

Corentin Ribeyre

Méthodes d'analyse supervisée pour l'interface syntaxe-sémantique

De la réécriture de graphes à l'analyse par transitions

Data-driven methods for syntax-semantic interface

From graph rewriting to transition-based parsing

Thèse dirigée par Laurence Danlos

Soutenue le 27 Janvier 2016

JURY

Pr. Laurence Danlos	Université Paris Diderot (Paris 7)	Directrice
Dr. Djamé Seddah	Université Paris Sorbonne (Paris 4)	Encadrant
Dr. Éric Villemonte de La Clergerie	Inria	Encadrant
Pr. Sylvain Kahane	Université Paris Ouest (Paris 10)	Examinateur
Pr. John A. Caroll	University of Sussex	Rapporteur
Pr. Paola Merlo	Université de Genève	Rapporteure

Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher.

Antoine de Saint-Exupéry
Terre des hommes

Résumé

Aujourd'hui, le volume de données textuelles disponibles est colossal. Ces données représentent une manne d'informations inestimables qu'il n'est pas possible de traiter manuellement. De fait, il est essentiel d'utiliser des techniques de Traitement Automatique des Langues pour arriver à extraire les informations saillantes et comprendre le sens sous-jacent. Cette thèse s'inscrit dans cette perspective et propose des ressources, des modèles et des méthodes pour permettre : (i) l'annotation automatique de corpus à l'interface entre la syntaxe et la sémantique afin d'en extraire la structure argumentale liant les prédicats (verbaux) à leurs arguments (ii) l'exploitation de ces ressources grâce à des méthodes efficaces.

Dans un premier temps, nous proposons un système de réécriture de graphes, ainsi qu'un ensemble de règles de réécriture manuellement écrites permettant l'annotation automatique de la syntaxe profonde du français. Grâce à cette approche, deux corpus ont vu le jour, à savoir le DeepSequoia, une version profonde du corpus Séquoia (Candito et Seddah, 2012b) et le DeepFTB, une version profonde du French Treebank (Abeillé et al., 2003) en dépendances (Candito et al., 2010).

Dans un second temps, nous proposons deux extensions d'analyseurs par transitions (Nivre, 2005) et les adaptons à l'analyse de graphes. Nous développons également un ensemble de traits linguistiquement riches issus d'analyses syntaxiques. L'idée est d'apporter des informations topologiquement diversifiées donnant à nos analyseurs les indices nécessaires pour une prédiction performante de la structure argumentale. Couplé enfin à un analyseur par factorisation d'arcs (McDonald et al., 2005; Martins et Almeida, 2014), cet ensemble de traits permet d'établir l'état de l'art sur le français et de dépasser celui préalablement établi pour les corpus DM (Flickinger et al., 2012; Ivanova et al., 2012) et PAS (Miyao et al., 2004) sur l'anglais (Oepen et al., 2014).

Enfin, nous explorons succinctement une méthode d'induction automatisant le passage d'une représentation surfacique (un arbre) vers une représentation sémantique (un graphe), méthode qui complète alors notre ensemble cohérent de ressources et modèles proposés pour l'analyse de l'interface syntaxe-sémantique sur le français et l'anglais.

Abstract

Nowadays, the amount of textual data has become so gigantic, that it is not possible to deal with it manually. In fact, it is now necessary to use Natural Language Processing techniques to extract useful information from these data and understand their underlying meaning. In this thesis, we offer resources, models and methods to allow : (i) the automatic annotation of deep syntactic corpora to extract argument structure that links (verbal) predicates to their arguments (ii) the use of these resources with the help of efficient methods.

First, we develop a graph rewriting system and a set of manually-designed rewriting rules to automatically annotate deep syntax in French. Thanks to this approach, two corpora were created : the DeepSequoia, a deep syntactic version of the Séquoia corpus (Candito et Seddah, 2012b) and the DeepFTB, a deep syntactic version of the dependency version (Candito et al., 2010) of the French Treebank (Abeillé et al., 2003).

Next, we extend two transition-based parsers (Nivre, 2005) and adapt them to be able to deal with graph structures. We also develop a set of rich linguistic features extracted from various syntactic trees. We think they are useful to bring different kind of topological information to accurately predict predicat-argument structures. Used in an arc-factored second-order parsing model (McDonald et al., 2005; Martins et Almeida, 2014), this set of features gives the first state-of-the-art results on French and outperforms the one established on the DM (Flickinger et al., 2012; Ivanova et al., 2012) and PAS (Miyao et al., 2004) corpora for English (Oepen et al., 2014).

Finally, we briefly explore a method to automatically induce the transformation between a tree and a graph. This completes our set of coherent resources and models to automatically analyze the syntax-semantics interface on French and English.

Remerciements

La thèse peut être considérée *a posteriori* comme un travail long, parfois laborieux, souvent enthousiasmant et qui, petit à petit et au fur et à mesure, projette le primo-chercheur au rang d'expert dans son domaine. Voilà pour la théorie. Elle est bien belle ! Mais la thèse c'est avant tout des gens, des rencontres, des discussions et beaucoup de temps, pour tout le monde. Ainsi, on [1] ne saurait trouver une thèse complète (et complètement finie) sans les quelques remerciements, que l'on appelle couramment d'usage, mais, qui pour ce manuscrit et ces trois dernières années passées à Alpage, auront le mérite d'être parfaitement sincères.

En premier lieu, on souhaiterait remercier le jury qui compose (ou a composé) cette thèse. Merci à Paola Merlo d'avoir accepté de relire le manuscrit et d'en être la rapporteure, merci aussi à elle de m'accueillir dans son équipe pour les deux prochaines années. Merci à John Carroll d'avoir accepté de relire le manuscrit et d'en être le rapporteur, merci à lui d'avoir fait l'effort de comprendre mon français parfois alambiqué. Merci aussi à Sylvain Kahane pour son suivi externe mais constant, durant ces trois dernières années, pour ses relectures, ses conseils et d'avoir accepté d'être présent dans le jury. Merci ensuite aux trois personnes sans qui cette thèse n'aurait pas pu voir le jour : Laurence Danlos pour m'avoir accueilli au sein d'Alpage à l'époque où elle était encore directrice et pour avoir accepté d'être ma directrice. On la remercie particulièrement pour son franc parlé et ses remarques éclairées sur nombre de points de cette thèse et pour m'avoir appris à servir un verre de vin. Djamé Seddah, ensuite, pour ses conseils, sa rigueur expérimentale, nos discussions jusqu'à pas d'heures, nos nuits deadlines, clopes (pour lui), pizzas, cafés. Merci encore pour les hacks Unix extrêmes et les démontages de Mac avec plus de poussières que de composants. Merci enfin pour sa disponibilité sans faille à toute heure du jour et surtout de la nuit. Eric (Villemonte) de La Clergerie enfin pour ses conseils, sa rigueur formelle, nos discussions les mardis et les mercredis sur les dernières technologies Web, ses explications mathématiques, pour m'avoir toujours permis de rentrer n'importe quand dans son bureau pour une aide, une question, avec toujours l'impression qu'on ne le dérangeait pas.

Dans un second temps, on souhaiterait remercier chaleureusement les permanents d'Alpage qui font vivre le laboratoire et sont toujours à l'écoute. On remercie en particulier Marie Candito pour m'avoir aidé longuement sur les problèmes linguistiques et pour avoir été très compréhensive au moment où je n'en pouvais plus. Merci aussi pour sa disponibilité, sa rigueur, mais encore pour m'avoir aidé à concevoir les règles d'OGRE, ce qui fut, sans l'ombre d'un doute, une tâche longue et difficile. Ensuite, on remercie aussi Benoît Crabbé pour cette fameuse soirée à COLING 2014 où j'ai compris ce que ça voulait dire "chanter Brel". Ensuite, on ne saurait avoir remercié toute l'équipe sans avoir dit merci aux doctorants et post-doctorants du laboratoire sans qui cette thèse n'aurait pas été la même. Merci à Charlotte, Valérie, Sarah, Maximin, Timothée, Pierre, Charles pour toutes les discussions et posters dans le bureau. Merci aussi à Christelle Guiziou, notre gestion-

1. La thèse a été rédigée sans l'usage de *on* pour des raisons stylistiques, mais comme on aime beaucoup ce clitique, on a choisi de l'utiliser massivement dans nos remerciements.

naire, qui a toujours été d'un grand secours dans le traitement des missions, avec qui on a longuement discuté de tout. Enfin, on remercie aussi Pascal Amsili. Merci pour nos longues discussions qui partaient dans tous les sens, pour m'avoir donné la possibilité de faire les cours de langages formels, mais aussi d'encadrer les projets à ses côtés. Merci encore pour sa disponibilité et merci surtout pour m'avoir donné ma chance au sein du cursus, soyons honnête cinq minutes, sans lui, je ne serai pas là à écrire ces remerciements aujourd'hui. Merci aussi d'être vintage dans son utilisation quotidienne de logiciels, *Mutt* sous Mac OS 10.10, ça n'a pas prix !

Ensuite, il y a les amis, bien sûr, ceux qui sont toujours là, plus ou moins loin, plus ou moins souvent, mais sur qui l'on peut compter. Merci à Chloé pour son soutien, son amitié, ses crumbles, ... Quelque part, on a été dans le même navire. Molière, s'il nous avait regardés, eût dit : « Mais qu'allaient-ils faire dans cette galère ? » Merci d'avoir été là dans les bons moments mais aussi dans les pires. Merci aussi de m'avoir pardonné ma mauvaise blague trois jours avant ta soutenance. Merci à Cloé pour tout et bien plus encore, pour m'avoir aidé quand j'étais au fond du gouffre, pour être celle qui appelle, parce qu'on ne peut pas prendre des nouvelles autrement, merci, vraiment... Merci encore à Ségolène dite Ségo et Salomé pour m'avoir fait oublier, à chaque fois, autour d'un verre ou d'un plat ce que c'était que de vivre à Paris. Merci pour votre soutien de jadis et naguère [2]. Merci à toi, Salomé, d'avoir essayé de comprendre ce que je faisais, inlassablement, au fil des années. Merci ensuite à Clarisse pour nos discussions téléphoniques ultra longues, pour ta gentillesse et pour être la seule à m'offrir de la noix de coco. Merci après à Pierre pour sa rigueur, son organisation, pour être sans doute celui qui comprend le mieux ce que je fais sans avoir jamais été du domaine. Merci aussi pour nos discussions, sur la vie (privée), la politique, la littérature et pour me permettre de parler informatique plus que de raison. Merci à Marion pour avoir toujours voulu savoir comment ça allait, pour m'avoir pardonné de la distraire si souvent quand elle bossait derrière moi, pour être là quand il faut, tout simplement. Merci aussi à Rafie pour prendre toujours des nouvelles et être compréhensive quand je n'en donne pas, merci d'avoir été si souvent attentive au téléphone et si gentille quand j'en avais besoin. Merci aussi d'être d'une disponibilité et d'une bonne humeur sans faille. Merci enfin à Marianne pour être toujours au top, pour nos longues discussions, pour sa gentillesse, sa disponibilité, pour s'intéresser aux gens comme elle le fait – chose rare mais essentielle aujourd'hui – et pour m'héberger n'importe quand et avoir cru que je pouvais être son ingénieur !

Après, il y a la famille, ceux qui nous aiment quand même. Merci d'abord à papa et maman, plus couramment Pascal et Catherine pour leur aide, leur soutien, leur disponibilité, leur gentillesse, pour tout et tellement plus. Merci à mes frères : Quentin d'abord d'avoir refait nombre de preuves avec moi et de m'avoir expliqué moults calculs. Merci aussi de m'avoir montré ce qu'était une descente du gradient avant même que je ne comprenne à quoi ça sert. Antonin ensuite, merci de m'envoyer les bandes-annonces de la nouvelle série tendance qui pourrait me plaire et merci de faire du feu dans la cheminée comme personne. Virgil enfin, merci de faire autant de câlins à son « gros Coco » et d'avoir installé

2. Au sens étymologique du terme

TeamViewer pour m'éviter d'avoir à traverser le jardin à chaque fois. Merci à mes sœurs maintenant : Garance et Apolline qui un jour m'ont demandé ce que je faisais exactement et qui ont eu l'air de comprendre mieux que certains, du haut de leurs dix ans. Merci aussi de garder mes compétences en électronique au top en cassant toujours quelque chose ou de compter les cadeaux de Noël comme personne. Merci à ma grand-mère dite « mamie » qui s'est toujours inquiétée de savoir si j'allais bien, si je ne manquais de rien, qui m'a toujours financé un livre quand j'en avais besoin et qui est toujours contente de me voir rentrer au fil des mois. Merci enfin à Clara, sans qui rien n'aurait été pareil. Merci pour son soutien sans faille au fil des années, même quand son ours était un peu trop grognon. Merci de prendre soin de moi quand je suis malade et merci d'aller braver le froid et la pluie pour m'acheter mes palmiers parce que « je rédige ». Merci d'être là au jour le jour tout simplement et de m'avoir appris à aimer le thé.

Enfin, merci à tous ceux qui ont un jour croisé ma route et qui m'ont apporté, de quelque manière que ce soit, beaucoup...

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des notions formelles

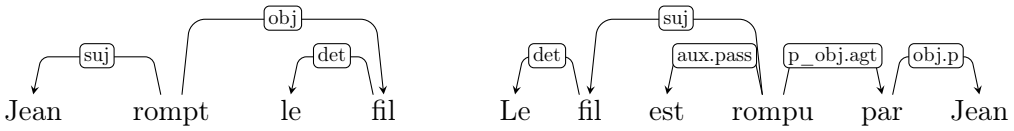
Chapitre

1 Introduction

Aujourd'hui, avec la développement rapide des « nouvelles technologies », beaucoup considèrent que la grande « révolution » du XXI^{e} siècle, c'est l'information (Curien et Muet, 2004). Toujours plus d'informations signifie des flux textuels ininterrompus agrégeant une masse gigantesque de connaissances. Les enjeux politiques et économiques des dix dernières années ont montré l'intérêt de traiter ces flux de manière automatique pour en retirer les informations saillantes. Cette tâche représente un défi majeur car elle est au carrefour entre la compréhension des langues naturelles et le développement de méthodes informatiques efficaces. En effet, l'essence même du langage est d'exprimer une pensée et de communiquer. Il est ainsi doté d'une *sémantique* et d'une *syntaxe*. Communiquer revient à établir une relation avec autrui, à transmettre *quelque chose* à *quelqu'un*. Pour que la communication entre deux personnes soit possible, il faut qu'ils partagent des associations entre des formes et des sens. La relation entre ces formes et ces sens peut être ambiguë (une forme peut avoir plusieurs sens) ; de même, un sens peut être exprimé de plusieurs manières (paraphrases). Explorer cette relation, c'est comprendre les structures qui la composent, à savoir la *syntaxe* et la *sémantique*, mais aussi la manière dont elles interagissent entre elles. C'est, selon nous, la clé de voûte permettant de passer d'un simple flux textuel à une information structurée, exploitable et qui fait sens.

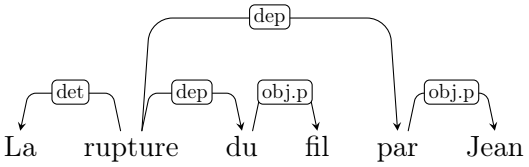
C'est la raison pour laquelle, nous situons notre travail à l'interface entre la syntaxe et la sémantique, à un niveau bien plus modeste et moins abstrait que celui de l'expression d'un sens sous forme d'une formule logique, mais plus complexe et plus général que l'utilisation d'une structure arborescente pour représenter la structure syntaxique d'un énoncé. En effet, le phénomène de paraphrase est largement répandu dans la langue naturelle. Par exemple, la phrase *Jean rompt le fil* peut être exprimée de différentes manières, toutes avec des représentations syntaxiques qui ne sont pas équivalentes comme le montre la Figure 1.1. Néanmoins, le sens exprimé par ces différents énoncés est globalement le même. Sans avoir recours à la logique du premier ordre, nous cherchons une représentation suffisamment expressive pour pouvoir rendre compte du dénominateur commun de ces énoncés :

la **structure argumentale** de la phrase encore appelée **structure prédicat-argument**.



(a) Construction active.

(b) Construction passive.



(c) Nominalisation verbale.

Figure 1.1: Diverses manières d'exprimer *Jean rompt le fil.*

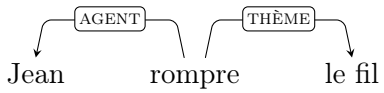


Figure 1.2: Structure argumentale uniforme pour les constructions de la Figure 1.1.

Elle a pour objet de neutraliser les alternances syntaxiques (active, passive, ...) en jeu dans les différentes phrases présentées à la Figure 1.1 et donc de produire une structure uniforme (Figure 1.2). La structure argumentale peut s'exprimer en terme de *réseau sémantique* (Mel'čuk et Polguère, 2008) ou en terme de prédicat logique de la manière suivante rompre'(AGENT = jean', THÈME = le_fil') [1], les deux représentations étant équivalentes. Par ailleurs, l'utilisation des termes AGENT et THÈME relève d'une formalisation de la structure argumentale dans laquelle les fonctions grammaticales sont abstraites au profit de rôles thématiques plus proches de la sémantique. Néanmoins, l'utilisation de tels rôles posent un certains nombres de problèmes (Jackendoff, 1972; Dowty, 1991; Saeed, 2003) qu'il est souvent préférable de pallier par l'utilisation des fonctions grammaticales comme le font la Grammaire Relationnelle de Perlmutter et Postal (1984) ou la f-structure de la théorie LFG (Kaplan et Bresnan, 1982; Dalrymple, 2001; Bresnan, 2001) : c'est exactement à ce niveau que se situe cette thèse de linguistique informatique.

Qui dit linguistique informatique dit recherche de méthodes opérationnelles permettant des applications en traitement automatique des langues (TAL). En effet, nous ne chercherons pas à redéfinir la notion de structure argumentale ou à apporter un éclairage nouveau aux diverses théories qui ont été élaborées au cours des cinquante dernières années. Notre but est différent et il est double. Premièrement, nous souhaitons présenter une méthodologie de pré-annotation permettant d'obtenir des structures argumentales, car le français,

1. Nous utilisons l'apostrophe pour dénoter le sens d'un prédicat et de ces arguments, comme il est d'usage en sémantique.

notre langue d'étude principale, n'a pas de ressources manuellement annotées à large couverture (et sur des domaines multiples) qui rendent compte de la structure argumentale de la phrase. Pour ce faire, nous avons participé à la création d'un schéma d'annotation modélisant en partie la structure argumentale de la phrase en français (Candito et al., 2014a; Perrier et al., 2014). Ce schéma d'annotation fait le choix de ne pas se préoccuper de la distinction donnée par les rôles thématiques et de conserver les fonctions grammaticales de la représentation syntaxique sous-jacente, il est donc au niveau de la **syntaxe profonde** d'un énoncé, cette représentation à mi-chemin entre syntaxe de surface et structure argumentale pleine et entière (comprenant l'explicitation non ambigüe des rôles thématiques après désambiguïsation des lemmes). Deuxièmement, nous souhaitons exploiter les données ainsi créées pour mettre au point des méthodes informatiques efficaces capables d'analyser automatiquement la structure argumentale d'une phrase ; cela passe par la réutilisation des méthodes d'analyse syntaxique et sémantique (Lluís et al., 2013; Henderson et al., 2013; Lluís et al., 2014; Martins et Almeida, 2014) ou des méthodes de réécriture de graphes (Rozenberg, 1997; Chiang et al., 2013; Peng et al., 2015). Ainsi, plusieurs contributions sont mises en avant dans cette thèse :

1. La création d'un système de réécriture de graphes linguistiquement motivé, appelé OGRE et qui montre de bonnes propriétés formelles sur les graphes linguistiques que nous analysons.
2. Son utilisation pour :
   - la pré-annotation automatique en syntaxe profonde du Séquoia Treebank (Candito et Seddah, 2012b), un corpus arboré multi-domaines librement disponible. La ressource produite est appelée DeepSequoia ;
   - l'annotation automatique en syntaxe profonde du French Treebank (Abeillé et al., 2003), un corpus arboré journalistique annotant des articles du journal *Le Monde*. [2]
3. La mise au point de méthodes d'analyse par transitions (Nivre, 2005) pour l'analyse de graphe généraux.
4. La création d'un jeu de traits syntaxiques (issus d'analyseurs en constituants et en dépendances variés) pour apporter du contexte à des analyseurs par transition et leur permettre de se rapprocher de l'état de l'art en prédiction de structures prédicat-argument sur l'anglais (Oepen et al., 2014) et le français.
5. L'extension d'un analyseur (Martins et Almeida, 2014) par factorisation d'arcs (McDonald et al., 2005) utilisant des techniques de décomposition duale (Martins et al., 2011) pour intégrer notre jeu de traits et présenter des résultats à l'état de l'art sur l'anglais et le français.
6. La mise au point **préliminaire** d'un système permettant l'induction automatique de règles de transformation en utilisant des techniques de fouille de sous-graphes fréquents (Jiang et al., 2013).

2. À partir de maintenant, nous abandonnons l'utilisation du terme de *corpus arboré* au profit de *treebank* ou simplement *corpus* quand le sens ne prête pas à confusion.

Enfin, notre thèse marque une étape préliminaire vers une interface syntaxe-sémantique à plus large échelle au sein du laboratoire Alpage. En effet, deux autres projets d'envergure ont été menés en parallèle de nos travaux : la création d'un VerbNet (Kipper-Schuler, 2006) du français, appelé Verbɜnet (Pradet et al., 2014) et la création d'un FrameNet (Baker et al., 1998) du français, appelé Asfalda (Candito et al., 2014a; Djemaa et al., 2016).

Plan de la thèse

La thèse se compose de huit chapitres répartis en deux parties, que nous décrivons ci-dessous :

Partie I : Positionnement du problème : La première partie est un exposé du problème linguistique que nous traitons et des travaux antérieurs qui y sont liés. Elle inclut les chapitres suivants :

— **Chapitre 2** qui revient sur la notion de structure argumentale et de syntaxe profonde et expose les enjeux de la thèse.

— **Chapitre 3** qui expose les ressources à large couverture déjà existantes sur des langues autres que le français, ressources qui sont manuellement annotées ou automatiquement acquises.

— **Chapitre 4** qui expose les ressources existantes sur le français en relation avec les représentations que nous cherchons à obtenir.

Partie II : Interface syntaxe-sémantique : des données aux modèles : Cette partie traite de nos contributions centrées autour du schéma d'annotation en syntaxe profonde, base du DeepSequoia. En effet, notre participation aux discussions sur la création de ce schéma a conditionné la mise au point du jeu de règles nécessaires à la pré-annotation automatique du DeepSequoia mais aussi la réflexion sur les méthodes les plus aptes à analyser automatiquement ce genre de structures. La partie est composée des chapitres suivants :

— **Chapitre 5** qui fait une description du point de départ de l'annotation profonde : le Séquoia treebank (Candito et Seddah, 2012b) converti automatiquement en dépendances surfaciques.

— **Chapitre 6** qui traite de l'annotation profonde du DeepSequoia (Candito et al., 2014a; Perrier et al., 2014). Le chapitre fait un tour d'horizon des phénomènes annotés et donne des scores inter-annotateurs qui sont comparés aux approches équivalentes.

— **Chapitre 7** qui expose la méthode de pré-annotation automatique du DeepSequoia au moyen d'un système de réécriture de graphes issu de la théorie des grammaires de graphes, une généralisation des grammaires hors contexte aux graphes (Rozenberg, 1997; Drewes et al., 1997; Ehrig et al., 2006) : au lieu de réécrire des symboles, ce genre de grammaires réécrit des sous-graphes. D'emblée, il est facile de voir pourquoi ce processus est intéressant lorsqu'il existe des corpus annotés en syntaxe de surface. Nous présentons notamment ogre, le système de réécriture mis au point durant

notre mémoire de Master (Ribeyre, 2012) et largement amélioré et enrichi durant la thèse. Nous donnons aussi une preuve de sa terminaison et de sa confluence. Enfin, nous effectuons une série d'expériences pour montrer l'intérêt de l'approche sur :

  - de nouvelles données de référence issu du French Treebank (Abeillé et al., 2003) ;
  - des sorties d'analyseurs surfaciques variés pour montrer que les règles écrites à la main généralisent correctement sur des sorties bruitées.

- **Chapitre 8** qui traite de l'analyse syntaxico-sémantique automatique de la syntaxe profonde avec une série d'expériences sur l'anglais et sur le français. Plus précisément, nous traitons en premier lieu des analyseurs par transitions et par factorisation d'arcs avant d'appliquer ces méthodes à la prédiction de la syntaxe profonde. Enfin, nous montrons que l'utilisation de traits syntaxiques issus de sources diverses (fragments d'arbres en constituants, étiquettes issues des dépendances surfaciques, ...) apporte le contexte manquant aux analyseurs par transitions pour améliorer les prédictions.
- **Chapitre 9** qui parle de l'induction automatique de règles de réécriture. Plutôt que d'écrire les règles manuellement, il est possible de trouver des méthodes astucieuses issues du domaine de recherche de sous-graphes fréquents (Jiang et al., 2013) pour automatiquement générer ces règles. Cette partie met en avant des techniques et des idées de réalisation plus qu'un système fonctionnel, puisque cette étape du travail a plus été vue comme un début de perspective que comme une fin en soi.

Publications

La thèse a déjà fait l'objet d'un certain nombre de publications pour chacune des contributions [3] :

- deux sur l'annotation en syntaxe profonde du DeepSequoia à laquelle nous avons activement participé (Candito et al., 2014b; Perrier et al., 2014) ;
- une sur la projection automatique du schéma en syntaxe profonde sur le French Treebank (Ribeyre et al., 2014a) ;
- deux sur le système de réécriture de graphes (Ribeyre et al., 2012; Ribeyre, 2013) ;
- deux sur l'analyse syntaxico-sémantique automatique par transitions et factorisation d'arcs au moyen de traits syntaxiques (Ribeyre et al., 2014b, 2015).

3. Nous avons aussi une publication suite à l'annotation d'un corpus syntaxique en constituants comportant des phrases issues de forums d'aide à l'utilisation d'un logiciel anti-virus en partenariat avec le centre ADAPT (Dublin City University) et Symantec (Kaljahi et al., 2015).

Première partie

État de l'art & Positionnement du problème

Chapitre

2 Structure argumentale de la phrase

Sommaire

Préambule

La grande majorité des sens lexicaux des langues naturelles possède la propriété d'être des prédicats. Nous définissons un prédicat comme un *lexème se construisant avec des arguments*. La définition de prédicat est particulièrement débattue au sein des théories linguistiques et plus généralement en philosophie du langage (Muller, 2013). Pour les tâches qui nous intéressent, à savoir produire un corpus annotant la structure argumentale de la phrase et son exploitation dans le cadre du Traitement Automatique des Langues, cette définition nous semble suffisante pour capturer une large part des prédicats des langues naturelles. Comme l'expliquent parfaitement Mel'čuk et Polguère (2008), il est utile de

modéliser les prédicats par des « micro-structures sémantiques » qui contiennent la représentation des arguments en question. Cette micro-structure peut être visualisée sous forme de graphe que l'on appelle *réseau sémantique* ou plus généralement *structure argumentale* ou encore *structure prédicat-argument*. Elle est constituée du prédicat lui-même connecté aux variables qui désignent les « positions disponibles » pour les sens correspondant aux arguments. Pour une phrase telle que *Jean donne un livre à Marie*, le réseau sémantique est illustré à la Figure 2.1.

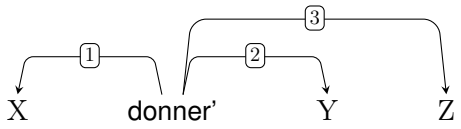


Figure 2.1: Exemple de réseau sémantique pour *X (Jean) donne Y (un livre) à Z (Marie)*. Nous dénotons un prédicat en utilisant une apostrophe indiquant son sens linguistique.

La notion de prédicat sémantique est directement héritée de la notion de prédicat logique. De fait, pour comprendre la notion de prédicat et d'arguments, il convient de repartir de la logique et donc des travaux en sémantique computationnelle. Ceux-ci nous permettront de mettre en évidence des points de comparaisons avec les structures prédicat-argument qui sont l'objet principal de cette thèse, mais aussi de montrer que le formalisme de la logique des prédicats dépasse largement les ambitions de notre travail. Une fois ces notions en place, nous discuterons de la notion de structure argumentale, intimement liée à la notion de Θ-rôles (ou *rôles thématiques*). Nous montrerons alors les avantages et les difficultés que posent ces notions et les raisons pour lesquelles nous lui préférons un niveau à l'intersection entre l'interface de la syntaxe et de la sémantique : le niveau syntaxique profond (ou *syntaxe profonde*) qui constitue l'étape initiale de nos travaux de recherche.

2.1 Sémantique et langue naturelle

Tout élément du langage naturel contient des expressions avec un sens. Ces mêmes expressions sont décomposables en parties, elles aussi avec un sens. Ce point de vue traditionnel est résumé par le principe de compositionnalité généralement attribué à Frege (1879).

« *Le sens d'une expression est fonction du sens de ces parties.* » Frege

Une fois que le sens des parties d'une expression complexe est posé et qu'a été explicitée l'articulation de ces parties entre elles, il est possible de donner le sens de l'expression tout entière. En effet, l'analyse du langage (permettant la génération d'un nombre infini de constructions grammaticalement correctes) induit nécessairement la capacité d'un locuteur à produire et à comprendre des formulations inédites pour lui. C'est l'objectif de la sémantique computationnelle, que d'aboutir à une représentation formelle d'un énoncé. Dans le cadre de cette thèse, notre énoncé sera la phrase. En effet, nous nous intéressons uniquement aux problèmes à l'interface entre la syntaxe et la sémantique et de la manière dont cette interface peut être définie. Cela suggère alors que les problèmes afférents au discours sont hors de propos.

Comme nous le suggérions dans le préambule, nous souhaitons définir les notions de prédicat et d'argument qui sont les atomes de la structure prédicat-argument. Le terme de prédicat et d'argument est héritée directement de la logique des prédicats qui s'inscrit dans la cadre d'une sémantique dite **véri-conditionnelle** consistant à établir d'abord les conditions de vérités de l'énoncé pour arriver finalement à produire une *formule logique*. Néanmoins, avant de parler de logique des prédicats, nous devons introduire la notion de logique propositionnelle qui définit la notion centrale de proposition.

2.1.1 Logique propositionnelle

Dans un premier temps, nous supposons que le sens des phrases est identifiable grâce à leurs conditions de vérité. Ainsi, pour une assertion telle que *La terre est ronde*, il faut regarder les conditions dans lesquelles cette phrase est soit vraie, soit fausse. Trouver si une phrase est vraie ou fausse, c'est faire de la sémantique dite *véri-conditionnelle*. Néanmoins, seule notre connaissance du monde permet de connaître la valeur de vérité de l'assertion *La terre est ronde*. Or, comme il n'est pas réaliste de formaliser la connaissance du monde dans son intégralité, nous considérons que l'énoncé *La terre est ronde* est atomique et n'est pas décomposable en propositions plus simples.

La définition d'une proposition est au cœur de la logique propositionnelle (appelée encore calcul propositionnel). En guise de simplification, il est possible de définir une proposition comme un énoncé qui est soit *vrai* soit *faux*. Ainsi *La terre est ronde* ou *Le livre est ouvert* sont deux propositions. La logique des propositions s'intéresse aux relations entre propositions et aux opérations sur ces dernières, sans analyser la composition non propositionnelle des propositions simples. Autrement dit, le calcul propositionnel ne rentre pas dans les détails de la composition des phrases elles-mêmes. De fait, une phrase optative

telle que *Que Dieu nous protège !* ou une phrase impérative comme *Tais-toi !* ne sont pas des propositions (dans le sens du calcul propositionnel). [1]

La logique des propositions possède alors un certain nombre d'outils formels :

— Des variables propositionnelles (P, Q, R, ...) qui symbolisent des propositions simples quelconques. Si la même variable apparaît plusieurs fois, elle symbolise à chaque fois la même proposition. De fait P = *La terre est ronde* est une proposition appelée P.

— Les opérateurs logiques sont les suivants : négation ($\neg$), conjonction ($\wedge$), disjonction ($\vee$), implication ($\rightarrow$), équivalence ($\leftrightarrow$). Les trois premiers sont standards et correspondent aux opérateurs utilisés dans le cadre de la logique booléenne. Pour les deux derniers, l'équivalence est définie comme telle : $P \leftrightarrow Q \equiv (\neg P \wedge \neg Q) \vee (P \vee Q)$ et l'implication comme cela : $P \rightarrow Q \equiv \neg P \vee Q$.

— Les signes de ponctuation se réduisent aux parenthèses ouvrante (et fermante) et organisent la lecture de manière à éviter toute ambiguïté. Dans l'expression $(\neg P \vee Q)$, seule P est niée, dans $\neg(P \vee Q)$, c'est la disjonction qui est niée complètement.

Quelques exemples de propositions :

— Jean dort ou Jean marche : $P \vee Q$ (où P = *Jean dort*, Q = *Jean marche*).

— Jean est mort, ces enfants sont orphelins : $P \rightarrow Q$ (où P = *Jean est mort* et Q = *Ses enfants sont orphelins*).

— Jeanne est enceinte, elle attend un bébé : $P \leftrightarrow Q$ (où P = *Jeanne est enceinte* et Q = *Elle attend un bébé*).

Afin de déterminer la condition de vérité d'une proposition, le calcul propositionnel autorise des procédures de décisions ou de tests. Celles-ci permettent de déterminer dans quels cas une expression est vraie et en particulier si elle est toujours vraie. Deux procédures de décision sont à notre disposition : la première est purement mécanique et repose sur l'application stricte des conditions de vérité à partir des tables de vérité, où deux propositions sont équivalentes si elles ont la même table de vérité. La seconde est appelée procédure axiomatique et permet à partir d'un axiome ($P \rightarrow (Q \rightarrow P)$) de démontrer son équivalence logique avec un théorème (c'est-à-dire sa simplification). Quelques axiomes constituent le point de départ dont la vérité est posée (et non démontrée). Il faut alors déterminer les règles de déduction permettant de manipuler les axiomes ou toute expression obtenue à partir de ceux-ci. L'enchaînement de ces déductions est une démonstration qui conduit à un théorème.

Cependant, la grande limitation de la logique des propositions réside dans le fait, qu'en se fondant sur des propositions atomiques, elle ne dispose d'aucun outil formel pour différencier les phrases suivantes :

— Tous les hommes sont mortels.

— Un homme est mortel.

1. De même, l'assertion *Dieu existe* n'est pas vérifiable en tant que telle, elle est donc considérée comme un fait qui peut être soit vrai, soit faux. Prouver l'existence de Dieu n'est pas à considérer ici.

— Jean est mortel.

En logique propositionnelle, ces trois énoncés sont représentés chacun par une proposition et sont donc atomiques bien qu'il existe des différences évidentes. En effet, les phrases données ci-dessus sont elles-mêmes structurées et donc décomposables en « fragments de sens » plus petit. Les considérer comme des atomes, c'est ne pas pouvoir distinguer la différence claire qu'il y a entre *Jean*, *un homme* et *tous les hommes*. Il faut donc trouver un moyen d'augmenter le pouvoir formel de notre outil d'analyse.

2.1.2 Logique des prédicats

La logique des prédicats est alors le prolongement de la logique propositionnelle. Elle introduit les notions de fonctions, de variables et de quantificateurs.

— $f(x)$: x est f.

— $\forall x.f(x)$: Tous les x sont f.

— $\exists x.f(x)$: Il existe un x qui est f.

Une fonction simple (encore dite atomique) est composée d'une variable x et d'une variable de prédicat f. f marque la place d'un prédicat quelconque et x celle d'un nom satisfaisant (ou non) ce prédicat. Il est alors possible de caractériser les phrases citées ci-dessus et qui ne pouvaient pas l'être via le seul calcul propositionnel :

— Tout homme est mortel : $\forall x(\text{homme'}(x) \rightarrow \text{mortel'}(x))$.

— Un homme est mortel : $\exists x(\text{homme'}(x) \rightarrow \text{mortel'}(x))$.

— Jean est mortel : mortel'(jean').

Une fonction est aussi appelé un *prédicat* et prend de un à n arguments. Pour le moment, nous n'avons abordé que des relations attributives qui portaient uniquement sur un seul terme. En logique, ce sont des **prédicats monadiques** (à un seul argument). Bien entendu, il existe aussi des **prédicats polyadiques** (à plus d'un argument). Un exemple pourrait être aimer'(jean', marie') représentant le sens de la phrase *Jean aime Marie*, où aimer' est un prédicat polyadique d'arité 2, c'est-à-dire comprenant deux arguments.

Une fois posées les bases de la logique des prédicats, il est nécessaire de s'intéresser aux liens qui unissent les énoncés aux conditions de vérité.

2.1.3 Interface syntaxe-sémantique ou comment lier phrases de la langue naturelle et conditions de vérité

Dans une langue formelle, la valeur de vérité d'une formule est définie récursivement à partir de celle de ses composants (c'est l'application stricte du principe de compositionnalité). Pour ce faire, il existe une règle pour chaque connecteur. Or, ces connecteurs n'apparaissent pas en tant que tels dans la langue. De fait, le lien entre phrases et conditions de vérité peut être obtenu de deux manières différentes : (i) utilisation directe des règles récursives applicables aux mots de la langue naturelle. Ce faisant, l'aspect syntaxique

de l'énoncé ne serait pas pris en compte ; or, les travaux de Tesnière (1959) et de Chomsky (1965) nous indiquent que la considération syntaxique est indissociable de la sémantique d'un énoncé. Chomsky (1955) disait d'ailleurs : "Linguistic theory has two major subdivisions : syntax and semantics" (ii) traduction vers une représentation intermédiaire, vers une langue formelle contenant de tels connecteurs, c'est la solution choisie par Montague (1970).

La grammaire de Montague prend en entrée uniquement des énoncés non ambigus. Si un énoncé tel que *La belle porte le voile* doit être analysé, nous aurons deux exécutions du processus d'analyse du modèle de Montague : l'une où *porte* est le sujet du verbe *voiler* et l'autre où *porte* est le verbe de la phrase. Il en résultera alors deux interprétations : (i) une belle femme porte un voile ; (ii) une belle porte cache quelque chose. Le processus d'interprétation sémantique en lui-même consiste à projeter une structure dans une autre structure.

2.1.3.1 Illustration du principe de compositionnalité dans la Grammaire de Montague

L'attribution d'un sens à un énoncé est régie par des principes généraux qui expliquent comment le sens d'une expression complexe dépend du sens de ses constituants. C'est le principe de compositionnalité de Frege (1879) que nous rappelions ci-dessus. Cependant, afin d'éviter l'apparition de non-sens évidents, il faut préciser davantage ce principe en ajoutant une relation fondamentale avec la syntaxe :

> « *Le sens d'une expression composée est une fonction du sens de ses parties et de la règle syntaxique par laquelle elles sont combinées.* »

Sans cette précision, on aboutirait à une situation où les phrases *Jean aime Marie* et *Marie aime Jean* auraient le même sens (Amsili et Bras, 1998).

Montague propose de considérer que la relation entre la syntaxe et la sémantique d'une langue naturelle sont du même ordre que celle régissant la sémantique et la syntaxe d'un langage formel. Il se place à l'interface entre la syntaxe et la sémantique. Nous reprenons alors les explications et l'argumentaire de Seddah (2004, pp. 45-55) qui décrit, au travers de quatre exemples comment la grammaire de Montague est l'application stricte du principe de compositionnalité de Frege (1879). Pour ce faire, il considère un fragment d'une grammaire hors contexte associé à un lexique auxquels il va progressivement ajouter les règles sémantiques et la sémantique de chaque mot du lexique telle qu'elle a été définie par Montague. Nous reproduisons à la Table 2.1 les règles de grammaire et le lexique que nous allons discuter ci-dessous.

Les exemples suivants sont alors étudiés les uns après les autres : *Jean dort*, *Un homme dort*, *Jean aime Marie* et *Jean aime une femme*. Pour commencer, il faut appliquer le principe de compositionnalité au premier exemple (*Jean dort*). Le principe nous dit que le sens d'un énoncé est une fonction de ses parties et de la règle syntaxique associée. Traditionnellement, l'analyse syntaxique de *Jean dort* est celle de la Figure 2.2.

Lexique syntaxique	Règles syntaxiques		
un.e : Det	S	→	NP VP
homme : N	VP	→	V
femme : N	VP	→	Vt NP
Jean : NPP	NP	→	NPP
Marie : NPP	NP	→	Det N
dort : V			
aime : Vt			

Table 2.1: Règles de grammaire et lexique donnés par Seddah (2004).

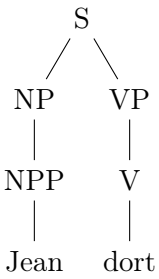


Figure 2.2: Analyse syntaxique en constituants pour *Jean dort.*

Par ailleurs, l'interprétation de cet énoncé est une valeur de vérité de type t [2]. Soit $\tau(S)$ le sens de la phrase, $\tau(NP)$ celui du syntagme nominal et $\tau(VP)$ celui du syntagme verbal. Nous appelons f la fonction d'interprétation en question. De fait : $\tau(S) = f(\tau(NP), \tau(VP))$. Selon le formalisme choisi, la fonction d'interprétation f peut se voir implémenter de différentes façons. Dans le système de Montague, c'est le λ-calcul [3] qui est utilisé. Il permet d'implémenter la fonction en tant que sens de l'application. La formule $\tau(S) = f(\tau(NP), \tau(VP))$ peut donc être interprétée de deux manières :

1. $\tau(S) = \tau(NP)[f(\tau(VP))]$ que l'on interprète comme le sens de NP prenant le sens de VP comme argument.
2. $\tau(S) = \tau(VP)[f(\tau(NP))]$ où cette fois-ci la relation argumentale est inversée.

L'idée de la grammaire de Montague est d'intégrer la fonction f dans le sens même de l'un ou l'autre des composants (NP ou VP). Cela se fait de la manière suivante : appelons $\tau(NP)' = f(\tau(NP))$ et $\tau(VP)' = f(\tau(VP))$, la formule $\tau(S)$ est donc équivalente à $\tau(S) = \tau(NP)[(\tau(VP)')]$ ou à $\tau(S) = \tau(VP)[(\tau(NP)')]$. La fonction a donc été intégrée dans le sens de l'un ou l'autre des arguments. Le principe de compositionnalité reste vérifié et il ne reste qu'à choisir quel composant est une fonction prenant l'autre comme argument. Cette question peut être résolue via l'utilisation des types sémantiques.

2. t est employée pour *truth value.*
3. Pour une présentation exhaustive du λ-calcul et des preuves afférentes, on pourra se reporter à Krivine (1997) ou moins formellement à Partee (1993).

Il faut d'abord montrer que le type de *Jean'* sera e car c'est un élément faisant partie d'un ensemble d'individus E. Il déduit alors que le type du verbe intransitif *dort* doit être $e \rightarrow t$ (une fonction qui prend un individu et renvoie une valeur de vérité, ce qui peut aussi s'écrire $\langle e, t \rangle$. Comme nous connaissons l'arbre syntaxique pour *Jean dort* et les types sémantiques associés, il ne reste plus qu'à définir son λ-terme $\lambda x.\text{dort'}(x)$ qui est une traduction quasi littérale de l'énoncé *Jean dort*. Nous aboutissons alors à l'analyse sémantique de la Figure 2.3.

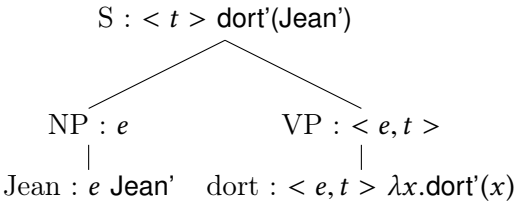


Figure 2.3: Arbre d'analyse sémantique pour *Jean dort*.

Par la biais du processus de β-réduction, l'application simple de $\lambda x.\text{dort'}(x)(Jean)$ qui est le résultat de l'application de la règle d'inférence $\tau(S) = \tau(VP)[\tau(NP)]$ est devenue dort'(Jean'). Le processus de β-réduction est le suivant :

1. $\tau(S) = \tau(VP)[\tau(NP)]$
2. $\tau(S) = \lambda x.\text{dort'}(x)(\text{Jean'})$
3. $\tau(S) = \text{dort'}(\text{Jean'})$

L'autre hypothèse ($\tau(S) = \tau(NP)[f(\tau(VP))]$) est alors considérée. Puisque le type sémantique de VP est $< e, t >$, la seule solution pour obtenir une nouvelle valeur de vérité, c'est de donner à NP le type $<< e, t >, t >$. Il est possible de s'interroger sur la pertinence d'un tel choix qui implique alors que *Jean* soit argument d'une fonction, mais si l'on considère que s'appeler *Jean* n'est pas une fin en soi d'un point de vue sémantique, mais que c'est une information qui doit nécessairement être associée à une propriété, cette interprétation s'impose alors d'elle-même. Dans ce cas, *Jean* devient l'argument de cette fonction et son syntagme nominal est ainsi associé sémantiquement à une fonction qui, étant donné une propriété, renvoie vrai si et seulement si cette propriété vaut pour *Jean*. Son type est donc $<< e, t >, t >$ et le λ-terme correspondant : $\lambda P.P(\text{Jean'}) :<< e, t >, t >$. L'analyse sémantique est alors celle représentée à la Figure 2.4.

Le processus de β-réduction est le suivant :

1. $\tau(S) = \tau(NP)[f(\tau(VP))]$
2. $\tau(S) = \lambda P.P(\text{Jean'})(\lambda x.\text{dort'}(x))$
3. $\tau(S) = \lambda x.\text{dort'}(x)(\text{Jean'})$
4. $\tau(S) = \text{dort'}(\text{Jean'})$

Nous notons alors que cette hypothèse donne la même interprétation ce qui ne permet pas de départager les deux hypothèses et de savoir laquelle préférer. Par ailleurs, notre

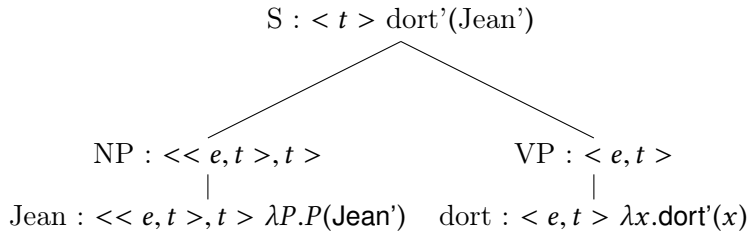


Figure 2.4: Arbre d'analyse sémantique pour *Jean dort* en considérant le syntagme nominal comme une fonction prenant le syntagme verbal comme argument.

intuition linguistique nous pousserait plutôt à considérer que le nom est l'argument du verbe et non l'inverse. En considérant l'exemple *Un homme dort*, nous verrons que cette hypothèse est valide. Dans cette exemple, la règle syntaxique sera $\mathsf{NP} \rightarrow \mathsf{Det\ N}$ et donc son type devrait être e ou $<< e, t >, t >$ comme précédemment, puisque c'est un syntagme nominal. Pour respecter le principe de compositionnalité, il est nécessaire de trouver les types du déterminant (Det) et du nom commun (N), puisque le syntagme nominal ne peut plus directement hérité le type de son fils, comme pour *Jean dort*. Seddah (2004) explique le processus par lequel Montague (1970) a montré que le déterminant a le λ-terme suivant : $\lambda Q.\lambda P.[\exists x.P(x) \wedge Q(x)] :<< e, t >, << e, t >, t >>$. Le déterminant a donc une portée supérieure à son syntagme. Il montre aussi que le NP ne peut pas avoir le type $< e >$. C'est l'une des contributions majeures de Montague, car nombre de logiciens ((Frege et Russell, 1994; Russell, 1969)) avaient mis en avant que ce problème ne permettait pas d'avoir une analyse strictement compositionnelle de la langue naturelle. Montague, en s'attachant à résoudre les problèmes de portées des quantificateurs et en donnant les règles sémantiques correctes pour chaque règle syntaxique, établit le principe de compositionnalité comme architecture de base de l'interprétation sémantique. L'analyse finale d'*un homme dort* est donnée à la Figure 2.5.

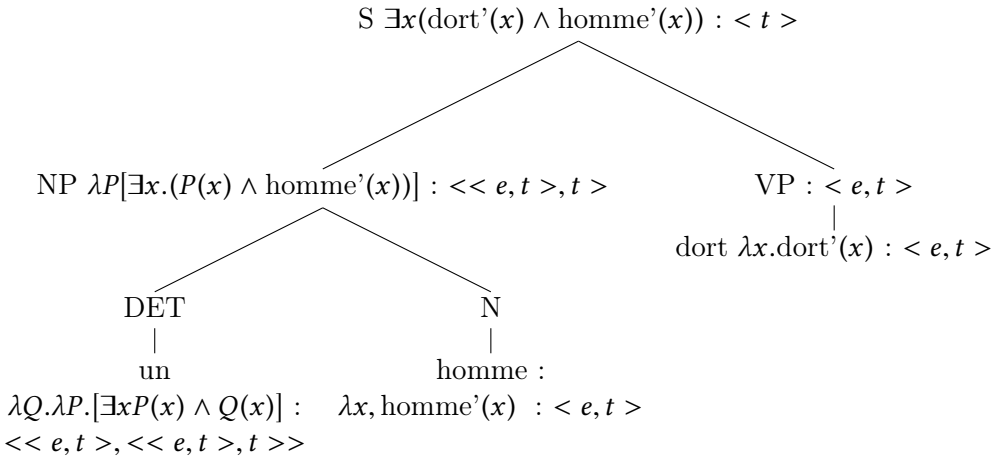


Figure 2.5: Arbre d'analyse sémantique pour *Un homme dort*.

Enfin, l'argumentaire est renforcé en prenant l'exemple d'un verbe transitif avec *Jean aime Marie* puis modifie le NP objet et le remplace par *une femme* pour montrer que l'analyse de *Jean aime une femme* est correcte et que la quantification porte bien sur l'ensemble de la phrase. À l'issue de ces observations, nous pouvons dresser le lexique sémantique et les règles associées (Table 2.2).

Lexiques			Règles			
Syntaxique	λ-terme	Types	Syntaxiques			Sémantiques
un.e : Det	$\lambda Q.\lambda P.[\exists x.P(x) \wedge Q(x)]$	$<< e,t >, << e,t >, t >>$	S	$\rightarrow$	NP VP	$\tau(S) = \tau(NP)[\tau(NV)]$
homme : N	λx.homme'(x)	$< e,t >$	VP	$\rightarrow$	V	$\tau(NP) = \tau(V)$
femme : N	λx.femme'(x)	$< e,t >$	VP	$\rightarrow$	Vt NP	$\tau(VP) = \tau(Vt)[\tau(NP)]$
Jean : NPP	λx.Jean'(x)	$< e,t >$	NP	$\rightarrow$	NPP	$\tau(NP) = \tau(NPP)$
Marie : NPP	λx.Marie'(x)	$< e,t >$	NP	$\rightarrow$	Det N	$\tau(NP) = \tau(Det)[\tau(N)]$
dort : V	λx.dort'(x)	$< e,t >$				
aime : Vt	$\lambda R.\lambda z.R(\lambda y$.aime'$(z)(y))$	$<<< e,t >, t >, < e,t >>$				

Table 2.2: Grammaire jouet et interface syntaxe-sémantique.

Nous avons alors produit un premier lexique syntaxico-sémantique ainsi que l'ensemble de règles associées, tel que Montague avait pu l'envisager. Cette idée a conditionné nombre de travaux et notamment ceux sur les grammaires combinatoires catégorielles (Steedman, 1996, 2000) (CCG) qui reposent aussi sur un isomorphisme entre la syntaxe et la sémantique via le λ-calcul.

2.1.3.2 Interface syntaxe-sémantique et grammaires catégorielles combinatoires

L'avantage des CCG résident dans leur traitement transparent de l'interface syntaxe-sémantique. La théorie permet le traitement de phénomènes complexes de coordinations elliptiques et de relatives, par exemple :

— Il paie 5 Millions d'euros à Google dès maintenant et un coût additionnel dans le futur.

— Qui doit et qui devrait légiférer sur le crime ici ?

Les CCG capturent directement les informations longue distance dans les phénomènes complexes illustrés ci-dessus. Elles n'ont donc pas besoin de mouvement syntaxique, ni de traces et encore moins d'éléments sans contribution (*null elements*). Par ailleurs, elles peuvent dériver une interprétation sémantique directement de la phrase analysée.

Les règles d'une CCG sont fondées sur le calcul catégoriel de Ajdukiewicz (1935) et de Bar-Hillel (1953) ainsi que sur la logique combinatoire de Curry et Feys (1958). Chaque mot du lexique d'une CCG est associé à une catégorie précise qui définit son comportement syntaxique. Ensuite, un ensemble de règles universelles sont établies afin de donner les possibles combinaisons d'un mot avec les autres constituants d'une phrase. En général, les catégories sont définies de la sorte :

1. **Catégories atomiques** Ce sont des catégories qui peuvent être mis en rapport avec les non-terminaux d'une grammaire hors contexte, par exemple S, NP, VP, etc.

2. **Catégories complexes** Ce sont des fonctions d'arité un marquées X/Y ou $X\backslash Y$ où Y est un argument et X le résultat. Les foncteurs ont une direction qui permet de différencier entre les arguments à droite du foncteur (indiqués par le slash /) et les arguments à sa gauche (indiqués par le slash inversé \). Nous suivons la notation donnée par Steedman (2000) pour représenter les catégories complexes. Les catégories complexes peuvent être composées pour former d'autres catégories encore plus complexes. Par exemple, la catégorie (S\NP)/NP représente la catégorie pour un verbe transitif : NP à droite du verbe, puis NP à gauche du VP pour marquer le sujet.

De plus, chaque catégorie syntaxique se voit associer une interprétation sémantique. Lors de la dérivation, l'analyse sémantique se fait de manière incrémentale. Dans le cas des CCG, l'interprétation sémantique est faite au moyen du λ-calcul. De fait, chaque catégorie syntaxique possède un λ-terme. Quelques exemples sont donnés en 2.1.

(2.1) *Jean* $\vdash$ NP : *Jean'*
Marie $\vdash$ NP : *Marie'*
dort $\vdash$ S\NP : $\lambda x.$*dormir'* x
aime $\vdash$ (S\NP)/NP : $\lambda x.\lambda y.$*aimer'* xy

L'application fonctionnelle est la base des CCG. Elle est directement héritée des grammaires AB (grammaires issues des travaux de Ajdukiewicz (1935) et de Bar-Hillel (1953), d'où le nom AB). Dans une grammaire AB, les catégories ne peuvent être combinées qu'en utilisant l'application fonctionnelle. En considérant des grammaires qui prennent en compte la direction de l'application – comme celles que nous considérons pour des traitements en linguistiques – la catégorie syntaxique utilise un slash ou un slash inversé pour la marquer. Cependant, l'interprétation sémantique est, quant à elle, insensible au sens de l'application.

Deux types d'application sont alors définies : *forward* et *backward*.

Définition 2.1 (Forward & Backward Application)**.**
Forward Application
$X/Y : f \quad Y : a \quad \Rightarrow \quad X : f(a)$
Backward Application
$Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : f(a)$

La règle d'application *forward* stipule que si un constituant porte la catégorie X/Y et qu'il est immédiatement suivi par un constituant de catégorie Y, il peut être combiné pour former un nouveau constituant de catégorie X. De la même manière, l'application *backward* permet à un constituant $X\backslash Y$ qui est immédiatement précédé par un constituant de catégorie Y de se combiner pour former un nouveau constituant de catégorie X. Un exemple de dérivation est donnée à la Figure 2.6.

Les CCG étendent les possibilités des grammaires AB grâce à un ensemble de règles fondées sur les combinateurs de la logique combinatoire (Curry et Feys, 1958). Ces règles combinatoires permettent des analyses syntaxiques plus intéressantes notamment en terme de dépendances longue distance dans les cas d'extractions et de coordinations, nous y reviendrons.

$$\dfrac{\dfrac{\text{Marie}}{NP : Jean'} \quad \dfrac{\dfrac{\text{aime}}{(S\backslash NP)/NP : \lambda x.\lambda y.aimer'\ xy} \quad \dfrac{\text{Jean}}{NP : Marie'}}{S\backslash NP : \lambda y.aimer'\ Jean'\ y}>}{S : aimer'(Marie', Jean')}<$$

Figure 2.6: Exemple de dérivation CCG.

Pour limiter l'expressivité de telles règles, Steedman (2000) pose les règles suivantes :

1. **Principe d'adjacence (*Principle of Adjacency*)** Les règles combinatoires ne peuvent être appliquées qu'à des entités adjacentes dans la chaîne d'entrée.
2. **Principe de cohérence (*Principle of Consistency*)** Toutes les règles combinatoires doivent être cohérentes avec la direction imposée par la fonction.
3. **Principe d'héritage (*Principle of Inheritance*)** Si la catégorie résultant d'une application est une fonction alors son slash (qui définit la direction de l'application) doit être le même que celui défini dans la fonction d'entrée.

Une fois ces règles établies, les CCG définissent de nouveaux foncteurs tels que ceux permettant la composition qui permet à deux foncteurs de se combiner en un nouveau foncteur.

Définition 2.2 (Composition)**.**
Les compositions possibles sont les suivantes :

(a) *Composition forward* :
$X/Y : f \quad Y/Z : g \quad \Rightarrow_{\mathbf{B}} \quad X/Z : \lambda x.f(g(x))$

(b) *Composition backward* :
$Y\backslash Z : g \quad X\backslash Y : f \quad \Rightarrow_{\mathbf{B}} \quad X\backslash Z : \lambda x.f(g(x))$

(c) *Composition croisée forward* :
$X/Y : f \quad Y\backslash Z : g \quad \Rightarrow_{\mathbf{B}} \quad X\backslash Z : \lambda x.f(g(x))$

(d) *Composition croisée backward* :
$Y\backslash Z : g \quad X\backslash Y : f \quad \Rightarrow_{\mathbf{B}} \quad X/Z : \lambda x.f(g(x))$

En plus des règles de composition, les CCG définissent des règles de montée de types :

Définition 2.3 (Montée de type)**.**
Les montées de type possibles sont les suivantes :

(a) *Montée de type forward* :
$X : a \quad \Rightarrow_{\mathbf{T}} \quad T/(T\backslash X) : \lambda f.f(a)$

(b) *Montée de type backward* :
$X : a \quad \Rightarrow_{\mathbf{T}} \quad T\backslash(T/X) : \lambda f.f(a)$

La composition et la montée de type sont conçues pour permettre des interactions et capturer les phénomènes longue distance de la langue telles que les extractions (2.7) ou les argument cluster coordinations (ACC, 2.8)

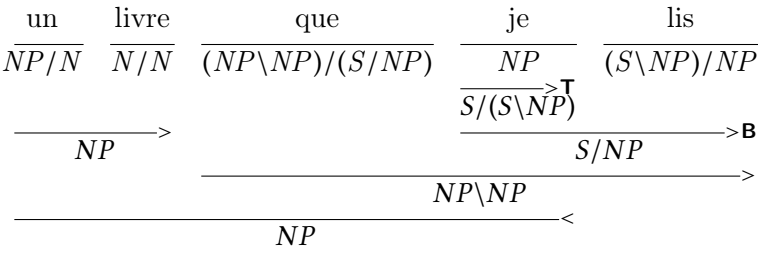


Figure 2.7: Exemple de dérivation pour une extraction en CCG.

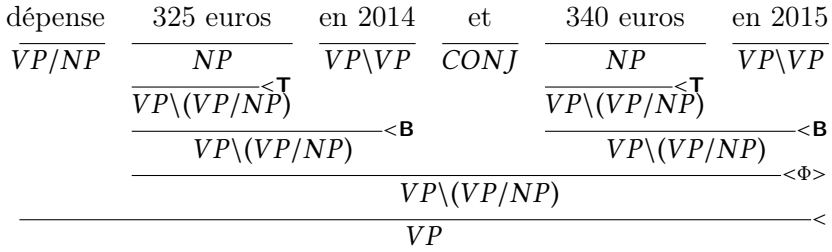


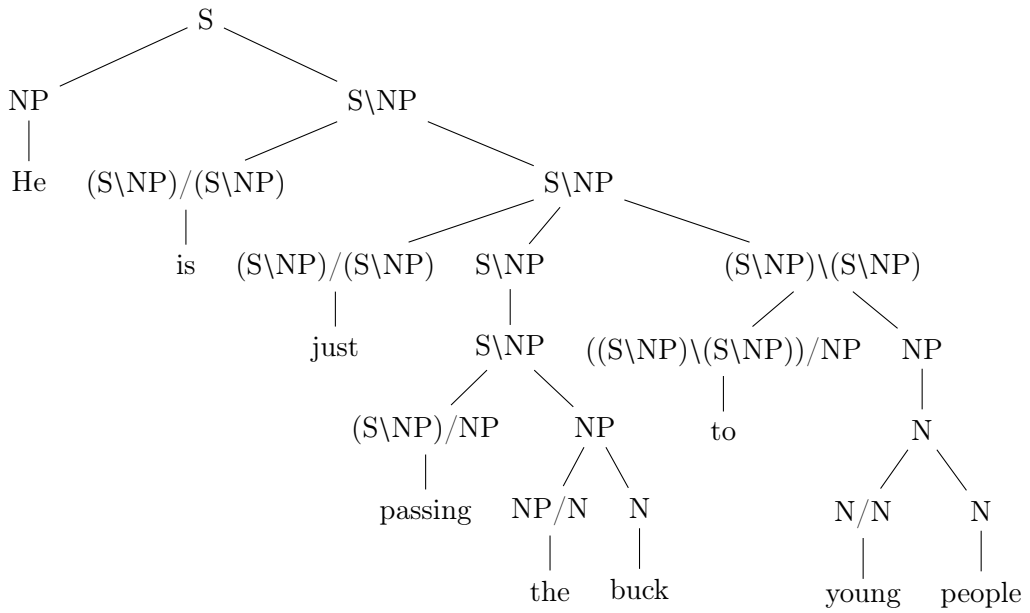
Figure 2.8: Exemple de dérivation pour un ACC en CCG.

Sur la Figure 2.8, un nouveau type de règle $< \Phi >$ est utilisé. C'est une règle qui permet de gérer la coordination :

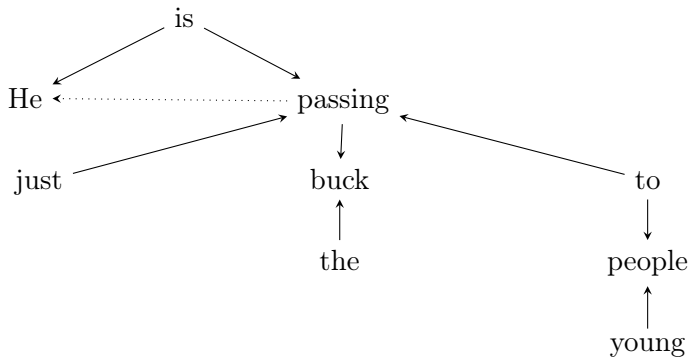
Définition 2.4 (CCG Coordination)**.**
X CONJ $X \quad \Rightarrow_{\Phi} \quad X$

Les CCG ont notamment été utilisées à large échelle pour créer une annotation semi-automatique du Penn Treebank (Marcus et al., 1993), appelé CCGBank (Hockenmaier et Steedman, 2007). Ce corpus utilise la puissance formelle de ces grammaires pour annoter les phénomènes longue distance et les coordinations elliptiques et permettre la récupération de la structure prédicat-argument via l'extraction du graphe de dépendances, une procédure ascendante qui retrace les étapes de dérivations grâce aux catégories CCG. Un exemple est donné à la Figure 2.9.

À travers l'exemple des CCG, nous voyons qu'il est possible d'avoir deux types de représentation sémantique : la première utilise le λ-calcul pour former une formule logique du premier ordre, la seconde utilise les dérivations pour construire la structure argumentale et représenter un graphe de dépendances. C'est la raison pour laquelle, nous pensons que le recours à la logique des prédicats n'est pas utile en première approximation et que l'utilisation de la structure argumentale est à même de représenter en partie le sens d'une



(a) Arbre de dérivation CCG.



(b) Graphe de dépendances.

Figure 2.9: Dérivation CCG et graphe de dépendances.

phrase. [4]

2.2 Structure argumentale et rôles thématiques

La structure argumentale indique le nombre d'arguments que possède une unité lexicale et leur relation sémantique envers cette unité lexicale, que l'on appelle prédicat. Il convient d'ores et déjà de remarquer que la notion de prédicat sémantique est une métaphore scientifique construite à partir de la notion de prédicat logique (décrite à la section 2.1). Il y a

4. Bien entendu, cela nous oblige à laisser de côté les problèmes de portée de la quantification, mais ces derniers dépassent largement le cadre de cette thèse.

un lien évident entre les notions d'argument d'un prédicat logique et d'argument d'un prédicat sémantique. Cependant, il ne s'agit là que d'une métaphore commode, comme celle de la valence syntaxique (sur le modèle de la valence atomique). La notion de prédicat sémantique n'est pas une notion logique et les lois de la logique formelle ne s'appliquent que de façon « métaphorique » sur les structures sémantiques prédicatives que manipule la sémantique linguistique (et non plus formelle) (Mel'čuk et Polguère, 2008). De plus, la notion de structure argumentale, quoique globalement partagée, n'est pas univoque et varie en fonction des théories surtout selon leur nature : plus syntaxique ou plus sémantique. Bresnan (2001), par exemple, l'a définie comme suit :

> « *Argument structure is an interface between the semantics and syntax of predicators (which we may take to be verbs in the general case) . . .Argument structure encodes lexical information about the number of arguments, their syntactic type, and their hierarchical organization necessary for the mapping to syntactic structure.* » (Bresnan, 2001)

2.2.1 Prédicats, arguments et modifieurs

Pour comprendre la notion de structure argumentale, il est aussi nécessaire de définir, le plus précisément possible, les notions de *prédicats, arguments* et *modifieurs*.

Une large part des unités lexicales des langues naturelles possèdent la propriété d'être des prédicats. Néanmoins, il existe des unités lexicales non prédicatives, que Mel'čuk et Polguère (2008) appellent des *noms sémantiques*. Ces noms sémantiques dénotent des entités conçues de manière autonome, sans référence à un fait spécifique et ne présupposent donc pas de participants (d'arguments). L'opposition prédicat / nom sémantique est fondée sur la conjonction de deux propriétés : (i) une propriété sémantique générale : dénoter un fait / une entité (ii) une propriété de combinatoire : contrôler / ne pas contrôler un nombre donné d'arguments. Nous pouvons illustrer ces notions en commençant par les prédicats (certains des exemples sont repris de (Mel'čuk et Polguère, 2008)) :

— *X* blesse *Y* (*Jean blesse Marie*).

— destruction de *X* par *Y* (*destruction de la ville par les romains*).

— *X* fidèle à *Y* (*Louise est fidèle à Marie*).

— *X* rapidement (*écrire rapidement*).

— *X* entre *Y* et *Z* (*une ville entre lac et montagne*).

— À bas *Y* ! (*À bas la malbouffe !*).

Comme le montrent Mel'čuk et Polguère (2008), n'importe quelle catégorie morpho-syntaxique peut comporter des prédicats. Pour illustrer les noms sémantiques, nous citerons des noms comme *Balzac*, *Canada*, *soleil*, ou encore *écureuil*. Tous ces noms dénotent des entités et ne contrôlent pas l'expression d'arguments. Du point de vue grammatical, ils ne sont lexicalisés que par des noms au sens large : noms communs, noms propres,

numéraux.[5] La distinction prédicat / nom sémantique ne semble pas poser de problèmes particuliers. Pourtant, elle est loin d'être exhaustive, car laissant de côté un nombre considérable de sens, car les deux propriétés qui opposent les prédicats aux noms sémantiques sont logiquement indépendantes. En effet, d'une part un sens peut dénoter un fait mais ne pas être un prédicat : il s'agit d'un prédicat non argumental (il ne contrôle pas de structure argumentale) comme le verbe *tonner* (*S'il tonne, la vendange sera bonne.*). D'autre part, un sens peut aussi être prédicatif sans pour autant dénoter un fait, c'est ce que Mel'čuk et Polguère (2008) appellent des quasi-prédicats comme par exemple *directeur* (X, le directeur de Y). Ainsi, même si la notion de prédicat est relativement aisée à définir, nous voyons qu'il existe beaucoup de cas où la frontière entre prédicat et nom sémantique est plus floue, ce qu'il faut prendre en compte lors de l'annotation d'un corpus au niveau argumental.

Une fois les prédicats définis, il faut s'attarder sur la notion d'argument qui peut être définie comme une expression qui complète le sens d'un prédicat. La plupart des prédicats prennent un (univalent), deux (bivalent) voire trois (trivalent) arguments. Nous pouvons donner deux critères pour reconnaître les arguments (repris de (Mel'čuk et Polguère, 2008)) :

1. du point de vue conceptuel, l'argument correspond à un participant de la situation dénotée par le prédicat ;
2. du point de vue syntaxique, une expression qui occupe une des positions dans la structure argumentale doit être exprimable dans la phrase sous le contrôle syntaxique du prédicat. Cela signifie qu'il est possible de construire des phrases où l'expression apparaît soit comme un dépendant syntaxique direct du prédicat, soit comme un dépendant syntaxique d'un collocatif verbal du prédicat.

Ces deux critères nous donnent encore un argument supplémentaire au niveau de l'interface entre la syntaxe et la sémantique. D'ailleurs, il est aisé de concevoir la manière d'exprimer les positions argumentales dans le cas des prédicats verbaux, puisqu'elles sont exprimées par les différents types de dépendants régis par le verbe (le sujet et les compléments). Des exemples sont donnés ci-dessous :

(2.2) **Jean** aime **Marie**.

(2.3) **Le jeune homme** parle [**à sa fiancée**].

(2.4) **Il** donne [**un livre**] [**à Marie**].

En ce qui concerne les noms, il faut aussi considérer d'autres types de structures, notamment les collocations verbales (verbes supports et verbes de réalisation) qui permettent d'exprimer les arguments auprès des noms prédicats : c'est par exemple le cas d'*ultimatum* et de *lancer un ultimatum* où le sujet du verbe *lancer* fait partie de la structure argumentale d'*ultimatum*.

Enfin, les arguments doivent être différenciés des **modifieurs** : alors qu'un prédicat a besoin des arguments pour compléter son sens, les modifieurs, apparaissant avec un

5. Pour plus de détails sur ces questions, voir Grimshaw (1990) qui explique que tous les noms ne portent pas de structure argumentale.

prédicat, lui sont optionnels.[6] Des exemples de modifieurs sont donnés en gras dans les phrases suivantes :

(2.5) Jean aime **vraiment** Marie.

(2.6) Le jeune homme parle à sa fiancée **à Strasbourg**.

(2.7) **Hier** il a donné un livre à Marie.

Une large littérature est dédiée aux problèmes de la reconnaissance d'arguments et de modifieurs. Le lecteur intéressé pourra se référer notamment à (Ágel, 2006) pour un panorama des tests permettant de différencier arguments et modifieurs, ainsi qu'une discussion associée.

2.2.2 Grille Thêta et rôles thématiques

Les travaux dans la tradition du Gouvernement et du Liage (Chomsky, 1981) décrivent la structure argumentale d'un prédicat comme une liste appelée **Grille Θ**. Les membres de la grille Θ sont appelés des rôles Θ ou des rôles thématiques. La grille Θ marque les arguments syntaxiques qui sont aussi des arguments sémantiques. Par ailleurs, il arrive souvent que les rôles thématiques possèdent une étiquette qui donne le type de relation sémantique, comme par exemple le rôle d'AGENT ou de PATIENT. Les rôles thématiques les plus fréquemment utilisés sont : AGENT, PATIENT, THÈME, EXPÉRIENT, BÉNÉFICIAIRE, INSTRUMENT, LOCATIF, BUT et SOURCE (voir Saeed (2003, pp. 149-154) pour des exemples et une discussion sur la distinction des différents rôles). D'emblée, il n'est pas aisé de savoir quel rôle donner pour un exemple précis. Considérons les trois exemples 2.8, 2.9 et 2.10.

(2.8) Jean porte le sac **au phare**.

(2.9) Sylvie **lui** achète une voiture.

(2.10) **Marie** reçoit un bouquet de fleurs.

Dans l'exemple 2.8, Saeed (2003) considère l'élément *au phare* comme un BUT, et l'élément *lui* dans l'exemple 2.9 comme un BÉNÉFICIAIRE, mais pour l'exemple 2.10, la question est plus difficile : *Marie* est-elle le BUT, BÉNÉFICIAIRE ou le RECEVEUR ? De même quand nous prenons l'exemple 2.11, *M. Cooper* est-il AGENT ou PATIENT ?

(2.11) **M. Cooper** a sauté de la falaise.

Si les rôles thématiques sont au cœur de la structure prédicat-argument, mais que la distinction des rôles ne va pas de soi, il faut alors se demander pourquoi il est utile de considérer la notion de structure argumentale.

6. Bien entendu, l'optionnalité n'est pas toujours une condition suffisante pour différencier un argument d'un modifieur, puisqu'il existe des arguments dits « optionnels », mais c'est une question en soi qui sort largement du cadre de cette thèse.

2.2.3 Pourquoi la structure argumentale ?

En fait, la question des rôles thématiques en soulève aussi une autre : celle de savoir comment s'articulent ces rôles et les fonctions grammaticales au sein de la structure syntaxique de la phrase. C'est là le propre de l'interface entre la syntaxe et la sémantique. Cette mise en correspondance (rôles thématiques / fonctions grammaticales) montre rapidement que structure syntaxique et structure argumentale ne sont pas isomorphes et ne recouvrent donc pas le même type d'informations. Considérons les exemples 2.12 et 2.13. Dans l'exemple 2.12, la fonction grammaticale de *cric* est sujet mais son rôle thématique est INSTRUMENT, alors que dans l'exemple 2.13, *Jean* est sujet et son rôle thématique est AGENT. Ainsi différents rôles peuvent occuper les mêmes fonctions grammaticales.

(2.12) **Le cric** a soulevé la voiture.

(2.13) **Jean** a soulevé la voiture.

De même, certains rôles peuvent être omis comme dans l' exemple 2.14 où seuls les rôles d'AGENT, de PATIENT et d'INSTRUMENT sont exprimés, contrairement aux exemples 2.15 où l'AGENT n'est plus exprimé et 2.16 où seul le PATIENT est exprimé.

(2.14) [Corentin]$_{\text{AGT}}$ casse [la glace]$_{\text{PAT}}$ [au marteau]$_{\text{INSTR}}$.

(2.15) [Le marteau]$_{\text{INSTR}}$ casse [la glace]$_{\text{PAT}}$.

(2.16) [La glace]$_{\text{PAT}}$ casse.

De plus, la question des changements de diathèse (encore appelée alternance syntaxique) est une question centrale de la notion de structure argumentale. La **diathèse** est un trait grammatical décrivant comment s'organisent les rôles thématiques dévolus aux arguments par rapport au procès exprimé par le verbe, c'est le cas en particulier pour les rôles d'AGENT et de PATIENT ou THÈME. La diathèse affecte la répartition syntaxique et le marquage morphologique de ces rôles sur le verbe et les différents arguments. Dans la suite de cette section, nous décrivons la diathèse passive, même si ce n'est pas la seule qui pose des problèmes au niveau de la correspondance syntaxe / structure argumentale (et plus généralement syntaxe / sémantique), mais nous aurons l'occasion d'y revenir en détails lors de l'annotation du DeepSequoia (partie 6, p. 71).

Pour illustrer la diathèse passive, considérons les exemples 2.17 et 2.18, le premier est à l'actif avec **chat** comme *sujet* et AGENT et **souris** comme *objet* et PATIENT. Le second est au passif : **chat** est *complément d'agent*, mais son rôle thématique d'AGENT ne change pas, alors que **souris**, toujours PATIENT, occupe la fonction *sujet*. Ici encore, nous remarquons que la structure argumentale est plus régulière que la structure syntaxique. Elle est une forme de généralisation et d'abstraction par rapport aux problèmes que pose l'analyse syntaxique dite **surfacique** (nous reviendrons sur ce terme).

(2.17) **Le chat** mange **la souris**.

(2.18) **La souris** est mangée **par la chat**.

Contrairement aux structures syntaxiques pour les deux exemples d'alternance active / passive, la structure argumentale pour le verbe *manger* restera toujours la même :

manger'(AGENT, PATIENT). Seule la mise en correspondance entre la structure syntaxique et la structure argumentale va changer. Cette question est au centre des problèmes abordés par cette thèse. Il faut aussi noter que la problématique a fait l'objet d'un certain nombre de travaux d'annotation. C'est le cas notamment de VerbNet (Kipper-Schuler, 2006). VerbNet est un lexique de verbes qui annotent les rôles thématiques de chaque verbe et étend les classes de verbes de Levin (1993). Chaque classe est décrite par des rôles thématiques, des restrictions de sélection sur les arguments verbaux et des cadres (*frames*) consistant en une description syntaxique et sémantique des prédicats.

D'autres travaux, cependant, ont fait le choix de se départir de la notion de rôles thématiques estimant que la généralisation pour chaque verbe (au-delà de la notion de proto-agent et proto-patient de Dowty (1991)) n'est pas possible. C'est notamment le cas du projet FrameNet (Fillmore, 1976; Baker et al., 1998) et son homologue français, Asfalda, en cours d'annotation (Candito et al., 2014a; Djemaa et al., 2016), pour lequel chaque cadre sémantique (par exemple *Commerce*) annote des participants (encore appelés *éléments du cadre*) qui sont bien plus précis que les rôles thématiques que nous avons décrits jusqu'alors. Pour le cadre *Commerce*, des éléments du type BUYER, SELLER, GOODS ou encore MONEY seront utilisés. De plus, FrameNet propose, en plus d'un lexique, une annotation en corpus.

2.3 De la structure argumentale à la syntaxe profonde

L'une des clés de l'annotation en rôles thématiques, c'est la désambiguïsation des lemmes et tout particulièrement des verbes. En effet, pour le verbe *décliner*, par exemple, nous pouvons admettre deux sens distincts, le premier, signifiant *baisser*, est illustré à l'exemple 2.19, tandis que le second, signifiant *refuser*, l'est à l'exemple 2.20.

(2.19) **Ses revenus** déclinent **de 5%** chaque année.

(2.20) **Jean** a décliné **cette offre d'emploi**.

Dans le premier cas, *ses revenus* est un THÈME alors que dans le deuxième, *Jean* est un AGENT. Cette tâche d'annotation peut s'avérer longue et complexe et requiert un travail conséquent. Reprenant, par ailleurs une partie de l'argumentaire de Dowty (1991), la distinction des rôles thématiques même les plus intuitifs (c'est-à-dire le rôle d'agent et celui de patient) n'est pas toujours chose facile. Pour bâtir une partie de son argumentaire, Dowty (1991, p. 556) considère les verbes qui réfèrent à une transaction commerciale, comme dans les deux exemples en 2.21.

(2.21) (a) Jean vend le piano à Marie.

(b) Marie achète le piano à Jean.

Dowty (1991) estime que l'acheteur comme le vendeur agissent de manière volontaire lors de la transaction, puisque les deux doivent effectuer des actions pour que la transaction ait lieu. De fait, il est possible de considérer *Jean* et *Marie* comme deux agents pour un même verbe, ce qui viole le critère Θ défini par Chomsky, où une seule expression nominale peut porter un et un seul rôle thématique pour un même prédicat. L'argumentaire de Dowty

(1991) est, bien entendu, plus étayé et les exemples donnés sont plus larges. Ils ont tous pour but de montrer les difficultés lors de la désambiguïsation des rôles thématiques. Or, nous rappelons que cette thèse poursuit deux buts différents : le premier est de produire une pré-annotation de qualité dans le cadre d'une collaboration sur la création d'un schéma en syntaxe profonde et son application en corpus.[7] L'idée est d'obtenir rapidement un nombre suffisant de phrases manuellement validées, ainsi qu'un processus de pré-annotation efficace afin de pouvoir projeter ces annotations sur d'autres corpus. Le deuxième est de trouver des méthodes intéressantes pour exploiter les données dans le cadre d'applications de Traitement Automatique des Langues. De plus, ce genre d'annotations propre à VerbNet est un projet qui a été mené en parallèle de notre thèse par Pradet et al. (2014) et Danlos et al. (2014). Ainsi, en première approximation, nous avons préféré nous restreindre encore un peu pour récupérer, non plus une structure argumentale annotée en rôles thématiques, mais une structure argumentale qui reflète la **syntaxe profonde** d'un énoncé.

2.3.1 Grammaire Relationnelle et syntaxe profonde

Pour donner une définition de la syntaxe profonde, nous nous sommes fondé sur les travaux de la Grammaire Relationnelle (Perlmutter et Postal, 1984).

2.3.1.1 Contexte

Avec la Grammaire Relationnelle, Perlmutter et Postal (1984) proposent une alternative à la Grammaire Transformationnelle et Générative (Chomsky, 1957). En effet, la grammaire transformationnelle argue que la grammaire des langues naturelles peut être caractérisée comme un système formel, appelé **grammaire générative**, qui est un modèle de la capacité humaine à produire et comprendre une infinité d'énoncés considérés comme des chaînes de symboles. Chomsky (1957) définit alors une hiérarchie comprenant des grammaires de type 3 (rationnelles), de type 2 (hors contexte), de type 1 (contextuelles) et de type 0 (Turing-complet). Il montre formellement que les grammaires rationnelles ne peuvent pas capturer la structure de l'anglais et montre intuitivement que les grammaires hors contexte, même si elles ont en principe la puissance formelle pour définir l'ensemble des énoncés d'une langue comme l'anglais, ne peuvent pas le faire de manière satisfaisante. Chomsky (1957) pose alors les bases de la grammaire transformationnelle dans laquelle une *structure profonde* alimente un système de « transformations » pour livrer une *structure de surface*.

Perlmutter et Postal (1984) se sont rendus compte que le traitement du passif dans la théorie de la Grammaire Transformationnelle était loin d'être satisfaisant surtout lorsque des langues autres que l'anglais étaient considérées. Ils ont alors cherché un moyen de caractériser ce phénomène à travers une étude multilingue, puisque il est présent dans de nombreuses langues. Pour donner vie à une telle approche, il faut trouver une représentation

7. Le travail est une collaboration avec Marie Candito, Djamé Seddah et Éric Villemonte de La Clergerie et une partie de l'équipe Sémagramme du LORIA (laboratoire nancéen dépendant de l'Inria) : Karën Fort, Bruno Guillaume et Guy Perrier

qui s'abstrait de l'ordre surfacique des éléments. En utilisant les éléments qu'ils avaient collectés sur diverses langues, Perlmutter et Postal (1984) postulent qu'une phrase est un **réseau de relations grammaticales**. Les relations qu'ils définissent sont les suivantes : *sujet de* (*subject of*), *objet direct de* (*direct object of*) et *objet indirect de* (*indirect object of*). Partant de ce principe, les auteurs admettent que l'objet direct d'une phrase active est le sujet surfacique de la phrase passive correspondante, mais que le sujet d'une phrase active n'est ni le sujet surfacique ni l'objet direct surfacique de la phrase au passif correspondante. De fait, Perlmutter et Postal (1984) mettent en place un système abstrait de relations qui laisse de côté l'ordre des mots, la morphologie verbale et le marquage des cas en les considérant comme des artefacts d'une représentation surfacique. [8]

2.3.1.2 Idées centrales

Perlmutter et Postal (1984) définissent trois relations (appelées aussi *relations termes* (*term relations*)) qui doivent être considérées comme les primitives de la théorie. Ces relations ne peuvent pas être considérées en terme de marquage casuel, ou d'ordre des mots et encore moins en termes de structure syntagmatique. En fait, les auteurs définissent un ordre d'oblicité sur ces relations : $1 < 2 < 3 < \text{non} - \text{termes}$, où 1 = *sujet de*, 2 = *objet direct de* et 3 = *objet indirect de*. Parmi les relations non-termes, il existe les relations obliques (correspondant à peu près aux adjoints) et la relation dite « chômeur », que nous explicitons ci-dessous. Les relations entre les prédicats et les termes sont représentées au moyen de Réseaux Relationnels (*Relational Networks*), qu'il est possible de définir comme un multi-graphe étiqueté et orienté. L'usage du multi-graphe est utile à cause de la notion de strate dans le réseau. La strate dite **initiale** est celle qui contient la représentation prototypique de la phrase, c'est *le niveau profond*. La strate **finale**, quant à elle, représente la réalisation de la phrase *en surface* Un exemple complet des diverses strates est donné à la Figure 2.10.

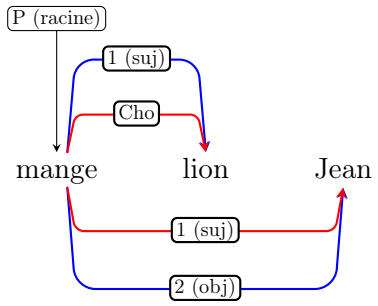


Figure 2.10: Exemple d'un réseau relationnel pour la transformation passif-actif et pour le couple de phrases : *Un lion mange Jean* / *Jean est mangé par un lion*.
Les arcs en rouge représentent la strate finale, alors que ceux en bleu représentent la strate initiale.

8. L'utilisation de relations grammaticales telles que celles décrites ci-dessus n'est pas nouvelle, car elles sont relativement standards dans diverses langues comme le français, l'anglais, l'allemand ou le latin. Cependant, Perlmutter et Postal (1984) utilisent ces relations dans un cadre formel, les plaçant au rang de théorie grammaticale à part entière.

Le Réseau Relationnel est régi par une série de contraintes et de règles. Les contraintes sont universelles et inviolables. Les règles, quant à elles, peuvent être adaptées en fonction de la langue. Les contraintes influent sur la topologie d'un tel réseau et des transformations possibles d'une strate vers une autre. Nous ne reprenons ici que les lois les plus basiques (et sans doute les plus essentielles) :

1. *The 1-Advancement Exclusiveness Law* : *Une proposition ne peut avoir au plus qu'un avancement vers 1.* Cela signifie que pour toute proposition, il n'est pas possible d'avoir plus d'un sujet (n°1 selon l'ordre d'oblicité défini ci-dessus).
2. *The Stratal Uniqueness Law* : *Au plus une expression nominale peut porter un terme donné dans un strate donnée.* Cette loi est proche du critère Θ. Elle indique que dans une strate donnée, il n'est pas possible d'avoir deux expressions nominales avec la même fonction grammaticale pour un même prédicat.
3. *The Chômeur Condition* : *Si une expression nominale i prend le terme d'une expression nominale j, alors j devient un* chômeur. Cette loi indique qu'en cas de changement de fonction grammaticale, puisqu'il n'est pas possible d'avoir deux fois la même fonction, alors l'expression nominale qui n'a pas changé devient chômeur.
4. *The Final 1 Law* : *Chaque proposition doit avoir un arc 1 dans la strate finale.* Cette loi indique que chaque proposition doit comportée au moins un sujet en surface.

Pour l'exemple de transformation passif-actif de la Figure 2.10, la loi n°1 est appliquée. elle permet à un terme (2 ou 3) d'être promu à un rang supérieur (d'après l'ordre d'oblicité donné ci-dessus). Les règles d'avancement sont $2 \rightarrow 1$, $3 \rightarrow 1$, $3 \rightarrow 2$, généralement. Cela permet à *Jean* de passer à l'état de *sujet de* (terme 1) à la voix passive. Cependant, deux termes 1 coexistent alors dans la représentation passive (*lion* et *Jean*), ce qui est rejeté par la loi n°2 et c'est donc la loi n°3 qui va s'appliquer et transformer *par le lion* en un *chômeur*. Enfin, la loi n°4, nous assure qu'il existe un sujet dans la phrase ainsi transformée. La Grammaire Relationnelle se construit donc autour d'étapes de transformations qui partent d'une représentation profonde (*initiale* dans la terminologie adoptée ici) pour arriver à une représentation surfacique (*finale*).

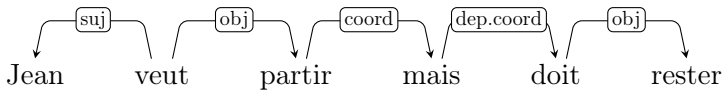
Nous ne sommes pas les premiers à réutiliser les notions de relations grammaticales **finales** / **initiales**. En effet, elles ont aussi été utilisées dans le cadre de l'élaboration de la hiérarchie de relations grammaticales (*grammatical relations* ou GR (Carroll et al., 1998)). GR est un formalisme qui a servi à l'annotation profonde (explicitation des sujets des verbes à contrôle, neutralisation de la diathèse passive, entre autres) d'un ensemble de 500 phrases issues du corpus multi-genres SUSANNE (Sampson, 1995) : le CBS 500. Le corpus permet l'évaluation d'analyseurs syntaxiques issus de théories diverses. Les relations finales et initiales sont utilisées pour marquer les changements de diathèse au sein de la théorie GR.

2.3.1.3 Cadre de sous-catégorisation et fonctions finales et canoniques

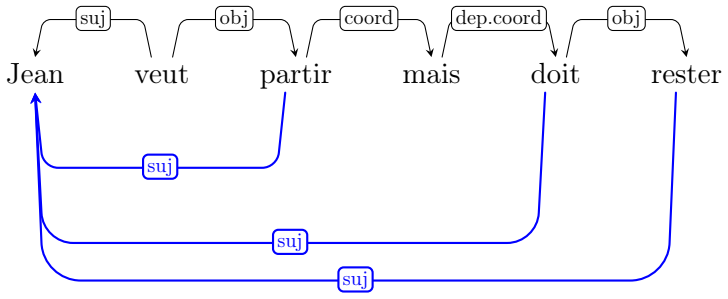
En effet, la syntaxe profonde poursuit deux buts : (i) l'explicitation des expressions qui remplissent le cadre de sous-catégorisation des prédicats [9] (ii) la neutralisation de l'alternance syntaxique (Candito et Perrier, 2015).

Explicitation des expressions qui remplissent le cadre de sous-catégorisation des prédicats. Pour ce faire, nous définissons le **cadre de sous-catégorisation final** comme le cadre de sous-catégorisation qui contient : (i) les fonctions observées associées aux arguments *exprimés* par le verbes (ii) dans le cas d'ellipses et/ou de verbes non conjugués, les fonctions des éléments qui seraient des argument du verbe si celui-ci était conjugué et utilisé sans ellipse. Cette formulation prend en compte le sujet (final) des infinitifs, des participes épithètes, des verbes coordonnés ou plus généralement tout argument partagé par plusieurs prédicats. Dans l'exemple 2.22, le cadre de sous-catégorisation final pour *partir* est partir'(sujet = Jean), et le cadre de sous-catégorisation final de *rester* est rester'(sujet = Jean), alors que la fonction grammaticale *sujet* entre *Jean* et *rester* n'est pas exprimée au niveau surfacique. Pour clarifier, la distinction surfacique / profond, nous donnons la représentation surfacique et profonde de l'exemple 2.22 à la Figure 2.11. Ainsi, les *arguments syntaxiques profonds* d'un verbe sont l'ensemble des expressions linguistiques qui portent une fonction grammaticale finale par rapport à ce verbe.

(2.22) Jean veut partir mais doit rester.



(a) Représentation surfacique

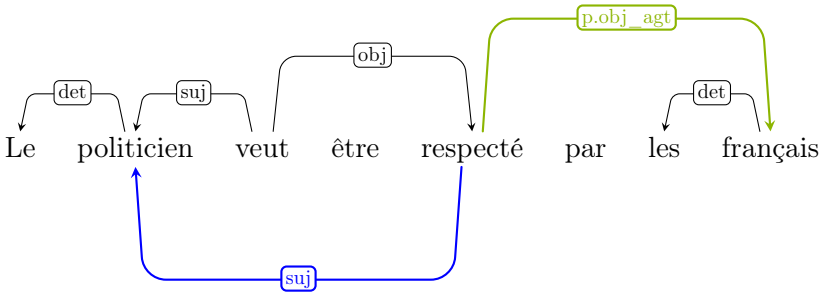


(b) Représentation profonde

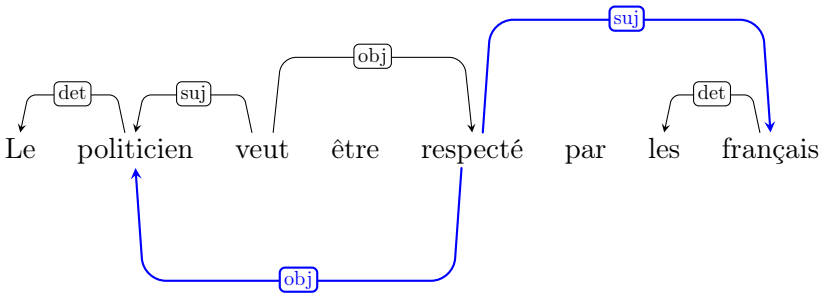
Figure 2.11: Distinction de l'analyse surfacique et de l'analyse profonde.

9. Le cadre de sous-catégorisation est complètement isomorphe à la grille Θ comme le rappelle Stowell (1981, pp. 35-37). C'est la raison pour laquelle, à partir de maintenant, nous n'emploierons plus que le terme de cadre de sous-catégorisation.

Neutralisation de l'alternance syntaxique. Pour capturer les régularités dues à l'alternance syntaxique et ce, sans avoir recours aux propriétés sémantiques ou aux rôles thématiques, mais en restant au niveau syntaxique, nous utilisons donc la distinction finale / initiale inspirée de la Grammaire Relationnelle. Nous rebaptisons le terme de fonction grammaticale *initiale* en fonction grammaticale *canonique* et les changements de diathèse sont alors des redistributions (Candito, 1999) des fonctions canoniques associées aux arguments syntaxiques. Le cadre de sous-catégorisation final est vu comme le résultat de l'application de 0 à n redistributions sur un **cadre de sous-catégorisation canonique**. Un exemple simple est celui d'un verbe au passif, comme par exemple *Le politicien veut être respecté par les français*. Elle est donnée à la Figure 2.12.



(a) Représentation profonde finale.



(b) Représentation profonde canonique.

Figure 2.12: Distinction de représentation profonde finale / canonique. Les différences sont marquées en bleu.

Pour un verbe transitif, le cadre de sous-catégorisation final est respecter'(sujet = politicien, complément d'agent = français), alors que le cadre de sous-catégorisation canonique est respecter'(sujet = français, objet = politicien). En effet, l'annotation finale marque *le politicien* comme sujet du verbe *respecter*, néanmoins l'alternance syntaxique due à la passivation du verbe donne un cadre de sous-catégorisation canonique où *le politicien* remplit la position **objet** du verbe *respecter* et non celle de **sujet**. [10]

10. À partir de maintenant, les arcs représentant des fonctions finales seront représentés en vert, alors que les arcs représentant des fonctions canoniques le seront en bleu. Lorsque la distinction final / canonique n'a pas de sens (c'est-à-dire que la fonction grammaticale reste la même après neutralisation de l'alternance syntaxique), nous garderons la couleur bleue.

Cette approche qui distingue cadres de sous-catégorisation *final* et *canonique* est la base de la définition de la syntaxe profonde utilisée lors de l'annotation du DeepSequoia (Candito et al., 2014a; Perrier et al., 2014). Néanmoins, d'autres théories existent pour représenter le niveau profond. Nous en considérons deux : la théorie de la Grammaire Lexicale Fonctionnelle ou LFG (Kaplan et Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) et la Théorie Sens-Texte ou MTT (Žolkovshij et Mel'čuk, 1967).

2.3.2 Syntaxe profonde dans la Grammaire Lexicale Fonctionnelle et la Théorie Sens-Texte

Nous choisissons ces deux théories pour leurs apparentes divergences en terme de représentations : la théorie LFG est issue de la tradition chomskienne de la grammaire générative qu'elle critique (Kaplan et Bresnan, 1982), alors que la MTT est à rapprocher des théories européennes sur les grammaires de dépendances avec des théories comme la Description Générative Fonctionnelle (Sgall et al., 1986) ou la *Word Grammar* (Hudson, 1984). Pour illustrer succinctement les particularités de ces deux théories, nous reprenons l'exemple de la Figure 2.12, *Le politicien veut être respecté par les français.*

2.3.2.1 Structure fonctionnelle et syntaxe profonde

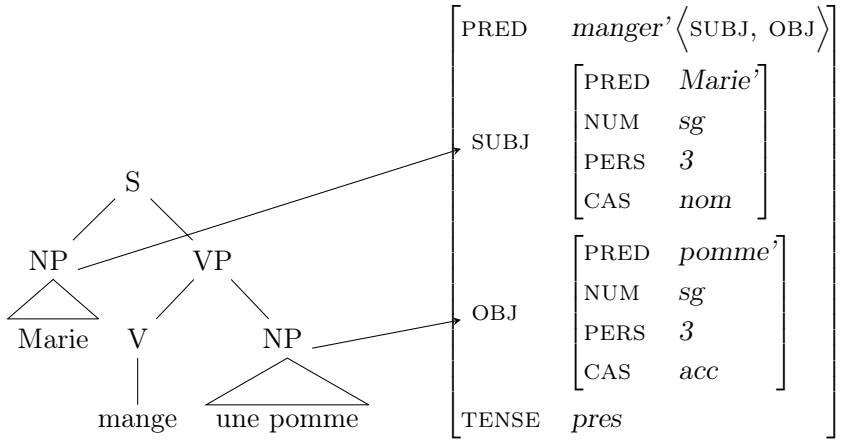
Avant toute chose, il convient de redonner quelques notions de la théorie. LFG sépare les informations concernant les structures linéaire et syntagmatique de l'analyse fonctionnelle d'une phrase. L'ordre des mots ainsi que les constituants sont représentés via la *c(onstituent)-structure* qui est une représentation arborescente classique. La *f(unctional)-structure* se présente comme une matrice attributs-valeurs (*attribute-value matrix*, appelée encore *structure de traits* ou *feature structure*) et encode les informations syntaxiques fonctionnelles sur les relations grammaticales, le temps et l'aspect, le cas, le nombre, la personne, etc. Nous représentons un exemple de c-structure (Figure 2.13 à gauche) et un exemple de f-structure (Figure 2.13, droite) pour la même phrase *Marie mange une pomme.*

La f-structure est une projection à partir de la c-structure puisqu'elle est liée à la c-structure via un système formel d'annotations qui forment des règles de réécritures. Un exemple est donné à la Figure 2.14 qui représente une grammaire syntagmatique classique à laquelle sont ajoutées des équations fonctionnelles. Ces équations donnent les clés pour calculer la f-structure (les informations concernant le genre, le nombre, la personne, etc. sont représentées dans le lexique). LFG utilise des méta-variables pour écrire les équations fonctionnelles :

(a) $\downarrow$ signifie « ce nœud ».

(b) $\uparrow$ signifie « mon parent » (c'est-à-dire le nœud dominant immédiatement).

Ainsi, l'équation $\uparrow=\downarrow$ signifie « toutes les informations que je possède appartiennent aussi à mon parent ». Cette équation décrit donc une tête. L'équation ($\uparrow$SUBJ$=\downarrow$) signifie « je représente la fonction sujet de mon parent ». Les équations fonctionnelles établissent un rapport avec la c-structure via une projection mathématique notée ϕ et représentée



(a) C-structure. (b) F-structure.

Figure 2.13: Exemple de représentation LFG pour *Marie mange une pomme*. La fonction ϕ est marquée par les flèches.

$$\text{S} \rightarrow \underset{(\uparrow\text{SUBJ}=\downarrow)}{\text{NP}} \quad \underset{\uparrow=\downarrow}{\text{VP}} \qquad\qquad \text{VP} \rightarrow \underset{\uparrow=\downarrow}{\text{V}} \quad \underset{(\uparrow\text{OBJ}=\downarrow)}{\text{NP}}$$

Figure 2.14: Annotations LFG pour un fragment simpliste du français.

par les flèches à la Figure 2.13. L'effet de cette projection réside dans le fait que les deux représentations (c-structure et f-structure) s'imposent des contraintes mutuelles. En effet, une analyse ne peut être correcte que si la f-structure est complète et cohérente et que la structure syntagmatique permet une telle structure fonctionnelle. Les différents fragments d'une même phrase sont combinés par unifications successives.

Il faut aussi noter que la f-structure est un niveau abstrait qui ne présuppose ni un ordre des mots particulier, ni une forme surfacique pour une phrase. C'est le rôle de la c-structure que d'encoder des informations propres à chaque langue. Par ailleurs, la f-structure inclut un nombre restreint de *fonctions grammaticales* (qu'il est possible de mettre en rapport avec les relations grammaticales de la Grammaire Relationnelle, cf. section 2.3.1, page 28). Dans l'exemple 2.13, elles sont deux : SUBJ et OBJ. Ces fonctions sont choisies comme potentiellement utilisables pour chaque langue. Elles sont au nombre de sept : SUBJ, OBJ, OBL(ique), COMP(lément), XCOMP(lément), ADJUNCT.

COMP et XCOMP sont les fonctions utilisées pour les propositions subordonnées dans une phrase. Le préfixe *X* devant un nom de fonction signifie généralement que la fonction est *ouverte*, c'est-à-dire que son sujet n'est pas réalisé au niveau de la c-structure et sera soumis à une équation de contrôle. Il existe des valeurs différentes selon la catégorie du syntagme en général associé. Abeillé (2007, p. 66) en donne une nomenclature. OBL est utilisé pour les arguments prépositionnels du verbe ou les verbes à double construction

objet comme dans l'exemple 2.23.

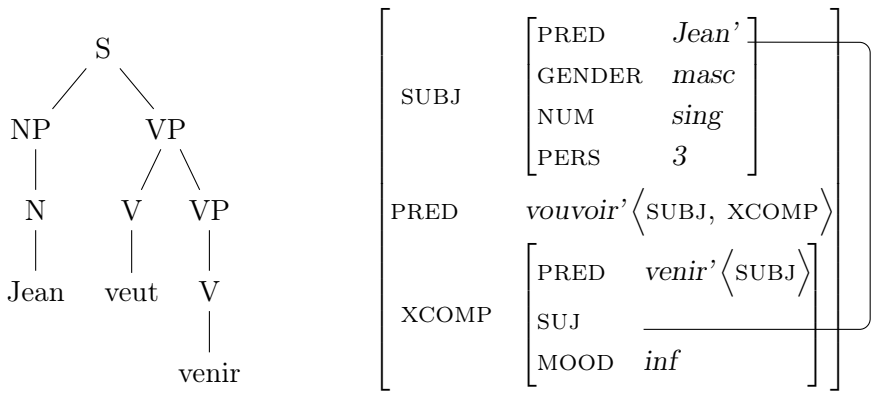
(2.23) John gave Mary a book.

Notre exemple présente deux phénomènes qui doivent être traités au niveau profond : le contrôle sujet et le passif. Concernant le contrôle sujet, LFG traite le phénomène par un partage de valeur entre les deux attributs SUJ (Dalrymple, 2001, pp. 313-360). Nous le notons grâce à un lien comme à la Figure 2.16. De plus, les équations fonctionnelles simplifiées pour le contrôle sujet sont indiquées à la Figure 2.15.

$$\begin{array}{lll} \text{VP} \rightarrow & \text{V} & \text{VP} \\ & \uparrow{=}\downarrow & (\uparrow\text{XCOMP}) = \downarrow \end{array}$$

$$\begin{array}{ll} \text{veut, V :} & (\uparrow\text{PRED}) = \text{vouloir}\langle\text{SUBJ, XCOMP}\rangle \\ & (\uparrow\text{SUBJ}) = (\uparrow\text{XCOMP SUBJ}) \end{array}$$

Figure 2.15: Équations fonctionnelles simplifiées pour les verbes à contrôle sujet.



(a) C-structure. (b) F-structure.

Figure 2.16: Représentation du contrôle en LFG.

Le lien créé entre *Jean* sujet de *vouloir* et le sujet attendu de *venir* permet de représenter une réentrance dans la structure de traits et modélise donc implicitement un graphe.

Concernant le traitement du passif, Bresnan (1982) propose une analyse lexicale qui établit la relation actif-passif au niveau des prédicats concernés et non au niveau des structures syntaxiques. Dans cette optique, il existe, pour chaque verbe transitif, une forme verbale passive en plus des formes verbales utilisées pour l'actif. Les différences de construction entre phrases actives et passives reflètent la différence qu'il existe entre la sous-catégorisation du passif et celle des formes actives : les verbes actifs sélectionnent (au niveau du trait PRED) un sujet et un objet direct alors que le passif sélectionne un sujet et un complément d'agent introduit par *par* (complément qui est optionnel). Le verbe *respecter* a ainsi deux formes données à la Figure 2.17.

L'analyse donnée suppose que l'auxiliaire du passif *être* a la même description fonctionnelle que le verbe copule, c'est-à-dire (↑PRED) = être'<XCOMP>SUBJ, en notant la fonction

respecte, V : (↑PRED) = respecter<SUBJ, OBJ>
(↑MOOD) = ind
(↑SUBJ NUM) = sing
(↑SUBJ PERS) = 3

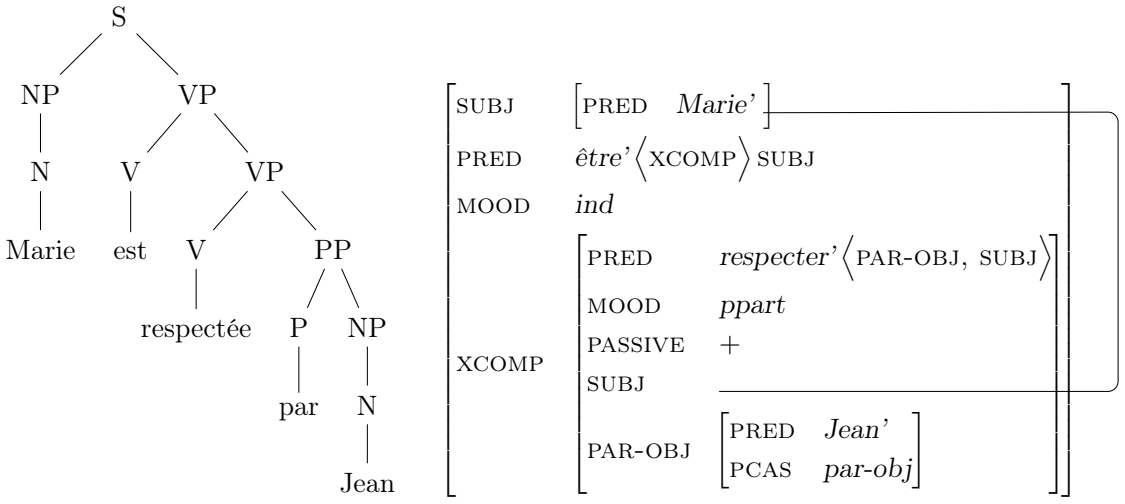
respecté, V : (↑PRED) = respecter<(PAR-OBJ,) SUJ>
(↑MOOD) = ppart
(↑SUBJ NUM) = sing
(↑SUBJ PERS) = 3
(↑PASSIVE) = +

Figure 2.17: Équations fonctionnelles pour le verbe respecter à l'actif et au passif.

sujet en dehors des chevrons car *être* ne lui assigne pas de rôle sémantique. Par ailleurs, le sujet de *respectée* est coïndicé avec le sujet de *être* (cf. Figure 2.19). De plus, nous définissons la règle syntagmatique et les équations fonctionnelles pour le VP sous *être* comme à la Figure 2.18.

VP → V VP
↑=↓ (↑XCOMP) =↓
(↓MOOD) = inf | ppart

Figure 2.18: Équations fonctionnelles pour un VP sous *être* (auxiliaire passif).



(a) C-structure. (b) F-structure.

Figure 2.19: Représentation du passif en LFG.

Les équations fonctionnelles à la Figure 2.17 ne sont pas à elles seules suffisantes pour représenter la diathèse passive. Il faut montrer la parenté sémantique du verbe entre l'actif

et le passif, puisque les deux phrases (*Jean respecte Marie* et *Marie est respectée par Jean*) auront une c-structure et f-structure différentes. Pour ce faire, il est nécessaire d'introduire la notion de **règle lexicale** qui définit la relation directement entre les fonctions grammaticales des traits PRED. La règle du passif est donc considérée comme un changement de fonction grammaticale (dans la description du cadre de sous-catégorisation de l'entrée lexicale) : le sujet devient complément prépositionnel dans le passif dit « long » (c'est-à-dire avec complément d'agent) ou disparaît dans le passif dit « court » et l'objet devient sujet. Ces changements se notent par une double flèche horizontale (⇒) et l'optionnalité est notée par une disjonction (|) entre rien (∅) et PAR-OBJ :

SUJ ⇒ ∅ | PAR-OBJ

OBJ ⇒ SUJ.

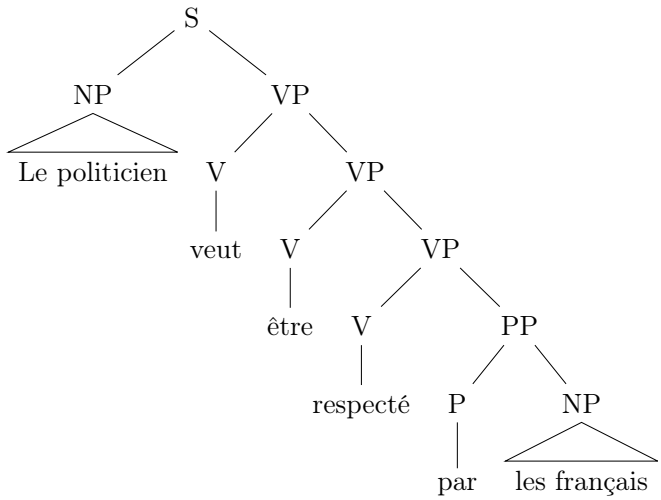
Il est désormais possible de montrer la représentation de *Le politicien veut être respecté par les français* (Figure 2.20). Nous remarquons que *le politicien*, sujet du verbe *vouloir*, remplit la fonction de sujet pour *être* mais aussi pour *respecter*. Cette analyse est conforme avec notre analyse donnée en Figure 2.12a. De plus, la distinction c-structure et f-structure et l'ajout des équations fonctionnelles permet une représentation flexible et indépendante de la langue tout en utilisant une représentation en constituants.

Après avoir présenté rapidement la théorie LFG et montré comment par l'utilisation d'une structure en constituants (la c-structure) et d'une structure fonctionnelle prenant la forme d'une matrice attribut-valeur (la f-structure), il était possible de représenter des phénomènes profonds comme l'explicitation du sujet d'un infinitif dans une construction à contrôle et la neutralisation de la diathèse passive, nous montrons maintenant comment la Théorie Sens-Texte, par l'utilisation des dépendances, donne une représentation tout aussi élégante de la même phrase.

2.3.2.2 Niveau profond et Théorie Sens-Texte en MTT

La théorie Sens-Texte (*Meaning-Text Theory*) est une théorie qui pose un cadre théorique pour la construction de modèles de la langue naturelle. On appelle ces modèles les **Modèles Sens-Texte**. Depuis ses débuts dans les années 1960, la Théorie Sens-Texte a mis l'accent sur le *sens* et considère la langue naturelle comme un outil dont la fonction première est de véhiculer le sens. La Théorie Sens-Texte a toujours pensé les relations comme la clé de voûte de la langue naturelle et a donc fait un usage intensif du concept de dépendance (et de syntaxe de dépendances) au détriment des constituants. La langue naturelle est considérée comme une mise en correspondance multivoque (*many-to-many mapping*) entre un ensemble infini mais dénombrable de sens et un ensemble infini mais dénombrable de textes. La correspondance entre les sens et les textes est multivoque dans la mesure où un sens donné peut être exprimé par différents textes (synonymie) et parallèlement un texte peut avoir plusieurs sens (ambiguïté, homonymie et polysémie).

Par ailleurs, étant donné la complexité de la correspondance *Sens-Texte*, des niveaux intermédiaires doivent être utilisés ; plus précisément un niveau syntaxique et un niveau morphologique. Ces deux niveaux correspondent à deux notions autonomes de la linguis-



(a) C-structure.

$$
\begin{bmatrix}
\text{SUBJ} & \begin{bmatrix}\text{PRED} & \textit{le politicien'}\end{bmatrix} \\
\text{PRED} & \textit{vouloir'}\langle \text{SUBJ}, \text{XCOMP}\rangle \\
\text{MOOD} & \textit{ind} \\
\text{XCOMP} & \begin{bmatrix}
\text{PRED} & \textit{être'}\langle \text{XCOMP}\rangle \text{ SUBJ} \\
\text{MOOD} & \textit{inf} \\
\text{SUBJ} & \boxed{1} \\
\text{XCOMP} & \begin{bmatrix}
\text{PRED} & \textit{respecter'}\langle \text{PAR-OBJ}, \text{SUBJ}\rangle \\
\text{MOOD} & \textit{ppart} \\
\text{PASSIVE} & + \\
\text{SUBJ} & \boxed{1} \\
\text{PAR-OBJ} & \begin{bmatrix}\text{PRED} & \textit{les français'} \\ \text{PCAS} & \textit{par-obj}\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

(b) F-structure.

Figure 2.20: Exemple de représentation LFG pour *Le politicien veut être respecté par les français.*

tique : la phrase et le mot. Par ailleurs, chaque niveau (ou domaine) – sauf celui sémantique – est subdivisé en deux sous-domaines : un profond et un surfacique. Le premier est orienté vers le sens (c'est-à-dire le fond de l'expression) et le second vers le texte (c'est-à-dire la forme de l'expression). Il y a donc sept niveaux :

1. Sémantique.
2. Syntaxique (profond et surfacique).
3. Morphologique (profond et surfacique).
4. Phonologique (profond et surfacique).

Une représentation d'un énoncé au niveau n est un ensemble d'objets formels appelés **structures**. La Théorie Sens-Texte distingue toujours une structure dite *centrale* : au niveau sémantique, c'est un graphe, au niveau syntaxique, c'est un arbre de dépendances, etc. Chaque niveau de représentation reflète un aspect précis de l'organisation d'un énoncé en mettant en exergue des relations linguistiques de différentes natures et, par conséquent, faisant usage de plusieurs formalismes. L'idée ici n'est pas d'unifier mais bien de distinguer. Pour ce faire, le Modèle Sens-Texte a recours à deux types de règles :

(a) Des règles de *correspondance* qui utilisent des correspondances entre des fragments de deux niveaux adjacents. C'est la majeure partie d'un Modèle Sens-Texte.

(b) Des règles d'*équivalence* qui établissent des équivalences entre les représentations d'un même niveau (par exemple des règles de paraphrases entre deux représentations profondes).

Mel'čuk distingue deux niveaux syntaxiques : le niveau profond (*deep syntactic structure*) et le niveau surfacique (*surfacic syntactic structure*). La structure syntaxique profonde est représentée par un graphe acyclique dirigé (DAG) dont les nœuds sont étiquetés par des lexèmes sémantiques comme le souligne Kahane (2003). En effet, à l'origine, cette représentation faisait usage d'un arbre de dépendances, mais les liens de coréférence explicités à ce niveau produisent un DAG et non un arbre.

De plus, les lexèmes sémantiques sont dits « pleins ». En effet, les prépositions, les conjonctions, les auxiliaires sont considérés comme sémantiquement vides et ne prennent pas part à la structure syntaxique profonde. Ils ne se retrouvent que dans la structure de surface. Enfin, une unité lexicale profonde peut être trois choses :

1. Un lexème.
2. Un phrasème (une expression idiomatique).
3. Une *fonction lexicale*.

Les fonctions lexicales sont des éléments fondamentaux de la Théorie Sens-Texte. Ce sont des outils formels qui décrivent les relations syntagmatiques et paradigmatiques entre lexèmes : collocations et dérivations sémantiques au sens large. Les fonctions lexicales sont des fonctions au sens mathématique du terme et peuvent donc prendre des arguments (qui sont des unités lexicales). La Théorie Sens-Texte distingue environ soixante fonctions lexicales standards qu'il est possible de classer selon trois axes :

1. La distinction syntagmatique / paradigmatique. Les fonctions syntagmatiques caractérisent un lexème alors que les fonctions paradigmatiques correspondent aux dérivés d'un lexème (au sens large incluant synonymes et antonymes, par exemple). La fonction lexicale MAGN(·) qui représente le sens d'*intense/très* rentre dans la première catégorie :
   — MAGN(vent) : fort, puissant ;
   — MAGN(pluie) : torrentielle ;
   — MAGN(pleuvoir) : des hallebardes.
2. Une distinction est faite entre les fonctions dites standards des fonctions non standards. Une fonction standard peut être appliquée à un nombre importants de mots-clés. Elles sont universelles dans le sens où elles s'appliquent à de nombreuses langues, c'est le cas de MAGN(·). En revanche, une fonction sera non standard si elle est applicable qu'à une poignée ou un seul mots-clé. Par exemple, la fonction LEAP(·) ne s'applique qu'à ANNÉE pour désigner une année bissextile.
3. Les fonctions lexicales sont divisées en fonctions *simples* et fonctions *complexes*. Les fonctions complexes résultent de la composition de fonctions simples.

Maintenant qu'est définie une unité lexicale profonde, nous nous tournons vers les relations de dépendances que portent les arcs du DAG de dépendances. Onze relations syntaxiques profondes sont utilisées, parmi lesquelles :

1. Six relations actancielles (I, II, ..., VI) qui correspondent grossièrement aux ARG_0, ..., ARG_n dans des théories comme PropBank (Palmer et al., 2005).
2. Deux relations attributives : $\text{ATTR}_{\text{rest}}$ et $\text{ATTR}_{\text{qual}}$. Elles sont utilisées pour marquer n'importe quel modifieur (circonstanciels ou attributifs).
3. Une relation d'adjonction (*appenditive relation*) : APPEND pour marquer les parenthèses, les interjections, les apostrophes comme dans l'exemple 2.24.
4. Deux relations de coordination : COORD et QUASI-COORD, la dernière étant utilisée pour des constructions spéciales du type : *Il est venu le 27 juin dans la soirée, vers vingt heures*. Dans cet exemple, la relation entre les compléments de temps sont QUASI-COORD puisque chaque conjoint est une élaboration sur le conjoint précédent.

(2.24) Où es-tu, Alice ?

Les relations syntaxiques profondes définies par la Théorie Sens-Texte se veulent suffisamment universelles pour décrire la syntaxe profonde de n'importe quelle langue.

La deuxième structure essentielle dans la représentation syntaxique de la Théorie Sens-Texte est la **structure syntaxique de surface**. Formellement, elle est représentée par un arbre de dépendances dont les nœuds sont étiquetés par les mots de la phrase et dont les arcs portent une étiquette syntaxique spécifique à chaque langue. Par ailleurs, la structure de surface conserve tous les mots de la phrase. Nous avons donc ici deux différences majeures avec la structure profonde décrite ci-dessus. Il est à noter que la structure syntaxique de surface est fortement liée à l'ordre linéaire des mots de la phrase, alors que son pendant

profond ne l'est pas. Ensuite, toutes les fonctions lexicales sont retransformées (au moyen de dictionnaires) en des lexèmes réels. Cependant, il n'existe pas une correspondance biunivoque (*one-to-one mapping*) entre les unités lexicales de la structure profonde et les mots de la structure de surface. Il arrive bien souvent que deux mots en surface ne correspondent qu'à un seul en profond, etc.

Il est alors possible de donner la représentation profonde pour *Le politicien veut être respecté par les français* (Figure 2.21). Le premier constat est que seuls les mots sémantiquement pleins sont conservés au niveau profond. Le participe est marqué comme étant un passif. Enfin, les relations actancielles sont utilisées pour marquer que *politicien* est agent de *vouloir* mais patient de *respecter* (dû à la diathèse passive) et que *français* est agent de *respecter*. Nous n'avons pas reproduit les traits morphologiques qui existent sur chaque lexème profond pour ne pas entraver la lecture. Concernant le traitement du verbe à contrôle, nous reprenons *quasi-dépendance* énoncé par Kahane (2001) et représenté sur le schéma par un arc en pointillé. Cette dépendance indique la relation entre le verbe contrôlé et son sujet.

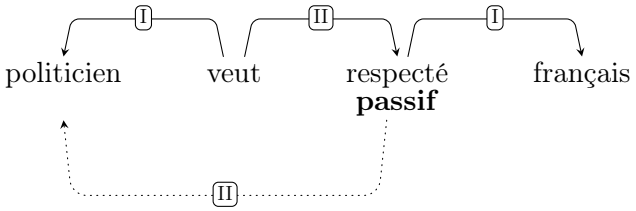


Figure 2.21: Représentation MTT au niveau profond pour l'exemple *Le politicien veut être respecté par les français.*

2.3.3 Discussion : la syntaxe profonde, un graphe de dépendances ?

Discuter la théorie LFG et la MTT n'est pas un choix anodin. Comme nous le disions page 33, les deux théories ont été choisies pour leurs divergences **apparentes**. En effet, la première fait le choix de deux structures : une structure en constituants pour représenter le niveau surfacique et une structure fonctionnelle pour encoder les informations profondes, alors que la seconde utilise un graphe de dépendances et une représentation à plusieurs niveaux. Cependant, l'utilisation de la structure fonctionnelle en LFG est une représentation sous forme de matrice attribut-valeur d'un graphe de dépendances. De fait, il est possible d'aller plus loin et de dire que les choix faits par les deux théories sont complémentaires et donnent lieu à une représentation similaire. En effet, lorsque l'explicitation des sujets non réalisés en syntaxe de surface est en jeu, la représentation sous-jacente ne peut plus être un arbre, puisqu'un même token est l'argument de plusieurs prédicats. L'utilisation d'une f-structure et de la réentrance (matérialisée par les liens sur les Figures 2.16 et 2.20) est une manière de représenter les gouverneurs multiples du graphe sémantique en MTT (Figure 2.21). Enfin, la représentation profonde que nous cherchons à produire est à mi-chemin entre la f-structure LFG et le graphe sémantique MTT : nous réutilisons les fonctions grammaticales utilisées au sein de la f-structure et la structure de graphe utilisée en MTT.

Chapitre

3 Ressources à large couverture pour l'interface syntaxe-sémantique

Sommaire

Introduction

Après avoir discuté des théories et positionné le problème qui nous intéresse, nous décrivons ici les diverses ressources créées dans le but de mieux comprendre la structure argumentale à l'échelle de la phrase mais surtout utilisables dans des tâches issues du Traitement Automatique des Langues. Dans un premier temps, nous nous concentrons sur les données existant dans des langues autres que le français. Nous explorons le cas du Prague Dependency Treebank (Hajič et al., 2000) pour le tchèque et de l'AnCora Corpus (Mille et al., 2013) pour l'espagnol. Nous nous attardons enfin sur les méthodes automatiques utilisées pour acquérir des ressources dites « profondes » à partir de corpus syntaxiques existants, telles que les corpus DM (Ivanova et al., 2012) et PAS (Miyao et al., 2004) que nous décrivons rapidement.

3.1 Structures prédicat-argument & Corpus en syntaxe profonde

Comme nous l'expliquions au chapitre précédent, le but de notre représentation profonde est l'explicitation des expressions qui remplissent le cadre de sous-catégorisation des prédicats et la neutralisation de l'alternance syntaxique sans avoir recours aux rôles thématiques qui demandent la désambiguïsation de chaque lemme en corpus. De fait, des ressources comme PropBank (Palmer et al., 2005) (pour les verbes) ou NomBank (Meyers et al., 2004) (pour les noms), quoique sémantiques et manuellement annotées, ne rentrent pas dans ce cadre théorique.

Néanmoins, l'influence de ces deux ressources est sans conteste puisque la majorité des approches d'analyse sémantique de surface dont nous nous inspirons ont émergé suite à leur création (Titov et al., 2009; Henderson et al., 2013; Lluís et al., 2013; Lluís et al., 2014). Le Proposition Bank (ou PropBank) (Palmer et al., 2005) est une annotation en rôles thématiques des verbes du Penn Treebank (Marcus et al., 1993) afin d'en déduire une structure prédicat-argument fine. Les arguments annotés sont de la forme $\textsc{Arg}_0$ à $\textsc{Arg}_n$ et l'annotation fait la différence entre arguments et modifieurs. PropBank considère les rôles verbe par verbe, en sachant que les arguments au-delà d'$\textsc{Arg}_1$ ne sont plus généralisables à l'ensemble des verbes (Palmer et al., 2005). En effet, $\textsc{Arg}_0$ et $\textsc{Arg}_1$ correspondent aux proto-roles d'agent et de patient/thème de (Dowty, 1991). Pour les autres arguments, la généralisation est moins claire, comme le montre Yi et al. (2007). Les rôles de PropBank, étant spécifiques à chaque verbes, conduisent à des inconsistances et font de la tâche de prédiction de tels rôles une gageure. Néanmoins, ce type d'annotation a eu une influence considérable dans le domaine (via deux campagnes d'évaluation (*shared tasks* (Surdeanu et al., 2008; Hajič et al., 2009)) et nous conduisent à nous positionner. Désireux de produire des modèles capables de prédire des annotations riches, susceptibles d'être vecteur d'une interface syntaxe-sémantique efficace, nous voulons aller au-delà en terme de précision de ce que certains ont pu considérer comme de la sémantique de surface (*shallow semantic*). C'est pourquoi nous nous concentrons ici sur le Prague Dependency Treebank et l'AnCora corpus.

3.1.1 Le Prague Dependency Treebank

Le Prague Dependency Treebank (Hajič, 1998) est l'un des corpus syntaxiques à large échelle (environ 49 000 phrases) les plus importants hors langue anglaise. En effet, il présente une annotation multi-strate qui permet de rendre compte des phénomène syntaxiques de la phrase, mais aussi de son interprétation sémantique via un niveau profond. Contrairement au Penn Treebank (Marcus et al., 1993), le Prague Dependency Treebank a fait l'objet d'une annotation plus longue et plus fine surtout en ce qui concernent les cas d'alternances syntaxiques, comme nous allons le voir. Il prend ses racines dans les théories des grammaires en dépendances, car la langue tchèque est une langue dont l'ordre des mots est relativement libre. De fait, il n'est pas souhaitable de faire une analyse en constituants,

puisque le non-projectivité serait trop importante.

Ainsi, le corpus est annoté selon trois niveaux différents. Le premier est l'annotation morphologique qui inclut les parties du discours (*parts of speech*), le second est une annotation surfacique en dépendances appelée *niveau analytique* et le troisième est un niveau profond qui suit la théorie de la Description Générative Fonctionnelle (*Functional Generative Description*, (Sgall et al., 1986)). Nous ne décrirons pas en détails le premier niveau d'annotations. Le lecteur intéressé pourra se référer à (Hajič et al., 2000).

Le niveau analytique (syntaxique) est un arbre de dépendances qui représente les relations syntaxiques de surface. Les principes de base adoptés pour l'annotation sont les suivants : (a) chaque mot et ponctuation sont représentés par exactement un nœud et aucun nouveau nœud n'est ajouté à ce niveau, seule la racine est marquée (b) l'annotation forme un arbre de dépendances étiqueté potentiellement non-projectif dont chaque nœud est composé de son token, de sa catégorie morpho-syntaxique et de son lemme.

Le troisième niveau d'annotations est le niveau tectogrammatical qui représente la syntaxe profonde de la phrase. C'est ce niveau qui nous intéresse particulièrement. Il a pour but de décrire le sens de la phrase et non plus sa syntaxe. L'analyse résultante est un arbre de dépendances **projectif**. Contrairement au niveau analytique, les structures arborées tectogrammaticales présentent les caractéristiques suivantes :

— Un nœud peut représenter plus qu'un mot. Par ailleurs, seuls les mots sémantiquement pleins représentent un nœud de l'arbre. Les autres (auxiliaires, prépositions, etc.) sont coindiciés avec les mots sémantiquement pleins.

— La non-projectivité n'est pas autorisée, les nœuds sont donc ordonnés de telle sorte que le critère de projectivité soit respecté.

— Les fonctions de surface sont remplacées par les fonctions tectogrammaticales (agent, patient, adresse, origine, thème, effet, etc.) au nombre de quarante.

— Chaque nœud de l'arbre contient le lemme du token et des traits morpho-syntaxiques.

— Un attribut est ajouté pour capturer l'articulation topic-focus (Sgall et al., 1986). Il peut prendre trois valeurs : T(opic), F(ocus), C(onstrast).

Ce corpus présente l'avantage d'avoir une annotation profonde fine avec des fonctions tectogrammaticales qui rendent compte de la complexité syntaxique de la langue. Néanmoins, la multiplication du nombre de fonctions rendent l'analyse syntaxique automatique difficile, comme a pu le montrer les résultats de la campagne d'évaluation 2014 SemEval d'analyse sémantique en dépendances à large couverture (Oepen et al., 2014) [1].

3.1.2 L'annotation MTT de l'AnCoRa Corpus

Un autre corpus a vu le jour récemment pour l'espagnol : l'AnCoRa-UPF (Mille et al., 2013). De même que pour le Prague Dependency Treebank, c'est un corpus multi-strate.

1. Le corpus utilisé durant cette campagne est une projection de la couche tectogrammaticale sur le Penn Treebank II (Marcus et al., 1993) appelé Prague Czech English Dependency Treebank (Cinková et al., 2009). Ce corpus a été partiellement validé à la main.

Cependant, la différence majeure réside dans le fait que l'AnCoRa-UPF corpus est annoté suivant la Théorie Sens-Texte. Au total, environ 3 500 phrases ont été annotées. Nous ne présentons ici que les deux niveaux d'annotations qui permettent l'extraction de structures prédicat-argument, à savoir le niveau profond et le niveau sémantique.

3.1.2.1 Niveau syntaxe profonde

À ce niveau, la représentation de la phrase est faite sous forme d'arbre de dépendances. Chaque nœud est une unité lexicale profonde (cf. section 2.3.2.2, page 37). Le corpus utilise les relations de dépendances profonde définies par la théorie, telles que APPEND, ATTR, COORD, I, II, III, etc. Par ailleurs, un certain nombre de nœuds présents au niveau surfacique sont supprimés, car inutiles au niveau profond, c'est le cas des prépositions, des auxiliaires, les déterminants définis et indéfinis et les pronoms relatifs. Enfin la structure profonde marque la diathèse passive en indiquant sur le verbe principal s'il est à l'actif ou au passif. Le corpus marque aussi les liens de coréférence grâce à la relation COREF.

En ce qui concerne l'annotation de ce niveau, un outil automatique permet de transformer le niveau surfacique en niveau profond en proposant une mise en correspondance des relations surfaciques et profondes. De plus, il est assez aisé de supprimer les mots vides de la structure profonde et ce de manière automatisée. Les annotateurs sont donc amenés à faire des corrections sur la structure, à marquer les liens de coréférence et à numéroter les arguments, ce qui réduit drastiquement le temps d'annotation.

3.1.2.2 Niveau sémantique

La structure sémantique est un graphe acyclique dirigé qui représente la structure argumentale de la phrase. Les nœuds au niveau sémantique sont les mêmes que ceux du niveau profond. Cependant, les auteurs choisissent d'ajouter des nœuds dits *méta* qui prennent en compte les informations syntaxiques présentes comme traits sur les nœuds profonds : le temps, le nombre, la possession. Par ailleurs, un nœud nommé ÉLABORATION est utilisé pour connecter les nœuds profonds reliés par l'étiquette ATTR ou APPEND. Enfin, les chiffres romains utilisés pour marquer le premier, second, troisième, ... arguments sont redéfinis grâce à la nomenclature utilisée par PropBank (ARG$_0$, ..., ARG$_n$). En ce qui concerne l'annotation de ce niveau, seul un outil automatique est utilisé, car toutes les informations sont déjà présentes au niveau profond, il est donc facile de transférer et de modifier l'arbre pour en faire un graphe.

3.2 Annotation automatique & extraction de grammaire

3.2.1 DeepBank & DM corpus

Le DeepBank (Flickinger et al., 2012) est une annotation dynamique (Oepen et al., 2002) du Penn Treebank via une grammaire HPSG (Pollard et Sag, 1994) manuelle à large couverture : l'*English Resource Grammar* (ERG, (Copestake et Flickinger, 2000)). L'annotation du corpus présente plusieurs étapes clairement définies et qui peuvent être réitérées

en fonction des améliorations faites sur la grammaire : dans un premier temps, le Wall Street Journal est analysé avec le PET parser qui utilise la grammaire ERG. Les cinq cent meilleures analyses sont conservées afin d'être désambigüisées manuellement. Les annotateurs passent la majeure partie de leur temps à regarder les arbres dans leur ensemble et décider si l'analyse est correcte ou non. Cela requiert au maximum $log(n)$ décisions, ce qui rend l'annotation extrêmement rapide. Dans un second temps, soit il reste un arbre et il est correct pour la phrase, soit aucun arbre ne demeure et la phrase est rejetée, ce qui indique une erreur ou un manque dans la grammaire d'origine. De plus, si l'annotateur ne peut désambigüiser la phrase, alors il peut donner une analyse comme correcte avec un score de confiance assez faible. Enfin, le corpus conserve la tokenisation de l'ERG ainsi que ses catégories morpho-syntaxiques qui sont fines.

L'idée sous-jacente de l'annotation est d'améliorer le corpus, comme la grammaire. Quand la grammaire est améliorée, il faut relancer l'analyse syntaxique d'une part, puis ré-appliquer les décisions des annotateurs, et enfin refaire la désambiguïsation des phrases qui ont été rejetées ou qui ont un score de confiance bas. Dès qu'une partie suffisante du corpus est annotée, le modèle de désambiguïsation est réentraîné, ce qui permet d'améliorer la sélection des analyses et donc de réduire le travail des annotateurs. Pour donner un ordre d'idée, ERG possède un taux de couverture de 80% sur les textes issus du Wall Street Journal.

Le DeepBank met à disposition l'analyse syntaxique, ainsi que l'analyse MRS de la phrase. Le corpus offre une analyse fine au niveau des catégories morpho-syntaxiques et des types de dépendances, mais aussi un graphe sémantique plus dense, car il n'est pas limité au traitement des seuls noms et verbes. Parmi les dépendances fines, nous citerons les distinctions faites au niveau des modifieurs nominaux (correspondant au NMOD pour le Penn Treebank en dépendances). Le DeepBank en offre trois :

— Adjoint avant la tête : The **big old cat** slept.

— Adjoint après la tête : The **cat we chased** ran.

— Adjoint après la tête qui est une relative réduite : The **cat in a tree** fell.

Bien entendu, le fait de proposer des analyses MRS confère au DeepBank une dimension sémantique loin de nos travaux. En effet, la MRS est connue pour gérer les ambiguïtés de portée sur les quantificateurs, chose que nous avons explicitement laissée de côté pour le moment. Nous nous attachons à décrire le DeepBank, car Oepen et Lønning (2006); Ivanova et al. (2012) l'ont transformé en un corpus de digraphes acycliques étiquetés présentant des caractéristiques syntaxico-sémantiques et appelé DM corpus. Le DM corpus résulte d'une double passe simplificatrice non réversible. La première phase transforme les représentations MRS en des structures de dépendances élémentaires qui ne gardent que les rôles sémantiques (ARG_0, ..., ARG_n, etc.) ou les propriétés MRS telles que TEMPS, NOMBRE, ASPECT, etc. La première passe simplificatrice transforme les formules MRS originales en structures de dépendances élémentaires avec des variables libres (*variable-free elementary dependency structures*) qui (i) fait correspondre chaque prédicat de la forme logique à un nœud dans le graphe de dépendances (ii) transforme les relations entre les arguments et les

prédicats en des arcs de dépendances. La seconde passe, effectuée par Ivanova et al. (2012) réduit les structures élémentaires en des formes strictement bi-lexicales liant prédicat et arguments entre eux. Bien que les étapes de conversion ne soient pas réversibles, les graphes de dépendances qui composent le DM corpus sont un sous-ensemble réel des informations contenues dans les formules logiques MRS. En effet, DM est un corpus à la frontière entre la syntaxe et la sémantique, il possède une annotation des arguments principaux de la phrase à la PropBank (Palmer et al., 2005) et NomBank (Meyers et al., 2004), mais est beaucoup plus dense, car son processus d'annotation conserve des informations syntaxiques telles que les coordinations. De plus, le corpus considère certains tokens comme étant sémantiquement vides, c'est notamment le cas des prépositions ou des ponctuations. Il est en ce sens à mi-chemin entre une représentation profonde et une représentation *à la* PropBank avec ses arguments numérotés. Il est donc un bon candidat pour comparer nos méthodes et nos travaux. Un exemple d'annotation est donné à la Figure 3.1.

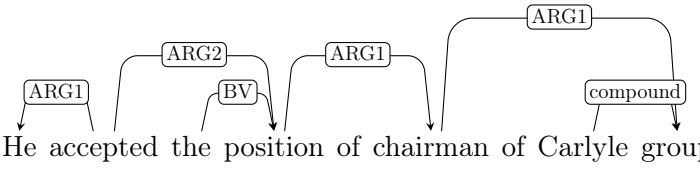


Figure 3.1: Exemple d'annotation du DM corpus pour une phrase du Penn Treebank.

3.2.2 PAS corpus & analyseur Enju

Contrairement au DeepBank qui a fait l'objet d'une validation manuelle, le principe d'Enju est l'induction d'une grammaire HPSG via un corpus existant, en l'occurrence le Penn Treebank (Marcus et al., 1993). Une fois la grammaire induite, un corpus peut ensuite être dérivé, corpus qui s'appuie sur la théorie HPSG (Pollard et Sag, 1994) pour donner des structures proches de la structure argumentale. Au départ le travail s'est uniquement concentré sur les verbes pour être étendu par la suite (Miyao et al., 2004). L'algorithme utilisé pour extraire cette grammaire est relativement standard comparativement aux diverses méthodes de même types (notamment celles de Cahill (2004); Hockenmaier et Steedman (2007); Shen et al. (2007)). L'argument principal d'un tel algorithme est que l'extraction automatique à partir d'un corpus permet de créer un début de grammaire à faible coût qui pourra être améliorée par la suite. La méthode peut être divisée en deux passes :

1. **Externalisation** : Cette passe transforme les annotations du Penn Treebank en analyses HPSG en ajoutant des heuristiques d'annotations ce qui produit des dérivations partielles d'analyses HPSG. D'abord, chaque nœud de l'arbre est marqué comme *tête*, *argument* ou *modifieur* en utilisant des tables de tête-arguments et de tête-modifieurs. Ensuite l'arbre est transformé en arbre binaire, puis d'autres heuristiques sont utilisées afin d'analyser des constructions difficiles, telles que les coordinations ou les relatives. Enfin, des catégories HPSG sont assignées aux non-terminaux.

2. **Extraction** : Dans cette passe, les entrées lexicales de la future grammaire sont directement extraites de l'arbre résultant après externalisation.

La grammaire ainsi extraite a permis de créer un analyseur syntaxique appelé Enju, capable de produire des structures prédicat-argument. Il est alors possible de ré-analyser le Penn Treebank afin d'obtenir un corpus en structure prédicat-argument appelé pas corpus et qui possède une annotation des arguments des prédicats via une numérotation *à la* PropBank (Arg_1, ..., Arg_n). Ce corpus n'a pas été validé à la main. Il présente une annotation dense qui le rapproche d'une annotation syntaxique profonde. Enju et le pas corpus considèrent que toutes les catégories morpho-syntaxiques peuvent avoir des arguments et ne marquent pas ou peu de mots sémantiquement vides. Un exemple de sortie est donnée à la Figure 3.2.

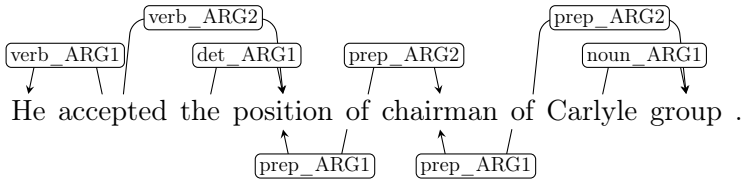


Figure 3.2: Exemple de sortie via l'analyseur Enju composant le corpus pas.

3.3 Synthèse des différentes ressources

Nous nous proposons de faire une synthèse des différentes ressources que nous avons présentées afin de pouvoir les comparer sur le plan des caractéristiques de taille, d'annotation, etc. Pour ce faire, nous donnons à la Table 3.1 les caractéristiques de chacune des approches. Nous laissons PropBank de côté en considérant, comme nous l'avons dit en le décrivant, qu'il est quelque peu éloigné des considérations de cette thèse.

	Validation	# Phrases	# Étiquettes	Alt. synt.	Sujets inf.	LDD	Coord. elliptiques
Prague	+	49 000	40	+	+	+	−
AnCora	+	3 500	11	+	+	−	−
DM	−	35 000	52	−	+	−	−
PAS	−	35 000	43	−	+	−	−
CCGBank	−	49 000	NA	−	+	+	+

Table 3.1: Comparaison des caractéristiques de chaque ressource.
LDD signifie dépendances longue distance et Alt. synt. signifie alternance syntaxique.

La Table 3.1 montre différentes choses. Sinon le corpus de l'espagnol (AnCora-UPF), tous les corpus présentent une taille similaire de quelques dizaines de milliers de phrases. Le nombre d'étiquettes utilisées durant l'annotation est très variable mais reste stable (sauf pour l'espagnol qui utilise les quelques étiquettes profondes définies par la mtt). Il faut tout de même noter que le nombre d'étiquettes n'est pas une indication de la difficulté de l'analyse. En effet, dans certains corpus comme dm ou pas, un nombre réduit d'étiquettes (une dizaine) couvrent 90% des annotations du corpus, ce qui n'est pas le cas pour le Prague

Dependency Treebank. Par ailleurs, sans surprise, le nombre de corpus automatiquement annotés (colonne *Validation*) est supérieur à ceux validés manuellement. Enfin, au niveau des structures profondes (et par transitivité des structures prédicat-argument) l'accent est surtout mis sur l'alternance syntaxique et le sujets des infinitives. Les dépendances longue distance (et *a fortiori* les coordinations elliptiques) sont en général laissées de côté, car bien trop rares. En effet, il faut rappeler que les corpus ont été créés pour servir deux buts : l'analyse linguistique et l'analyse automatique. Les phénomènes dits « complexes » (longue distance), même s'ils apportent des informations intéressantes (Rimell et al., 2009), ne sont pas discriminants sur un F_1-score global donné après analyse automatique.

Dans ce chapitre, nous n'avons discuté que des ressources dans des langues autres que le français, il est désormais temps de donner un panorama des ressources existantes en syntaxe profonde pour le français.

Chapitre

4 Syntaxe profonde pour le français

Sommaire

Introduction

Le panorama des ressources en syntaxe profonde, manuellement validées ou automatiquement acquises, est important lorsqu'il s'agit de langues autres que le français. Dans ce chapitre, nous nous intéressons aux données existantes du français. Nous verrons que très peu de ressources manuellement annotées existent et qu'en général l'annotation est soit partielle, soit trop petite pour être utilisée par des analyseurs automatiques sans avoir recours à des techniques complexes d'apprentissage automatique. Nous partirons d'une ressource de type PropBank créée par transfert linguistique (Van der Plas et al., 2010) pour discuter l'acquisition automatique de ressources en syntaxe profonde via des méthodes d'acquisition automatique fondées sur des surcouches LFG (Schluter et van Genabith, 2008), des grammaires de logique typée (Moot, 2015) ou d'un système de réécriture de graphes (Guillaume et Perrier, 2012).

4.1 Acquisition d'un PropBank du français

Van der Plas et al. (2010) ont créé une corpus PropBank de 1 000 phrases manuellement annotées pour le français. L'annotation n'est pas faite *ex nihilo*, puisque les auteurs utilisent les idées de transfert multi-lingue présentes dans les travaux de Padó (2007) pour FrameNet (Baker et al., 1998). L'objectif du travail est de voir s'il est possible de transférer les annotations PropBank de l'anglais au français et de voir si la tâche est bien définie linguistiquement et suffisamment générale pour être portée à d'autres langues.

Van der Plas et al. (2010) utilisent le corpus Europarl, un ensemble de textes parlementaires européens alignés pour chaque langue parlée dans l'Union Européenne. Ils travaillent exclusivement sur des traductions directes entre l'anglais et le français et sur des phrases allant de un à quarante tokens. Les annotateurs, au nombre de quatre, doivent trouver pour chaque verbe français, sa correspondance en anglais afin de pouvoir annoter les arguments de ce verbe en réutilisant la nomenclature ProBank. S'il est impossible de trouver une correspondance du verbe en anglais, ils doivent utiliser une étiquette factice (*dummy label*). Un total de 1040 phrases sont annotées dont 130 par les quatre annotateurs et le reste par une seule et même personne (ce qui est tout de même susceptible d'introduire un biais d'annotation).

Le travail cherchant à montrer la validité de l'annotation PropBank d'une langue à une autre, le calcul de l'accord inter-annotateur est important. Après discussions et consensus sur certains problèmes, l'accord inter annotateur (mesuré en terme de F-score bien meilleur que le fameux κ pour cette tâche) va de 91% à 95%. L'utilisation de l'étiquette factice (qui touche 130 prédicats) est utilisée notoirement pour les cas d'expressions polylexicales (*multi-word expression*) qui ont un équivalent unique en anglais et les cas de collocations et d'expressions idiomatiques.

L'approche utilisée est intéressante et donne d'excellents résultats, mais elle présente quelques désavantages : le choix du corpus en est une puisqu'il faut un corpus aligné entre les langues avec des phrases relativement courtes (jusqu'à quarante tokens) ; enfin l'analyse se concentre (comme PropBank) uniquement sur la structure prédicat-argument des verbes donnant une bonne première approximation, mais dans le cadre qui nous intéresse ici, nous souhaiterions aller plus loin pour annoter au moins la structure prédicat-argument des verbes et des adjectifs. Néanmoins, un peu plus de 1 000 phrases sont mises à disposition formant un corpus de référence manuellement validé – *a priori* le seul jusqu'ici sur ce type d'annotation.

4.2 Annotation automatique en F-structures

Schluter et van Genabith (2008) propose d'adapter l'algorithme d'annotation automatique en F-structures LFG de Cahill (2004). L'idée première est d'induire automatiquement une grammaire profonde LFG, mais l'annotation automatique permet d'obtenir un corpus en constituants avec des structures fonctionnelles qui peuvent servir à extraire des cadres de sous-catégorisation, travail qui s'inscrit parfaitement dans le cadre d'une annotation

automatique profonde via la théorie LFG. Nous présenterons d'abord le corpus qui a servi de point de départ au travail : le Modified French Treebank (Schluter et van Genabith, 2007) (ou MFT) avant de parler succinctement de l'architecture d'annotations (voir (Cahill, 2004) et (Schluter et van Genabith, 2008) pour plus de détails).

Le Modified French Treebank (Schluter et van Genabith, 2007) est un corpus de 4 740 phrases environ issues du French Treebank (Abeillé et al., 2003). C'est une réannotation partielle des arbres en constituants déjà annotés dans le French Treebank. L'idée était d'annoter avec plus de cohérence certains phénomènes et d'apporter des modifications au schéma d'annotation pour permettre un meilleur apprentissage via des analyseurs statistiques. Les modifications structurelles sont les suivantes :

— L'annotation de relatives qui propage des informations vers le pronom relatif, aboutissant à une catégorie PPrel ou NPrel plus facilement discriminée par un analyseur stochastique.

— L'annotation des VN et des V selon trois sous-catégories : *finite, part* ou *inf* (donnant VNinf, VNfinite, VNpart, etc.).

— La réannotation de la coordination que les auteurs jugent incohérente dans le French Treebank. Ils lui préfèrent une annotation unifiée où la coordination est un syntagme unique avec les conjoints comme filles.

— L'utilisation de chemins fonctionnels pour les dépendances longue distance. C'est un des pré-requis à la résolution des phénomènes longue distance lors de l'annotation automatique en f-structures. C'est sans doute la modification la plus importante faite sur le corpus et qui permet une résolution automatique de très bonne qualité.

L'algorithme d'annotation automatique est relativement proche des approches du même genre (Hockenmaier et Steedman, 2007; Shen et al., 2007). C'est un processus en quatre étapes : (a) annotation automatique en f-structures des arbres en constituants (b) extraction des équations fonctionnelles obtenues à l'étape précédente et utilisation d'un solveur de contraintes pour vérifier qu'elles produisent une f-structure bien formée (c) extraction des cadres de sous-catégorisation via les f-structures (d) extraction des dépendances longue distance. Normalement, la première étape est souvent réalisée au moyen d'heuristiques plus ou moins complexes, néanmoins dans le cas qui nous intéresse, Schluter et van Genabith (2008) utilisent les informations déjà présentes dans le MFT afin d'aider l'algorithme. Ce faisant, ils évitent d'avoir recours à des heuristiques et garantissent une annotation de qualité (environ 96% de F-score). Enfin, l'algorithme permet d'extraire automatiquement les dépendances longue distance et de les annoter automatiquement au niveau des f-structures, c'est le cas pour les clivées, certaines coordinations elliptiques comme les *gapping*, les *argument cluster coordinations* ou les *right node raisings*.

Le travail présenté ici fait partie d'une longue tradition déjà observée sur l'anglais d'annotation automatique (ou semi-automatique) de corpus existants afin d'obtenir des strates plus profondes et d'extraire des structures prédicat-argument. Nous verrons, au cours des sections suivantes, comment ce genre d'approches a été réutilisées sous d'autres formes, notamment par l'utilisation de grammaires catégorielles ou d'un système de réécriture.

4.3 Grammaires catégorielles & TLGBank

Sandillon-Rezer (2013) et Moot (2015) proposent deux méthodes relativement proches pour créer un corpus arboré suivant les préceptes des grammaires catégorielles (grammaires A/B, (Ajdukiewicz, 1935; Bar-Hillel, 1953)). La première s'appuie sur le corpus Séquoia libre de droit (Candito et Seddah, 2012b), un corpus arboré du français annotant quatre domaines différents : l'*Est Républicain*, journal lorrain relatant de nombreux faits divers, Europarl, des débats parlementaires, EMEA, un corpus médical et une fraction de la Wikipédia française. Le second, appelé TLGBank et dérivé du French Treebank (Abeillé et al., 2003), utilise un formalisme proche des grammaires catégorielles appelées grammaires de logique typée (*type-logical grammars*) qui sont plus expressives que les grammaires A/B.

Les grammaires de logique typée sont plus expressives que les grammaires A/B de la même manière que les Grammaire catégorielle combinatoire (Steedman, 2000). Les règles d'inférence sont données en détails dans (Moot, 2015, annexe). Les algorithmes d'annotation des corpus sont similaires à ce qui a été fait pour l'anglais notamment par Hockenmaier et Steedman (2007), à savoir :

1. découpage des mots composés (propres au French Treebank)
2. Binarisation des arbres en conservant la distinction entre ajouts et arguments. Les arguments portent des catégories fondées sur leur étiquette syntaxique (par exemple NP pour les syntagmes nominaux, ...) en utilisant une table de percolation des têtes (Magerman, 1995).
3. Traitement spécialisé pour les clusters de verbes (par exemple Elle *a pu être considérée* comme malade). Un exemple est donné à la Figure 4.1.
4. Transformation de l'annotation des coordinations.
5. Insertion de traces à aux endroits nécessaires et traitement des pronoms relatifs et clitiques.

Suite au travail de Sandillon-Rezer (2013), 3 200 phrases du corpus Séquoia ont été annotées avec le formalisme des grammaires A/B. Par ailleurs, Moot (2015) obtient 90,6% de supertags corrects via les outils de Curran et al. (2007) entraînés sur le French Treebank (validation croisée de dix échantillons) contre 88,1% de supertags corrects sur l'*Est Républicain*. Enfin, sinon quelques corrections effectuées sur les coordinations et les clusters verbaux (Moot, 2015), le TLGBank n'est pas un corpus manuellement validé. Enfin, nous dressons une comparaison du travail de (Moot, 2015) par rapport au CCGBank. La différence majeure, outre la différence de formalisme, réside dans le fait que le CCGBank possède un nombre de règles logiques plus important que les grammaires de logique typée, de fait le lexique du CCGBank est moins grand que celui du TLGBank.

4.4 Annotation profonde automatique du French Treebank

Guillaume et Perrier (2012) propose l'utilisation d'un système de réécriture de graphes, appelé GREW (Guillaume et al., 2012), pour l'annotation semi-automatique du French Tree-

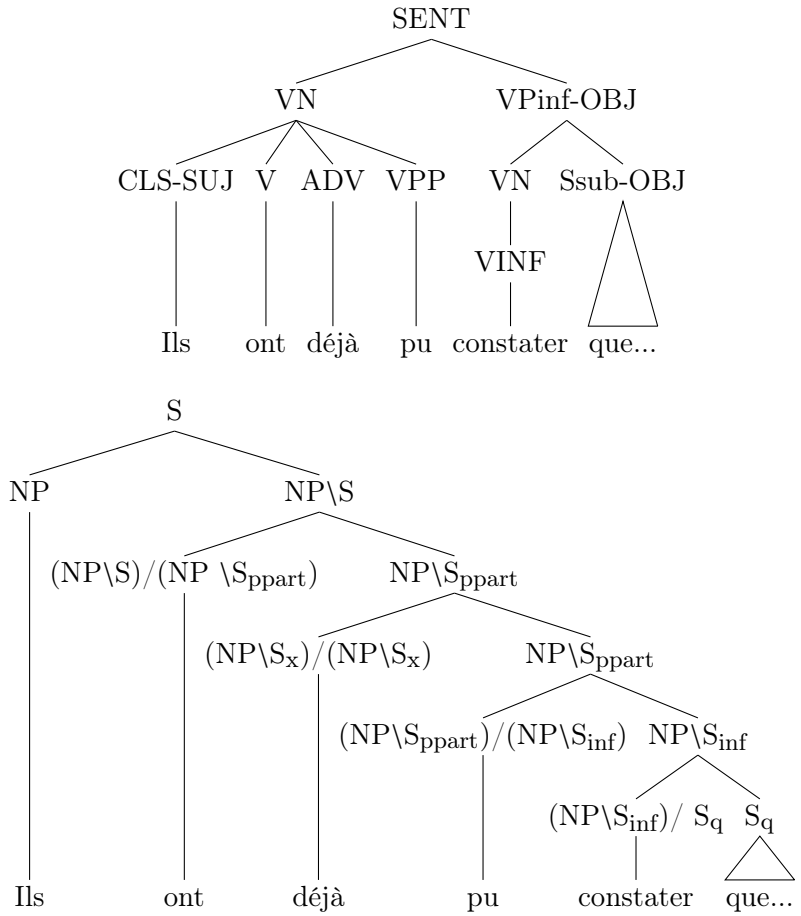


Figure 4.1: Traitement des clusters verbaux lors de la transformation en grammaire de logique typée.

bank en dépendances via des structures de type MRS en dépendances. L'idée est similaire au DeepBank (Flickinger et al., 2012) : ajouter des informations de sémantique de surface sur un corpus présentant une annotation surfacique. Pour ce faire, les auteurs utilisent un système de réécriture composé de 562 règles réparties en 34 modules qui sont ensuite regroupés en six paquetages :

— INIT qui transforme le format de dépendances en un format compatible avec le système de réécriture.

— VERB qui normalise le noyau verbal : les auxiliaires sont traités comme sémantiquement vides, les verbes pronominaux sont marqués comme tel, ainsi que l'alternance syntaxique pour les voix moyenne, active et passive.

— SUJ traite les sujets des verbes (infinitifs) et des adjectifs.

— ARGV qui explicite les arguments des verbes et s'occupe de l'alternance syntaxique.

— PRO qui détermine les antécédents des pronoms relatifs et traite les pronoms réflexifs.

— sem qui calcule les relations sémantiques entre les prédicats.

Un exemple de sortie des différents paquetages est donné à la Figure 4.2.

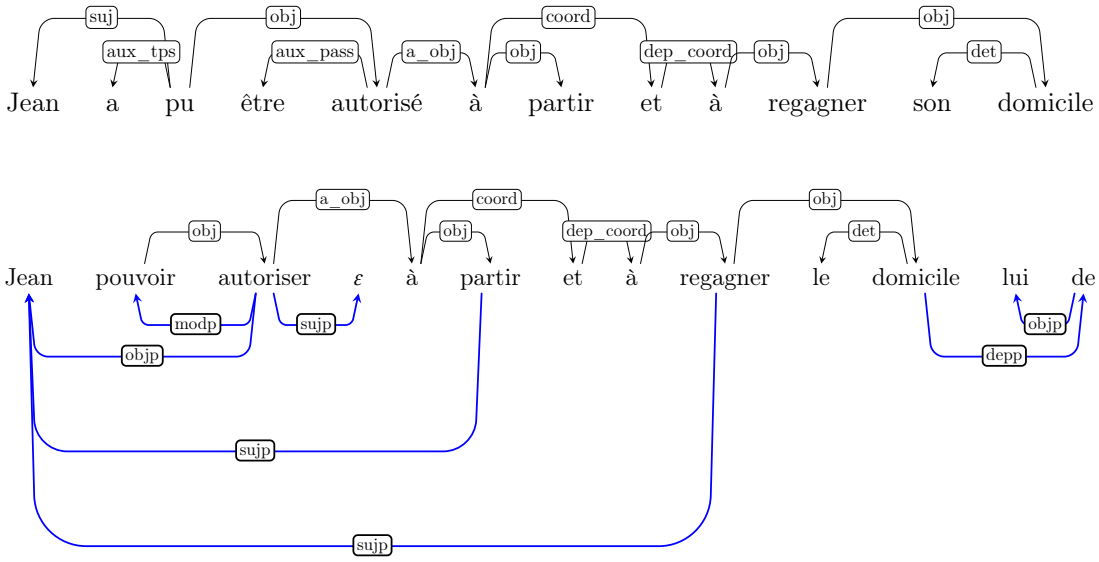


Figure 4.2: Exemple de traitement via la réécriture pour une phrase du French Treebank : en haut la structure d'origine, en bas en bleu, les dépendances ajoutées au niveau profond.

Le travail est fait en deux temps : dans un premier temps une annotation profonde est ajoutée au French Treebank (Abeillé et al., 2003) avant de l'utiliser pour la transformer en structure DMRS (Copestake et al., 2005) avec une notation numérotée des arguments. L'approche qui nous intéresse tout particulièrement est la première étape qui prend en compte les changements de diathèse, ce qui est crucial pour expliciter les cadres de sous-catégorisation canoniques des verbes, des noms et des adjectifs. Hormis le fait que les annotations sont automatiques et n'ont pas été validées à la main, le plus gros problème du travail réside dans la gestion complexe du système de réécriture : le nombre de règles rend difficile le maintenance d'un tel système. Par ailleurs, certains choix faits par les auteurs sont parfois inconsistants, c'est notamment le cas en ce qui concerne les ponctuations qui sont conservées, alors qu'elles peuvent être considérées comme sémantiquement vides (c'est-à-dire inutile pour la structure prédicat-argument). Néanmoins, le travail est un pas en avant vers une sémantique de surface pour le français là où peu de travaux existent.

4.5 Synthèse des ressources sur le français

Dans cette section, nous nous proposons comme pour les ressources présentées au chapitre précédent, de faire une synthèse comparative. Comme au chapitre précédent, nous laissons de côté l'annotation PropBank de Van der Plas et al. (2010), puisque nous nous intéressons plus à l'annotation en syntaxe profonde sur des structures prédicat-argument qui vont plus loin que la structure argumentale verbale. Nous reprenons à la Table 4.1 les ressources existant dans d'autres langues auxquelles nous ajoutons celles sur le français.

	Validation	# Phrases	# Étiquettes	Alt. synt.	Sujets inf.	LDD	Coord. elliptiques
Prague	+	49 000	40	+	+	+	−
AnCora	+	3 500	11	+	+	−	−
DM	−	35 000	50	−	+	−	−
PAS	−	35 000	40	−	+	−	−
CCGBank	−	49 000	NA	−	+	+	+
MFT-LFG	|	4 700	NA	−	+	+	+
TLGBank	|	12 300	NA	−	+	+	+
(Guillaume et Perrier, 2012)	−	12 300	±30	+	+	−	−

Table 4.1: Comparaison des caractéristiques de chaque ressource. En haut les ressources pour des langues différentes du français et en bas les ressources pour le français. L'utilisation du *pipe* (|) note une validation manuelle partielle.

À la suite de l'étude des ressources existantes sur le français, nous nous rendons compte qu'il manque une ressource à large échelle, idéalement couvrant plusieurs domaines et manuellement validée permettant d'extraire la structure prédicat-argument d'une phrase. La ressource devra prendre en compte les problèmes d'alternance syntaxique aussi bien que l'annotation des phénomènes manquants dans les corpus déjà existants (sujets des infinitives, des participiales, etc.), les phénomènes longue distance et les coordinations elliptiques. Dans la suite de ce travail, nous allons présenter le schéma d'annotation profonde que nous avons mis en place. Ce travail fait suite à l'annotation automatique du French Treebank par Guillaume et Perrier (2012) qui a été profondément complétée pour donner naissance au DeepSequoia, le premier corpus à présenter une annotation manuelle à large échelle multi-domaines en syntaxe profonde pour le français.

Deuxième partie

Interface syntaxe-sémantique : des données aux modèles

Chapitre

5 Représentation surfacique en dépendances : point de départ de l'annotation profonde

Introduction

Sommaire

Le corpus Séquoia est un corpus arboré hors domaine librement disponible (Candito et Seddah, 2012b). Il a été annoté en suivant le schéma d'annotation du French Treebank en constituants (Abeillé et al., 2003). Contrairement au French Treebank, il présente des sous-corpus qui s'éloignent du style journalistique [1]. Le corpus a été pensé pour sa diversité syntaxique tout en considérant la possibilité de faire des expériences d'analyse syntaxique automatique de type adaptation de domaine avec des approches semi-supervisées.

Suite à l'annotation en constituants, le corpus a fait l'objet d'une conversion automatique en dépendances en suivant la méthodologie décrite dans (Candito et al., 2010). Cette conversion produit nécessairement des arbres projectifs. De fait, ces arbres ne sont pas satisfaisants dans le cadre des phénomènes d'extractions et plus généralement de dépendances longue distance et ont donc fait l'objet d'une ré-annotation manuelle décrite dans (Candito et Seddah, 2012a). Le schéma d'annotation en dépendances **de surface** résultant est le point de départ pour la définition du schéma d'annotation en dépendances profondes

1. Nous rappelons que le French Treebank est un corpus qui annote des phrases issues du journal *Le Monde*.

(Candito et al., 2014b; Perrier et al., 2014) et il est donc important de commencer par détailler les étapes de mise au point du corpus en dépendances de surface.

5.1 Constitution du corpus

Le corpus Séquoia (Candito et Seddah, 2012b) comporte 3 099 phrases de quatre origines différentes :

1. Europarl (Koehn, 2005) (561 phrases).
2. Le journal lorrain l'*Est Républicain* (524 phrases).
3. Une fraction de la Wikipédia française (996 phrases).
4. Une fraction de phrases issues de l'agence européenne du médicament (EMEA) (Tiedemann, 2009) (1 018 phrases).

Les sous-corpus ont été choisis pour leur diversité variable par rapport au French Treebank (Abeillé et al., 2003).

— **Europarl** (Koehn, 2005) constitue un corpus très utilisé en Traitement Automatique des Langues et les débats parlementaires qu'il contient montrent *a priori* des caractéristiques syntaxiques qui peuvent différer du type journalistique par l'emploi fréquent de la première personne et du vocatif, relativement rares dans le cadre d'un article de presse.

— **L'*Est Républicain*** (Gaiffe et Nebhi, 2009) est un corpus rassemblant deux années du quotidien régional du même nom. La particularité du journal réside dans le fait que ses articles sont souvent des faits divers, des avis de décès ou des inaugurations en tout genre.

— **FrWiki** sont un regroupement de dix-neuf articles issus de la Wikipédia française. Ces articles concernent des « affaires » sociales ou politiques célèbres, récentes pour la plupart. Le style est assez narratif et se rapproche du style d'une encyclopédie.

— **EMEA** (Tiedemann, 2009) contient des documents concernant des médicaments, essentiellement des rapports publics d'évaluation, chaque rapport étant dédié à la justification de l'autorisation ou l'interdiction de la mise sur le marché d'un médicament. Le corpus est particulièrement éloigné du style journalistique, puisqu'il traite du domaine médical et que le genre textuel est proche de celui du rapport scientifique. Il contient une terminologie spécialisée (protocoles de test et administration de médicaments, descriptions de maladies, etc.). Ce corpus comporte aussi de nombreux verbes à l'impératif (pour les instructions d'utilisation), la description des dosages et un usage fréquent de précisions apparaissant entre parenthèses. (Candito et Seddah, 2012b) ont extraits deux rapports issus du corpus OPUS constitué par Tiedemann (2009).

5.2 Conversion en dépendances de surface

Le schéma d'annotation est une conversion automatique des arbres en constituants, eux seuls ayant été manuellement corrigés par des annotateurs experts (Candito et al., 2009). De fait, la représentation surfacique peut contenir des erreurs ou des approximations qui

ont été corrigées lors du passage vers la syntaxe profonde. La conversion est fondée sur la technique de percolation des têtes décrites par (Magerman, 1995) et c'est une procédure en plusieurs temps :

1. Annotation des têtes via une table de percolation des têtes en utilisant une version améliorée de Dybro-Johansen (2004) et Arun et Keller (2005).
2. Extraction des dépendances bi-lexicales. Les étiquettes fonctionnelles présentes dans la représentation syntagmatique sont utilisées comme étiquette de la dépendance si elles existent. Or, dans le schéma d'annotation du French Treebank, seuls les syntagmes dépendants d'un verbe à un mode autre que le participe passé reçoivent une étiquette grammaticale. Lors de la conversion en dépendances, des heuristiques permettent d'étiqueter les dépendances restantes. Ce sera le cas pour les dépendants de gouverneurs non-verbaux, les coordinations et les dépendants des participes passés. Par exemple, *vêtements achetés par les Français*, le syntagme prépositionnel *par les Français* ne possède pas d'étiquette fonctionnelle, de fait une heuristique donnera une étiquette *p.obj*, indiquant que c'est l'objet du participe.

Candito et al. (2009), qui ont créé le protocole de conversion du schéma d'annotation French Treebank vers des dépendances bi-lexicales, font état d'une évaluation des erreurs produites par une conversion automatique (ne produisant que des arbres projectifs). Pour ce faire, ils annotent 120 phrases du French Treebank converti en dépendances (FTB-120) afin de corriger les erreurs dues à la transformation. Les résultats font état d'un UAS [2] de 98,78% et d'un LAS [3] de 98%, montrant une très bonne conversion automatique et un faible taux d'arcs non projectifs dans le corpus.

Nous donnons à la Table 5.1, le jeu d'étiquettes de dépendances (fonctions grammaticales) utilisées lors de la conversion.

5.2.1 Choix lors de la conversion en dépendances

Certains choix sont par ailleurs spécifiques à la conversion en dépendances :

— Les auxiliaires de temps, passif ou causatif sont marqués comme dépendant du participe ou de l'infinitif qu'ils introduisent. Dans la phrase *Jean a mangé une pomme*, l'auxiliaire de temps *a* est dépendant de *mangé* (en bleu sur la Figure 5.1).

— Les prépositions et les conjonctions de subordination sont traitées comme la tête des compléments qu'ils introduisent et, ce quelle que soit la catégorie de celui-ci. Dans la phrase *Le chat de ma mère*, *de* est la tête de *ma mère* (en rouge sur la Figure 5.1). Il en va même de même pour une phrase comme *Je rêve d'aller à Nancy* où *aller* est objet de la préposition *d'* et *d'* est *de_obj* du verbe *rêve*.

2. *Unlabeled Attachment Score* : c'est une mesure d'évaluation très utilisée pour l'analyse en dépendances qui calcule le nombre de dépendances correctement attachées (tête et dépendant) sans prendre en compte l'étiquette de la dépendance.

3. *Labeled Attachment Score* : c'est une mesure d'évaluation similaire à l'UAS mais qui prend en compte l'étiquette de la dépendance.

Fonction	Utilisation
aff	Pour un clitique figé et le clitique se ni réfléchi ni réciproque
aff.demsuj	Fonction du clitique se par rapport au verbe pour exprimer le neutre ou le moyen
ato	Attribut de l'objet
ats	Attribut du sujet
aux.caus	Auxiliaire du causatif ou du complexe se faire + inf
aux.pass	Auxiliaire du passif
aux.tps	Auxiliaire de temps
a_obj	Complément indirect non locatif introduit par *à*
coord	Relation portée par un coordonnant, avec comme gouverneur le premier conjoint
dep	Relation sous-spécifiée, pour les dépendants prépositionnels de gouverneurs non verbaux et non adjectivaux (pas de gestion de la distinction argument / ajout pour ces cas)
dep.coord	Relation portée par un coordonné (sauf le premier), avec comme gouverneur le coordonnant immédiatement précédent
det	Déterminant
de_obj	Complément indirect non locatif introduit par de
dis	Fonction uniquement finale, pour les dépendants disloqués (apparaissant en tête ou fin, avec un clitique de reprise : Paul, sa moto, il l'a cassée)
mod	Modifieur de gouverneur verbal ou adjectivalmodifieur, autre qu'une relative, d'un autre type de gouverneur pour peu que l'on puisse repérer structurellement qu'il s'agit d'un modifieur (adjectif épithète par exemple)
mod.cleft	La subordonnée dans le cas d'une clivée
mod.rel	Dépendance d'une relative par rapport à son antécédent
obj	Objet de verbe
obj.cpl	Objet de complémenteur
obj.p	Objet de préposition
ponct	Relation portée par tout dépendant typographique, sauf pour les virgules jouant le rôle de coordonnant (qui porte la relation coord)
p_obj.agt	Fonction uniquement finale : complément d'agent en par ou de en cas de passif, ou en par dans le cas de causatif
p_obj.o	Complément indirect du verbe ou de l'adjectif autre
suj	Sujet d'un verbe ou d'un adjectif

Table 5.1: Fonctions grammaticales utilisées dans la représentation en dépendances surfaciques.

— Nous avons mentionné que le schéma du French Treebank ne fournit les fonctions grammaticales que pour les dépendants de verbe. Pour les dépendants de gouverneurs d'autres catégories, la conversion assigne soit une fonction générique DEP, soit une fonction plus précise, mais uniquement dans les cas où la fonction grammaticale peut être déduite sans considérer les mots sémantiquement pleins (c'est-à-dire en ne considérant que la syntaxe et les mots grammaticaux). C'est par exemple le cas des adjectifs et participes épithètes, qui portent la fonction MOD (en violet Figure 5.1). En revanche, par exemple, les compléments prépositionnels des noms sont notés DEP.

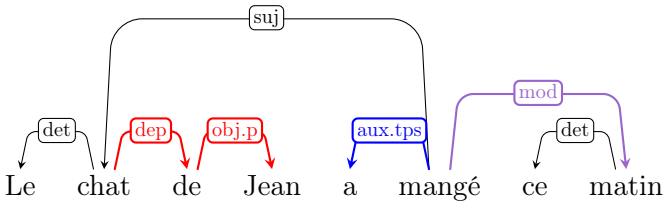


Figure 5.1: Exemple de phrase avec des annotations propres à la conversion en dépendances.

5.2.2 Annotation des dépendances longue distance

Une dernière étape a été l'annotation des gouverneurs en cas d'extraction longue distance (Candito et Seddah, 2012a). En effet, le schéma d'annotation syntagmatique du French Treebank ne comporte pas de traces. Aussi, il n'est pas possible de retrouver automatiquement et précisément quelle est la tête d'un syntagme apparaissant en position « non locale » (c'est-à-dire en position extraite). La conversion en dépendances va systématiquement rattacher la tête d'un syntagme extrait à un gouverneur local. Par exemple, dans l'arbre syntagmatique du NP *un sentiment que tous semblaient partager* (Figure 5.2), le pronom relatif porte la fonction grammaticale OBJ, mais il n'y a aucune indication qu'il s'agit de l'objet de *partager*. À la conversion en dépendances, c'est la tête locale *semblaient* qui sera prise comme le gouverneur de *que*. La correction manuelle fournit le gouverneur correct *partager*, créant ainsi un arc non projectif (en vert, Figure 5.2).

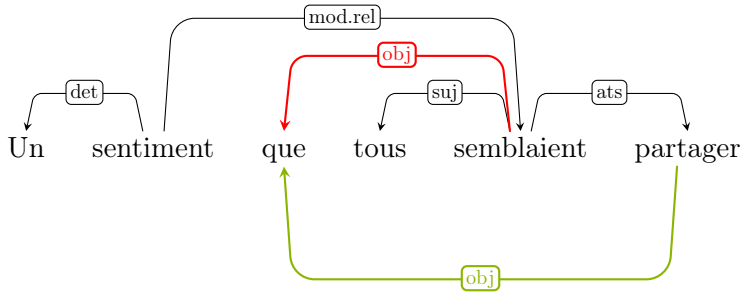


Figure 5.2: Conversion automatique qui amène une dépendance incorrecte entre *que* et *semblaient* (en rouge). Dépendance correcte (en vert) entre *que* et *partager*.

Les auteurs choisissent de faire une annotation manuelle en vérifiant un à un les mots *qu-* et le clitique *en* (des cas de dépendances non locales comme la topicalisation, par exemple, n'ont pas été corrigés manuellement). Ils obtiennent moins de 0.15% de dépendances corrigées. Le phénomène est cependant important pour le pronom relatif *que* (un tiers des occurrences) et le clitique *en* (un cinquième des occurrences). Candito et Seddah (2012a) reportent un nombre de 63 extractions de ce type dans le corpus Sequoia et 555 dans le French Treebank.

5.3 Schéma d'annotation résultant

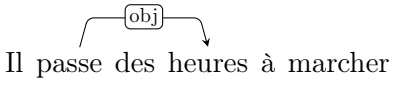
Nous donnons ici une description très succincte de la représentation obtenue pour divers phénomènes syntaxiques. Pour une description plus complète des arbres en dépendances de surface, le guide le plus complet est en fait le guide d'annotation des dépendances profondes (Candito et Perrier, 2015) détaillant les différences entre schéma de surface et schéma profond.

5.3.1 Phénomènes hérités de l'annotation du French Treebank en constituants

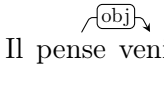
La fonction SUJ est utilisée pour marquer le sujet d'une phrase finie. Elle est utilisée pour les cas simples : *La **fille vient***, pour le verbe d'une relative et le pronom relatif : *La fille **qui viendra***. Dans le cas d'un *il* explétif dans une interrogative comme *Quand Paul viendra-t-il ?*, la fonction SUJ est utilisée deux fois, une fois pour marquer le sujet (ici **Paul**), une nouvelle fois pour le **il** explétif.

La fonction OBJ est utilisée pour noter les objets qui sont soit nominaux, soit pronominaux, soit clitiques ou encore phrastiques. De même, les interrogatives indirectes sont notées comme objets (5.1 d). Par ailleurs, certains adjectifs et compléments nominaux obligatoires sont objets. En général il s'agit de compléments indiquant une quantité (5.1 e). Dans le même esprit, les compléments introduits par *à* portent la fonction A.OBJ et ceux introduits par *de*, la fonction DE.OBJ. Enfin, la fonction P_OBJ est utilisée dans le French Treebank pour les autres compléments prépositionnels. Elle a été scindée en deux dans le corpus Séquoia : P_OBJ.AGT pour les compléments d'agent (Fig. 5.1f) et P_OBJ.O pour les autres cas, en particulier les locatifs obligatoires (Fig. 5.1i).

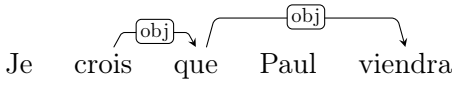
(5.1) (a)



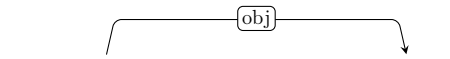
(b)



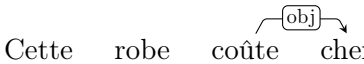
(c)



(d)

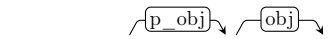


(e)



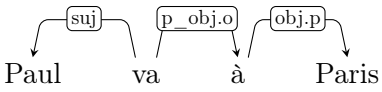
(f)

(g)



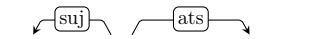
(h)

(i)

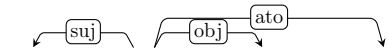


Les fonctions ATS et ATO expriment respectivement l'attribut du sujet et celui de l'objet. La conversion de surface conserve la copule comme tête.

(5.2) (a)

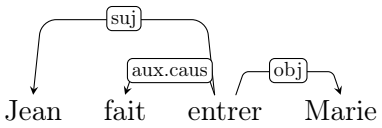


(b)



Concernant les changements de diathèse, seules les fonctions finales (cf. section 2.3.1.2, p. 30 pour plus de détails sur la distinction finales / canoniques) sont annotées. Dans le cas du passif, comme nous l'avons vu dans l'exemple 5.1 f, le complément d'agent est marqué POBJ.AGT et le sujet passif est noté SUJ. Dans le cas du causatif (5.3 a), tous les dépendants sont rattachés à l'infinitif et une relation AUX.CAUS est notée entre faire et l'infinitif.

(5.3) (a)

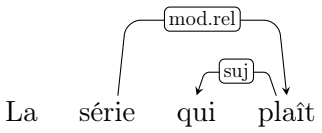


On considère maintenant les choix de conversion concernant :

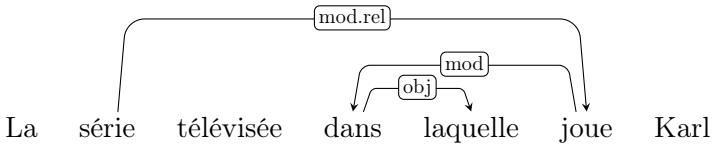
— Les relatives (5.4 a, b) pour lesquelles une relation MOD.REL est présente entre l'antécédent et le verbe de la relative, mais où la relation entre l'antécédent et le pronom relatif est inexistante.

— Les clivées (5.4 c) pour lesquelles le foyer de la clivée est attribut du sujet, la subordonnée est dépendante de la copule et où la relation syntaxique entre le verbe de la subordonnée et le foyer de la clivée n'est pas noté.

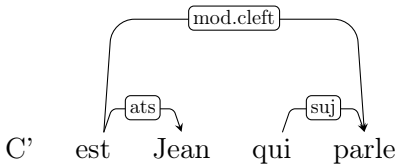
(5.4) (a)



(b)



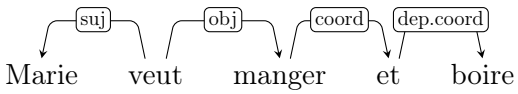
(c)



5.3.2 Coordinations

Enfin, l'annotation des coordinations est aussi un point central, surtout lorsqu'il s'agit de passer au niveau profond. Comme nous l'avons dit ci-dessus, la conversion pose le premier conjoint de la coordination comme tête, ce qui revient à favoriser une annotation proche des choix de la Théorie Sens-Texte [4]. Ce choix, bien que raisonnable, est problématique dans les cas où la distinction entre *un étudiant et un professeur intelligent**s*** et *un étudiant et un professeur intelligen**t*** est importante, puisque dans les deux cas, *intelligent* est dépendant de *professeur*. Par ailleurs, l'annotation surfacique ne donne pas accès aux informations d'ellipses, que ce soit pour les partages de sujet, les réductions de structures conjointes (*argument cluster coordination*), les mises en facteur droit (*right node raising*) [5], les phrases trouées (*gappings*) [6], etc. Des exemples sont donnés en 5.5.

(5.5) (a)

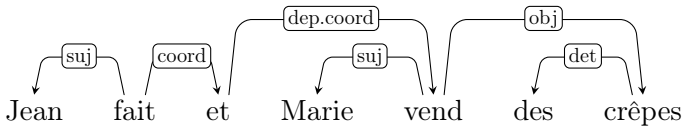


4. cf. (Popel et al., 2013) pour une comparaison de l'annotation des coordinations dans différents corpus arborés.

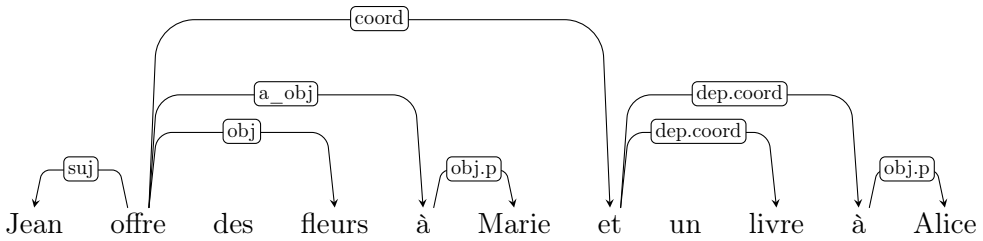
5. Pour une étude des right node raisings en français, voir Mouret et Abeillé (2011). Un débat existe sur le fait de considérer ce genre de construction comme un partage ou comme une ellipse, voir Barros et Vicente (2011) pour une discussion à ce propos.

6. Les traductions sont repris de Desmets (2008), mais nous leur préférerons la terminologie anglaise plus fréquente même en français.

(b)



(c)



Comme le montrent les exemples 5.5, les cas de partage ne sont pas explicités au niveau surfacique (puisque cela requiert des graphes). Il n'est pas possible de savoir dans l'exemple 5.5 a que *Marie* est le sujet de *manger* et de *boire*. De même, impossible de connaître la relation entre le verbe *fait* et *des crêpes* dans le cas de right node raising (Fig. 5.5 b). Ces informations seront explicitées au niveau profond, comme nous allons le voir au chapitre suivant.

Chapitre

6 De la surface à la syntaxe profonde : le DeepSequoia

Sommaire

Introduction

Dans ce chapitre nous décrivons la représentation profonde que nous avons considérée et autour de laquelle nous avons bâti systèmes et méthodes. L'annotation du DeepSequoia, le nom donné au corpus Séquoia annoté en syntaxe profonde, qui en résulte est le fait de deux équipes de recherche, une à Nancy, Sémagramme et une à Paris, Alpage. [1] La création du guide d'annotation et l'annotation en elle-même est le fait d'un projet commun auquel nous avons activement participé.

1. Les personnes impliquées sont les suivantes : Marie Candito, Karën Fort, Bruno Guillaume, Guy Perrier, Djamé Seddah et Éric Villemonte de La Clergerie.

Les choix théoriques de l'annotation que nous décrivons à la section 6.1 sont en partie fondés sur la Grammaire Relationnelle de Perlmutter et Postal (1984) que nous avons détaillée à la section 2.3.1 (p. 28). Par ailleurs, nous discuterons de la campagne d'annotation à la section 6.2 qui est, à notre connaissance, unique en son genre, car faisant l'objet d'une pré-annotation et d'une annotation en double aveugle. En effet, le corpus a été créé au moyen d'une pré-annotation complète à base de de règles que nous avons mises au point et qui a permis de limiter la tâche des annotateurs. Enfin, après avoir donné l'accord inter-annotateur sur le corpus, nous discuterons de quelques choix formels qui ont un impact sur les méthodes à mettre en œuvre pour analyser automatiquement ce genre de représentation, avant de comparer le schéma et son processus d'annotation aux ressources existantes (section 6.3).

6.1 Le schéma d'annotation en syntaxe profonde

6.1.1 Principes généraux

Comme nous le rappelions à la section 2.3.1.3, l'objectif principal de la représentation profonde est de généraliser sur la variation syntaxique sans faire de distinctions ni de généralisations purement sémantiques. Pour ce faire, les notions de cadre de sous-catégorisations finale et canonique sont utilisées ainsi que la distinction initialement proposée par (Perlmutter et Postal, 1984) de fonctions grammaticales finales et initiales (appelées canoniques dans le cas de la représentation profonde du DeepSequoia). Ces notions permettent de représenter les changements de diathèse comme des redistributions (Candito, 1999) des fonctions grammaticales sous-catégorisées par un lexème.

Nous avons déjà défini le cadre de sous-catégorisation final à la section 2.3.1.3 p.31, nous rappelons cependant que son utilité est d'expliciter les sujets finaux des infintinitifs, des participes épithètes, les sujets finaux des verbes coordonnés et plus généralement tout argument partagé par plusieurs prédicats (section 6.1.2). Le cadre de sous-catégorisation canonique, quant à lui, est utilisé pour neutraliser la variation syntaxique due aux changements de diathèse. Les changements de diathèse sont considérés comme des redistributions des fonctions canoniques associées aux arguments syntaxiques. Suivant la Grammaire Relationnelle (Perlmutter et Postal, 1984), le cadre de sous-catégorisation final est vu comme résultant de l'application de 0 à n redistributions sur un cadre de sous-catégorisation canonique. Étant donné une occurrence de verbe, le cadre de sous-catégorisation canonique peut donc par définition être obtenu par application inverse des redistributions appropriées. Un exemple simple est le cas d'un verbe au passif dont le cadre de sous-catégorisation final est (sujet, par-objet) et le cadre de sous-catégorisation canonique est (sujet, objet) (section 6.1.3).

L'annotation en syntaxe profonde ne considère que les redistributions qui comportent un marquage morpho-syntaxique (typiquement l'auxiliaire pour le passif, ou le clitique sémantiquement vide *se* pour les alternances moyennes et neutres). Les alternances syntaxiques sans marquage morpho-syntaxique ne sont pas capturées et donnent lieu à des cadres de sous-catégorisation canoniques différents.[2] C'est le cas par exemple avec une alternance causative/inchoative comme *X coule Y* / *Y coule* : la sous-catégorisation canonique est couler'(sujet = X, objet = Y) pour *X coule Y*, et la sous-catégorisation canonique est couler'(sujet = Y) pour *Y coule* : l'argument sémantique « l'entité qui coule » est donc tantôt sujet canonique, tantôt objet canonique.

En revanche, pour l'alternance moyenne (par exemple dans « On avale facilement ce médicament » / « Ce médicament s'avale facilement ») ou bien l'alternance neutre (qui *efface* l'actant agentif ou causal, comme dans « Cela dissout le médicament » / « Le médicament se dissout (de lui-même) »), qui sont toutes deux marquées par le clitique *se*, le lien entre les deux formulations est capturé par redistribution, et pour ces deux alternances, l'objet direct dans la version transitive (médicament) est, dans la version intransitive, le sujet final mais l'objet canonique. Les resdistributions considérées sont les suivantes : le passif,

2. Ces alternances, en l'absence de marquage formel, sont considérées comme relevant de l'analyse sémantique.

l'impersonnel, le moyen, le neutre et le causatif et elles peuvent interagir (cf. (Candito, 1999) pour une étude des interactions entre redistributions).

Par ailleurs, dans la représentation profonde visée, à la manière des corpus DM (Ivanova et al., 2012) ou PAS (Miyao et al., 2004) ou encore de l'AnCora-UPF (Mille et al., 2013), certains mots sont considérés comme sémantiquement vides et ne portent aucune dépendance (cf. section 6.1.5). La représentation relève alors en partie de la sémantique, puisque n'explicitant pas les informations de portée, les informations de coréférence non déterminables syntaxiquement et les informations de structure informationnelle. L'intérêt d'une telle représentation est de préparer l'analyse sémantique en donnant accès aux structures argumentales. De manière générale, le corpus annote la structure argumentale des verbes et des adjectifs. Les noms sont laissés de côté pour le moment et prévus pour un travail futur.

Dans la suite de cette section, nous décrivons les phénomènes au centre de la représentation profonde.

6.1.2 Explicitation des sujets finaux

L'une des caractéristiques principales du schéma d'annotation profond est l'explicitation du sujet final des verbes non conjugués (infinitifs et participes). En effet, c'est une particularité des infinitifs et des participes qu'un de leurs arguments ne soit pas localement exprimé. Cet argument est systématiquement le sujet final de l'infinitif ou du participe, mais peut correspondre à une fonction canonique autre que le sujet, pour peu que l'infinitif ou le participe apparaisse dans une diathèse non canonique. Parmi les sujets à annoter, il y a notamment :

— Le sujet des participes et des adjectifs épithètes (Figure 6.22).

— Le sujet des participes et des adjectifs dépendants du verbe qui en général possèdent la fonction ATS, ATO ou MOD. Au niveau profond, comme le montre la Figure 6.1, la relation avec le verbe est conservée, ce qui fait que l'on reste au niveau syntaxique.

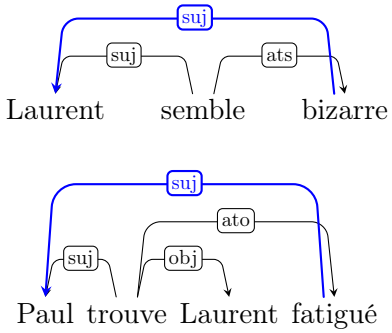


Figure 6.1: Marquage des sujet finaux pour les participes et adjectifs dépendants du verbe.

— Le sujet des participiales et adjectivales adnominales (Figure 6.2) où le participe n'a pas de sujet final réalisé localement. Un dépendant du verbe principal correspond alors au sujet du participe. Le sujet final du participe est explicité au niveau profond.

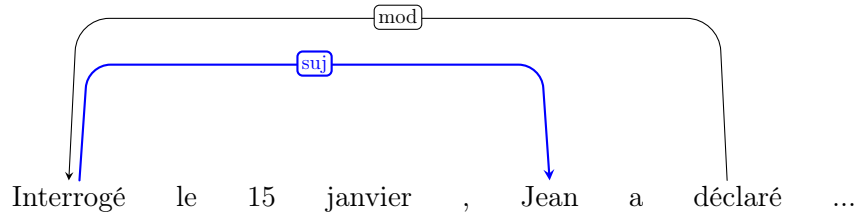
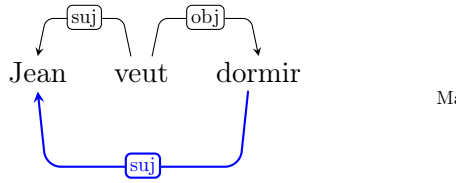
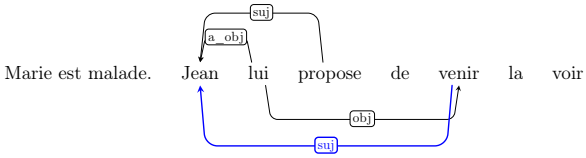


Figure 6.2: Marquage des sujet finaux pour les participiales et les adjectivales « détachées ».

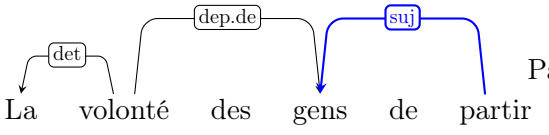
— Le sujet des infinitifs, comprenant notamment le contrôle obligatoire (Fig. 6.3a), le contrôle arbitraire (Baschung, 1996) (Fig. 6.3b), les infinitives gouvernées par un nom (Fig. 6.3c), par un adjectif (Fig. 6.3d), par une préposition non régie (Fig. 6.3e) et le *tough movement* (Fig. 6.3f).



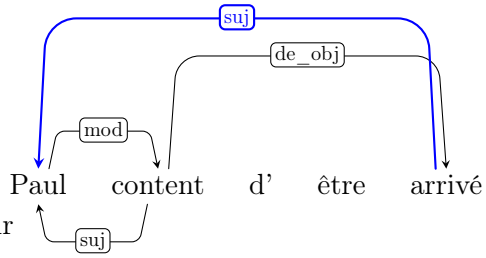
(a) Contrôle obligatoire.



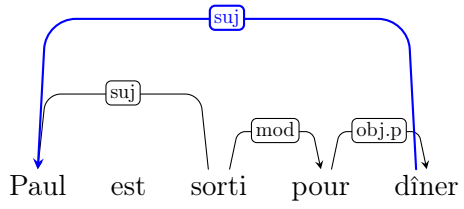
(b) Contrôle arbitraire.



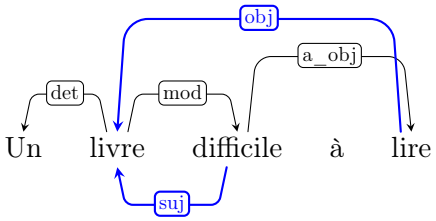
(c) Infinitif gouverné par un nom.



(d) Infinitif gouverné par un adjectif.



(e) Infinitif gouverné par une prép. non régie.



(f) Tough movement.

Figure 6.3: Marquage des sujets finaux pour les infinitifs.

6.1.3 Neutralisation de l'alternance syntaxique

Suite à l'explicitation des sujets finaux, le schéma traite principalement les cas d'alternance syntaxique, primordial pour récupérer une structure argumentale. L'alternance syntaxique peut être définie comme une relation entre deux constructions mettant en jeu le même prédicat (c'est-à-dire le même verbe) avec un appariement différent et éventuellement une morpho-syntaxe différente (clitique *se*, auxiliaire + participe, etc.) (Candito, 1999). Les diathèses traitées sont les suivantes :

— La diathèse passive, comme à la Figure 6.4, où *politicien* est l'objet canonique de *être respecté*, verbe au passif.

— La diathèse impersonnelle, comme à la Figure 6.5. Ici, il est important de distinguer les tournures impersonnelles (par exemple *il pleut*, *il s'agit d'un faux*, etc.), d'un changement de diathèse impersonnelle. Par ailleurs, la diathèse passive peut aussi interagir avec la diathèse impersonnelle pour donner une diathèse impersonnelle passive comme dans l'exemple *Il vous est demandé de régler le problème.*

— La diathèse causative, (Figure 6.17) où la fonction ARGC est utilisée pour marquer le causateur.

— Le *se* moyen. Il s'agit d'une construction productive, pour tout transitif à sujet canonique agentif. Syntaxiquement, le sujet final correspond à l'objet canonique, le sujet canonique n'est pas exprimable et le verbe porte un *se*, mais sémantiquement, le sujet canonique est interprétable : il existe un participant même s'il n'est pas exprimé et son interprétation est résolvable en contexte, ou, la plupart du temps, est générique comme en 6.6 où quelqu'un mange le carpaccio.

— Le *se* neutre. Comme dans la construction moyenne, un transitif apparaît avec son objet canonique en position sujet, et un clitique *se*. Mais dans le cas du neutre, l'agent (sujet canonique) n'est pas interprétable. Par exemple, à la Figure 6.7, il n'est pas facile d'interpréter que quelqu'un ou quelque chose a cassé la branche. Par ailleurs, le neutre est possible pour une classe sémantique de verbes beaucoup plus restreinte que pour le moyen (aspect inchoatif). La représentation profonde considère cependant que la différence entre *se* neutre et *se* moyen est d'ordre sémantique, et elle n'est pas désambigüisée.

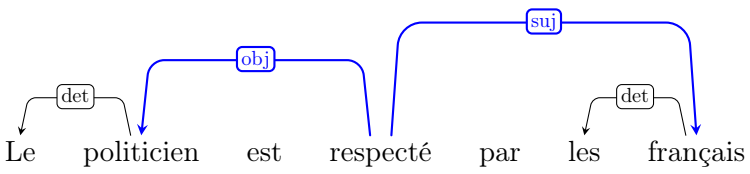


Figure 6.4: Exemple de diathèse passive.

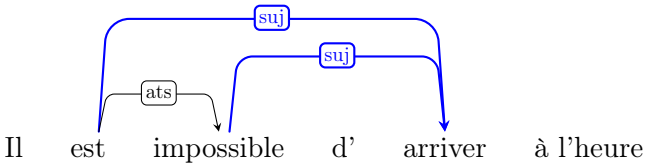


Figure 6.5: Exemple de diathèse impersonnelle.

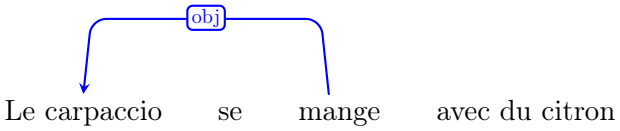


Figure 6.6: Exemple de *se* moyen.

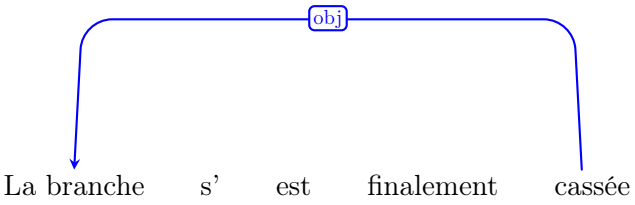


Figure 6.7: Exemple de *se* neutre.

6.1.4 Traitement de la coordination

La coordination est un autre phénomène qui a fait l'objet de traitements particuliers au niveau profond. Dans le cadre général, la coordination est distribuée dans certains cas, c'est-à-dire que les dépendances portées par le premier conjoint sont explicitées sur tous les conjoints suivants. Plus précisément : seules les dépendances sortantes du premier conjoint sont distribuées (c'est-à-dire que le conjoint est le gouverneur), et non les dépendances entrantes. En outre, la distribution est faite seulement dans le cas où cela est sémantiquement correct (cf. 6.8 pour un exemple de cas où la distribution serait sémantiquement fausse).

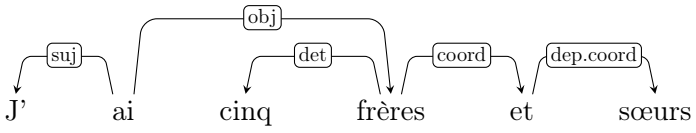


Figure 6.8: Exemple de coordination où la distribution serait fausse.

Deux grands principes sont alors utilisés pour décider des cas de « distribution » :

— **Principe 1** : Distribution des dépendances de coordonnés vers leur dépendant (module les cas où c'est sémantiquement faux). Exemples :
  — *Paul dîne et dort* : *Paul* est noté suj de *dîne* (déjà présent en surface) et de *dort* (à rajouter en représentation profonde).

— *Paul veut partir et revenir* : *partir et revenir* sont gouverneurs de *Paul* : *Paul* est noté suj de *partir* et de *revenir*

Ce choix permet de compléter la structure argumentale des prédicats apparaissant comme n-ième conjoint (non premier) d'une coordination : par exemple pour 6.9 cela permet d'expliciter la structure argumentale de *boit* qui en surface apparaît sans sujet.

— **Principe 2** : Les dépendances de coordonnés vers leur gouverneur ne sont pas distribuées. Dans l'exemple 6.10, *Paul, Kim et Marie* dépendent au niveau profond (suj) de *sourient*, il n'y a pas de distribution. La représentation est donc la même qu'au niveau surfacique. En effet, dans ce cas, le gouverneur a déjà tous ses dépendants (cf. le lien vers le premier conjoint). Distinguer une lecture distributive contre une lecture collective de la coordination est considéré comme relevant de la représentation purement sémantique : dans *Paul et Marie sourient*, il y a l'équivalence sémantique avec *Paul sourit* et *Marie sourit*, alors que dans *Paul et Marie partent pour Tokyo*, il n'est pas possible d'avoir une lecture collective où ils partent ensemble. De même, *Paul et Marie se disputent* ne peut pas être réduit à *Paul se dispute* et *Marie se dispute*.

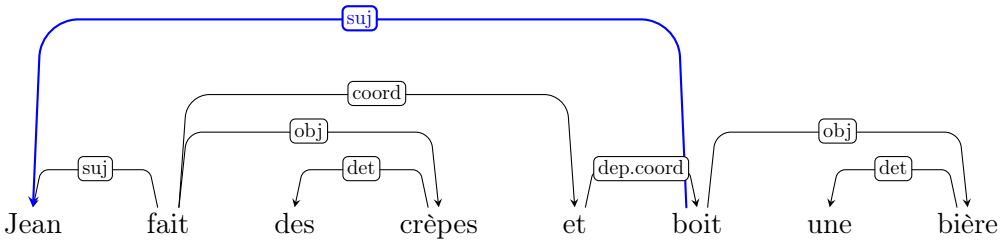


Figure 6.9: Distribution de la coordination pour expliciter la structure argumentale du verbe coordonné.

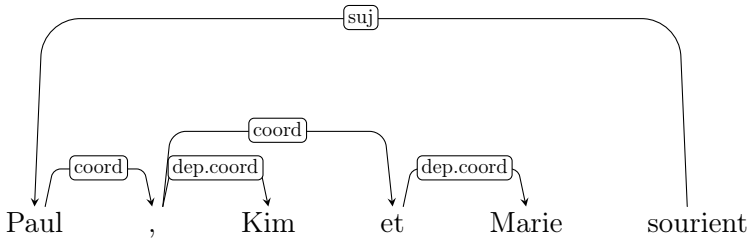


Figure 6.10: Non distribution de la coordination selon le principe 2.

6.1.4.1 Cas des coordinations elliptiques complexes

Les principes généraux utilisés lors de l'annotation profonde étant posés, nous pouvons regarder le traitement des coordinations elliptiques : right node raisings, argument clusters et gapping, notamment. Pour les cas de right node raising, la coordination est distribuée

si ce n'est pas sémantiquement faux (principe 1), un exemple est donné à la Figure 6.11, où l'objet *des crêpes* est distribué sur les deux verbes.

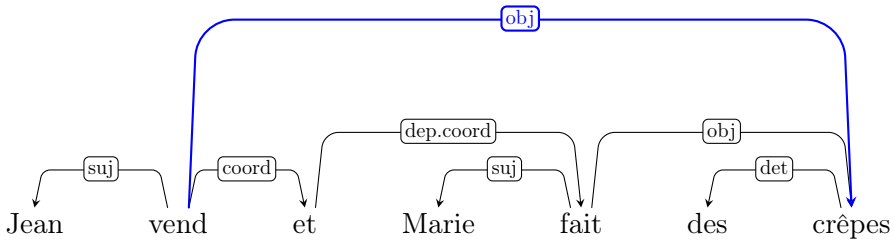


Figure 6.11: Exemple d'annotation profonde des right node raisings.

En ce qui concerne les cas d'argument clusters, nous rappelons qu'un des objectifs du schéma d'annotation est d'expliciter les structures argumentales de prédicats. Cet objectif nécessiterait d'expliciter plusieurs instances de prédicats pour une seule occurrence effective. Cependant dans cette version du schéma d'annotation, l'ajout de nœuds (à des fins d'explicitation de l'ellipse) n'est pas autorisée. Lee traitement fait en représentation de surface est utilisé au niveau profond : aucune répétition du prédicat, certes, mais les cas d'argument clusters sont tout de même repérable. En effet, le fait que le coordonnant introduisant le deuxième conjoint est rattaché sur le verbe est hérité du schéma d'annotation du French Treebank (ce qui diffère de ce qui est fait habituellement sur le premier conjoint) : donc en 6.12, le *et* est rattaché au verbe *offre*. Ensuite, les éléments introduits par le coordonnant (le groupe conjoint *une casquette* et *au garçon*) dépendent du coordonnant en utilisant les fonctions grammaticales pertinentes par rapport au verbe et non pas comme habituellement avec une relation DEP.COORD. De fait, *casquette* est OBJ de *et*, et *garçon* est A_OBJ de *et*. Le même principe est utilisé pour les gappings (Fig 6.13).

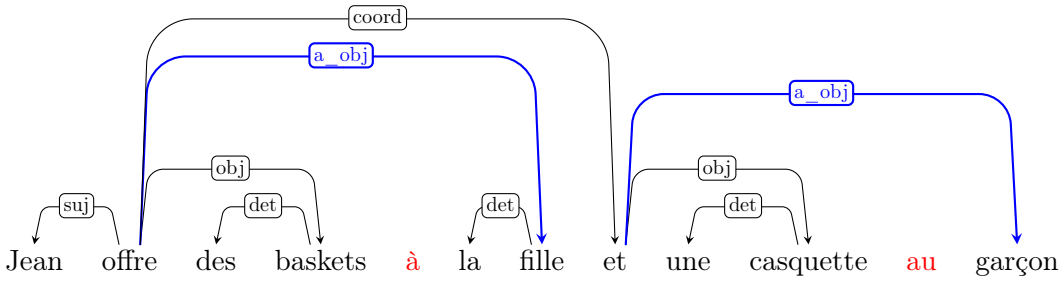


Figure 6.12: Exemple d'argument cluster coordination en profond (les mots vides sont notés en rouge).

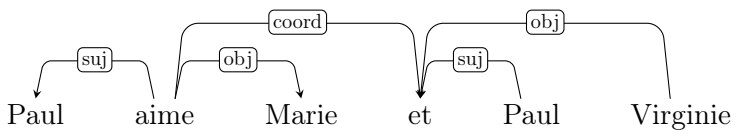


Figure 6.13: Exemple de head gapping en profond.

6.1.5 Marquage des tokens sémantiquement vides

Le dernier phénomène massif pris en compte dans le schéma concerne les tokens sémantiquement vides. La représentation profonde ne conserve pas tous les tokens au niveau du graphe. Néanmoins, ils ne sont pas supprimés, seuls arcs entrants et sortants le sont afin de déconnecter ces tokens du graphe, à la manière des corpus DM (Ivanova et al., 2012) et PAS (Miyao et al., 2004). Les arcs profonds attachés à ces tokens sont déplacés sur les tokens sémantiquement pleins de la manière suivante :

— La dépendance portée par une préposition régie pointe au niveau profond vers l'objet de la préposition (Fig. 6.3c).

— La dépendance portée par le complémenteur *que* introduisant une phrase pointe au niveau profond vers la racine de cette phrase, à savoir le verbe fini (Fig. 6.14).

— La dépendance portée par le complémenteur *à* ou *de* introduisant une infinitive pointe au niveau profond vers l'infinitif (Fig. 6.3f).

— La dépendance portée par *que* ou *qui* dans une clivée est déplacée au niveau profond vers le foyer de la clivée (Fig. 6.15).

— La dépendance du clitique réfléchi *se* est transférée au niveau profond vers l'élément qui est sujet canonique dans le cas général (Fig. 6.18). Le transfert sera différent dans le cas du construction causatif avec *se faire*. Dans ce cas, la dépendance vers le clitique réfléchi est transférée sur l'argument causateur (Fig. 6.16).

— La préposition introduisant un complément d'agent voit ses dépendances déplacées vers son dépendant (le complément d'agent) (Fig. 6.16).

— Le complémenteur *si* introduisant une interrogative indirecte comme dans l'exemple *Jean ne sait pas s'il viendra* donne, au niveau profond, une dépendance directe entre le verbe enchâssant et le verbe de l'interrogative (un marquage du caractère interrogatif de la phrase est fait sur le premier mot via un trait).

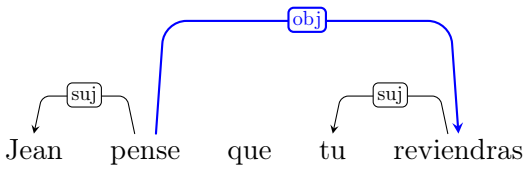


Figure 6.14: Suppression du complémenteur *que* et déplacement des dépendances.

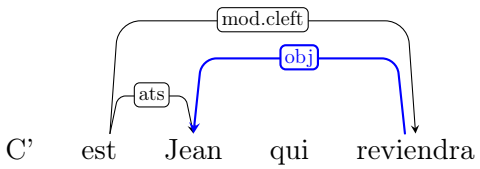


Figure 6.15: Suppression de *qui* dans une clivée et déplacement des dépendances.

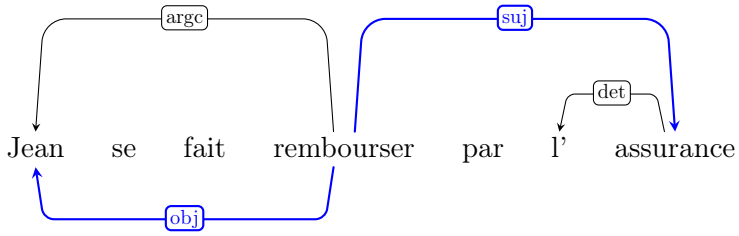


Figure 6.16: Suppression dans les constructions avec « se faire ».

Afin de clore cette section, nous donnons un récapitulatif des fonctions grammaticales utilisées au niveau profond (Table 6.1). Nous utilisons la couleur bleu pour marquer une fonction canonique uniquement.

6.2 Campagne d'annotation

Comme nous le disions en préambule, l'annotation de corpus est le fait de deux équipes, Alpage à Paris et Sémagramme à Nancy. Comme c'est un processus connu pour être long et coûteux, il nous a fallu déterminer un protocole suffisamment strict pour faciliter la tâche des annotateurs tout en leur proposant un guide d'annotation de qualité que nous décrivons maintenant.

6.2.1 Méthodologie et protocole d'annotation

L'annotation suit un protocole strict qui n'a pas de précédent à notre connaissance, puisque les deux tâches de pré-annotation et d'annotation ont été faites en double aveugle, avec une passe d'adjudication par chaque équipe puis une passe d'adjudication finale entre les deux équipes. Nous limitons ainsi le biais d'annotation à chaque étape.

6.2.1.1 Mise au point du guide par annotation préalable

Pour déterminer les difficultés d'annotation et voir sa complexité tout en mettant à jour le guide d'annotation, nous avons extrait aléatoirement des phrases du corpus Séquoia. Elles ont été choisies au sein de chaque domaine donnant lieu à un corpus de 250 phrases appelé *mini référence*.[3] Les phrases ont été choisies en conservant une homogénéité de longueur et nous nous sommes assurés que des phrases suffisamment longues étaient capturées par le processus de sélection. En effet, nous supposons que dans des phrases longues (plus de 40 mots), la probabilité de trouver des phénomènes complexes auxquels le guide n'apporte pas de réponse est forte. La mini référence a alors été pré-annotée puis validée manuellement par quatre des porteurs du projet et le guide a été mis à jour au fur et à mesure après discussion des problèmes avec l'ensemble des membres du projet.

3. Le corpus Séquoia contient quatre domaines, mais emea a été scindé en deux sous corpus (dev et test) ce qui donne 250 phrases et non 200.

Fonction	Utilisation
ARG	Utilisée dans le cas de prépositions « liées » : dans *de Charybde en Scylla*, *en Scylla* est dépendant de type ARG de la première préposition *de*
ARGC	Fonction uniquement canonique : pour l'argument causateur dans les constructions causatives
ARG.COMP	Relation d'une comparative avec son gouverneur qui est un adverbe ou adjectif exprimant la comparaison
ARG.CONS	Relation d'une consécutive avec son gouverneur adverbial ou adjectival
ATO	Attribut de l'objet
ATS	Attribut du sujet
A_OBJ	Complément indirect non locatif introduit par à
COORD	Relation portée par un coordonnant, avec comme gouverneur le premier conjoint
DEP	Relation sous-spécifiée, pour les dépendants prépositionnels de gouverneurs non verbaux et non adjectivaux (pas de gestion de la distinction argument / ajout pour ces cas)
DEP.COORD	Relation portée par un coordonné (sauf le premier), avec comme gouverneur le coordonnant immédiatement précédent
DET	Déterminant
DE_OBJ	Complément indirect non locatif introduit par de
MOD	Modifieur de gouverneur verbal ou adjectival, autre qu'une relative, d'un autre type de gouverneur pour peu que l'on puisse repérer structurellement qu'il s'agit d'un modifeur (adjectif épithète par exemple)
MOD.APP	Apposition à un groupe nominal
MOD.CLEFT	La subordonnée dans le cas d'une clivée
MOD.COMP	Modifieur introduisant une comparaison
MOD.INC	Proposition incise
MOD.REL	Dépendance d'une relative par rapport à son antécédent
MOD.REL.PART	Dépendance de la tête nominale d'une relative avec ellipse introduite par dont par rapport à l'antécédent
MOD.SUPER	Modifieur exprimant le superlatif
MOD.VOC	Vocatif
OBJ	Objet de verbe
OBJ.CPL	Objet de complémenteur
OBJ.P	Objet de préposition
PONCT	Relation portée par tout dépendant typographique, sauf pour les virgules jouant le rôle de coordonnant (qui porte la relation coord)
P_OBJ.O	Complément indirect du verbe ou de l'adjectif autre
SUJ	Sujet d'un verbe ou d'un adjectif

Table 6.1: Fonctions grammaticales utilisées dans la représentation en dépendances profondes (Fonctions canoniques uniquement en bleu).

6.2.2 Phénomènes annotés manuellement en amont

Au démarrage de la campagne d'annotation, certains phénomènes connus pour être rares (causatifs) ou difficiles (clitiques réfléchis, *il* impersonnel) ont été annotés manuellement en amont. Nous les détaillons rapidement ci-dessous.

6.2.2.1 Constructions causatives

L'annotation suit celle du French Treebank dans lequel le verbe *faire* + *infinitif* est considéré comme un verbe complexe causatif. La conversion en dépendances de surface, comme expliqué à la page 64, donne *faire* comme un dépendent (AUX.CAUS) de l'infinitif. Au niveau profond, l'auxiliaire est considéré sémantiquement vide et les fonctions canoniques sont explicitées. En effet, le causatif a la particularité d'introduire un argument causateur supplémentaire. ARGC est la fonction canonique de cet argument causateur. Un exemple est donné à la Figure 6.17. Les fonctions canoniques permettent d'expliciter le fait que c'est Marie qui entre, mais que Paul est celui qui cause l'action.

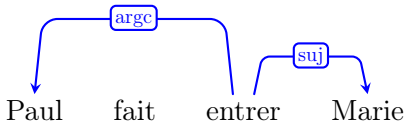


Figure 6.17: Exemple d'annotation du causatif en syntaxe profonde.

6.2.2.2 Clitiques réfléchis

L'annotation des réfléchis a fait l'objet d'un guide d'annotation spécifique (Candito, 2003) réalisé en parallèle de l'annotation du DeepSequoia et de l'annotation du projet Asfalda (French Framenet, Candito et al. (2014a)). Nous ne reprenons ici que les informations essentielles et ce de manière succincte. Le clitique *se* est toujours considéré comme sémantiquement vide et il n'appartient pas à la représentation profonde. Cependant, quatre classes d'occurrences de *se* sont considérées, car elles ont un impact sur la représentation profonde.

1. **Vrai réfléchi ou réciproque** : Le test principal est repris de (Creissels, 2007) : c'est la possibilité que [*x se* **verbe** *w*] puisse devenir [*x* ne **verbe** que lui-même] *ou* [*x* ne **verbe** qu'à lui-même]. Nous donnons aussi un test moins restrictif : celui de pouvoir forcer une lecture réciproque, en remarquant que si le réciproque est possible, alors le réfléchi doit l'être aussi. En surface le clitique a la fonction OBJ ou A_OBJ, en profond ce lien est déporté sur le sujet, donnant la représentation à la Figure 6.18
2. **Réfléchi intrinsèque** : [*se* + **verbe**] est considéré comme un seul lexème (utilisé pour les verbes pronominaux, l'antipassif, l'autocausatif (Creissels, 2007)). En surface, la fonction est AFF. Au niveau profond, le *se* est sémantiquement vide et le verbe porte un lemme profond *se_X* (cf. section 1, p. 117, pour plus de détails

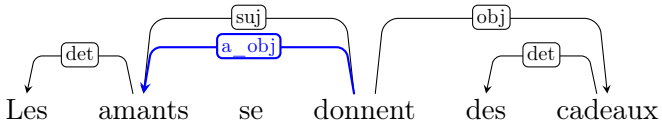


Figure 6.18: Exemple d'annotation d'un vrai réfléchi ou réciproque en syntaxe profonde.

sur les lemmes profonds). Dans l'exemple *Il se souvient de Capri, mais c'est fini*, le clitique *se* est sémantiquement vide et le verbe porte le lemme profond *se_souvenir*.

3. **Se moyen ou se neutre** : Nous ne distinguons pas le moyen du neutre dans le corpus. Dans ce cas-là, le clitique *se* porte la fonction aff.demsuj (qui disparaît en profond) et le sujet final du verbe est son objet canonique. Un exemple est donné à la Figure 6.19.

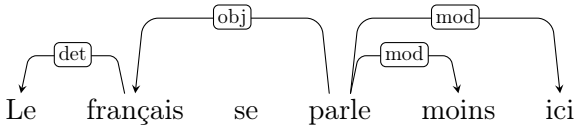


Figure 6.19: Exemple d'annotation d'un se moyen ou neutre en syntaxe profonde.

4. **Construction avec partie inaliénable.** : Les occurrences sont de la forme [x *se* **verbe** y] où x est sujet et y objet. y correspond à une partie inaliénable de x, ce qui se traduit par l'équivalence [x **verbe** *possessif* y]. Au niveau profond, un lien dep.de entre x et y est utilisé (pour pouvoir reconstruire y *de* x). Un exemple est donné à la Figure 6.20 qui peut être paraphrasé en *Le patient a fracturé sa hanche* (x = patient et y = hanche).

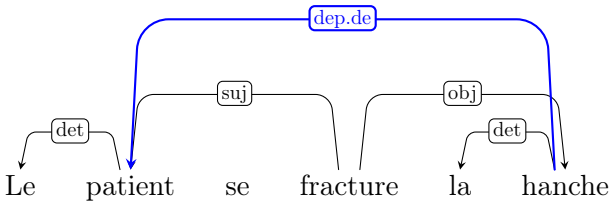


Figure 6.20: Exemple d'annotation d'une construction avec partie inaliénable en syntaxe profonde.

6.2.2.3 Le clitique *il* impersonnel

Le caractère référentiel de tous les « il » du corpus a été manuellement annoté. En outre, le cas des verbes *intrinsèquement impersonnels* (*il pleut*) a été distingué du cas des verbes analysables comme une alternance syntaxique (Fig. 6.21). Le traitement de la diathèse impersonnelle est résumée à la page 76 avec les autres changements de diathèse.

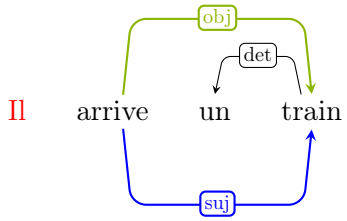


Figure 6.21: Exemple d'annotation des clitiques impersonnels en syntaxe profonde. Le token en rouge est considéré comme sémantiquement vide. Nous avons noté la fonction finale en vert et la fonction canonique en bleu (seule cette dernière est conservée au niveau profond).

6.2.2.4 Réduction du biais d'annotation grâce à l'annotation en double aveugle

À la suite de la construction du guide d'annotation, du choix des phrases formant la mini-référence et de l'annotation manuelle des phénomènes décrits ci-dessus, le processus d'annotation a été réalisé en trois étapes.

Pré-annotation du DeepSequoia. Nous sommes partis du postulat que l'annotation en syntaxe profonde pouvait être faite en grande partie automatiquement, ce qui permet de limiter la tâche des annotateurs et de leur faire gagner du temps. En effet, certaines informations lexicales telles que les verbes à contrôle ou les verbes à montée peuvent être annotés avec une grande précision via des règles. Par ailleurs, rediriger les arcs des mots sémantiquement vides vers les mots sémantiquement pleins est aussi une passe d'annotation que des règles peuvent automatiquement gérer. Pour toutes ces raisons, nous avons décidé d'utiliser deux systèmes par règles : un système à réalisé à Nancy et appelé grew et décrit aux sections 7.1.3.1 (pour les bases formelles, p. 100) et 7.1.3.2 (pour les utilisations linguistiques, p. 101) ; un autre réalisé dans le cadre de notre mémoire de recherche de Master (Ribeyre, 2012) et largement amélioré durant cette thèse (sec. 7.1.3.3, p. 105 pour les bases formelles et sec. 7.2.1, p. 7.2.1, pour les utilisations linguistiques). L'idée d'utiliser deux systèmes sans s'échanger d'informations sur les règles écrites permet de limiter le biais de pré-annotation et globalement réduit le biais d'annotation.

Validation des données pré-annotées. Une fois les données pré-annotées deux équipes d'annotateurs ont validé et corrigé chaque phrase : une équipe à Nancy (trois étudiants en linguistique à Nnacy) et une autre à Paris (deux annotateurs experts à Paris). Chaque équipe a annoté phénomène par phénomène afin d'obtenir le corpus le plus homogène possible. Les phénomènes sont annotés du plus simple au plus complexe, permettant une progression dans la formation des annotateurs tout en affinant le guide au fur et à mesure. [4]

4. Pour aider à l'annotation, nous avons développé un outil permettant de visualiser et de modifier

Scripts de contrôle et adjudication. Après l'annotation, des scripts de contrôle ont assuré que le corpus ne présentait pas d'anomalies de formatage et/ou des incohérences au niveau des étiquettes des relations (c'est-à-dire relations qui n'existent pas, relations impossibles à trouver entre certaines paires de mots, etc.). Enfin, les deux équipes, celle de Nancy et celle de Paris, ont adjudiqué les divergences d'annotation pour obtenir la ressource finale.

À notre connaissance ce type de protocole d'annotation est inédit et n'a jamais été employé avant. En effet, en général, si une pré-annotation est faite, elle n'est exécutée que par un seul système et non deux. Souvent encore, seule une équipe d'experts annote le corpus sans se concerter et non deux équipes. De fait, l'emploie de ce protocole permet de limiter le biais à chaque phase d'annotation du corpus le rendant plus homogène et moins propice à la propagation d'erreurs.

6.2.3 Accord inter-annotateur

Pour estimer la difficulté de la tâche d'annotation (mais aussi la difficulté de la tâche d'analyse automatique qui nous intéresse en Traitement Automatique des Langues), nous reportons l'accord inter-annotateur entre Paris et Nancy à la Table 6.2. Nous calculons un F_1-score pour deux raisons : la première provient du fait que le F_1-score est utilisé pour l'évaluation des analyseurs automatiques sur ce genre de tâche, ce qui nous donne une idée de la marge haute qu'aucun système automatique n'a de chance de dépasser et la seconde réside dans le fait que le κ (Siegel et Castellan, 1988) habituellement utilisé pour le calcul de l'accord inter-annotateur est inadéquat sur des tâches de catégorisation où le nombre d'items n'est pas fixe (Burchardt et al., 2006, note n°2).

	Tout	Profond seuls ($\notin$ surface)
LF	96.00	94.23
UF	97.48	95.77
# arcs	59 235	12 878

Table 6.2: Accord inter-annotateur pour l'annotation du DeepSequoia.
LF représente le F_1-score étiqueté, UF non étiqueté.

À la Table 6.2 deux types d'accord inter-annotateur ont été distingués. Le premier, appelé *tout*, signifie que tous les arcs du corpus sont pris en compte avec les fonctions canoniques. En effet, un certain nombre d'arcs présents en surface ne sont pas modifiés lors de l'annotation profonde et sont donc aussi capturés lors du calcul de l'accord inter-annotateur. Le second, appelé *profond seuls* ne calcule l'accord que sur les arcs qui ne sont pas déjà réalisés en surface (sujets des infinitives, des participiales, des adjectives, etc.). Cette métrique montre un accord inter-annotateur plus faible, mais aussi plus réaliste, car les arcs surfaciques non modifiés ne sont pas pris en compte.

les graphes de dépendances. L'outil est librement disponible en ligne (`https://github.com/Cocophotos/DepAnnotator`).

On peut comparer notre accord inter-annotateur à celui de divers autres travaux tels que l'annotation tectogrammaticale du Prague Dependency Treebank (Hajič, 1998; Hajičová et Pajas, 2000) et l'annotation MTT de l'AnCora-UPF (Mille et al., 2013). Nous donnons l'accord inter-annotateur de ces deux corpus à la Table 6.3.

	LF	UF
PDT	84.40	91.10
AnCora	89.40	96.15
DeepSequoia	94.23	95.77

Table 6.3: Comparaison de l'accord inter-annotateur du DeepSequoia, du Prague Dependency Treebank (niveau tectogrammatical) et de l'AnCora-UPF [5].

Mille et al. (2013) donne un accord inter-annotateur calculé sur la surface mais en prenant en compte les étiquettes présentes en surface et celles en au niveau profond (79 étiquettes sont donc considérés). Ils rappellent par ailleurs que l'ensemble de ces étiquettes permettant de dériver automatiquement le graphe profond, l'accord inter-annotateur peut donc être considéré pour le niveau profond. Nous remarquons que les scores non étiquetés sont très similaires à ceux obtenus pour le DeepSequoia. La différence sur les scores étiquetés s'explique sans doute par le nombre d'étiquettes considérées pour le Prague Dependency Treebank (40 étiquettes) et l'AnCora-UPF (79 étiquettes) contre 34 pour le DeepSequoia.

6.3 Aspects formels et comparaison avec d'autres ressources

Nous proposons maintenant de discuter certaines particularités du DeepSequoia et notamment le fait que sa représentation profonde génère des cycles. Enfin, nous comparons ses caractéristiques avec celles des autres corpus que nous avons décrits précédemment et dont la synthèse se trouve à la page 57.

6.3.1 Le niveau profond est-il un graphe cyclique ?

Contrairement à tous les autres ressources en syntaxe profonde, le DeepSequoia présente la particularité d'avoir des graphes cycliques. Il est alors intéressant de se demander dans quelle mesure ces cycles sont nécessaires pour représenter la structure argumentale de la phrase. Ils sont de deux sortes : des cycles directs ($A \rightarrow B\ B \rightarrow A$) ou des cycles indirects.

6.3.1.1 Cycles directs dans le DeepSequoia

L'annotation des adjectifs prédicatifs génèrent le plus grand nombre de cycles directs. Notre représentation profonde conserve la dépendance de surface entre le nom et son adjectif marquant que l'adjectif est un modifieur (étiquette MOD) du nom. Au niveau profond,

5. LF signifie *Labeled F_1-score* et représente le F_1-score en prenant en compte le relations de dépendances, UF pour *Unlabeled F_1-score* ne prend pas en compte ces relations.

une relation de type sujet est ajoutée entre l'adjectif et le nom (étiquette suj) permettant de différencier les cas : *Cette chose est impossible* (existence de la copule) et *Une chose impossible* (la copule n'existe pas). Cependant, en conservant le fait que l'adjectif est un modifieur du nom, le corpus génère un nombre important de cycles directs. Un exemple est donné à la Figure 6.22.

A posteriori, nous pensons que cette annotation n'est ni nécessaire ni intéressante à conserver. En effet, un arc est une relation binaire entre deux termes A et B, qu'il est possible de considérer comme $A \rightarrow B$ (ou l'inverse en fonction du sens de l'arc). Un cycle direct induit une relation d'équivalence entre deux termes A et B, c'est-à-dire $A \leftrightarrow B$. Or, il est assez facile de retrouver cette relation d'équivalence à partir d'une des deux relations d'implication : en effet, si $A \rightarrow B$, alors il sera possible de dire $B \rightarrow A$ (ou vice-versa), le cycle direct n'est donc pas justifié. Par ailleurs, l'existence d'une relation cyclique directe entre deux mots ne permet plus de discriminer qui est l'argument et qui est le prédicat, chose contre-intuitive lorsqu'il s'agit de modéliser la structure argumentale. Enfin, il est tout à fait possible de modéliser ce type de cycles en utilisant une étiquette appropriée telle que suj.m (pour suj et mod).

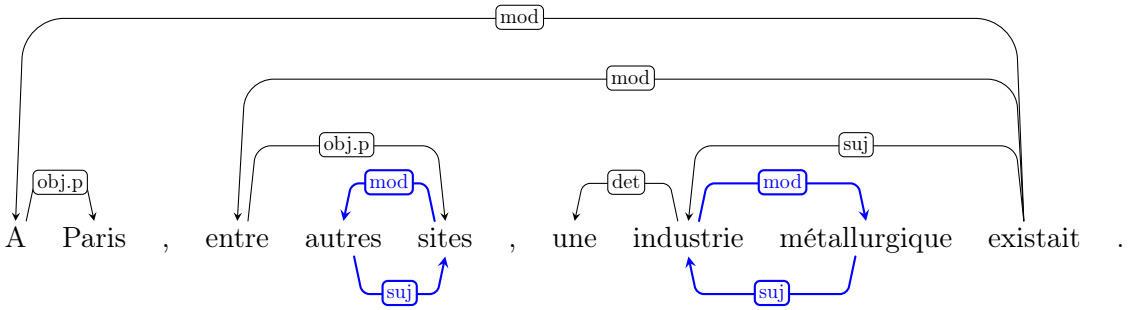


Figure 6.22: Exemple de cycle direct dans le DeepSequoia (en bleu).

6.3.1.2 Cycles indirects dans le DeepSequoia

De même, le DeepSequoia contient un certain nombre de cycles indirects dont nombre sont causés par le traitement des propositions relatives (exemple 6.23).

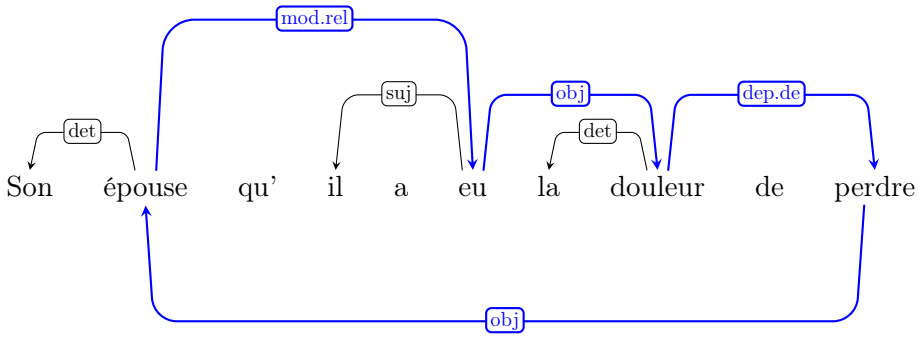


Figure 6.23: Exemple de cycle indirect dans le DeepSequoia mettant en jeu une relative (en bleu).

Le cycle est visible à cause du chemin entre MOD.REL, OBJ, DEP.DE et OBJ. Il n'est pas possible de supprimer l'arc OBJ qui explicite le cadre sous-catégorisation du verbe *perdre*. Cependant, l'arc MOD.REL qui explicite une relation entre l'antécédent et le verbe de la relative n'est pas nécessaire pour retrouver la structure argumentale de la phrase. En effet, la relative est un phénomène purement syntaxique, car la paraphrase *Il a eu la douleur de perdre sa femme* représente un sens similaire et une structure argumentale similaire. De fait, MOD.REL peut être supprimé, ce qui évitera ce type de cycles.

La question de savoir si la représentation profonde est un graphe cyclique relève donc, selon nous, de la définition que nous donnons du niveau profond. C'est un niveau qui mélange syntaxe et sémantique et en fonction des applications, il peut être intéressant de converser certains phénomènes syntaxiques. Par exemple, il peut être intéressant de faire la différence entre *Marie que Jean a vue* et *Jean a vu Marie* (voir'(Jean, Marie), car la structure informationnelle n'est pas la même, le focus n'est pas mis sur la même entité. De même, au niveau syntaxique, il peut être intéressant de pouvoir différencier *Une chose impossible* et *Cette chose est impossible*. Néanmoins, pour les applications qui nous intéressent, ce genre de distinctions n'est pas nécessaire, de même que les cycles qu'il engendre.

6.3.2 Comparaison avec les ressources existantes

Après avoir discuté des cycles du DeepSequoia, il est utile de le comparer aux ressources existantes et que nous avons déjà décrites dans les chapitres précédents. Pour ce faire, nous complétons la synthèse de la page 57 en l'augmentant du corpus que nous avons construit et présenté tout au long de ce chapitre. Elle est donnée à la Table 6.4.

	Validation	# Phrases	# Étiquettes	Alt. synt.	Sujets inf.	LDD	Coord. elliptiques
Prague	+	49 000	40	+	+	+	−
AnCora	+	3 500	11	+	+	−	−
DM	−	35 000	52	−	+	−	−
PAS	−	35 000	43	−	+	−	−
CCGBank	−	49 000	NA	−	+	+	+
MFT-LFG	|	4 700	NA	−	+	+	+
TLGBank	|	12 300	NA	−	+	+	+
(Guillaume et Perrier, 2012)	−	12 300	±30	+	+	−	−
DeepSequoia	+	3 099	34	+	+	+	|

Table 6.4: Comparaison des caractéristiques des ressources existantes avec le DeepSequoia. En haut les ressources pour des langues différentes du français et au milieu les ressources pour le français.
L'utilisation du *pipe* (|) note une validation manuelle partielle.

Le corpus est comparable à la couche tectogrammaticale du Prague Dependency Treebank. La majeure différence réside dans le fait que le Prague Dependency Treebank marque la structure informationnelle de la phrase (Hajičová et al., 1998), ce que notre représentation profonde ne fait pas. Cela mis de côté notre représentation explicite le cadre de sous-catégorisation des verbes et adjectifs et neutralise les changements de diathèse, comme le fait le Prague Dependency Treebank. Une autre différence majeure se retrouve dans le choix des relations utilisées par le Prague Dependency Treebank : elles sont proches des

rôles thématiques (AGENT, PATIENT, etc.) que nous avons choisis de ne pas utiliser pour le moment. De fait, notre représentation profonde possède un jeu de relations moins important et plus proche de la syntaxe. Enfin, la couche tectogrammaticale annote des arbres projectifs, l'ordre des mots n'est donc plus respecté. Cela est possible dans les langues à morphologie riche, où le système casuel est très développé, car il est possible distinguer *Jean aime Marie* de *Marie aime Jean* facilement, ce n'est pas le cas pour le français. Par ailleurs, il semble clairement plus naturel d'utiliser des graphes qui offrent une représentation plus souple et tout aussi bien maîtrisée. Nous pouvons aussi comparer le DeepSequoia avec l'AnCora-UPF. Ici, les différences sont nombreuses : l'AnCora-UPF ne possède qu'un petit nombre d'étiquettes et n'annote pas ou peu de phénomènes longue distance. Par ailleurs, le corpus utilise des arguments numérotés à la manière de PropBank (Palmer et al., 2005) et NomBank (Meyers et al., 2004).

Finalement, le DeepSequoia est une ressource qui se situe à mi-chemin entre l'AnCora-UPF et le Prague Dependency Treebank. Il est l'une des premières ressources manuellement annotées et en syntaxe profonde pour la langue française. Il présente aussi l'avantage d'être libre de droit et hors domaine. Cette dernière caractéristique peut aussi être vue comme un inconvénient, car la plupart des travaux auxquels nous souhaitons nous comparer sont évalués sur le Penn Treebank, un corpus journalistique. De fait, il est utile – sinon nécessaire – de posséder un tel corpus à titre de comparaison. Il était difficile de répliquer notre processus d'annotation sur les 22 000 que compte le French Treebank, car l'annotation manuelle est coûteuse en temps. De fait, nous avons ré-utilisé nos techniques de pré-annotation pour effectuer une annotation automatique de ce corpus que nous décrivons maintenant.

Chapitre

7 Vers plus de données : pré-annotation automatique par règles de réécriture

Sommaire

Introduction

L'annotation du DeepSequoia présentée au chapitre précédent a été effectuée via une passe de pré-annotation utilisant deux systèmes de réécriture de graphes : GREW (Bonfante et al., 2010, 2011b,a) développé à Nancy et OGRE (Ribeyre et al., 2012; Ribeyre, 2013) développé dans le cadre de notre mémoire de Master (Ribeyre, 2012) et amélioré de manière conséquente au cours de cette thèse. La réécriture de graphes est une approche qui s'inscrit plus généralement dans la théorie des grammaires de graphes (issue des langages formels), théorie présentée à la section 7.1. L'idée est néanmoins simple et intuitive : au lieu de réécrire des symboles comme le font les grammaires hors contexte, des sous-graphes

sont réécrits. Cependant, contrairement à la réécriture de symboles (et plus généralement de chaînes de caractères), la réécriture de graphes pose des problèmes en terme de complexité puisque la majorité des algorithmes utilisés sont soit NP-difficiles, soit NP-complet. Néanmoins, nous nous montrons que les graphes prédicat-argument considérés dans cette thèse ont des propriétés très particulières qui les rapprochent des arbres et permettent de garantir de bonnes performances en pratique même lors de l'utilisation d'algorithmes NP-complet. Une fois ces bases formelles posées, nous discutons des propriétés de notre système de réécriture qui tire partie des applications linguistiques que nous traitons pour proposer une approche en deux temps permettant de limiter le nombre de règles à écrire et améliorer drastiquement la maintenabilité du système sur le long terme. Nous démontrons que le système termine et qu'il est confluent tout en montrant que la seconde phase de réécriture est comparable à l'écriture d'une forme de méta-grammaire (Candito, 1999) pour les graphes. Enfin, nous appliquerons notre système à trois tâches complémentaires :

1. La pré-annotation du DeepSequoia pour laquelle nous proposons une évaluation donnant d'excellents résultats (section 7.2.1).
2. La projection du schéma d'annotation profond via les règles de réécriture sur le French Treebank (Abeillé et al., 2003) pour obtenir plus de données (section 7.2.2) et pouvoir conduire des expériences d'analyse automatique de la syntaxe profonde en français.
3. L'utilisation des règles de réécriture pour produire des structures prédicat-argument via les sorties d'analyseurs syntaxiques de surface (section 7.3) ; le but étant de montrer la viabilité de l'approche sur des sorties prédites et donc bruitées.

7.1 Un système formel complet d'annotation automatique de structures profondes

7.1.1 Notations formelles indispensables

Nous avons besoin de notation formelle pour définir les graphes de dépendances et certaines opérations qui leur sont liés.

Définition 7.1 (Graphe orienté)**.**
$G = \langle V, E \rangle$ un graphe orienté avec :

- V est un ensemble de nœuds.
- $E \subseteq V \times V$ est un ensemble d'arcs, où chaque arc est un paire ordonnée (u, v). Un arc (u, v) pourra aussi être écrit $u \rightarrow v$. u est le nœud initial ou source et v le nœud terminal ou cible. L'arc $a = (u, v)$ est dit **sortant** en u et **incident** en v.

Une boucle est un arc (u, u) reliant un sommet u à lui-même. Un graphe orienté est dit **élementaire** s'il ne contient pas de boucle.

Définition 7.2 (Graphe orienté et étiqueté)**.**
$G = \langle V, E, L \rangle$ un graphe orienté et étiqueté avec :

- V est un ensemble de nœuds.
- L est un ensemble d'étiquettes.
- $E \subseteq V \times L \times V$ est un ensemble d'arcs étiquetés, où chaque arc est un triplet ordonné (u, l, v). Un arc (u, l, v) pourra aussi être écrit $u \xrightarrow{l} v$.

Un graphe orienté est un **p-graphe** s'il comporte au plus p arcs entre deux nœuds. Nous utilisons les notions de multi-graphe et de p-graphe (avec $p > 1$) indifféremment, de même pour les notions de 1-graphe et de graphe.

Définition 7.3 (Sous-graphe)**.**
Un graphe $H = \langle V', E_H, L_H \rangle$ est un **sous-graphe** de $G = \langle V, E, L \rangle$ si $V' \subseteq V$, $E_H \subseteq E$ et $L_H \subseteq L$.

Un **sous-graphe induit** de G pour un sous-ensemble $V' \subset G$ est un sous-graphe H de G ayant pour arcs uniquement ceux de G joignant les nœuds de V' :

$$H = \langle V', E_H, L \rangle \text{ avec } V' \subset V \text{ et } E_H = \{(u, l, v) \in E, u \in V', l \in L \text{ et } v \in V'\}.$$

Définition 7.4 (Isomorphisme de graphe)**.**
Un graphe $G_1 = \langle V_1, E_1, L \rangle$ est isomorphe à un graphe $G_2 = \langle V_2, E_2, L \rangle$, si et seulement si une bijection $f : V_1 \rightarrow V_2$ existe telle que : $\forall (u, l, v) \in E_1 \Leftrightarrow (f(u), l, f(v)) \in E_2$. La bijection f est un isomorphisme entre G_1 et G_2.

Un graphe G_1 est un *sous-graphe* isomorphe à un graphe G_2, si et seulement s'il existe un sous-graphe $g \subseteq G_2$ tel que G_1 est isomorphe à g. Dans ce cas, on appelle g un *enchâssement* (*embedding*) de G_1 dans G_2.

Définition 7.5 (Degré (entrant / sortant))**.**
Le **degré** (ou **valence**) d'un nœud d'un graphe est le nombre d'arcs reliant ce sommet, les boucles sont comptées deux fois. Le degré d'un nœud u est noté $\deg(u)$. Dans un graphe orienté, on distingue le **degré entrant** $d^-(u)$, c'est-à-dire le nombre d'arcs dirigés vers le nœud u et le **degré sortant** de ce nœud $d^+(u)$, c'est-à-dire le nombre d'arcs sortant de u. On a $\deg(u) = d^+(u) + d^-(u)$: le degré d'un nœud u est la somme de son degré entrant et sortant.

Définition 7.6 (Chemin dans un graphe)**.**
Un chemin d'un nœud u vers un nœud v dans un graphe orienté G est une séquence $\langle v_0, v_1, \ldots, v_k \rangle$ de nœuds, tels que $u = v_0$, $v = v_k$ et $(v_{i-1}, v_i) \in E$ pour tout $i \in \{1, \ldots, k\}$. La longueur du chemin est le nombre d'arcs dans le chemin, soit k. S'il existe un chemin entre u et v alors v est **accessible** à partir de u. Dans le cas d'un graphe non orienté, un chemin s'appelle une **chaîne**. De plus, un chemin $\langle v_0, v_1, \ldots, v_k \rangle$ forme un **circuit** si $v_0 = v_k$ et si le chemin comporte au moins un arc ($k \geq 1$).

Propriété 7.1 (Propriété de connexité)**.**
Un graphe orienté est dit **connexe** si chaque nœud est **accessible** à partir de n'importe quel autre. Autrement dit, si pour toute paire de nœuds $(u, v) \in A$, il existe un chemin entre u et v.

Propriété 7.2 (Propriété d'acyclicité)**.**
Un graphe acyclique est un graphe sans circuit.

Pour une phrase $S = ({w_0}^1, w_1, \ldots, w_n)$, une séquence strictement ordonnée, avec $w_0, \ldots, w_n$ les tokens dans la phrase et un graphe orienté $G = \langle V, E \rangle$ avec $V = S$, nous supposons une relation d'ordre total strict $<$ sur les nœuds V du graphe, ordre qui correspond à l'ordre des mots dans la phrase. Cette relation d'ordre nous permet de comparer les nœuds entre eux, pour deux nœuds $u, v \in V$ avec $u < v$, $\min(u, v) = u$ et $\max(u, v) = v$.

Propriété 7.3 (Propriété de non-planarité)**.**
Deux arcs (i, j) et (k, l) avec $\min(i, j) < \min(k, l)$ se croisent si et seulement si $\min(i, j) < \min(k, l) < \max(i, j) < \max(k, l)$. Un graphe est dit non-planaire s'il existe au moins deux arcs (i, j) et (k, l) qui se croisent.

À noter que la définition de la non-planarité en Traitement Automatique des Langues diffère de celle donnée en théorie des graphes : un graphe est non planaire si au moins deux arcs se croisent dans un demi-plan, alors qu'en théorie des graphes, le plan entier est pris en compte.

7.1.2 Introduction aux grammaires de graphes

Les grammaires de graphes (Pfaltz et Rosenfeld, 1969; Rozenberg, 1997) sont une extension des grammaires hors contexte aux graphes.[2] Au lieu de réécrire des symboles à

1. w_0 est un nœud virtuel qui sert de racine.
2. Hors de la théorie des langages formels, le terme *réécriture de graphes* est plus couramment employé. Nous utiliserons les deux termes indifféremment l'un de l'autre.

chaque étape de dérivation, un sous-graphe est réécrit. Les grammaires de graphes ont été d'abord inventées pour résoudre des problèmes d'analyse sur des images (Pfaltz et Rosenfeld, 1969). Par la suite, un nombre important de travaux se sont développés afin de proposer des théories et des algorithmes applicables à un ensemble important de problèmes complexes, tels que la reconnaissance de motifs, ou la gestion de bases de données (Baresi et Heckel, 2004).

Les deux méthodes les plus utilisées pour réécrire un graphe sont le remplacement de nœuds ou le remplacement d'arcs. Dans le premier cas, un nœud d'un graphe donné est remplacé par un nouveau sous-graphe qui se connecte alors au graphe d'entrée (Rozenberg, 1997) (appelé aussi *graphe hôte*). Dans le second cas, un arc est remplacé par un nouveau sous-graphe (Drewes et al., 1997). Ce sous-graphe est accolé au graphe hôte et fusionne avec certains nœuds de l'hypergraphe que l'on appelle *nœuds externes*. De manière plus générale, la théorie peut être formalisée selon le principe des *approches algébriques*, principe selon lequel un sous-graphe est remplacé par un autre, au lieu de remplacer un nœud par un sous-graphe ou un arc par un sous-graphe. [3]

7.1.2.1 Confluence et terminaison

D'autre part, deux propriétés importantes dominent tout système de réécriture : la propriété de **terminaison** et la propriété de **confluence**. Ces deux propriétés permettent d'assurer que le système fonctionne et qu'il fait exactement ce pour quoi il a été créé. La première propriété (celle de **terminaison**) assure que le système termine en un temps fini, c'est-à-dire qu'il ne part pas dans un cycle infini de réécritures. La deuxième propriété (celle de **confluence**) garantit que la réécriture donnera toujours la même structure en sortie et quelque soit l'ordre d'application des règles, en cas de non déterminisme. La confluence est une propriété importante, car elle assure une meilleure maintenabilité du système de règles. Sans la garantie de cette propriété, soit le système est structuré en modules au sein desquels l'application des règles est confluente, c'est notamment le cas de GREW (Bonfante et al., 2011a), soit le système produit l'ensemble de tous les graphes possibles pour un graphe hôte et un ensemble de règles données, à la manière d'une forêt de dérivation(Billot et Lang, 1989) produite par certains analyseurs syntaxiques. Dans le premier cas, cela signifie que chaque ajout de règles peut violer la confluence et qu'il faut prouver à chaque ajout que le module est confluent ; dans le second cas, il est important d'avoir un modèle de désambiguïsation pour trouver le graphe le plus probable, étant donné le graphe hôte et l'ensemble de règles.

7.1.2.2 Isomorphisme de (sous)-graphes et NP-complétude

Un autre point central des grammaires de graphes et plus généralement des problèmes de réécriture de graphes réside dans la détection d'un isomorphisme entre graphes ou sous-

3. Pour dresser un parallèle avec des méthodes plus connues en TAL, nous citons XMG (Parmentier et Le Roux, 2005; Crabbé et al., 2013), une méta-grammaire extensible permettant de générer des grammaires d'arbres telles que les Grammaires d'arbres adjoints (*Tree Adjoining Grammar*, TAG, et des formalismes connexes) ou encore les grammaires d'interactions (Guillaume et Perrier, 2009).

graphes, c'est-à-dire savoir si deux graphes sont identiques ou si un sous-graphe est contenu dans un graphe (pour une description formelle de l'isomorphisme voir définition 7.4, p. 93). Le problème de l'isomorphisme de graphe est connu pour n'avoir ni algorithme en temps polynomial, ni une preuve de son caractère NP-complet (Schöning, 1988) contrairement au problème d'isomorphisme de sous-graphe qui est, quant à lui, NP-complet, à comparer avec le fait qu'il existe des solutions en temps linéaire (Hopcroft et Tarjan, 1972) pour le calcul de l'isomorphisme d'arbres.

Le problème n'ayant pas d'algorithme en temps polynomial, de nombreuses approches ont vu le jour afin de trouver l'algorithme le plus rapide possible. La Table 7.1 décrit différents algorithmes utilisés aujourd'hui dans ce domaine.

Algorithmes	Techniques utilisées	Isomorphisme
Ullmann (1976)	backtracking + lookahead	(Sous) Graphe
SD (Schmidt et Druffel, 1976)	matrice de distance + backtracking	Graphe
Nauty (McKay, 1981)	théorie des groupes + étiquetage canonique	Graphe
VF (Cordella et al., 1998)	parcours profondeur + règles de plausibilité	(Sous) Graphe
VF2 (Cordella et al., 2001)	VF + structures de données avancées	(Sous) Graphe

Table 7.1: Algorithmes de test d'isomorphisme de (sous) graphe les plus connus.

Tous les algorithmes s'emploient à réduire l'espace de recherche d'une manière ou d'une autre. L'algorithme d'Ullmann (1976) emploie une procédure de retour en arrière (*backtracking*) et de regard en avant (*lookahead*). L'algorithme SD, quant à lui, utilise une matrice de distance représentant le graphe avec une procédure de retour en arrière (Schmidt et Druffel, 1976). L'algorithme *Nauty* (McKay, 1981) utilise la théorie des groupes pour transformer le graphe en une forme canonique qui permet de vérifier l'isomorphisme de manière efficace, même si certaines formes canoniques prennent un temps exponentiel à être créées. Les algorithmes VF (Cordella et al., 1998) et son implémentation plus efficace, VF2 (Cordella et al., 2001), utilisent un parcours en profondeur associé à des règles de plausibilité, toujours dans le but de restreindre l'espace de recherche.

Tout ce que nous venons de présenter s'applique sur des problèmes où les graphes sont généraux : cycliques, non planaires, parfois même complets. Néanmoins, les graphes représentant la structure argumentale sont très particuliers. D'une part, ils présentent la particularité d'avoir un ordre fixe (suivant l'ordre des mots), d'autre part ils sont loin d'être complets. Ces deux propriétés ne sont qu'une vue d'ensemble grossière de l'intégralité des propriétés caractérisant ce type de graphes, c'est pourquoi nous les développons plus précisément ci-après.

7.1.2.3 Formellement qu'est-ce qu'un graphe linguistique ?

Plusieurs propriétés apparaissent naturellement pour les graphes que nous traitons. Nous en distinguons quatre importantes :

1. **Ordre fixe des nœuds** qui suivent l'ordre linéaire de la phrase. Cette propriété est discutée par certains corpus comme le Prague Dependency Treebank (Hajič, 1998) où

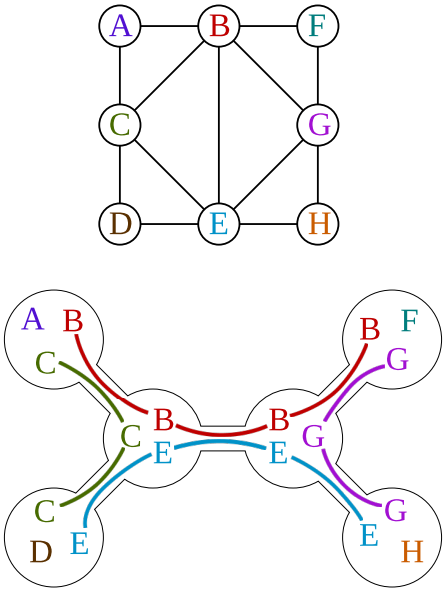


Figure 7.1: Exemple de décomposition arborescente pour le graphe composé de 8 nœuds (de A à H).

l'ordre n'est plus considéré pour respecter le critère de projectivité. Néanmoins, elle est utile pour l'analyse automatique par transitions (Nivre, 2005). Ce type d'analyse est souvent décrit comme une analyse incrémentale gauche-droite, construisant le sens d'un énoncé à la manière dont le fait le cerveau humain lors de la lecture d'une phrase (Roark et al., 2009). Si l'ordre n'est pas respecté, la lecture incrémentale devient difficile voire impossible.

2. **Non-planarité**. Cette propriété (Propriété 7.3, p. 94) donne une mesure de complexité de la structure prédicat-argument, informellement définie au chapitre 2, page 9 et donc de la phrase étudiée. Elle est directement liée à la notion de non-projectivité pour les arbres de dépendances (Kuhlmann et Satta, 2009; Kuhlmann, 2013). La question est cependant beaucoup plus cruciale, car le nombre d'arcs étant plus important, la non-planarité est plus fréquente dans un graphe que dans un arbre.
3. **Degré entrant et sortant**. L'analyse automatique cherche à prédire pour une phrase donnée, sa structure prédicat-argument. Cela revient à prédire les arcs qui composent la structure. Le degré entrant et sortant (Définition 7.5, p. 94) est donc un bon indicateur de complexité du graphe prédicat-argument.
4. **Largeur arborescente** (*treewidth*). La dernière propriété est sans doute l'une des plus intéressantes. La largeur arborescente mesure à quel point un graphe est proche

d'un arbre. Plus la largeur est faible, plus le graphe ressemble à un arbre. La largeur arborescente est définie de la manière suivante : Étant donné un graphe $G = (V, E)$, une décomposition arborescente de G est un couple (X, T), où $X = \{X_1, \ldots, X_n\}$ est une famille de sous-ensembles de sommets de V, et T est un arbre dont les nœuds sont ces sous-ensembles X_i, tels que :

a) l'union de tous les X_i de X est égale à V ($\bigcup_i X_i = V$) ;

b) pour toute arête (v, w) de E, il existe un nœud X_i de l'arbre T qui contient v et w ;

c) si X_i et X_j contiennent un même sommet v, alors tous les nœuds X_z de T sur le chemin non dirigé entre X_i et X_j contiennent v.

Cette dernière condition est équivalente au fait que tous les nœuds X_i de l'arbre T contenant un nœud v de G induisent un sous-arbre de T. En général, il existe plusieurs décompositions arborescentes. La taille t d'une décomposition arborescente est $t = \max_{X_i \in X} |X_i| - 1$, c'est-à-dire la taille de son plus grand ensemble moins un. La largeur arborescente (*treedwith*) $tw(G) = \min_{(X,T)} t$, c'est-à-dire la plus petite décomposition arborescente de G. (*treewidth*) du graphe. Un exemple de décomposition arborescente est donné à la Figure 7.1.

Nous avons calculé ces propriétés sur quatre corpus différents : le DM corpus (Ivanova et al., 2012) (décrit page 47), le PAS corpus (Miyao et al., 2004) (décrit page 49), le DeepSequoia (Candito et al., 2014b) et le DeepFTB (Ribeyre et al., 2014a) (décrit page 120 et qui est une projection de la pré-annotation du DeepSequoia sur le French Treebank). Nous utilisons ces quatre corpus car ils ont une représentation relativement équivalente et font l'objet d'expériences d'analyse automatique au chapitre 8, section 8.3 (p. 148). Les statistiques sont données Table 7.2.

	NON-PLANAR.		DEG. ENTRANT	DEG. SORTANT	LARG. ARBO.
	% Arcs	% Phrases	Min/Max	Min/Max	Min/Moy/Max
DM	3.45	41.94	0 / 9	0 / 5	0 / 1.30 / 3
PAS	3.92	45.97	0 / 14	0 / 4	1 / 1.72 / 3
DeepSequoia	3.91	35.14	0 / 7	0 / 32	1 / 1.42 / 3
DeepFTB	3.74	44.14	0 / 12	0 / 35	1 / 1.50 / 4

Table 7.2: Statistiques sur les graphes de différents corpus en syntaxe profonde.

Plusieurs constats peuvent être faits : premièrement, le taux d'arcs croisés est suffisamment important pour qu'il soit intéressant de le prendre en compte. Nous verrons notamment que c'est un problème lors de l'analyse par transitions, car la gestion de la non-planarité doit faire l'objet d'une action spéciale apportant un fort non-déterminisme et rendant l'analyse plus sensible à la propagation d'erreurs. Deuxièmement, le degrée sortant ne semble pas être une mesure convaincante sur les graphes prédicat-argument vu la divergence entre les schémas d'annotation. Néanmoins, cette divergence peut être expliquée par la présence dans les corpus en français d'énumérations rares où la première entité

de l'énumération est la tête des suivantes. Dans le cas d'énumérations longues, le degré sortant augmente rapidement.

Le degrée entrant, quant à lui, donne une assez bonne généralisation, car il modélise la notion de multi-gouverneurs. Par ailleurs, le calcul des degrés entrant et sortant permet de capturer la présence de nœuds sémantiquement vides, ce qui se vérifie pour tous les corpus considérés. Cette information est essentielle, car elle met en évidence l'une des généralisations les plus fortes au niveau de tous les schémas d'annotation : entre la syntaxe et la sémantique et au sein de certaines catégories morpho-syntaxiques, il n'est plus utile de considérer certains nœuds. Néanmoins, pour les langues configurationnelles où le système casuel n'est pas très développé, l'ordre des mots importe (pour distinguer par exemple *Jean aime Marie* de *Marie aime Jean*). De fait, il peut être important sinon nécessaire de conserver ces nœuds vides dans le graphe. Cela a pour effet de produire des graphes déconnectés qui posent des problèmes lors de l'induction automatique de règles de réécriture (cf. chapitre 9, p. 175 pour plus de détails), mais aussi au niveau de l'analyse par transition : il faut en effet prévoir une action spécifique qui est capable de gérer ces cas (cf. chapitre 8, p. 129 pour plus de détails).

Enfin, la largeur arborescente est sans doute la métrique qui donne le plus d'informations. Elle indique, par exemple, qu'en moyenne les graphes prédicat-argument sont relativement proches d'arbres. En effet, une largeur arborescente nulle signifie que le graphe est un arbre. Nos graphes ayant une largeur arborescente moyenne d'environ 1.50, nous pouvons conclure qu'en moyenne, nous sommes très proche d'arbres. De plus, cette propriété est souvent utilisée en théorie des graphes pour les schémas d'approximation polynomiaux (Williamson et Shmoys, 2011) : quand la largeur arborescente est connue, elle est fixée comme constante et il est alors possible de proposer des algorithmes polynomiaux pour des problèmes NP-difficiles. Au vu des largeurs arborescentes que nous considérons, nous pouvons espérer que les algorithmes NP-difficiles que nous utilisons seront suffisamment rapides en pratique.

Cette propriété n'est pas anodine puisqu'elle conditionne la viabilité de notre approche de réécriture. Nous souhaitons développer des méthodes efficaces, que ce soit en terme de qualité de production (maximiser le F_1-score sur un corpus) ou en terme de rapidité d'exécution. Or, l'isomorphisme de (sous)-graphes, qui est NP-complet, est le problème central de la réécriture de graphes, là où l'isomorphisme de (sous)-arbres a une solution en temps linéaire. Nos graphes étant en moyenne proches d'arbres, nous pouvons espérer, qu'en pratique, nos temps d'analyse soient quasi linéaires. De fait, les approches par réécriture de graphes sont une approche complémentaire au parsing en TAL.

Après avoir explicité certaines propriétés formelles de nos graphes prédicat-argument, nous nous attachons à décrire les approches algébriques de réécriture, car c'est le point de départ de notre travail sur la réécriture. Nous parlerons aussi succinctement des grammaires par remplacement d'hyperarcs (*hyperedge replacement grammars* ou HRG), seul formalisme autre que les approches algébriques à avoir été utilisé en Traitement Automatique des Langues (Jones et al., 2012; Chiang et al., 2013; Peng et al., 2015).

7.1.3 Réécriture de graphes et applications linguistiques

7.1.3.1 De l'approche par somme amalgamée double à l'approche par somme amalgamée simple

L'approche Double Pushout (DPO) ou, en français, par somme amalgamée double (Rozenberg, 1997) est très répandue dans les systèmes de réécriture de graphes. Elle fait partie d'un groupe plus général de théories appelées *approches algébriques* (Ehrig et al., 2006), fondées sur la théorie des catégories. [4] Intuitivement une règle de transformation est définie comme $p = L \leftarrow K \rightarrow R$, les flèches représentant des morphismes totaux sur les grpahes. Une règle consiste en trois graphes L, K et R. L est le graphe de partie gauche de la règle. Il formule la pré-condition selon laquelle la règle est applicable et R formule les post-conditions. K est l'intersection éventuellement vide entre L et R et dénote l'interface qui reste inchangée durant la transformation. K est souvent appelé l'*invariant* ou le *graphe de recollement* (*gluing graph*). De fait, $L - K$ (ou $L \backslash K$) est la partie qui sera supprimée du graphe d'entrée et $R - K$ (ou $R \backslash K$) la partie qui sera ajoutée. La règle p est alors applicable sur le graphe hôte G pour donner le graphe H en plusieurs étapes :

1. Trouver un morphisme m de L dans G. Intuitivement, L est un sous-graphe qui est reconnu (*matché*), modulo renommage via m, dans G.
2. Supprimer la partie de G qui correspond à $L \backslash K$ modulo le morphisme m. Cela nous donne une structure $D = G \backslash m(L \backslash K)$.
3. Construire $H = D \cup (R \backslash K)$.

K sert donc d'interface et contient les nœuds et les arcs qui seront préservés lors de l'application de la règle. Le graphe K est nécessaire pour attacher le motif *matché* à son contexte : s'il est vide, le *match* ne peut désigner qu'un composant connecté de G.

Ces étapes sont communes à toutes les approches de transformation de graphes. L'approche algébrique résume ces étapes en une simple construction d'un diagramme de somme amalgamée double (*double pushout*) qui facilite de nombreuses preuves (preuves qui auraient été complexes sans passer par la théorie des catégories). Plus formellement, une transformation directe de graphe avec p et m est définie comme suit. Étant donnée une production $p = \langle L \leftarrow K \rightarrow R \rangle$ et un graphe de contexte D qui inclut l'interface K, le graphe hôte G est défini comme $G = L +_K D$, c'est-à-dire que G est le recollement (*gluing*) de L et de D selon K (exprimé par $+_K$). Le graphe cible H est alors noté $H = R +_K D$. Plus précisément, les morphismes de graphe $K \rightarrow L$, $K \rightarrow R$ et $K \rightarrow D$ sont utilisés pour exprimer comment K est inclus dans L, R et D respectivement. Cela permet de définir les constructions de recollement $G = L +_K D$ et $H = R +_K D$ comme les sommes amalgamées (*pushouts*) (1) et

4. La théorie des catégories en mathématiques formalise la structure mathématique et ses concepts en terme de collections d'*objets* et de *flèches* (aussi appelées *morphismes*). Une catégorie a deux propriétés basiques : la possibilité de composer les morphismes de manière associative et l'existence d'un morphisme d'identité pour chaque objet. La théorie des catégories peut être utilisée pour formaliser des concepts comme les ensembles, les anneaux et les groupes. À noter que la catégorie des ensembles est sans doute la plus intuitive pour comprendre la notion de catégories. En effet, un ensemble est un objet dans la théorie et les flèches (ou morphismes) sont des fonctions (injective, surjective et bijective correspondant respectivement à un monomorphisme, épimorphisme et à un isomorphisme).

(2) sur la Figure 7.2, ce qui donne une somme amalgamée double (ou *double pushout*). Le morphisme de graphe résultant $R \rightarrow H$ est appelé co-image (*comatch*) de la transformation $G \Rightarrow H$.

Pour appliquer la production p avec une image m de L dans G (donné par un morphisme de graphe $m : L \rightarrow G$ à la Figure 7.2), il faut en premier lieu construire le graphe de contexte D tel que le recollement $L +_K D$ de L et D via K est égal à G. Dans un second temps, il faut construire le recollement $R +_K D$ de R et D via K qui génère alors le graphe cible H. On a donc une transformation par somme amalgamée double via p et m.

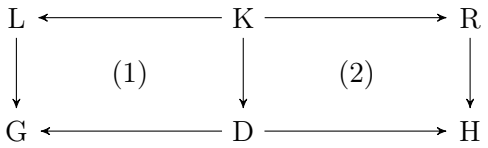


Figure 7.2: Approche par somme amalgamée double (Approche *double pushout*).

Pour le moment, nous n'avons décrit que l'approche DPO. Mais, une autre approche qui ne requiert pas l'explicitation de l'interface K existe : l'approche par somme amalgamée simple (*(*single pushout, SPO). Le passage d'une approche à l'autre est triviale : en effet, la production $p = \langle L \leftarrow K \rightarrow R \rangle$ peut aussi être considérée comme un morphisme de graphe partiel $p : L \rightarrow R$. De fait, l'étape de réécriture n'est plus définie que par une somme amalgamée simple (Fig. 7.3) représente ces opérations. Les morphismes horizontaux sont partiels et les verticaux sont totaux. L'approche SPO peut alors être vue comme une généralisation de l'approche DPO.

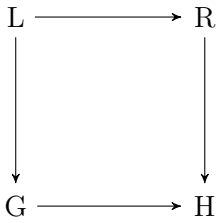


Figure 7.3: Approche par somme amalgamée simple (Approche *single pushout*).

7.1.3.2 Utilisation en Traitement Automatique des Langues

GREW (Bonfante et al., 2010, 2011a) est un système de réécriture de graphes fondé sur l'approche SPO. Il est structuré en modules, car le système n'est pas globalement confluent. La modularisation est donc utilisée pour garantir cette propriété au sein de chaque module et assurer une cohérence dans le cadre de traitements linguistiques. GREW a notamment été utilisé pour diverses applications linguistiques :

1. La transformation d'analyses syntaxiques en dépendances vers des graphes sémantiques (transformation en Dependency Minimal Recursion Semantics DMRS, une extension de la MRS (Copestake et al., 2005)).

2. L'ajout d'informations en syntaxe profonde sur des analyses syntaxiques de surface.
3. La pré-annotation du corpus DeepSequoia (Candito et al., 2014b; Perrier et al., 2014).

D'un arbre syntaxique en dépendances à un graphe sémantique

Bonfante et al. (2010) s'intéressent au passage d'un arbre syntaxique à un graphe sémantique en enrichissant le schéma d'annotation passage [5] (Paroubek et al., 2009) puis de transformer l'arbre en une structure dmrs qui décrit un graphe dont les nœuds sont des prédicats et les arcs des relations entre ces prédicats. La relation est précisée par une étiquette apposée sur l'arc. Pour réussir la réécriture, Bonfante et al. (2010) ajoutent une surcouche au format passage qu'ils nomment dsc (dépendances syntaxiques complètes) afin de se rapprocher le plus possible du graphe d'arrivée et d'avoir suffisamment d'informations pour arriver à produire le dmrs. C'est une sorte de niveau intermédiaire entre la surface et la dmrs qu'il est possible de comparer au niveau profond. À titre indicatif, nous décrivons ici certains des ajouts effectués au schéma d'annotation passage :

— Pour modéliser la quantification, ils ajoutent la relation det entre le déterminant et le nom.
— Pour les propositions relatives, ils ajoutent la relation ant entre le pronom relatif et son antécédent, puisque cette relation dépend uniquement de la syntaxe.
— Ils ajoutent la relation aux ou passif en fonction de l'auxiliaire.
— Ils gèrent le contrôle sujet au moyen de Dicovalence (Eynde et Mertens, 2006).

Les règles mises en place correspondent toutes à des phénomènes linguistiques fortement représentés en corpus. Une fois les dépendances syntaxiques complètes recouvrées, les règles de transformer vers la dmrs sont en général assez simples, montrant que le niveau profond se rapproche de celui de la dmrs.

Annotation d'un corpus arboré en syntaxe profonde Bonfante et al. (2011b) s'intéressent à l'enrichissement d'un corpus arboré surfacique, le **French Treebank en dépendances** (dont l'annotation est décrite au chapitre 5 (p. 61) et diffère largement du schéma d'annotation passage) (Abeillé et al., 2003; Candito et al., 2009), pour produire une représentation en syntaxe profonde. Les règles utilisées sont de quatre types :

1. Règles grammaticales d'actants (elles déterminent syntaxiquement les sujets des infinitives, des participes et des adjectifs).
2. Règles lexicales d'actants (extraites de Dicovalence (Eynde et Mertens, 2006) qui déterminent les sujets ou objets d'infinitifs compléments de verbes ou d'adjectifs à contrôles).

5. passage est un schéma d'annotation en groupes syntaxiques (*chunks*) reliés par des relations syntaxiques. Le schéma propose 6 types de groupes syntaxiques (groupe nominal, prépositionnel, adjectival, adverbial, verbal introduit par une préposition et le noyau verbal). Il propose 14 relations syntaxiques telles que la relation d'auxiliaire, de sujet, d'objet, d'attribut du sujet ou de l'objet, etc. passage reprend des idées développées dans gr (Carroll et al., 1999) et poursuit en partie les mêmes buts : l'évaluation d'analyseurs syntaxiques issus de théories diverses.

3. Règles d'antécédents de la relative.
4. Règles de coordinations (partage du sujet et intéraction avec la montée et le contrôle essentiellement).

Nous ne développerons pas ici toutes les règles définies et leur application, nous allons simplement nous concentrer sur les phénomènes complexes, tels que l'antécédent des relatives et la coordination.

Récupération de l'antécédent de la relative. Le problème est non-trivial comme le démontre Bonfante et al. (2011b). En effet, l'annotation du French Treebank en dépendances donne un lien MOD.REL entre l'antécédent et le verbe et un lien X entre le relatif et un autre mot rattaché via un nombre n d'arcs successifs au verbe (en bleu sur la Figure 7.4). Il existe donc un chemin de longueur k non connu à l'avance. Le seul moyen de pouvoir retrouver le relatif et son antécédent est alors d'avoir une règle (Figure 7.5) où l'antécédent est lié au pronom relatif par un chemin de taille k avec un arc étiqueté MOD.REL.

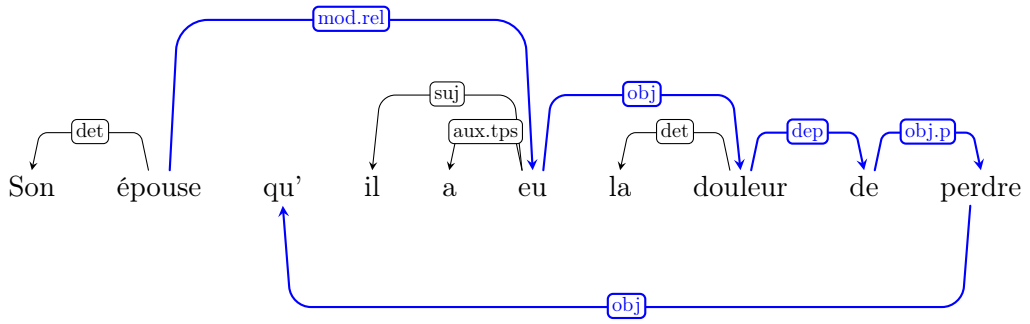


Figure 7.4: Exemple de relative dans le French Treebank.

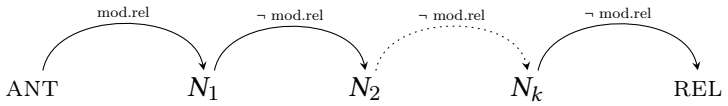


Figure 7.5: Représentation d'une règle généraliste pour la relative et son antécédent.

Cependant, cette règle n'est plus locale et pour un système comme GREW le principe de localité est important afin d'optimiser les calculs. D'autre part, une telle règle ne prend pas en compte les contraintes d'îlots [6]. Pour pallier ce problème, GREW utilise quatre règles différentes qui sont une décomposition de la règle présentée à la figure 7.5. Ces règles s'appliquent dans un ordre précis et autant de fois que nécessaire pour remonter à l'antécédent et pouvoir créer un arc ANT.REL entre le pronom relatif et l'antécédent. La démarche n'est pas sans poser problème : quatre règles sont en effet nécessaires pour retrouver l'antécédent d'un relatif et l'ordre d'application des règles est extrêmement contraint. Il faut une règle récursive qui s'applique si et seulement si une autre règle ne peut s'appliquer. Ce type

6. Voir (Ross, 1967) et (Kahane, 2000) pour plus de détails sur ces questions.

d'application ne permet pas une souplesse dans l'écriture des règles, bien qu'elle soit assez efficace pour repérer ce type de cas.

Coordination et partage de sujets. grew est obligé de traiter ce phénomène dans tous les modules. Il y a en effet une interaction forte entre le sujet des infinitives (dû au contrôle ou à la montée) et un partage du sujet (Exemple 7.1).

(7.1) Jean pense partir aujourd'hui et **pouvoir** rentrer demain [7].

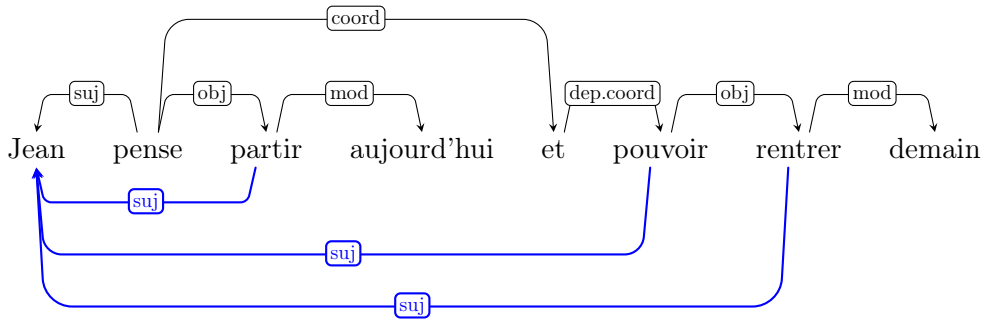


Figure 7.6: Représentation profonde de l'exemple 7.1.

Il faut déterminer le sujet de « partir » avant d'en déduire que « pouvoir » partage ce sujet. Cela incite donc à appliquer les règles de coordination après le module sujet. Cependant, pour déterminer le sujet de « rentrer », il est nécessaire de connaître le sujet de « pouvoir ». De ce point de vue, il faudrait donc appliquer les règles relatives à la coordination avant le module sujet. Les auteurs particularisent chaque règle de coordination dans chaque module pour pouvoir traiter ce genre de phénomènes, ce qui amène à ajouter autant de règles de coordinations qu'il y a de modules et de types de coordination. Encore une fois la gestion des règles est complexe des règles est complexe, ce qui pose des problèmes au niveau de la maintenabilité du système.

Pour finir, Bonfante et al. (2011b) donne une idée succincte de la pertinence de grew grâce à une analyse d'erreurs qui indiquent que nombre d'entre elles résident dans l'annotation automatique des antécédents des relatives. Pour le reste, les auteurs imputent ces erreurs à des problèmes d'annotation dans le French Treebank ou des manques lexicaux dans Dicovalence (Eynde et Mertens, 2006) – utilisé pour l'ajout de données morphologiques et des phénomènes de montée et de contrôle.

Pré-annotation du DeepSequoia. Les travaux sont les précurseurs des travaux sur l'annotation en syntaxe profonde du corpus Séquoia. grew a été utilisé pour faciliter l'annotation du DeepSequoia en réutilisant et complétant les règles préalablement écrites pour pré-annoter les phénomènes les plus massifs dans le corpus (sujets finaux des verbes non

7. Cette construction est particulièrement difficile, car c'est une factorisation de syntagmes verbaux. L'annotation en syntaxe profonde requiert la gestion de règles pour le contrôle, la montée mais aussi pour le partage du sujet.

finis, changements de diathèse, etc.). La pré-annotation a demandé un total d'environ 290 règles de réécriture et donne de très bons scores (Table 7.3) qui sont en rapport avec les méthodes de même catégorie comme l'annotation en F-structure du MFT (Schluter et van Genabith, 2007) produisant une annotation automatique avec un F_1-score de 96% (Schluter et van Genabith, 2008).

	LP	LR	LF	UP	UR	UF
Arcs profonds	98.43	97.71	98.07	99.63	98.90	99.26

Table 7.3: Scores de pré-annotation du DeepSequoia pour grew (Précision, rappel, F_1-score étiqueté (LP, LR, LF) et non étiquetés (UP, UR, UF)).

La pré-annotation du DeepSequoia a été menée au sein des deux équipes en charge du projet. De fait, grew est le système utilisé à Nancy, alors qu'ogre que nous présentons maintenant est celui qui a été utilisé à Paris.

7.1.3.3 Optimized Graph Rewriting Engine

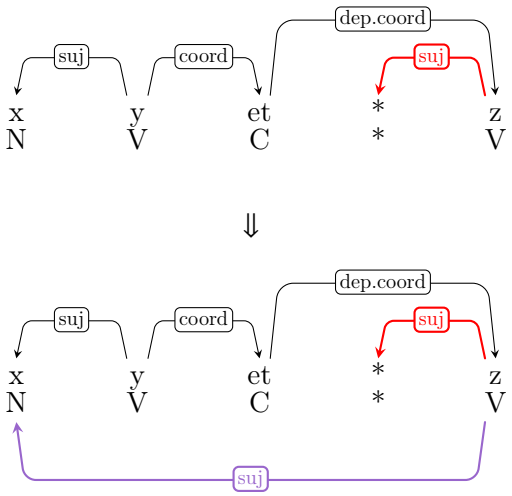
ogre (Optimized Graph Rewriting Engine) est un système de réécriture de graphes développé dans le cadre de cette thèse avec pour ambition de pallier au maximum les problèmes récurrents des systèmes de réécriture classiques, à savoir la confluence et la maintenabilité des règles écrites à la main. Pour ce faire, le système divise le travail de réécriture en deux phases : une phase similaire à l'approche spo (et à grew) et une seconde phase de propagation d'informations dans le graphe d'entrée. [8]

Description d'une règle de réécriture. ogre s'éloigne de l'approche traditionnelle en matière de règles de réécriture pour des raisons de simplicité d'utilisation et d'efficacité. Une règle de réécriture est donc subdivisée en quatre structures distinctes décrites ci-dessous. Nous donnons un exemple de règle à la Figure 7.7 ainsi que son application sur un sous-graphe.

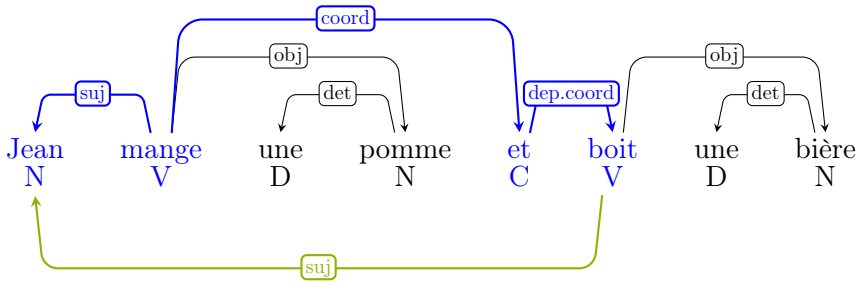
Il y a donc quatre structures différentes pour chaque règle :

1. Le motif de graphe (ou *match*) à trouver au sein du corpus à réécriture. C'est la seule structure obligatoire.
2. Les commandes de réécriture qui permettent d'ajouter, de modifier ou de supprimer un arc ou un nœud.
3. Les conditions négatives d'application (*negative application conditions*) (Habel et al., 1996) qui permettent de modifier le motif en ajoutant des contraintes : recherche d'un sous-graphe représentant une coordination pour laquelle le premier conjoint porte un sujet **mais pas le second**. La partie en gras représente la condition d'application négative.

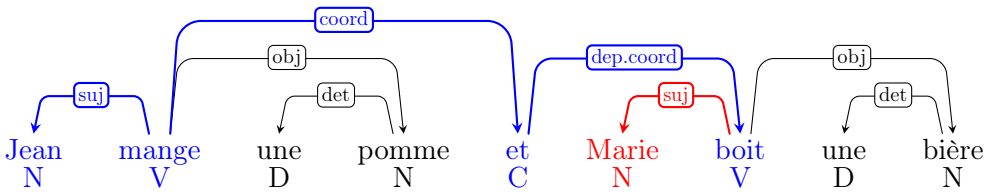
8. La deuxième phase du système est inspirée d'idées issues de l'écriture de règles pour la transformation des dérivations tag de FrMG (Villemonte de La Clergerie, 2010) vers des dépendances surfaciques suivant le schéma d'annotation du French Treebank en dépendances (Candito et al., 2010).



(a) Règle de réécriture avec contrainte négative d'application en rouge et ajout d'arc en violet. x, y, z, *et* sont des noms de variables (une variable $*$ est anonyme). Les catégories morpho-syntaxiques restreignent le *matching* en imposant que x, y, z et *et* portent bien ces catégories dans le graphe hôte. $*$ est un joker signifiant que la catégorie n'importe pas.



(b) Graphe du corpus où la règle peut s'appliquer.



(c) Graphe du corpus où la règle ne peut pas s'appliquer (à cause de la contrainte négative d'application illustrée en rouge).

Figure 7.7: Règle d'OGRE et exemple d'application. La règle cherche un sous-graphe représentant potentiellement un partage du sujet (comme dans l'exemple 7.7b). La condition négative impose que le deuxième conjoint de la coordination n'ait pas déjà un sujet. Le matching est représenté en bleu et la condition négative en rouge. L'arc ajouté après application de la commande de réécriture est représenté en vert.

4. Les déclencheurs que nous décrivons en détails ci-dessous.

Ce découpage permet une grande flexibilité dans la mesure où le système peut être utilisé aussi bien pour de la réécriture que pour du *matching* simple dans des arbres et des graphes de dépendances. Par exemple, il est possible de l'utiliser pour faire de l'extraction d'informations dans des graphes prédicat-argument en cherchant un motif particulier à extraire. De fait, OGRE est sur les arbres et les graphes, ce que Unitex (Paumier, 2006; Paumier et Martineau, 2015) est aux chaînes de caractères.

Première phase de réécriture : une adaptation de l'approche SPO. La première phase est similaire à l'approche SPO, à ceci près que les règles ne sont appliquées qu'une et une seule fois et ce, sur le **graphe d'entrée G_0 seulement**. Cela permet de garantir la terminaison du système, les règles ne s'appliquant qu'une fois, il est certain que le système termine. La réécriture s'effectue alors comme suit. Soit $\mathcal{R}$ un ensemble de règles de réécriture $L \rightarrow R$ avec L la partie gauche de la règle et R la partie droite de la règle comme pour l'approche SPO. Pour chaque règle $r \in \mathcal{R}$ et un ensemble $\mathcal{M}_r$ de morphismes m de L dans G_0, définissant les points d'application de la règle dans le graphe G_0, en fonction de L, nous définissons trois ensembles :

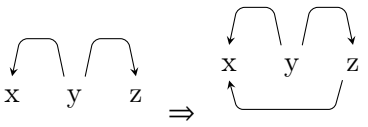
1. $C_r = L_r \cap R_r$, l'ensemble des éléments communs à la partie gauche et la partie droite : l'interface.
2. $A_{r,m} = m(R_r \backslash C_r)$, l'ensemble des éléments qu'il faut ajouter à G_0 avec $m \in \mathcal{M}_r$.
3. $S_{r,m} = m(L_r \backslash C_r)$, l'ensemble des éléments qu'il faut supprimer de G_0, avec $m \in \mathcal{M}_r$.

Une fois ces ensembles définis, il est possible de dénoter le graphe $G_{\mathcal{R}}$, comme le graphe résultant de l'application de tous les $A_{r,m}$ et de tous les $S_{r,m}$ pour toutes les règles $r \in \mathcal{R}$ et tous les morphismes $m \in \mathcal{M}_r$ et . Plus précisément, nous définissons :

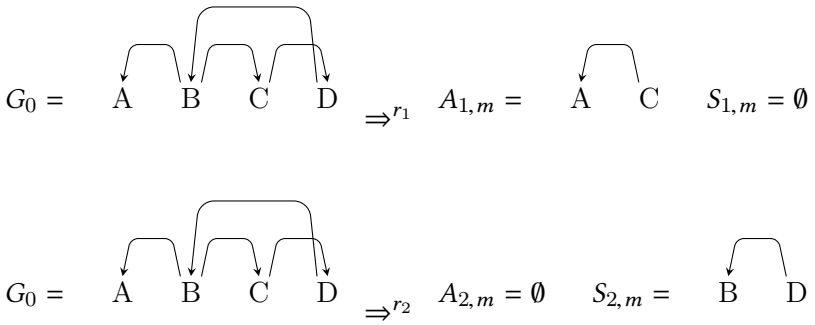
1. $A = \bigcup_{\substack{r \in \mathcal{R} \\ m \in \mathcal{M}_r}} A_{r,m}$, comme l'ensemble de tous les éléments qu'il faut ajouter à G_0.
2. $S = \bigcup_{\substack{r \in \mathcal{R} \\ m \in \mathcal{M}_r}} S_{r,m}$, comme l'ensemble de tous les éléments qu'il faut supprimer de G_0.

Le graphe de sortie $G_{\mathcal{R}} = (G_0 \cup A) \backslash S$, c'est-à-dire l'union du graphe d'entrée et de tous les éléments à ajouter auquel nous retranchons tous les éléments à supprimer. Lors de la réécriture, il est possible que deux règles r_1 et r_2 soient contradictoires (c'est-à-dire que l'une ajoute un arc et l'autre l'enlève). Cependant, la sémantique définie ici force une certaine interprétation. Il est bien entendu possible de définir un autre type de fusion pour obtenir $G_{\mathcal{R}}$. Dans le cadre qui nous intéresse (la pré-annotation d'un corpus en syntaxe profonde), les suppressions sont rares et les conflits entre les règles sont inexistants. Un exemple d'application est donné à la Figure 7.8.

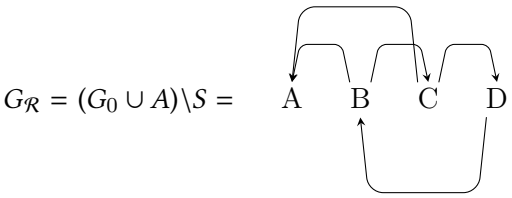
Cependant, la première phase du système est trop restrictive pour l'annotation linguistique recherchée. En effet, il peut être utile de s'appuyer sur une réécriture faite précédemment sur le graphe pour en effectuer une nouvelle. C'est par exemple le cas à la Figure 7.5 où l'arc se déplace dans le graphe par réécriture successive. Dans le cas de l'annotation en



(a) Règle r_1 qui ajoute un arc entre x et z. (b) Règle r_2 qui supprime un arc entre u et w.



(c) Application des deux règles.



(d) Graphe de sortie G_n.
$A = A_{1,m} \cup A_{2,m} = A_{1,m}$ et $S = S_{1,m} \cup S_{2,m} = S_{2,m}$

Figure 7.8: Processus de réécriture avec un ensemble de deux règles $\mathcal{R} = \{r_1, r_2\}$ sur le graphe hôte G_0 pour produire le graphe de sortie $G_{\mathcal{R}}$.

syntaxe profonde, ce phénomène de propagation d'informations n'est valable que sur les arcs. C'est la raison pour laquelle, il n'est pas nécessaire d'avoir un système de réécriture général (c'est-à-dire qui puisse opérer des réécritures complexes sur les nœuds et les arcs d'un même graphe). En effet, dans un souci d'analyse automatique, on cherche surtout à prédire des dépendances entre les mots et donc la réécriture d'arcs est plus importante que la réécriture de nœuds. Il est donc intéressant d'explorer une phase qui n'opérerait que sur les arcs du graphe et permettrait cette propagation d'informations.

Système de propagation. La deuxième phase de notre système de réécriture est une phase de propagation d'informations dans le graphe en se fondant exclusivement sur des transformations effectuées sur les arcs. La propagation d'informations est une tâche importante lors du passage d'un niveau surfacique à un niveau profond. Pour en donner une bonne

intuition, nous allons considérer deux exemples issus du schéma d'annotation du DeepSequoia. Le premier exemple, illustrant une cascade de verbes à contrôle et à montée, est donné à la Figure 7.9. Le second exemple, illustré à la Figure 7.10, est un exemple de propagation des informations dans le cadre d'une préposition sémantiquement vide.

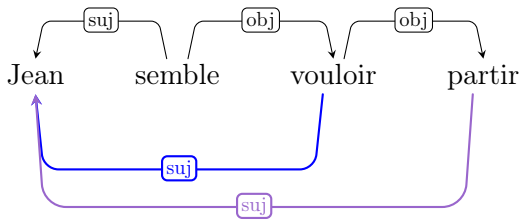


Figure 7.9: Cascade de verbes à contrôle et à montée.
L'arc bleu correspond à la première étape de propagation et l'arc violet à la seconde.

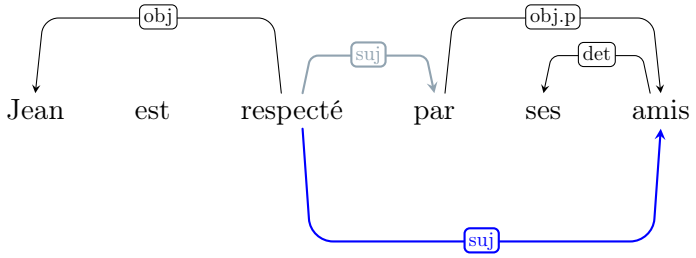


Figure 7.10: Déplacement sur un mot sémantiquement plein.
L'arc bleu est l'arc ajouté par propagation. Dans le cadre d'un déplacement, l'arc censé être déplacé prend la couleur grise pour des raisons de confluence expliquées plus loin dans la section.

Dans les exemples précédents se dessinent la notion de propagation d'informations sur les arcs. L'exemple à la Figure 7.9 montre bien l'ajout **successif** des arcs SUJ entre les différents verbes à l'infinitif et ce grâce à l'existence d'un arc SUJ entre le verbe de la principale et *Jean*. Ce phénomène est massif dans les tâches linguistiques qui nous intéressent et ne touchent que les arcs du graphe et non les nœuds. C'est pourquoi, il nous a semblé judicieux d'exploiter cette propriété.

Pour ce faire, le système de propagation utilise des *déclencheurs*. Intuitivement, un déclencheur est porté par un arc et permet l'exécution d'une action (ajout, déplacement, suppression d'un arc) en fonction d'une configuration dans le graphe. Un exemple est donné à la Figure 7.11 [9] où trois déclencheurs *share down* marqués $s_{\downarrow}$(suj) ont été ajoutés au graphe d'entrée. Le déclencheur $s_{\downarrow}$(suj) entre *semble* et *vouloir* (en orange), ajoute un arc SUJ entre *Jean* et *vouloir* (en orange) suite à la présence de l'arc étiquete $s_{\downarrow}$(suj). De même, puisqu'il existe un déclencheur $s_{\downarrow}$(suj) entre *vouloir* et *diriger* (en violet), alors un arc SUJ est ajouté entre *Jean* et *diriger*. Il en va de même pour le déclencheur en bleu sur la Figure 7.11. Les déclencheurs sont créés lors de la première phase de réécriture. La description d'une règle inclut en effet une structure particulière qui définit sur quels arcs les déclencheurs doivent être posés.

9. C'est une variation de l'exemple à la Figure 7.9 montrant ainsi la complexité que peut prendre la

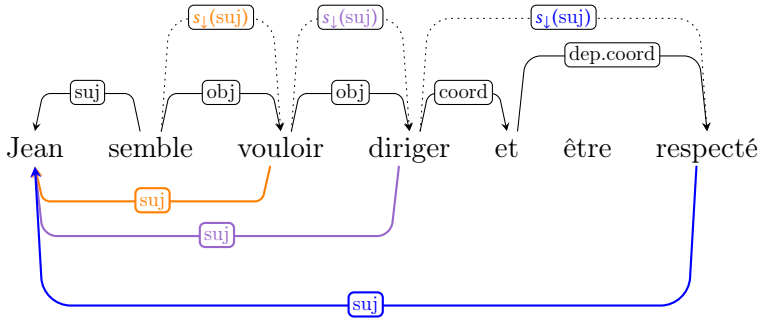


Figure 7.11: Système de propagation via des déclencheurs.

Plus généralement, les déclencheurs peuvent être considérés comme des méta-règles qui permettent de générer une grammaire par remplacement d'arcs (*edge replacement grammar*) (Drewes et al., 1997) (voir aussi section 7.1.4, p. 115). Chaque application d'un déclencheur est une étape de dérivation. L'avantage de cette approche est comparable à l'écriture d'une méta-grammaire (Candito, 1996) qui évite de multiplier le nombre de règles, car un arc portant un déclencheur correspond à un ensemble de règles ou d'application de règles. Pour garantir de bonnes propriétés, le système impose que les arcs du graphe soient bicolore, c'est-à-dire qu'il existe un ensemble $C = \{\bullet, \bullet\}$ et que chaque arc du graphe soit alors soit de couleur noire ($\bullet$), soit de couleur grise ($\bullet$). En effet, les arcs de couleur grise seront supprimés à la fin de la propagation. Ensuite, le système définit l'ensemble de déclencheurs suivants : $D_L = \{s_\downarrow(\mathrm{l}), s_\uparrow(\mathrm{l}), r_\downarrow(\mathrm{l}), \vec{a}(\mathrm{l}) \mid l \in L\}$ avec L les étiquettes du graphe. Ainsi, il est possible de définir un p-graphe étiqueté coloré à déclencheurs.

Définition 7.7 (p-graphe étiqueté coloré à déclencheurs)**.**
Un p-graphe étiqueté coloré à déclencheurs est un p-graphe étiqueté $G = \langle V, E, L' \rangle$ avec :

- V l'ensemble des nœuds du graphe.
- $L' = L \cup D_L$ l'ensemble d'étiquettes représentant les relations de dépendances et les déclencheurs.
- $E \subseteq V \times L' \times C \times V$ est un ensemble d'arcs, où chaque arc est un quadruplet ordonné (u, l', c, v) aussi dénoté par $u \xrightarrow{l', c} v$.

Le p-graphe contient deux contraintes supplémentaires :

1. tout arc déclencheur soit grisé, $\forall e = (u, l', c, v) \in E,\ l' \in D_L \implies c = \bullet$.
2. il n'est pas possible d'avoir un même arc e de couleurs différentes (si $(u, l, c, v) \in E$ et $(u, l, c', v) \in E$ alors $c = c'$).

Ensuite, soit $\sim$ la projection du graphe $G = \langle V, E, L' \rangle$ vers un graphe $\widetilde{G} = \langle V, \widetilde{E}, L' \rangle$, tel que $(u, l', v) \in \widetilde{E}$ existe si $(u, l', c, v) \in E$ est un arc de G. $\widetilde{G}$ est donc une projection de G

propagation d'informations.

sans information de couleur. Notons que, d'après les contraintes sur le p-graphe étiqueté et coloré de la définition 7.7, nous avons $|\widetilde{G}| = |G|$.

Enfin, nous définissons $G_{\bullet} = \langle V, E_{\bullet}, L' \rangle$, avec $E_{\bullet} = \{(u, l', c, v) \in E \mid c = \bullet\}$ (resp. $G_{\bullet} = \langle V, E_{\bullet}, L' \rangle$, avec $E_{\bullet} = \{(u, l', c, v) \in E \mid c = \bullet\}$), comme le sous-graphe de G pour lequel tous les arcs sont noirs (resp. gris).

Avant propagation, tous les arcs de E qui possède une étiquette issue de L sont noirs ($\bullet$). Un arc gris ($\bullet$) indique qu'il doit être supprimé à la fin des étapes de propagation : soit cet arc porte un déclencheur, soit il a été modifié par un déclencheur. [10]

Descriptions des déclencheurs. Certains déclencheurs peuvent transformer un arc noir en arc gris, indiquant qu'il doit disparaître à la fin de la propagation. L'utilisation de deux couleurs est la condition *sine qua none* pour garantir la confluence du système de propagation tout en proposant des déclencheurs qui peuvent déplacer ou supprimer des arcs. Chaque déclencheur est associé à une règle de réécriture $r : L \to K$ et possède les propriétés suivantes :

1. Il ne peut pas être déplacé d'un arc à un autre.
2. Il ne peut pas ajouter de nœuds.
3. Il ne peut pas supprimer d'arcs.
4. Son domaine d'application est restreint : sur l'arc lui-même ou sur les arcs voisins.

Nous utilisons quatre types de déclencheurs associés à leur règle de réécriture [11] :

1. *share down* $s_{\downarrow}(\mathrm{l})$: si $y \xrightarrow{s_{\downarrow}(\mathrm{l})} z$ alors :
   - $L = \left\langle \{x, y, z\}, \left\{ y \xrightarrow{l,\bullet} x \right\} \right\rangle$.
   - $R = \left\langle \{x, y, z\}, \left\{ y \xrightarrow{l,\bullet} x, z \xrightarrow{l,\bullet} x \right\} \right\rangle$.

   Ce déclencheur est utilisé à la Figure 7.11 et illustré à la Figure 7.12a. L'idée est de dupliquer un arc en fonction de l'étiquette qu'il possède. C'est utile dans le cadre de l'interaction entre les verbes à contrôle/montée et la coordination par exemple.
2. *share up* $s_{\uparrow}(\mathrm{l})$: si $x \xrightarrow{s_{\uparrow}(\mathrm{l})} y$) alors :
   - $L = \left\langle \{x, y, z\}, \left\{ y \xrightarrow{l,\bullet} z \right\} \right\rangle$.
   - $R = \left\langle \{x, y, z\}, \left\{ y \xrightarrow{l,\bullet} z, x \xrightarrow{l,\bullet} z \right\} \right\rangle$.

   Ce déclencheur est illustré à la Figure 7.12b. Il est similaire à $s_{\downarrow}(\mathrm{l})$ dans la mesure où un arc est dupliqué en fonction de l'étiquette qu'il possède, seule change la configuration du sous-graphe d'entrée.

10. À noter que la couleur grise est reprise des grammaires unifiées de polarité (Kahane et Lareau, 2005), dans lesquelles elle indique une polarité grise qui marque qu'un arc est invisible à la fin des étapes de dérivation. Une alternative possible aurait été de remplacer l'information de couleurs par la gestion d'un ensemble d'arcs à éliminer.

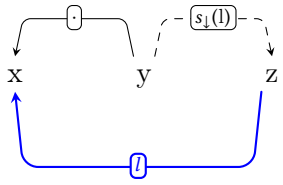
11. En réalité, seuls $s_{\downarrow}()$, $s_{\uparrow}()$ et $\vec{a}()$ ont été utilisés pour pré-annoter le DeepSequoia.

3. *redirect down* $r_{\downarrow}(\mathrm{l})$: si $y \xrightarrow{r_{\downarrow}(\mathrm{l})} z$) alors :
   - $L = \left\langle \{x, y, z\}, \left\{ x \xrightarrow{l,\bullet} y \right\} \right\rangle$.
   - $R = \left\langle \{x, y, z\}, \left\{ x \xrightarrow{l,\bullet} y, x \xrightarrow{l,\bullet} z \right\} \right\rangle$.

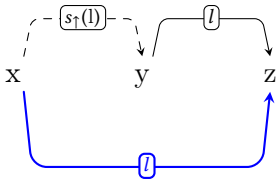
   Ce déclencheur est illustré à la Figure 7.12c. Il permet de déplacer un arc en fonction de son étiquette, cela s'avère utile lorsqu'un nœud est considéré comme sémantiquement vide, mais qu'il n'est pas souhaitable de perdre ses arcs entrants. Ils sont donc redirigés vers les tokens sémantiquement pleins, comme le fait le DeepSequoia (cf. chap. 6, p. 6).

4. *multi-edge* $\vec{a}(\mathrm{l})$: si $x \xrightarrow{\vec{a}(\mathrm{l})} x$) alors :
   - $L = \left\langle \{x, y\}, \left\{ x \xrightarrow{l_0,\bullet} y \right\} \right\rangle$ avec $l_0 \neq l$.
   - $R = \left\langle \{x, y\}, \left\{ x \xrightarrow{l_0,\bullet} y, x \xrightarrow{l,\bullet} y \right\} \right\rangle$.

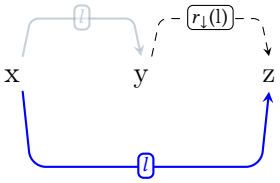
   Ce déclencheur est illustré à la Figure 7.12d. Il permet d'ajouter un arc d'étiquette différente entre deux nœuds. Ce processus est intéressant pour modéliser une représentation multi-strates à la manière des fonctions finales et des fonctions canoniques dans l'annotation du DeepSequoia ou encore des fonctions analytiques et tectogrammaticales pour le Prague Dependency Treebank (cf. sec. 3.1.1 p. 44).



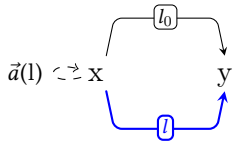
(a) Déclencheur *share down* $s_{\downarrow}(\mathrm{l})$.



(b) Déclencheur *share up* $s_{\uparrow}(\mathrm{l})$.



(c) Déclencheur *redirect down* $r_{\downarrow}(\mathrm{l})$ (l'arc entre x et y est grisé).



(d) Déclencheur *multi-edge* $\vec{a}(\mathrm{l})$.

Figure 7.12: Types de déclencheurs (déclencheur en pointillés, modifications en bleu).

Comme nous le disions ci-dessus, les déclencheurs évitent de devoir multiplier le nombre de règles. Ceci assure une meilleure maintenabilité du système de réécriture. Mais ce n'est pas le seul avantage : les propriétés de **terminaison** et de **confluence** sont aussi garanties. Soit G_0 le graphe étiqueté coloré en entrée, D_L l'ensemble de déclencheurs défini page 110 et G_s le graphe étiqueté et coloré en sortie. De plus, nous définissons une étape de dérivation i comme l'étape résultant de l'application **effective** d'un déclencheur d sur le graphe G_i (c'est-à-dire que d ajoute un arc au graphe).

Lemme 7.1 (Monotonie croissante)**.**
Pour tout graphe G_i à l'étape i de dérivation, nous avons :

1. *Stabilité sur les nœuds de G_i ($V_i = V_{i+1}$).*
2. *Monotonie croissante stricte sur les arcs de $\widetilde{G}_i$ ($\widetilde{E}_i \subset \widetilde{E}_{i+1}$).*
3. *Monotonie croissante sur les arcs de $G_{\bullet i}$ ($G_{\bullet i} \subseteq G_{\bullet i+1}$).*

Stabilité sur les nœuds. Cette propriété est déduite naturellement de la définition des déclencheurs (définition 7.1.3.3, p. 111) : aucun déclencheur d ne peut ajouter de nœuds dans le graphe, donc le nombre de nœuds reste stable durant toute la séquence de dérivation.

Monotonie croissante sur les arcs grisés $E_{\bullet n} \in G_{\bullet n}$. Les déclencheurs sont définis de sorte qu'ils ne changent que les arcs noirs en arcs gris (et non l'inverse). De fait, soit $E_{\bullet i}$, l'ensemble des arcs gris à chaque étape de dérivation i, à l'étape $i+1$, nous avons $E_{\bullet i} \subseteq E_{\bullet i+1}$.

Monotonie croissante stricte sur les arcs $\widetilde{E}_n \in \widetilde{G}_n$. D'après la définition d'un déclencheur, il n'est pas possible de supprimer d'arcs. De fait, soit $\widetilde{E}_i$ l'ensemble des arcs à l'étape de dérivation i, à l'étape $i+1$, $|\widetilde{E}_{i+1}| > |\widetilde{E}_i|$. Plus précisément, chaque déclencheur $d \in D_L$ ajoute un arc e à $\widetilde{G}_i$, de fait $\widetilde{E}_{i+1} = \widetilde{E}_i \cup \{e\}$.

Ces trois propriétés de monotonie nous permettent alors de démontrer ce lemme.

Proposition 7.1 (Terminaison)**.**
Il n'existe pas de séquences infinies de dérivation partant de G_0.

Preuve. Nous supposons qu'une séquence infinie S_∞ de dérivation existe pour n'importe quel G_0. Dans un premier temps, nous nous occupons de la projection $\widetilde{G}_0$ du graphe G_0. D'après le lemme 7.1, nous savons que l'ensemble de nœuds est stable et égal à V_0. Le nombre de graphes qu'il est possible de construire sur V_0 est fini et est de taille N, avec $N < \infty$.

Par ailleurs, le nombre d'arcs dans le graphe $\widetilde{G}_0$ croît strictement. De fait, $\widetilde{G}_n \neq \widetilde{G}_i$, $\forall i < n$. De fait, à N étapes de dérivation, le graphe $\widetilde{G}_s$ peut avoir au plus $|\widetilde{E}_0| + N$ arcs. Il est donc fini et la séquence de dérivation aussi.

Un raisonnement similaire peut être fait pour tous les graphes $G_{\bullet n}$, il n'existe donc pas de séquences infinies de dérivation partant de G_0.

Proposition 7.2 (Confluence)**.**
Pour un G_0 donné, il existe un et un seul G_s, quelque soit l'ordre d'application des déclencheurs de D_L.

Preuve. Il faut montrer via une étude par cas que pour chaque paire de déclencheurs (a, b), l'application de a puis b donnant le graphe $G_{a,b}$ et identique à l'application de b puis a donnant le graphe $G_{b,a}$ (Figure 7.13). Autrement dit, il faut montrer que $G_{a,b} = G_{b,a}$.

Nous savons que l'application d'un déclencheur de D_L n'est pas fonction de la couleur de l'arc, de fait, il est possible de raisonner sur $\widetilde{G}$ sans perte de généralisation. De plus, les déclencheurs de D_L ne peuvent influer que sur les arcs non déclencheurs de $\widetilde{E}_L = \widetilde{E} \backslash \widetilde{E}_{D_L}$: l'ensemble des arcs qui sont étiquetés par L. Lors de l'application du déclencheur a, le

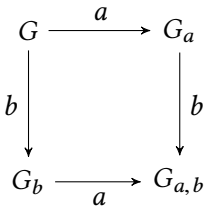


Figure 7.13: Diagramme commutatitif de l'application d'une paire (a, b) de déclencheurs.

lemme 7.1 montre que $\widetilde{E}_{L,i+1} = \widetilde{E}_{L,i} \cup \{e_a\}$, avec e_a l'arc ajouté par le déclencheur a. De fait, après l'application du déclencheur a, l'ensemble d'arcs $\widetilde{E}_{L,i}$ est inchangé. Il est donc possible d'appliquer le déclencheur b indépendamment de a.[12] Il est facile de vérifier que cette propriété est vraie pour chaque paire (a, b) de déclencheur, le système est donc confluent.

À la fin de la propagation, G_s est trivialement transformé en $G_{\bullet_s}$ pour ne conserver que les arcs noirs. Le système de propagation permet donc une gestion efficace et prouvée de la réécriture d'arcs via un algorithme de point fixe : tant que les déclencheurs s'appliquent, l'algorithme s'exécute. L'algorithme 1 présente la propagation.

Algorithme 1 Algorithme de point fixe pour la propagation.

```
1: Fonction PROPAGATION(G : graphe d'entrée, Déclencheurs : ensemble non ordonné de déclencheurs)
2:     G' ← ∅
3:     Répéter
4:         G' ← copie de G
5:         Pour d dans Déclencheurs Faire
6:             G ← APPLY(G, d)
7:         Fin Pour
8:     Jusqu'à ce que |G'| = |G|        ▷ |G| décrit la taille de G (i.e. son nombre d'arcs)
9: Fin Fonction
```

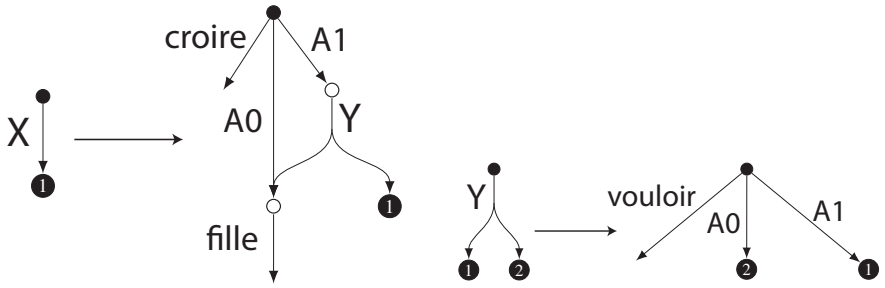
Le déclencheur d associé à la règle de réécriture $r : L \to R$ ne peut s'appliquer que si le sous-graphe L est trouvé dans le graphe d'entrée, la fonction Apply ne fait donc pas nécessairement de modifications sur le graphe G. Si l En tirant partie des propriétés connues des graphes que nous traitons, nous avons été capable de mettre au point un système pertinent, montrant de bonnes propriétés d'application qui permet de réduire le nombre de règles. ogre prend un parti pris de simplicité pour se limiter à des graphes linguistiques. Mais il est possible d'envisager l'utilisation d'approches plus puissantes dont certaines ont déjà été appliquées à des problèmes de Traitement Automatique des Langues.

12. À noter que la réciproque est trivialement vraie.

7.1.4 Approches complémentaires aux approches algébriques : les grammaires par remplacement d'hyperparcs

L'une des autres approches utilisée en TAL pour l'analyse de graphes via des grammaires s'appuie sur les grammaires de remplacement d'hyperarcs (ou *hyperedge replacement grammars* HRG). Ce formalisme se veut une transposition aux graphes des grammaires hors contexte. En effet, dans une grammaire hors contexte, les non-terminaux sont des symboles devant être réécrits dans le but de former une chaîne de caractères par dérivations. Dans une HRG, les non-terminaux sont des hyperarcs et les étapes de réécriture remplacent ces hyperarcs par des sous-hypergraphes. Un hyperarc est ici une généralisation d'un arc dans un graphe qui connecte n nœuds d'un graphe avec $n \geq 1$. Dans le cas des hypergraphes dirigés, chaque hyperarc possède un et un seul nœud source et zéro ou plusieurs nœuds cibles. Sur la Figure 7.14a, l'hyperarc étiqueté *croire* ne connecte qu'un et un seul nœud, alors que l'hyperarc *A1* connecte deux nœuds et l'hyperarc Y trois.

Une règle d'une HRG possède un hyperarc en partie gauche et un hypergraphe en partie droite comme à la Figure 7.14. Pour savoir comment fusionner l'hypergraphe dans le graphe d'entrée, la théorie introduit la notion de *nœuds externes* (nœuds numérotés sur fond noir à la Figure 7.14) qui sont coindexés entre la partie gauche et la partie. Pour décrire le mécanisme de réécriture, il est utile de définir $H[e/R]$ comme l'hypergraphe obtenu après remplacement de l'hyperarc $e = (v_1 \cdots v_n)$ par l'hypergraphe R. Les nœuds externes de R fusionnent avec les nœuds de e de telle sorte que R se connecte à $H[e/R]$ aux mêmes nœuds auxquels e est connecté dans H. Étant donné un hypergraphe quelconque H avec un arc e, s'il existe une production $p : l_H(e) \to R \in G_P$ et que $|X_R| = |e|$, $H \Rightarrow_P H[e/R]$ est définie comme une dérivation directe indiquant que p peut dériver $H[e/R]$ à partir de H en une seule étape. $H \Rightarrow_G^* R$ indique que R est dérivable à partir de H par G en un nombre fini d'étapes de réécriture. Une séquence de dérivations est donnée à la Figure 7.15.



(a) Règle 1 de la grammaire. (b) Règle 2 de la grammaire.

Figure 7.14: Deux règles d'une HRG.

Les HRG peuvent être vues comme le pendant des grammaires d'arbres (TAG, grammaires d'interactions, etc.) pour les hypergraphes. De fait, les propriétés de tels systèmes sont parfaitement connues et il est possible de les adapter aux tâches de prédiction de structures prédicat-argument. Cependant, le fait de travailler avec des (hyper)graphes posent des problèmes importants en terme de complexité algorithmique pour le parsing, car le

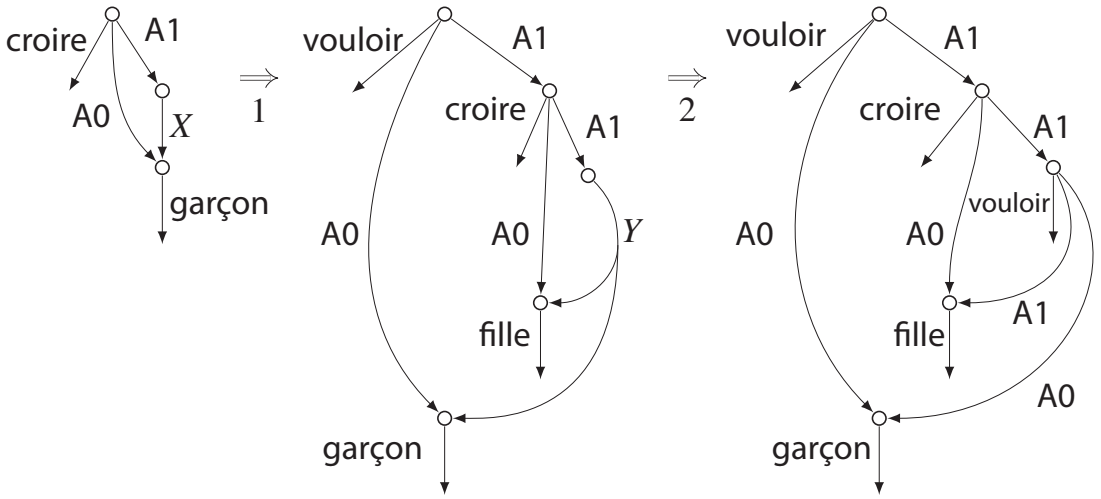


Figure 7.15: Étapes de dérivation via la HRG ci-dessus, pour la phrase *Le garçon veut que la fille croit qu'il la veut.*, donnant une représentation PropBank simplifiée (repris de (Chiang et al., 2013)).

problème est connu pour être NP-complet (Lautemann, 1990). La complexité peut être bornée comme l'a proposé récemment Chiang et al. (2013) en adaptant l'algorithme de Lautemann (1990) aux problèmes pour lesquelles la largeur arborescente est fixée et faible. Chiang et al. améliorent l'algorithme initialement proposé, algorithme qui est une généralisation de l'algorithme CKY appliqué aux HRG. Pour ce faire, Chiang et al. adopte une technique similaire au *rank-minimization problem* pour les *Linear Context-Free Rewriting Systems* pour lequel (Gildea, 2011) a montré que le problème était lié à la notion de largeur arborescente. L'algorithme développé par Chiang et al. (2013) a une complexité en temps de $O((3^d n)^{k+1})$ avec d le degré maximal du graphe d'entrée et k la largeur arborescente maximale de la grammaire. La complexité en espace est de $O((2^d n)^{k+1})$.

Enfin, l'induction de telles grammaires est encore balbutiante surtout dans le domaine du TAL, où – à notre connaissance – seule l'approche de Peng et al. (2015) utilisant des méthodes d'échantillonnage de Monte-Carlo par chaînes de Markov (Andrieu et al., 2003) a donné des résultats encourageants. Par ailleurs, il est assez complexe de mettre en place manuellement des règles de réécriture avec ce genre de grammaires, dans la mesure où la généralisation à des hypergraphes n'est pas aisée à appréhender, puisque très peu intuitive. Pour toutes ces raisons, nous avons décidé de suivre les approches algébriques où un sous-graphe en remplace un autre, car cela donne des systèmes de réécriture beaucoup plus simples et plus intuitifs comme nous allons le voir avec OGRE dans le cadre de l'annotation automatique de treebanks.

7.2 Annotation automatique de treebanks par réécriture de graphes

7.2.1 Pré-annotation automatique du DeepSequoia

Les règles de transformation pour la pré-annotation sont organisées en cinq modules distincts qui sont des ensembles de règles. Chaque module est appliqué de manière séquentielle. Cette distinction permet d'annoter phénomène par phénomène et d'utiliser le système de réécriture en parallèle des corrections manuelles, à la manière de l'annotation dynamique du DeepBank (Flickinger et al., 2012). Pour donner une idée de la complexité de la pré-annotation, les cinq modules contiennent respectivement 19, 40, 21, 39 et 36 règles pour un total de 155 règles (contre 290 environ pour GREW, cf. page 105). Quand bien même le nombre de règles est assez faible, la maintenabilité demande une certaine dose d'expertise comme pour beaucoup de travaux du même genre.

7.2.1.1 Modules de pré-annotation

Chaque module effectue une tâche distincte :

1. **Lemme profond et mode profond**. Ce module rend explicite les lemmes et les modes profonds pour les verbes en convertissant le mode des auxiliaires en traits. Par exemple, sur la Figure 7.11 p. 110, le verbe *respecté* est un participe passé au niveau surfacique, mais au niveau profond c'est un infinitif passé (*être respecté*). Plus généralement, le mode profond est le mode du premier auxiliaire : *a été mangé*, aura un mode profond *indicatif*, *fasse manger*, *subjonctif*, etc. L'idée est de simplifier l'écriture des règles dans les modules suivants, mais aussi de ne pas perdre d'information, car les auxiliaires sont sémantiquement vides. Les lemmes profonds ont le même but, ils sont surtout remarquables pour les verbes intrinsèquement impersonnels, comme dans l'exemple *Il se souvient d'elle* où le verbe porte le lemme profond *se_ souvenir*, dans la mesure où le clitique *se* n'appartient pas à la représentation profonde.
2. **Explicitation des sujets finaux**. Ce module fait un usage intensif de la deuxième phase du système de réécriture, surtout en ce qui concerne les cascades de prédicats et/ou de coordinations (Figure 7.11) où deux verbes à montée et contrôle sont présents ainsi qu'une coordination elliptique. Les verbes à contrôle et à montée étant des phénomènes lexicaux, nous les avons extraits de Dicovalence (Eynde et Mertens, 2006) avant d'affiner notre liste sur le corpus de développement du DeepSequoia. Ces informations sont directement intégrées dans les règles de réécriture.
   Un des problèmes majeurs de ce module d'annotation concerne le cas des prépositions non régies, pour lesquelles il est difficile d'inférer automatiquement le sujet de l'infinitif. En effet, une même préposition ne force pas le sujet du verbe principal comme sujet du verbe à l'infinitif. Dans l'exemple 7.2, le sujet de l'infinitif est bien le sujet du verbe principal (*Paul* est bien le sujet de *souper*) mais dans l'exemple 7.3 ce n'est pas le cas (*Jean* n'est pas le sujet de *laver*, c'est *Sophie* le sujet de *laver*). Néanmoins, certaines prépositions telles que *sans*, *avant (de)*, *après*, *au lieu de* ont

une tendance importante à privilégier comme sujet de l'infinitif, le sujet du verbe modifié par la préposition, ce qui permet de le repérer automatiquement de manière précise.

(7.2) Paul$_i$ était sorti pour souper$_i$.

(7.3) Jean paie Sophie$_i$ pour laver$_i$ Marie.

3. **Neutralisation de l'alternance syntaxique**. Ce module gère l'alternance syntaxique (diathèse passive, causative, impersonnelle, etc.). Il fait notamment usage du déclencheur *multi-edges* pour ajouter les fonctions canoniques nécessaires lors de changements de diathèse.

   Un point important à noter est qu'une séparation nette entre le module précédent et celui-ci n'est pas possible, en particulier à cause des verbes à contrôle. En effet, la généralisation syntaxique applicable aux verbes à contrôle mélange fonctions finales et canoniques. Un verbe à contrôle impose quel argument *canonique* est le sujet *final* de l'infinitif. Le verbe *condamner*, par exemple, est un verbe à contrôle objet : son objet *canonique* est le sujet *final* de la proposition infinitive qu'il introduit. Dans l'exemple *La cour condamne Jean à travailler toute sa vie*, l'objet *Jean* est le sujet **final** du verbe *travailler*. Si le verbe *condamner* est mis au passif : *Jean est condamné à travailler toute sa vie*, il est toujours vrai que l'objet **canonique** du verbe *condamner* est sujet de l'infinitif. C'est là toute la nécessité d'une distinction selon l'axe final / canonique, qui permet une bonne généralisation de ces cas complexes.

4. **Comparatifs, superlatifs et interrogatives**. Ce module pré-annote les comparatifs, les superlatifs et les interrogatives en marquant notamment l'argument profond du comparatifs, des superlatifs et en supprimant les *il* explétifs dans les interrogatives.

5. **Suppression des mots sémantiquement vides**. Ce module supprime automatiquement les mots considérés comme sémantiquement vides et redirige les arcs profonds sur les mots sémantiquement pleins (Figure 7.10, p. 109 et section 6.1.5, p. 80).

7.2.1.2 Évaluation du système de règles

La création des règles de réécriture pour la pré-annotation du DeepSequoia avait deux buts : simplifier la tâche d'annotation et pré-annoter (ou à défaut annoter) tout corpus qui possède le même schéma d'annotation que celui du French Treebank converti en dépendances. L'idée est d'appliquer le schéma d'annotation profond le plus rapidement possible. De fait, les règles ont été écrites de manière à limiter au maximum les informations lexicales afin qu'elles ne soient pas fortement liées à un corpus. Néanmoins, il est important de tester leur efficacité en présentant une évaluation des règles après correction manuelle des annotations sur le DeepSequoia.

Pour ce faire, il a été nécessaire de diviser le corpus DeepSequoia en quatre parties : la partie qui servira à l'entraînement (*train*) de modèles statistiques, une deuxième sur laquelle les règles ont été mises au point (la mini-référence ou *miniref*), une troisième qui

est un second corpus de développement (*dev2*) et qui a permis d'améliorer les règles et une quatrième, le corpus de test (*test*) utilisé pour évaluer les systèmes existants et futurs sur des données inconnues. Le découpage du corpus avec le nombre de phrases et de tokens, ainsi que le nombre de tokens profonds (les mots vides ne sont pas comptés) est donné Table 7.4.

Ens.	# Phrases	# Tokens	# Tok. profonds	# Arcs
Train	2 202	47 415	40 792	57 423
Dev-1 (Miniref)	247	5 852	5 038	7 122
Dev-2	250	5 360	4 606	6 503
Test	400	8 411	7 264	10 195

Table 7.4: Découpage du DeepSequoia pour les évaluations et expérimentations.

Fondée sur la réécriture de la surface de référence (*gold*), cette évaluation constitue un plafond difficile de dépasser lors de la prédiction automatique d'un tel schéma via des méthodes d'analyse sémantique (*semantic parsing*) ou des méthodes de réécriture. La Table 7.5 rapporte l'évaluation de l'application des règles sur le DeepSequoia en terme de précision, rappel et F_1-score lorsque sont considérées les étiquettes sur les arcs (c'est un score étiqueté ou *labeled score*[13]) et lorsque ces étiquettes ne sont pas considérées (un score non étiqueté ou *unlabeled score*). De plus, deux catégories d'arcs sont distinguées : l'ensemble des arcs de la représentation profonde (*arcs profonds*) et seulement les arcs qui ont été ajoutés par le schéma (*arcs profonds non réalisés en surface*). En effet, dans l'exemple à la Figure 7.16, l'arc suj entre *Jean* et *veut* est un arc hérité de la représentation de surface mais qui reste valide en profond, il en va de même pour l'arc obj. L'arc suj entre *Jean* et *dormir* a été ajouté au niveau profond, il appartient donc à la seconde catégorie.

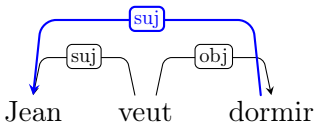


Figure 7.16: Distinction entre arcs profonds et arcs profonds non réalisés en surface.

Cette évaluation en deux temps permet de distinguer une évaluation globale du schéma produit par les règles et une évaluation centrée uniquement sur les ajouts. En effet, comme bon nombre d'arcs sont hérités de la représentation surfacique, sans cette distinction, les scores présentés ne reflèteraient pas complètement la difficulté de la tâche de pré-annotation.

Globalement, les règles donnent des résultats excellents sur le DeepSequoia, ce qui montre l'efficacité de notre approche de pré-annotation automatique. Ces scores sont comparables aux scores donnés pour grew (Table 7.3, p. 105). Le travail de l'annotateur en

13. LP, LR et LF signifient respectivement Labeled Precision, Labeled Recall, Labeled F_1-score. UP, UR et UF sont les scores non étiquetés.

Test	# arcs de référence	LP	LR	LF	UP	UR	UF
Arcs profonds (tout)	8259	99.5	99.2	99.4	99.5	99.3	99.4
Arcs profonds ($\notin$ surface)	1806	98.1	97.3	97.7	98.3	97.5	97.9

Table 7.5: Évaluation des règles sur le DeepSequoia (Précision, Rappel et F_1-score (non) étiquetés.

est donc grandement facilité dans la mesure où le nombre d'erreurs à corriger est restreint. Nous pouvons comparer cette pré-annotation automatique à celle effectuée sur le Modified French Treebank par Schluter et van Genabith (2008) qui rapporte un F_1-score de 96% légèrement inférieur au notre de 97.90%.

Néanmoins, cette évaluation en elle-même ne répond pas au problème du sur-apprentissage (*overfitting*) sur le DeepSequoia. En effet, à force d'itérations sur la mini-référence pour mettre au point les règles, il est possible d'avoir spécialisé les règles sur les domaines du corpus Séquoia. De fait, il est important de vérifier dans quelle mesure les règles sont généralisables sur d'autres corpus lorsque le vocabulaire et le style ne sont pas les mêmes. Par exemple, il n'est pas évident qu'elles puissent s'appliquer correctement sur un corpus comme celui du French Treebank. Là où le Séquoia est un corpus balancé avec quatre domaines différents et des styles variés, le French Treebank présente une certaine homogénéité car les articles sont tous issus d'un même journal.

7.2.2 Projection des annotations sur le French Treebank

Comme nous le suggérions précédemment, le but de cette projection est triple. Tout d'abord, nous voulons étendre le corpus d'entraînement et de test afin de pouvoir entraîner des systèmes stochastiques. En effet, le corpus Séquoia ne contient que 3 200 phrases, il est donc difficile d'entraîner des systèmes avec une forte exactitude. Ensuite, la majorité des travaux sur le français ont été conduits sur le French Treebank. Pour être comparable, il est important de disposer d'une annotation sur ce treebank. Enfin, nous souhaitons évaluer nos règles sur un corpus différent de celui pour lequel elles sont prévues au départ. Il est donc nécessaire de regarder les forces et les faiblesses d'une telle approche.

7.2.2.1 Évaluation quantitative de la projection

Afin de conduire nos expériences, nous avons utilisé le French Treebank dans sa version SPMRL 2014 (Seddah et al., 2014) avec catégories morpho-syntaxiques et traits morphologiques prédits. Le découpage du corpus est donné à la Table 7.6. Il a par ailleurs été nécessaire d'adapter quelques peu les règles, puisque le corpus en l'état n'annotait pas les distinction entre p_obj et p_obj.agt [14], ce qui ne permettait pas de repérer correctement

14. Cette distinction a été introduite lors de l'annotation du corpus Séquoia (Candito et Seddah, 2012b) et n'est donc pas présente dans l'annotation du French Treebank.

les passifs et provoquait des erreurs trop fréquentes.

Ens.	#Phrases	#Tokens
Train	14 759	443 113
Dev	1 235	38 820
Test	2 541	75 216

Table 7.6: Découpage du French Treebank SPMRL 2014.

Pour évaluer les règles, nous avons manuellement annoté 200 phrases issues de l'ensemble de développement en collaboration avec Marie Candito. Nous avons procédé comme suit : nous appliquons les règles sur les arbres surfaciques de référence augmentés des informations nécessaires à la pré-annotation profonde (marquage des clitiques impersonnels, du causatif et des réfléchis). Une fois ces règles appliquées, nous obtenons une représentation profonde. Cependant, lors de l'annotation manuelle des 200 phrases pour évaluation, nous avons aussi corrigé des erreurs dans la représentation surfacique.[15] Ces corrections sur la surface nous permettent d'obtenir une *surface corrigée* sur laquelle nous réappliquons les règles. Ainsi, notre évaluation se fait en deux temps : (a) évaluation de la syntaxe profonde sans surface corrigée (*Profond prédit 1*), (b) évaluation de la syntaxe profonde avec surface corrigée (*Profond prédit 2*). Globalement, les évaluations sont bonnes (Table 7.7) avec plus de 97% de LF, nous laissant penser que les règles ne sont pas liées au corpus Séquoia et ne se fondent pas sur des idiosyncrasies propres à ce corpus. En effet, l'évaluation de *Profond prédit 1* donne une estimation de la qualité de l'ensemble du DeepFTB puisqu'il y a un mélange d'erreurs dues au règles mais aussi dues à la conversion automatique de la surface en dépendances. Cependant, il est utile de voir comment se comporte l'application des règles sur une représentation surface sans erreurs, car l'évaluation entre la surface corrigée et non corrigée nous montre que la conversion n'est pas parfaite, ce qui pénalise notre système. La ligne *Profond prédit 2* nous montre alors que les résultats sont comparables à ceux de l'application des règles sur le DeepSequoia (environ 99% de LF) nous laissant penser que les règles généralisent parfaitement à d'autres corpus lorsque l'annotation surfacique est parfaite. En effet, l'avantage de l'approche par réécriture est de ne pas s'appuyer exclusivement sur le lexique (sauf dans le cas des phénomènes lexicalisés tels que le contrôle ou la montée) et de pouvoir s'appuyer sur l'intégralité des informations contenues dans le graphe : informations sur les traits morphologiques, les catégories morpho-syntaxiques qu'elles soient fines (différence entre nom propre et nom commun par exemple) ou grossières (une seule catégorie pour les noms), mais surtout la structure arborescente surfacique donnant des indices nécessaires à la transformation d'un arbre en graphe profond. Pour nos expériences futures, le Deep French Treebank (DeepFTB) sera suffisant en terme de qualité.

15. La représentation en dépendances est obtenue par conversion automatique et n'est donc pas exempt d'erreurs (Candito et al., 2010).

FTB (200 dev.)	# arcs référence	LP	LR	LF	UP	UR	UF
Surf. non corrigée vs. Surf. corrigée	6170	98.7	98.0	98.4	100.0	99.4	99.7
Profond prédit 1	6012	97.5	97.1	97.3	98.9	98.4	98.7
Profond prédit 2	6012	99.5	99.3	99.4	99.6	99.4	99.5

Table 7.7: Évaluation des règles sur le French Treebank (Précision, Rappel et F_1-score (non) étiquetés).
Profond prédit 1 : Application des règles sur surface non corrigée,
Profond prédit 2 : Application des règles sur surface corrigée.

7.2.2.2 Évaluation qualitative de la projection

Le nombre d'erreurs étant relativement faible, nous les avons considérées une à une pour produire une analyse qualitative de l'application des règles. Cette analyse révèle que certains phénomènes ne sont pas (correctement) gérés, en raison de leur complexité ou de leur caractère souvent ambigu : (i) les compléments prédicatifs nominaux dans des phrases comme *C'est une femme Capitaine* où une relation ARG doit être ajoutée entre *femme* et *Capitaine* n'est pas automatiquement annotée (ii) les coordinations elliptiques ne sont totalement annotées par les règles et ce pour le *gapping* ou les *argument cluster coordinations*. Enfin, l'annotation automatique des sujets des infinitifs est la cause principale des erreurs. Nous les classons en deux catégories :

1. **Les verbes à contrôle ou à montée** qui ne sont pas présents dans nos ressources lexicales : *annoncer* et *continuer de* en sont deux exemples. En effet, contrairement à des travaux comme (Chrupala et van Genabith, 2007), nous avons choisi de ne pas acquérir les verbes sur corpus mais grâce à des lexiques, il est donc possible d'avoir des problèmes de couverture.
2. **Le contrôle arbitraire** n'est pas non plus géré correctement. En effet, les règles utilisent toujours le sujet du verbe principal comme sujet de l'infinitif, même s'il est clair que cette règle est trop simpliste, puisque dans un exemple comme *Ils ont reçu les élèves pour visiter le château*, le sujet de l'infinitif est l'objet du verbe principal et non son sujet.

7.2.2.3 Comparaison avec d'autres approches

Pour finir, il est possible de comparer notre approche avec trois autres : l'annotation automatique du CCGBank (Hockenmaier et Steedman, 2007), l'annotation LFG automatique du Modified French Treebank (Schluter et van Genabith, 2007, 2008) et celle de treebanks de l'allemand (Rehbein et van Genabith, 2009). Pour la première approche, Hockenmaier et Steedman (2007) rapportent un taux de couverture de 94% lorsque les annotations sont appliquées à un ensemble inconnu de phrases, contre 100% pour notre approche. Schluter et van Genabith (2008) rapportent une pré-annotation de 96% de F_1-score en l'appliquant sur une annotation de référence là où Rehbein et van Genabith (2009) donnent des scores

allant de 95% à 97% sur deux treebanks de l'allemand. [16] Même si la couverture et le F_1-score ne sont pas deux mesures comparables, elles donnent une estimation de la qualité des deux approches et permettent de voir que nous sommes comparables avec un F_1-score de 97.3%.

Ces résultats sont très encourageants et démontrent la capacité de notre approche à s'appliquer sur des domaines et des corpus différents. En effet, nos règles de réécriture (Ribeyre et al., 2014a) sont conçues pour être faiblement lexicales, hormis les informations de contrôle et de montée, elles s'appuient au maximum sur des informations présentes au niveau des catégories morpho-syntaxiques, parfois au niveau des traits morphologiques et surtout au niveau de la structure syntaxique de surface. Peu d'informations issues des tokens ou des lemmes sont utilisées.

Cependant, nous n'avons testé notre approche que sur des données de référence (*gold*). En l'état, il n'y a aucune raison de croire que l'approche est efficace lorsque les règles sont appliquées sur des sorties prédites d'analyseurs surfaciques.

7.3 Réécriture de sorties d'analyseurs surfaciques : une approche utile ?

Pour démontrer le bien fondé de l'approche par réécriture dans le cadre de la prédiction de structures profondes, nous avons utilisé des sorties d'analyseurs surfaciques puis les avons réécrites grâce aux règles d'OGRE. L'idée est de voir la généralisation de notre approche sur des sorties prédites (et donc bruitées) mais aussi de comprendre quelles informations sont réellement utiles pour bien induire le passage d'un niveau de surface au niveau profond.

7.3.1 Protocole expérimental

Pour ce faire, nous avons considéré les données SPMRL 2014 sur le français (Seddah et al., 2014) avec catégories morpho-syntaxiques et traits morphologiques prédits. Ces données ont la particularité de considérer les expressions polylexicales (*multi-word expression*) directement au niveau de la structure syntaxique via une étiquette DEP_CPD (cf. Figure 7.17). La prédiction de ce genre d'expressions est un challenge en soi et peut dégrader les performances de l'analyseur utilisé (Candito et Constant, 2014). Pour pallier en partie ce problème, les données SPMRL 2014 mettent à disposition deux traits morphologiques supplémentaires appelés *mwehead* et *pred*. Le premier donne la tête de l'expression polylexicale et le second indique si un token est issu d'une expression polylexicale. Ces traits sont prédits (Seddah et al., 2014), afin de respecter un scénario réel de prédiction lors de l'analyse syntaxique surfacique. Ce protocole va alors nous donner une vision réelle de la performance de nos règles.

16. Nous laissons de côté l'évaluation de Rehbein et van Genabith (2009) sur le TiGerDB bien inférieure aux deux autres présentées. En effet, comme les auteurs l'expliquent le TiGerDB annote des informations supplémentaires qu'il est difficile de projeter sur le TiGer Treebank, les scores sont donc moins bons.

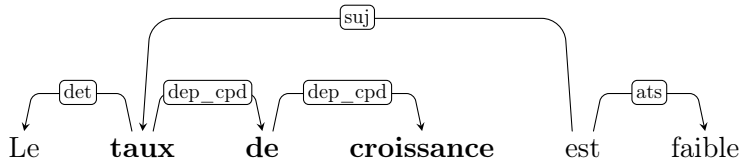


Figure 7.17: Exemple d'annotations d'expressions polylexicales dans les données SPMRL 2014 pour le français.

Nous considérons trois analyseurs de surface différents, choisis pour leur divergence dans les approches utilisées. Le premier est un analyseur stochastique par factorisation d'arcs (*arc-factored graph-based parser*), appelé Mate (Bohnet, 2010). Le deuxième est un analyseur stochastique par transitions bénéficiant d'une recherche par faisceaux (*transition-based beam-search parser*), appelé DyALog-SR (Villemonte de La Clergerie, 2013a). Le dernier est un analyseur s'appuyant sur une méta-grammaire TAG appelé FrMG (Villemonte de La Clergerie, 2010) [17]. Tous les analyseurs sont utilisés sans modification préalable ni ajustement de traits et ou de paramètres. Nous pensons que la diversité d'approches dans l'analyse surfacique nous donnera des informations complémentaires, que ce soit au niveau de la qualité de la réécriture, mais aussi au niveau du type d'analyseur à utiliser pour obtenir une réécriture de qualité.

Les expériences se déroulent ainsi : une fois l'analyse de surface prédite, nous appliquons les règles afin de produire une sortie en syntaxe profonde que nous évaluons.

7.3.2 Évaluations détaillées : des résultats encourageants

Nous reportons les résultats (ensemble de développement) sur la syntaxe de surface à la Table 7.8 et sur la syntaxe profonde à la Table 7.9.

dév.	LAS	UAS
FrMG	82.78	85.71
DyALog-SR	83.48	87.17
Mate	**84.31**	**87.81**

Table 7.8: Évaluation de la prédiction surfacique pour les trois analyseurs considérés (dév SPMRL 2014 (Seddah et al., 2014)).

Malgré la disparité dans l'ordre de grandeur des performances de surface par rapport à celles au niveau profond, le différentiel de points entre les différents analyseurs est conservé en profond (±3 points). Par ailleurs, la qualité de la prédiction profonde est corrélée à la qualité surfacique, ce qui s'explique notamment par notre approche qui s'appuie énormé-

17. Cet analyseur est capable de produire des dépendances surfaciques par conversion des sorties TAG en dépendances, voir (Villemonte de La Clergerie, 2010) pour plus de détails.

dév.	LP	LR	LF	UP	UR	UF
FrMG	79.31	79.80	79.55	82.26	82.76	82.51
DyALog-SR	80.03	80.52	80.27	83.72	84.24	83.98
Mate	80.94	81.37	**81.16**	84.56	85.00	**84.78**

Table 7.9: Évaluation de la prédiction profonde à partir des trois analyseurs surfaciques et des règles d'ogre (dév. SPMRL 2014 (Seddah et al., 2014)).

ment sur la structure surfacique pour fonctionner. Néanmoins, sans optimisation aucune sur les traits et les paramètres utilisés, nos scores sont tout à fait convaincants et apportent une approche valable pour la prédiction du niveau profond en français.

Nous souhaitons pousser plus loin nos investigations et regarder quantitativement comment notre système de réécriture se comporte. Pour ce faire, nous choisissons d'abord de ne considérer que les arcs profonds non réalisés en surface (cf. évaluation du DeepSequoia (p. 120) et de la projection vers le DeepFTB (p. 122). En effet, comme nous l'avons expliqué auparavant, l'intersection entre la surface et la structure profonde est importante, ce qui implique que les performances du passage surfacique / profond ne peuvent pas seulement être considérées en ne regardant qu'un F_1-score global. Nous reportons ces nouveaux scores à la Table 7.10.

dév.	LP	LR	LF	UP	UR	UF
FrMG	**81.98**	76.55	79.17	**84.33**	78.74	**81.44**
DyALog-SR	79.44	77.64	78.53	81.23	79.39	80.30
Mate	79.68	**79.10**	**79.39**	81.42	**80.82**	81.12

Table 7.10: Évaluation de la prédiction profonde à partir des trois analyseurs surfaciques et des règles d'ogre sur les arcs profonds non réalisés en surface (dév).

Les résultats (Table 7.10) montrent que la qualité de prédiction surfacique n'est pas totalement corrélée à la prédiction de la syntaxe profonde. En effet, FrMG – dont les performances sont les plus faibles en surface – permet une meilleure génération de la structure profonde après réécriture que DyALog-SR. Mais globalement, les tendances restent les mêmes. Nous soulignerons cependant que l'approche utilisée par les différents analyseurs a un impact important sur les résultats de réécriture : (i) FrMG, issu d'une grammaire tag, gère mieux les phénomènes de partage du sujet ou de contrôle ; (ii) Mate, un analyseur par factorisation d'arcs, considère la structure dans son ensemble (et non par transitions successives comme le fait DyALog-SR). Pour avoir une vue plus détaillée de ce phénomène, nous donnons le détail des performances par étiquette (en ne considérant que les 5 étiquettes les plus fréquentes et qui représentent 91% de la masse des dépendances). Les résultats sont présentés à la Figure 7.18.

Certaines tendances sont visibles notamment en ce qui concerne les objets indirects

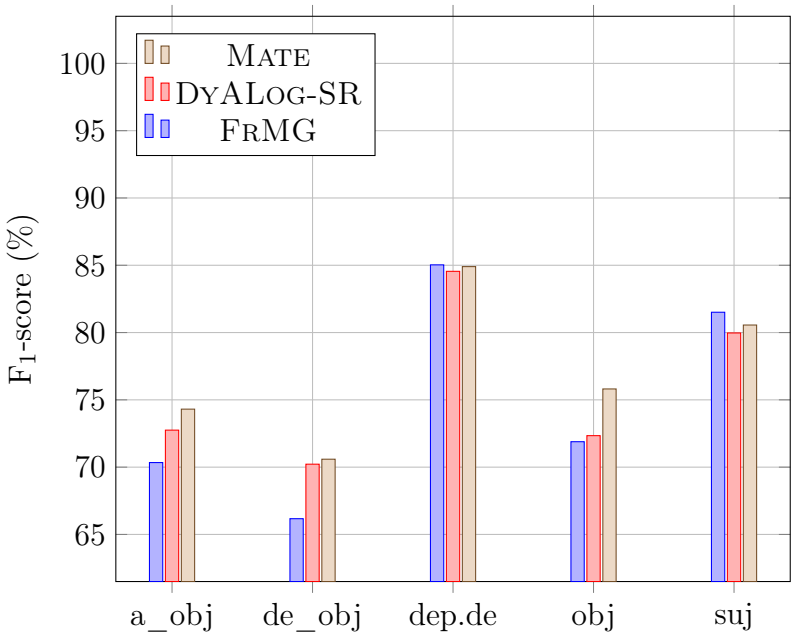


Figure 7.18: F_1-score pour les étiquettes les plus fréquentes du DeepFTB (arcs profonds non réalisés en surface pris en compte sur l'ensemble de développement).

(a_obj et de_obj) d'une part et les sujets et objets d'autre part. Pour les premiers, ce sont les informations de surface qui sont cruciales. En effet, les arcs profonds a_obj et de_obj sont directement hérités d'un déplacement dû à une préposition sémantiquement vide (Figure 7.19). Si la prédiction surfacique n'est pas bonne, le système de réécriture ne peut pas ajouter l'arc profond. Or, les attachements prépositionnels sont souvent des décisions difficiles à prendre pour certains analyseurs, ce qui explique les scores plus faibles pour FrMG.

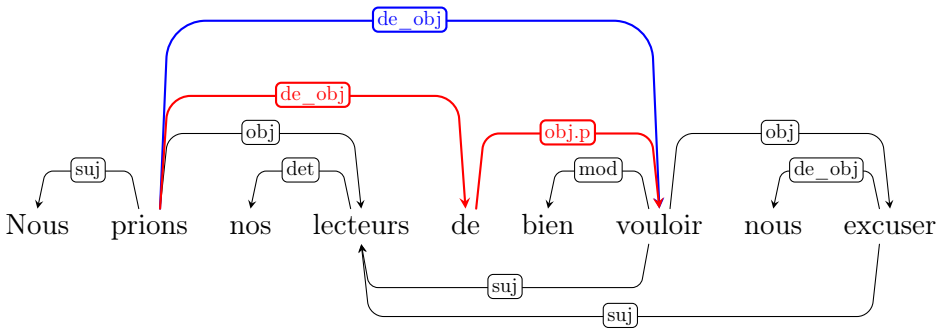


Figure 7.19: Représentation profonde du DeepSequoia pour un objet indirect et une préposition sémantiquement vide (de_obj profond en bleu et surfacique en rouge).

L'autre point central est la prédiction des sujets et des objets directs. Un système tel que FrMG permet à notre système de mieux récupérer les sujets non explicités en surface

(contrôle, montée, ellipses, etc.). FrMG s'appuie sur les informations du LEFFF (Sagot, 2010) pour prendre certaines de ses décisions, c'est notamment le cas pour le contrôle. De fait, il n'est pas étonnant que pour une phrase où un phénomène de contrôle apparaît, FrMG produise une représentation surfacique meilleure [18] que les deux autres analyseurs qui n'ont pas cette information à l'entraînement. Enfin, concernant les objets directs profonds, le problème est le même que pour les objets indirects : certaines prédictions de surface sont essentielles lors du passage vers le niveau profond. Le problème est illustré à la Figure 7.20, où la prédiction surfacique de l'arc OBJ conditionne directement le passage vers le profond.

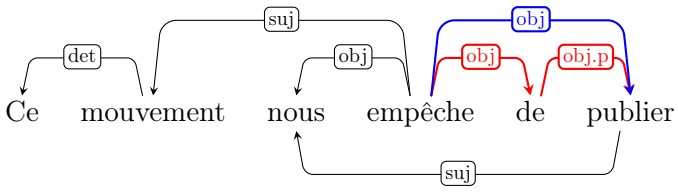


Figure 7.20: Représentation profonde du DeepSequoia pour un objet direct et une préposition sémantiquement vide (obj profond en bleu et surfacique en rouge).

Par ces diverses évaluations, nous mettons en lumière deux points distincts : (i) la prédiction de surface est directement corrélée à la qualité de la transformation profonde, que ce soit parce qu'une large part des arcs prédits en surface se retrouve dans la structure profonde ou parce que la gestion des prépositions sémantiquement vides rend crucial une bonne prédiction surfacique (ii) l'utilisation d'informations complémentaires (informations sur le contrôle, la montée, etc.) peuvent avoir un impact intéressant sur la manière dont la structure profonde sera générée par les règles de transformation. Quoi qu'il en soit, l'utilisation de règles de réécriture appliquées à des sorties surfaciques prédites est une approche tout à fait valable et utile pour prédire le niveau profond.

7.3.3 Comparaison avec les travaux antérieurs

Cependant, l'approche n'est pas nouvelle, puisqu'elle a déjà fait l'objet de plusieurs travaux utilisant des formalismes différents. En effet, notre approche peut être largement comparée à l'écriture d'une grammaire (ou comme nous l'avons montré pour le système de propagation, d'une méta-grammaire) à la main pour enrichir des corpus arborés existants. Ce type de méthodes a longtemps été utilisé et a donné lieu à divers corpus de référence tels que le PARC 700 (King et al., 2003) [19], ou encore le CBS 500 (Carroll et al., 1999) (cf. page 30). Nous évaluons donc notre approche contre un analyseur issu d'une grammaire LFG manuelle mais utilisant un modèle de désambiguisation stochastique (King et al., 2003) et analyseur qui utilise une variation des *Definite Clause Grammars* (Pereira et Warren, 1980) couplé à un analyseur probabiliste LR (Carroll et Briscoe, 1996). Les résultats des divers analyseurs sont reportés à la Table 7.11. Nous remarquons que nos résultats sont

18. À noter que la qualité de cette représentation est largement dépendante de la qualité du Lefff.
19. Le PARC 700 est une annotation LFG manuelle de 700 phrases dans le but d'évaluer les performances d'un analyseur issu d'une grammaire manuelle LFG.

comparables à ceux reportés par (King et al., 2003) (et ce eu égard à la taille des données) mais largement inférieurs à ceux de (Carroll et al., 1998), car lors de l'évaluation, ces derniers considèrent comme correcte une relation grammaticale parente. Par exemple, dans la hiérarchie GR, CLAUSAL est le parent de XCOMP et COMP, si l'analyseur renvoie CLAUSAL au lieu de COMP, alors l'évaluateur considère que l'étiquette est correcte [20].

	Taille des données	F_1-score
GR (Carroll et al., 1998)	500	88.15
PARC 700 (King et al., 2003)	700	79.50
OGRE+Mate	2541	81.16
OGRE+Mate (profonds seuls)	2541	79.39

Table 7.11: Comparaison de l'approche par réécriture à des approches par grammaires manuelles.

Ainsi, l'approche par réécriture est une approche valide pour trois tâches distinctes : (i) la pré-annotation d'un corpus arboré pour lequel il n'existe pas de données dans le schéma d'annotation voulu ; (ii) la projection de ces annotations sur un autre corpus arboré présentant des caractéristiques structurelles communes ; (iii) la prédiction d'un niveau profond à partir de sorties d'analyseurs surfaciques. De fait, deux nouvelles questions se posent alors :

1. Est-il souhaitable d'utiliser des méthodes directes d'analyse automatique (*semantic parsing*) pour prédire ce type de graphes ? Et si oui quelles méthodes est-il nécessaire d'utiliser ?
2. Pouvons-nous apprendre automatiquement le passage du niveau surfacique au niveau profond sans avoir recours à l'écriture manuelle de règles de transformation ?

Nous explorons ces deux questions dans les chapitres suivants, d'abord en étendant des techniques éprouvées d'analyse syntaxique en dépendances et ensuite de manière plus prospectif en combinant la réécriture de graphe et l'apprentissage automatique.

20. Nous n'avons pas cette distinction lors de notre évaluation, car l'abstraction produite par le passage au niveau profond la rend moins cruciale que pour le niveau surfacique où, par exemple, la distinction entre les différents types d'auxiliaires (AUX.TPS, AUX.PASS, AUX.CAUS) peut être hiérarchisée sous une même étiquette AUX. La seule hiérarchisation possible serait au niveau de la distinction entre les différents types de modifieurs (MOD) : MOD.APP (apposition), MOD.INC (incise) et MOD.VOC (vocatif).

Chapitre

8 Parsing syntaxico-sémantique automatique

Sommaire

Introduction

Après avoir présenté des résultats encourageants de transformation d'analyses surfaciques prédites via les règles de réécriture ayant servi à la pré-annotation automatique du corpus en syntaxe profonde, nous nous sommes tournés vers des approches directes d'analyse de la structure argumentale de la phrase. Nous développons, dans ce chapitre, deux

approches complémentaires fondées sur les approches par transitions qui ont montré leur efficacité en analyse syntaxique (Nivre, 2005) et ont été naturellement portées à l'analyse sémantique. Nous décrivons alors deux jeux différents de transitions implémentés au sein des analyseurs suivants : (i) une extension du modèle de Sagae et Tsujii (2008), appelé S&T parser, qui utilise un décodage glouton et un jeu de traits riches pour l'analyse de graphes acycliques (DAGs) planaires ; (ii) une extension du modèle de Villemonte de La Clergerie (2013b), appelé DyALog-SR, qui utilise une recherche par faisceaux associée à un jeu de traits riches pour explorer correctement l'espace de recherche. L'analyseur est aussi capable de traiter une forme réduite de non-planarité en réutilisant des actions introduites dans Titov et al. (2009) pour l'analyse sémantique de surface.

Nous proposons de comparer leur efficacité sur trois corpus dont les schémas d'annotation sont fondés sur des présupposés différents mais partageant néanmoins des caractéristiques communes : le DM corpus (cf. section 3.2.1, p. 47), le PAS corpus (cf. section 3.2.2, p. 49) et le DeepFTB (cf. section 7.2.2, p. 120). Nous avons déjà présenté certaines de leurs caractéristiques à la Table 7.2 de la page 98.

	DM		PAS		DeepFTB	
	Train	Dev	Train	Dev	Train	Dev
# Phrases	32 389	1 614	32 389	1 614	14 759	1 235
# Tokens	742 736	36 810	742 736	36 810	457 872	40 055
% Tokens vides	21.65	21.58	4.30	4.25	11.97	12.19
# Arcs	559 882	27 778	723 437	35 573	424 808	37 110
% Arcs croisés	3.44	3.29	3.93	3.26	3.70	3.87
# Étiquettes	51	35	42	39	26	25
# Étiquettes$_{95\%}$	9	9	17	17	12	12

Table 8.1: Statistiques sur les graphes pour les corpus DM, PAS et DeepFTB. Étiquettes$_{95\%}$ représente le nombre d'étiquettes totabilisant 95% des arcs du corpus.

Les statistiques (Table 8.1) montrent trois propriétés importantes :

1. **Tokens sémantiquement vides.** Les trois corpus présentent des chiffres assez divergents. En effet, tous ne considèrent pas les mêmes tokens comme étant sémantiquement vides. DM considère que les auxiliaires, les prépositions et certaines conjonctions de subordination sont sémantiquement vides, là où PAS ne marque que les ponctuations finales de la sorte. Le DeepFTB, quant à lui, marque comme sémantiquement vides les mêmes types de catégories que DM bien qu'il conserve la ponctuation au sein du graphe sémantique. Les taux de tokens sémantiquement vides indiquent que la prise en compte de cette propriété au sein de l'analyse est cruciale pour avoir une analyse de qualité.
2. **Arcs croisés.** La densité d'arcs non-planaires dans les trois corpus est globalement la même. Nous constatons qu'il est intéressant d'en tenir compte lors de l'analyse,

mais que ça n'est pas l'aspect le plus crucial pour obtenir de bonnes performances, étant donné sa fréquence faible.

3. **Nombre d'étiquettes de dépendances.** Ce paramètre semble montrer (en considérant le corpus de développement surtout) que globalement les corpus ont le même nombre d'étiquettes. Néanmoins, ces chiffres globaux ne rendent pas compte de la réalité linguistique qui sous-tend chacun des schémas d'annotation. En effet, la distribution des étiquettes n'est pas la même : neuf étiquettes regroupent 95% de la masse des dépendances pour DM, contre douze pour le DeepFTB et dix-sept pour PAS. Cette propriété linguistique est illustrée en détails Table 8.17. Certaines fréquences sont intéressantes à relever : (i) les expressions polylexicales sont aussi fréquentes sur DM que sur le DeepFTB (9% contre 12%) (ii) la distribution des dépendances marquant la coordination sont similaires (2.50% environ) sur les trois corpus (_and_c sur DM, coord_* sur PAS et coord/dep.coord sur le DeepFTB) (iii) DM présente un marquage des marqueurs locatifs et temporels que les autres corpus ne distinguent pas. Pour le DeepFTB, ces phénomènes sont marqués MOD comme les autres modifieurs.

DM est un corpus dont le schéma d'annotation cherche clairement à se rapprocher d'une analyse sémantique alors que PAS est plus proche d'une analyse de type syntaxe profonde. Le DeepFTB est un corpus qui se situe entre DM et PAS.

DM	%	PAS	%	**DeepFTB**	%
ARG1	39.82	adj_ARG1	13.62	mod	19.16
ARG2	24.17	noun_ARG1	10.62	det	13.61
compound	12.58	prep_ARG2	10.31	ponct	12.62
BV	10.69	prep_ARG1	10.12	suj	11.83
poss	2.36	verb_ARG2	9.61	dep_cpd	9.28
_and_c	2.23	det_ARG1	9.35	obj.p	6.55
loc	1.53	verb_ARG1	9.13	obj	6.28
ARG3	1.35	punct_ARG1	5.48	dep.de	6.18
times	0.93	aux_ARG2	3.10	root	3.33
appos	0.85	aux_ARG1	3.08	dep.coord	2.50
mwe	0.81	coord_ARG2	2.49	coord	2.26
conj	0.61	coord_ARG1	2.49	dep	1.62
subord	0.45	comp_ARG1	1.91	*a_obj*	1.07
neg	0.39	conj_ARG1	1.26	*mod.rel*	0.90
_or_c	0.37	poss_ARG2	0.92	*ats*	0.89
plus	0.22	poss_ARG1	0.92	*de_obj*	0.80
_but_c	0.16	conj_ARG2	0.80	*obj.cpl*	0.44
TOTAL	95.67		95.21		95.20

TABLE 8.2: Distribution des étiquettes de dépendances dans chaque corpus (dev+train pris en compte).

Les cellules en italique ne sont pas prises en compte dans le total.

De fait, ces trois corpus ont un ensemble de propriétés communes qui permet de les

positionner le long d'un axe syntaxe / sémantique intéressant, ce qui sera intéressant pour mener des comparaisons. Dans ce chapitre nous testons à construire des modèles d'analyse automatique supervisée, où nous montrons que les analyseurs par transitions ne sont pas à même de capturer toute la richesse des représentations et qu'ils manquent d'informations contextuelles pour prendre leurs décisions. De fait, nous nous appuyons sur le fait que la syntaxe et la sémantique sont en général interdépendantes et que les structures prédicat-argument se recoupent dans une certaine mesure avec les analyses de surface. Nous extrayons des traits syntaxiques issus de trois analyseurs surfaciques, deux en dépendances et un en constituants, puis exploitons les propriétés de chaque structure pour apporter ce contexte manquant aux analyseurs et leur donner les informations permettant une prédiction plus précise de la structure profonde. Pour étayer notre argumentaire, nous réalisons une analyse détaillée des résultats et montrons l'utilité d'une telle approche sur les constructions linguistiques difficiles.

Enfin, puisque nos deux analyseurs sont fondés exclusivement sur des méthodes par transitions, qui n'ont, par construction, pas une vue globale de la phrase à analyser, nous nous tournons vers des méthodes plus globales à base de factorisation d'arcs. Plus particulièrement, nous décrivons l'analyseur de Martins et Almeida (2014), le TurboSemanticParser (ou TParser), qui présente un décodage par décomposition duale (Martins et al., 2011) lui permettant d'intégrer des traits d'ordre supérieur afin d'améliorer la prédiction sémantique. Nous montrons alors que l'apport des traits syntaxiques est toujours intéressante sur ce genre d'analyseur et lui permet de dépasser l'état de l'art.

8.1 Analyse par transitions

L'analyse par transitions est une méthode d'analyse syntaxique populaire et efficace dans le domaine du *parsing* (Nivre, 2005; Hall et Nivre, 2008; Nivre, 2009; Zhang et Nivre, 2011; Bohnet et Nivre, 2012; Zhu et al., 2013). L'idée est d'analyser le texte en entrée en une seule passe, sans retour en arrière. L'analyseur construit la structure analysée (arbre ou graphe) de façon incrémentale, de bas en haut et de gauche à droite. À chaque moment de l'analyse, l'analyseur accumule une liste de sous-arbres ou de sous-graphes par rapport au texte en entrée.

L'analyse par transitions a été d'abord utilisée pour analyser des arbres en dépendances avant d'être généralisée aux graphes (Sagae et Tsujii, 2008; Titov et al., 2009; Choi et Palmer, 2011; Henderson et al., 2013; Ding et al., 2014). L'attrait d'une telle méthode réside dans sa complexité algorithmique. En effet, il est possible d'analyser une phrase en temps linéaire [1] par rapport au nombre de tokens dans la phrase, complexité que d'autres approches ne sont pas capables d'atteindre (Nivre, 2005).

8.1.1 Définitions

Formellement, un *système par transitions* est une machine abstraite consistant en un ensemble de *configurations* (aussi appelées *états*) et un ensemble de *transitions* entre les configurations. L'intuition est qu'une séquence de transitions valides, démarrant à la *configuration initiale* pour une phrase donnée et terminant dans l'une des *configurations finales* définit un arbre valide de dépendances pour une phrase donnée (Kübler et al., 2009). Nous présentons d'abord l'approche utilisée pour l'analyse d'arbres, avant de l'étendre aux graphes.

Définition 8.1 (Configuration dans le cadre de l'analyse par transitions)**.**
Une *configuration* pour une phrase $S = w_0, w_1, \ldots, w_n$ et un arbre étiqueté $G = \langle V, E, L \rangle$ (déf. 7.2, p.93) [2] est un triplet $c = \langle \sigma, \beta, A \rangle$ où

— σ est une pile de mots $w_i \in V$.

— β est une file de mots $w_i \in V$.

— A est un ensemble d'arcs de dépendances étiquetés $(w_i, l, w_j) \in E$ encore écrits $u \xrightarrow{l} v$.

Une configuration représente une analyse partielle de la phrase d'entrée où les mots dans la pile σ sont partiellement analysés, les mots dans la file β sont les mots restants à

1. Cela n'est valable que pour les arbres, puisque pour les graphes la complexité atteint $O(n^2)$ avec n le nombre de tokens dans la phrase. Cependant, cette complexité reste bien inférieur aux approches par factorisation d'arcs (McDonald et al., 2005; Kuhlmann, 2014).

2. Nous rappelons que la séquence S et l'ensemble de nœuds $(V, <)$ ordonné strictement sont identiques et que l'ensemble des arcs E est défini comme $V \times L \times V$ où L est un ensemble de relations de dépendances. Plutôt que la notion d'ensemble de nœuds, nous lui préférerons ici l'utilisation la notion de phrase, car elle dénote une réalité importante pour l'analyse par transitions, contrairement aux approches par grammaires de graphes que nous avons décrites au chapitre précédent où cette notion était plus secondaire.

analyser et l'ensemble d'arcs A représente l'arbre de dépendances partiellement créé. Les configurations *initiales* et *finales* sont définies comme suit :

Définition 8.2 (Configuration initiale & finale)**.**
Pour n'importe quelle phrase $S = w_0 w_1 \cdots w_n$,

— Une configuration *initiale* $c_0(S)$ est $([w_0]_\sigma, [w_1, \cdots, w_n]_\beta, \emptyset)$.

— Une configuration *finale* est une configuration de la forme $(\sigma, []_\beta, A)$ pour n'importe quelle σ et A, où $[]_\beta$ dénote la file vide.

Une fois l'ensemble de configurations défini, ensemble qui inclut une configuration initiale unique et un ensemble de configurations finales, il est alors possible de définir un ensemble de *transitions* entre les configurations. Formellement, une transition est une fonction partielle de configurations à configurations (c'est-à-dire une transition fait correspondre une configuration c_i à une nouvelle configuration c_{i+1}, mais une transition peut être indéfinie pour certaines configurations). Une transition correspond à une action d'analyse : ajouter un arc, modifier la pile, modifier la file, etc. que nous détaillons plus loin.

L'analyse par transitions est un cadre théorique et non une méthode en soi ne produisant qu'un seul type d'analyse. En effet, il existe de nombreux ensembles de transitions (encore appelés jeux de transitions). Tous ces jeux ont des fonctions précises et permettent l'analyse des arbres, ou encore des DAGs et des graphes, comme nous le montrerons plus loin. Nous utilisons le symbole $\mathcal{T}$ pour l'ensemble de transitions possibles dans un système par transitions. Dans le but de générer des analyses complètes, nous introduisons aussi la notion de *séquence de transitions*.

Définition 8.3 (Séquence de transitions)**.**
Une *séquence de transitions* pour une phrase $S = w_0 w_1 \cdots w_n$ est une séquence de configurations $C_{0,m} = (c_0, c_1, \cdots, c_m)$ telle que :

— c_0 est une configuration initiale $c_0(S)$ pour S.

— c_m est une configuration finale.

— Pour chaque i tel que $1 \leq i \leq m$, il existe une transition $t \in \mathcal{T}$ telle que $c_i = t(c_i - 1)$.

8.1.2 Jeux de transitions et non-déterminisme

Pour pouvoir analyser un arbre, il faut définir des actions qui définissent les transitions à effectuer pour passer d'une configuration à une autre. Table 8.3, nous donnons un jeu de transitions possible pour analyser une structure arborescente (comme celle présentée à la Figure 8.1).

L'action Shift supprime le premier token w_i de la file et l'ajoute sur la pile σ. Les actions Left-Reduce$_l$ et Right-Reduce$_l$ ajoutent un arc entre deux tokens w_i et w_j adjacents et au sommet de pile. Chacune des deux actions supprime un des tokens de la

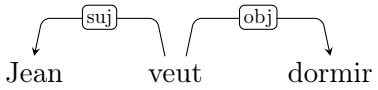


Figure 8.1: Arbre pour *Jean veut dormir*.

Transition				Pré-conditions
Shift	$(\sigma, w_i\|\beta, A)$	$\Rightarrow$	$(\sigma\|w_i, \beta, A)$	$\|\beta\| \neq 0$
Left-Reduce$_l$	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_i, \beta, A \cup \{(w_i, l, w_j)\})$	$i \neq 0$
Right-Reduce$_l$	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_j, \beta, A \cup \{(w_j, l, w_i)\})$	

Table 8.3: Jeu de transitions pour l'analyse d'arbres.
La notation $\sigma|w_i$ représente la pile après y avoir ajouté w_i et $w_i|\beta$ représente la file après y avoir ajouté w_i.

pile. Parmi les actions présentées ici, il est nécessaire de distinguer les actions que nous appelons **atomiques**, des actions appelées **instanciables**. Une action atomique n'est pas définie sur un ensemble L de relations de dépendances, c'est le cas de shift. Toutes les autres actions présentées ici sont instanciables : elles peuvent exister pour chaque relation de dépendances (Left-Reduce$_{\text{suj}}$, Left-Reduce$_{\text{obj}}$, etc.). [3] Une séquence de transitions possibles pour l'analyse de *Jean veut dormir* est donnée à la Table 8.4.

Transitions	Configurations		
	σ	β	A
	[]	[Jean, veut, dormir]	$\emptyset$
shift	[Jean]	[veut, dormir]	$\emptyset$
shift	[Jean, veut]	[dormir]	$\emptyset$
left-reduce$_{\text{suj}}$	[veut]	[dormir]	$A_1 = \{(\text{veut}, \text{suj}, \text{Jean})\}$
shift	[veut, dormir]	[]	A_1
right-reduce$_{\text{obj}}$	[veut]	[]	$A_2 = A_1 \cup \{(\text{veut}, \text{obj}, \text{dormir})\}$

Table 8.4: Exemple d'analyse avec le jeu de transition de la Table 8.3 pour la phrase *Jean veut dormir*.

L'analyse de *Jean veut dormir* révèle certains points intéressants de ce type de méthodes. Le point essentiel est qu'il n'existe pas une seule séquence de transitions pour aboutir à une configuration finale. Il n'existe pas non plus qu'une seule configuration finale. L'analyse par transitions pour les langues naturelles est donc par essence non-déterministe. Après le deuxième shift, nous pouvions soit en réaliser un troisième, soit effectuer l'action

3. Cette distinction a un impact en TAL, lors de l'apprentissage de la meilleure séquence de transitions, plus le nombre d'étiquettes est grand, plus le nombre de classes à prédire par un classifieur donné est élevé. Il peut donc être intéressant de faire attention aux actions utilisées et de privilégier des actions atomiques.

left-reduce pour créer l'arc suj, par exemple. De fait, pour faire de l'analyse déterministe, il faut trouver un moyen de prédire, pour n'importe quelle configuration non terminale c, quelle est la configuration suivante. Pour ce faire, il est nécessaire d'avoir recours à un **oracle**, c'est-à-dire une fonction $O : \mathcal{C} \rightarrow \mathcal{T}$ qui associe une configuration à une transition telle que $O(c) = t$ si et seulement si t est une transition applicable à partir de la configuration c et que t est correcte. Trouver la transition optimale $t = O(c)$ est un problème difficile pour l'analyse des langues naturelles, l'apprentissage automatique supervisé est donc utilisé pour résoudre ce problème.

Grâce à l'oracle, l'algorithme d'analyse est simplifié, comme le montre l'algorithme 2. Pour l'analyse d'arbre, l'algorithme est en temps linéaire si le calcul de l'oracle et l'application des transitions peuvent être évalués en temps constant.

Algorithme 2 Analyse déterministe fondé sur un oracle.

1: $c \leftarrow ([w_0]_\sigma, [w_1, \ldots, w_n]_\beta, \emptyset])$
2: **Tant que** c n'est pas terminale **Faire**
3: $\quad t \leftarrow O(c)$
4: $\quad c \leftarrow t(c)$
5: **Fin Tant que**
6: **Retourne** c

L'utilisation de l'apprentissage automatique supervisé est rendu nécessaire, puisqu'en réalité, il est extrêmement difficile de trouver des oracles satisfaisants pour les problèmes sur l'analyse de langue naturelle. De plus, l'apprentissage automatique supervisé est justifié par le développement de ressources manuellement annotées qui ont permis de modéliser ce problème dans un cadre de classification automatique.

8.1.3 Apprentissage automatique supervisé et décodage

Il convient de distinguer deux étapes : (i) l'*entraînement* d'un modèle en se fondant sur des données annotées (ii) l'utilisation du modèle pour trouver la meilleure séquence de transitions, appelée *décodage*.

8.1.3.1 Entraînement d'un classifieur multi-classes en ligne

L'analyse en dépendances utilise généralement des modèles discriminants, ce qui permet de réduire le problème d'apprentissage à une tâche de classification multi-classes (Nivre, 2005).

Dans la tâche de classification multi-classes, l'idée est d'assigner une classe pour chaque exemple en entrée à l'aide d'un corpus d'entraînement T de la forme $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ où $\mathbf{x}_i$ est le i$^{\text{ème}}$ exemple et y est l'étiquette du i$^{\text{ème}}$ exemple.

Dans le cas qui nous intéresse, y_i est un arbre ou un graphe et $\mathbb{x}_i$ la phrase correspondante. Bien entendu, l'analyse étant réalisée au moyen d'une séquence d'actions en partant d'une configuration initiale pour arriver à une configuration finale, il est nécessaire de définir un correspondance biunivoque (*one-to-one mapping*) entre la structure y_i et cette

séquence d'actions $\mathcal{A} = (a_1, \ldots, a_m)$. Pour ce faire, nous utilisons l'oracle de la manière suivante :

$$T_p = \{((x, c), a) \mid O(c) = a, x \in T\}$$

où T_p est un nouvel ensemble d'entraînement qui, pour chaque configuration c et phrase x, donne une action à exécuter grâce à l'oracle (Nivre, 2005). Le problème de classification peut alors être résumé comme suit : étant donnée une phrase $\mathbf{x}$ et un arbre $\mathbf{y}$, on cherche :

$$\hat{y} = \arg\max_{\mathbf{y} \in \Phi(\mathbf{x})} s(\mathbf{x}, \mathbf{y})$$

où $\Phi(\mathbf{x})$ est l'ensemble de tous les arbres $\mathbf{y}$ possibles étant donnée la phrase $\mathbf{x}$, $s(\cdot)$ est une fonction qui donne un score pour l'arbre $\mathbf{y}$ et $\hat{y}$ est donc le meilleur arbre possible pour cette phrase. La fonction $s(\mathbf{x}, \mathbf{y})$ est décomposable en

$$s(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{m} \text{score}(\mathbf{x}, c_i)$$

avec c_i la configuration à l'étape i. La fonction score peut alors être décomposée comme le produit scalaire d'un vecteur de poids $\mathbf{w} \in \mathbb{R}^D$ et d'une fonction de traits $f(\mathbf{x}, c_i)$ représentant un vecteur de $\{0, 1\}^D$ pour la phrase $\mathbf{x}$ et la configuration courante c_i.

$$\sum_{i=0}^{m} \mathbf{w} \cdot f(\mathbf{x}, c_i)$$

Plus précisément, $f(\mathbf{x}, c_i)$ représente des traits booléens valués à partir de c_i : $f(\mathbf{x}, c_i) = (f_1(\mathbf{x}, c_i), f_2(\mathbf{x}, c_i), \ldots, f_D(\mathbf{x}, c_i))$. Dans cette formulation, chaque fonction f_i est par exemple de la forme :

$$f(\mathbf{x}, c_i) = \begin{cases} 1 \text{ si } c_i.\sigma[0] = \text{Jean et si } c_i.\beta[0] = \text{veut} \\ 0 \text{ sinon} \end{cases}$$

Apprendre le vecteur $\mathbf{w}$ est généralement réalisé au moyen d'un algorithme de classification. Dans le cadre de l'analyse en dépendances, il arrive le plus souvent que cet apprentissage soit fait « en ligne » (*online learning*), c'est-à-dire que chaque exemple d'entraînement $(\mathbf{x}_i, y_i)$ est considéré et le vecteur $\mathbf{w}$ est mis à jour après chaque exemple, contrairement à l'analyse par lot (*batch learning*) où l'ensemble d'exemples d'entraînement T est évalué avant de mettre à jour le vecteur $\mathbf{w}$. L'apprentissage en ligne peut être réalisé par un perceptron moyenné (Rosenblatt, 1958; Freund et Schapire, 1999), structuré (Collins, 2002) ou non. Le processus d'apprentissage est répété un certain nombre de fois via des itérations successives.

Une fois l'apprentissage effectué, trouver le meilleur arbre pour une phrase donnée parmi tous les arbres possibles est un problème dont la solution s'obtient en temps exponentiel. Il existe donc deux méthodes approximatives couramment utilisées lors de la prédiction de la meilleure analyse : une analyse dite gloutonne (*greedy search*) et une analyse par faisceaux (*beam-based search*), qui réduisent drastiquement la complexité de départ et permettent une analyse en temps linéaire.

8.1.3.2 Deux techniques de décodage pour l'analyse par transitions : de la méthode gloutonne à la recherche par faisceaux

L'analyse gloutonne (*greedy parsing*) va produire la meilleure séquence de transitions pour une phrase donnée en prenant chaque décision localement. Pour une configuration c à l'étape i, l'algorithme calcule la meilleure transition possible en prenant en compte c (c'est-à-dire les symboles au-dessus de la pile, les symboles dans la file et la structure partielle A). Une fois cette transition prédite, elle est appliquée et l'algorithme recommence jusqu'à arriver à une configuration finale. L'analyse est gloutonne dans le sens où chaque décision est un optimum local et qu'aucune garantie n'est prise sur le fait que l'ensemble des décisions permettra d'arriver à un optimum global.

L'analyse par faisceau (*beam-based parsing*), quant à elle, permet une exploration plus large de l'espace de recherche en évaluant à chaque étape i jusqu'à b décisions, où b est la taille du faisceau (généralement entre 8 et 64). L'arbre est le produit du meilleur chemin exploré lors de l'analyse, ce qui a pour effet de réduire la propagation d'erreurs, un problème courant dans le cadre d'une analyse gloutonne. L'algorithme permettant une analyse par faisceau est donné via l'algorithme 3, il est issu de (Zhang et Clark, 2011).

Algorithme 3 Analyse par faisceau. Adapté de (Zhang et Clark, 2011)

```
1: Fonction BEAM-SEARCH(c_0, agenda, candidates, b)
2:     candidates ← c_0
3:     agenda ← ∅
4:     Tant que Vrai Faire
5:         Pour can in candidates Faire
6:             agenda ← INSERT(POSSIBLECONFIGURATIONS(candidate), agenda)
7:         Fin Pour
8:         best ← NBEST(AGENDA, 1)
9:         Si ISTERMINAL(best) Alors
10:            Retourne best
11:        Fin Si
12:        candidates ← NBEST(AGENDA, B)
13:        agenda ← ∅
14:    Fin Tant que
15: Fin Fonction
```

L'analyse par faisceau est généralement réalisée via un algorithme d'apprentissage supervisé en ligne. Souvent, un perceptron structuré est utilisé (Collins, 2002). Par ailleurs, la mise à jour du vecteur de poids $\mathbf{w}$, à l'entraînement, est effectuée de deux manières différentes :

— via une mise à jour à la première erreur (*early update*). Cette méthode popularisée par Collins et Roark (2004) consiste à faire une mise à jour lorsque l'analyse correcte (*gold*) n'est plus dans le faisceau puis de passer à la phrase suivante.

— via une mise à jour par violation maximale (*max violation*) (Huang et al., 2012). Les auteurs montrent que la méthode est une généralisation de l'*early update*. Durant l'entraînement, à la fin de l'analyse d'une phrase, si la configuration de référence (*gold*) est sortie du faisceau, l'algorithme d'apprentissage est mis à jour en pénalisant la configuration la plus mauvaise sur l'ensemble de l'analyse. Il y a donc eu une violation maximale sur l'ensemble des configurations évaluées à l'entraînement. Une illustration des deux principes de mise à jour est représentée à la Figure 8.2.

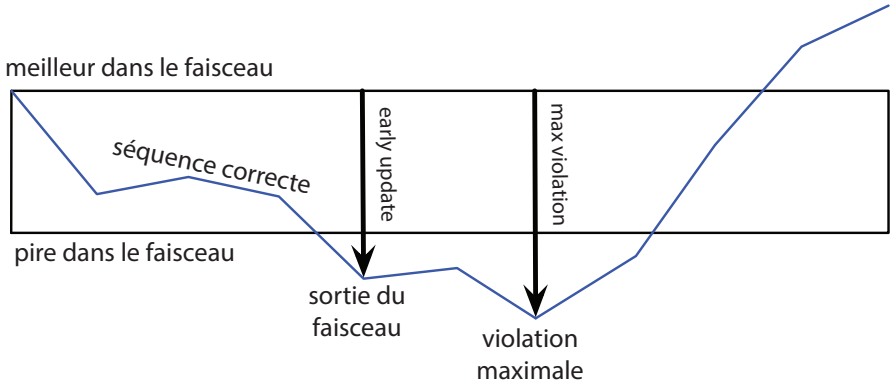


Figure 8.2: Illustration des deux mises à jour couramment répandu pour l'apprentissage automatique et l'analyse par faisceaux en analyse de dépendances (repris de Huang et al. (2012)). La courbe bleue représente la séquence correcte de transitions.

Maintenant que nous avons considéré l'apprentissage d'un oracle et les techniques les plus couramment utilisées pour décoder la meilleure structure étant donné un vecteur de poids $\mathbf{w}$ et une phrase, nous allons nous intéresser à l'extension des méthodes d'analyse sur les arbres pour les graphes et présenter deux analyseurs utilisés durant la thèse ainsi que leur jeu de transitions.

8.1.4 Analyse de graphes : extension de deux analyseurs par transitions existants

Nous avons utilisé deux analyseurs par transitions. Le premier est une extension du modèle proposé par Sagae et Tsujii (2008), le S&T parser, capable d'analyser les DAGs planaires. Nous l'étendons afin de prendre en compte nos représentations linguistiques. Le second est une extension de l'analyseur de Villemonte de La Clergerie (2013a), appelé DyALog-SR ou DSR. Cette extension présentée dans (Ribeyre et al., 2014b) est capable d'analyser certains graphes non planaires grâce à des idées issues de Titov et al. (2009).

8.1.4.1 Analyse de DAGs planaires partiellement connectés : une extension du modèle de Sagae et Tsujii (2008)

Nous étendons l'analyseur de Sagae et Tsujii (2008) pour l'adapter à nos données et à nos besoins. Tout d'abord, nous changeons l'algorithme d'apprentissage automatique. Là où

les auteurs avaient choisi un modèle par maximum d'entropie relativement lent à entraîner, nous lui préférons l'algorithme en ligne du perceptron moyenné (Freund et Schapire, 1999).

Jeu de transitions pour l'analyse de DAGs planaires non connexes. Ensuite, l'amélioration majeure concerne le jeu de transitions initialement prévu pour l'analyse de DAGs planaires connectés (Figure 8.3), que nous avons complété pour les graphes non connexes. Dans la partie supérieure de la Figure, nous reproduisons le système de transitions qui permet de générer des arbres. Ces actions ne sont pas suffisantes pour l'analyse de graphes, car LEFT-REDUCE et RIGHT-REDUCE suppriment un des tokens de la pile. Ce faisant, si le token supprimé possède plusieurs gouverneurs (c'est-à-dire si la structure à analyser est un graphe), il n'est alors plus possible de les attacher. De fait, Sagae et Tsujii (2008) utilisent deux autres actions LEFT-ATTACH$_l$ et RIGHT-ATTACH$_l$ pour gérer ce cas de figure. En effet, ces deux actions suppriment les deux tokens au sommet de la pile, créent un arc, puis soit elles remettent les deux tokens dans la pile (LEFT-ATTACH$_l$), soit seulement l'un des deux tokens (w_i) est remis sur le dessus de la file β (RIGHT-ATTACH$_l$), ce qui permet à w_i d'être attaché à d'autres tokens et donc d'avoir plusieurs gouverneurs. Un exemple d'analyse pour l'annotation profonde de *Jean veut dormir* (Figure 8.4) est donné à la Table 8.5.

	Transition				Pré-conditions
ARBRE	SHIFT	$(\sigma, w_i\|\beta, A)$	$\Rightarrow$	$(\sigma\|w_i, \beta, A)$	$\|\beta\| \neq 0$
	LEFT-REDUCE$_l$	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_i, \beta, A \cup \{(w_i, l, w_j)\})$	$i \neq 0 \wedge \nexists w_i \xrightarrow{l} w_j$
	RIGHT-REDUCE$_l$	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_j, \beta, A \cup \{(w_j, l, w_i)\})$	$\nexists w_j \xrightarrow{l} w_i$
GRAPHE					
	LEFT-ATTACH$_l$	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_i\|w_j, \beta, A \cup \{(w_i, l, w_j)\})$	$i \neq 0 \wedge \nexists w_i \xrightarrow{l} w_j$
	RIGHT-ATTACH$_l$	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_j, w_i\|\beta, A \cup \{(w_j, l, w_i)\})$	$\nexists w_j \xrightarrow{l} w_i$

FIGURE 8.3: Jeu de transitions pour l'analyse de DAGs planaires (repris de Sagae et Tsujii (2008)). La notation $\sigma|w_i$ représente la pile après y avoir ajouté w_i et $w_i|\beta$ représente la file après y avoir ajouté w_i.

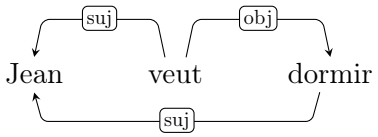


FIGURE 8.4: Graphe pour *Jean veut dormir*.

Bien que ce jeu de transitions soit capable d'analyser des DAGs, sa principale limitation réside dans son incapacité à traiter des nœuds déconnectés du graphe.[4] Or, nos représentations présentent un fort taux de nœuds dits sémantiquement vides et il est donc

4. Par ailleurs, nous relâchons la contrainte pour permettre à l'analyseur de gérer les cycles directs, ce qui sera utile pour l'analyse du DeepFTB.

Transitions	Configurations		
	σ	β	A
	[]	[Jean, veut, dormir]	$\emptyset$
SH	[Jean]	[veut, dormir]	$\emptyset$
SH	[Jean, veut]	[dormir]	$\emptyset$
LA_{SUJ}	[veut, Jean]	[dormir]	$A_1 = \{(\text{veut}, \text{suj}, \text{Jean})\}$
SH	[veut, Jean, dormir]	[]	A_1
LR_{SUJ}	[veut, dormir]	[]	$A_2 = A_1 \cup \{(\text{dormir}, \text{suj}, \text{Jean})\}$
RR_{OBJ}	[veut]	[]	$A_3 = A_2 \cup \{(\text{veut}, \text{obj}, \text{dormir})\}$

Table 8.5: Exemple d'analyse avec le jeu de transition de Sagae et Tsujii (2008) pour la phrase *Jean veut dormir*.
SH = shift, LA = left-attach, LR = left-reduce, RR = right-reduce.

nécessaire de prendre en compte cette particularité. Prouver que le jeu de transitions de Sagae et Tsujii (2008) n'est pas capable d'analyser des DAGs partiellement connectés est trivial. Il suffit de montrer que pour un tel graphe, il n'existe pas d'actions permettant de supprimer de la pile σ (ou de la file β) un nœud sans lui attacher un arc. De fait, il n'est pas possible de gérer le cas où un nœud existe mais est déconnecté du graphe. Pour pallier cette limitation, nous proposons une nouvelle action, appelée POP_0 (Figure 8.5), permettant de supprimer un token au sommet de la pile. Un exemple d'analyse pour la phrase *Jean est respecté* (Figure 8.6) est donné à la Table 8.6.

Transition				Pré-conditions
POP_0	$(\sigma \vert w_i, \beta, A)$	$\Rightarrow$	(σ, β, A)	$\nexists(w_i, l, w_j) \wedge \nexists(w_j, l, w_j)\ \forall l, w_j$

Figure 8.5: Transition pour gérer les DAGs partiellement connectés (intégrée dans le S&T parser).

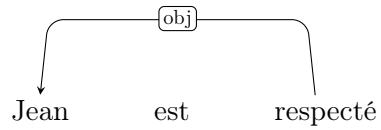


Figure 8.6: Graphe pour *Jean est respecté*.

Traits par défaut utilisés par l'analyseur. Les traits implantés au sein de l'analyseur sont relativement basiques et correspondent en partie à ce qui était proposé par Sagae et Tsujii (2008). Afin d'en donner une définition, nous définissons Word_{β_i} (resp. Lemma_{β_i} et POS_{β_i}), le token (resp. lemme et catégorie morpho-syntaxique) à la position i dans la file. Nous faisons de même pour σ_i, qui représente la position i dans la pile. Soit $d_{i,j}$ la

Transitions	Configurations		
	σ	β	A
	[]	[Jean, est, respecté]	$\emptyset$
SH	[Jean]	[est, respecté]	$\emptyset$
SH	[Jean, est]	[respecté]	$\emptyset$
P_0	[Jean]	[respecté]	$\emptyset$
SH	[Jean, respecté]	[]	$\emptyset$
LR_{OBJ}	[respecté]	[]	$A_1 = \{(\text{respecté}, \text{obj}, \text{Jean})\}$

Table 8.6: Exemple d'analyse avec le jeu de transition de Sagae et Tsujii (2008) et l'action POP_0 pour la phrase *Jean est respecté*.
SH = shift, P_0 = pop_0, LR = left-reduce.

$\text{Word}_{\sigma_1,\sigma_2,\sigma_3}$	$\text{Lemma}_{\sigma_1,\sigma_2,\sigma_3}$	$\text{POS}_{\sigma_1,\sigma_2,\sigma_3}$
$\text{Word}_{\beta_1,\beta_2}$	$\text{Lemma}_{\beta_1,\beta_2}$	$\text{POS}_{\beta_1,\beta_2,\beta_3}$
$\text{leftPOS}_{\sigma_1,\sigma_2}$	$\text{rightPOS}_{\sigma_1,\sigma_2}$	$\text{leftLabel}_{\sigma_1,\sigma_2}$
$\text{rightLabel}_{\sigma_1,\sigma_2}$	a	$d_{12}\ d'_{11}$

Table 8.7: Traits par défaut de l'analyseur.

distance entre un mot Word_{σ_i} et Word_{σ_j}. Nous définissons de même $d'_{i,j}$ comme la distance entre Word_{β_i} et Word_{σ_j}. De plus, nous définissons $\text{leftPOS}_{\sigma_i}$ (resp. $\text{leftLabel}_{\sigma_i}$) comme la catégorie morpho-syntaxique (resp. l'étiquette si elle existe) du mot immédiatement à la gauche de σ_i ; de même pour $\text{rightPOS}_{\sigma_i}$ (resp. $\text{rightLabel}_{\sigma_i}$). Finalement, a représente l'action précédente qui a été prédite par le parser. La Table 8.7 liste les traits par défaut. Nous utilisons des unigrammes, mais aussi des bigrammes et des trigrammes de traits comme il est d'usage de le faire.

Dans la Table 8.7, $\text{X}\sigma_i, \sigma_j, \sigma_k$ est une notation qui exprime que, pour un ensemble $F = \{\text{X}_{\sigma_i}, \text{X}_{\sigma_j}, \text{X}_{\sigma_k}\}$ fixé, un nouvel ensemble $G = \mathcal{P}_1(F) \cup \mathcal{P}_2(F) \cup \mathcal{P}_3(F)$ est créé, G étant l'union des 1-combinaisons, 2-combinaisons et 3-combinaisons de F. Par exemple, $\text{Word}_{\sigma_1,\sigma_2,\sigma_3}$, signifie que les traits suivants sont utilisés : Word_{σ_1}, Word_{σ_2}, Word_{σ_3}, $\text{Word}_{\sigma_1,\sigma_2}$, $\text{Word}_{\sigma_1,\sigma_3}$, $\text{Word}_{\sigma_2,\sigma_3}$, $\text{Word}_{\sigma_1,\sigma_2,\sigma_3}$.

Prédiction des graphes par décodage glouton. Enfin, l'analyseur proposé utilise un décodage glouton. L'espace de recherche n'est donc pas exploré de manière efficace. Or, comme il a été démontré pour l'analyse en dépendances syntaxiques, une exploration efficace de l'espace de recherche est cruciale (Zhang et Nivre, 2011) pour obtenir des prédictions précises. C'est la raison pour laquelle, il nous a semblé intéressant d'utiliser aussi un analyseur qui présente un système de décodage par faisceaux.

8.1.4.2 Analyse de graphes non-planaires : une extension du modèle de Villemonte de La Clergerie (2013a)

Le deuxième analyseur dont nous avons fait usage au cours de cette thèse est une extension de Villemonte de La Clergerie (2013a), un parseur utilisant la recherche par faisceaux, initialement développé pour l'analyse d'arbres en dépendances.

Jeu de transitions pour l'analyse de graphes non-planaires. Parmi les améliorations proposées, quatre nouvelles actions sont proposées (Ribeyre et al., 2014b) (Table 8.8) en plus de celles de Sagae et Tsujii (2008) : notre action POP_0, une action POP_1, l'action SWAP utilisée par Titov et al. (2009) et une action NOOP. [5] L'utilisation des deux actions POP évite d'employer LEFT-REDUCE_l et RIGHT-REDUCE_l. En effet, LEFT-REDUCE et RIGHT-REDUCE sont décomposables en deux sous actions : une action d'attachement (à la manière des actions ATTACH) et une action de suppression de la pile (à la manière des actions POP). Ainsi, l'utilisation d'ATTACH et de POP est nécessaire et suffisante. De plus, POP permet d'avoir deux actions ATTACH symétriques.

Transition				Pré-conditions
SHIFT	$(\sigma, w_i\|\beta, A)$	$\Rightarrow$	$(\sigma\|w_i, \beta, A)$	$\|\beta\| \neq 0$
LEFT-ATTACH_l	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_j\|w_i, \beta, A \cup \{(w_i, l, w_j)\})$	$i \neq 0 \wedge \nexists w_i \xrightarrow{l} w_j$
RIGHT-ATTACH_l	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_j\|w_i, w_i\|\beta, A \cup \{(w_j, l, w_i)\})$	$\nexists w_j \xrightarrow{l} w_i$
POP_0	$(\sigma\|w_i, \beta, A)$	$\Rightarrow$	(σ, β, A)	
POP_1	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_i, \beta, A)$	
SWAP_σ	$(\sigma\|w_j\|w_i, \beta, A)$	$\Rightarrow$	$(\sigma\|w_i\|w_j, \beta, A)$	
NOOP	$(\sigma, [], A)$	$\Rightarrow$	$(\sigma, [], A)$	

TABLE 8.8: Jeu de transitions de DyALog-SR pour l'analyse de DAGs non connectés et non-planaires.
La notation $\sigma|w_i$ représente la pile après y avoir ajouté w_i et $w_i|\beta$ représente la file après y avoir ajouté w_i.

En ce qui concerne l'action SWAP_σ, elle est introduite par Titov et al. (2009) et échange deux tokens adjacents au sommet de la pile σ. L'analyseur effectue alors un réordonnancement en ligne (*online reordering*) des nœuds afin de planariser au mieux la structure. L'idée est une réminiscence de l'algorithme de planarisation décrit par Hajicová et al. (2004) où les arbres non-planaires sont transformés en arbres planaires en réordonnant récursivement leurs sous-arbres pour trouver un ordre linéaire des mots pour lequel l'arbre est planaire. Cependant, comme le font remarquer Nivre (2008) et Titov et al. (2009), cet ordre est garanti d'exister pour n'importe quel arbre. De fait, un arbre non-planaire peut toujours être rendu planaire. Ce n'est pas le cas pour un graphe. Dans le cas général, le théorème

5. Cette action permet de compenser le fait que les chemins induits par la recherche par faisceaux à l'apprentissage ne sont pas tous de même longueur. NOOP permet de garantir que tous les chemins ont la même longueur en produisant des configurations sans contribution pour simuler une transition. Sans cette action, l'analyseur présentait des scores plus faibles pour des faisceaux de grande taille.

de Kuratowski dicte qu'un graphe fini est planaire si et seulement s'il ne contient pas de sous-graphe qui est une expansion [6] de K_5 (le graphe complet à 5 sommets) ou $K_{3,3}$ (le graphe complet biparti à 3+3 sommets). La contrainte est encore plus forte dans le cas qui nous intéresse, puisque nous définissons la planarité comme la possibilité de dessiner le graphe dans un **demi-plan**, or le cas général parle de **plan**, ce qui rend plus libre cette notion. De fait, la transition Swap_σ n'est pas suffisante pour analyser tous les graphes (Titov et al., 2009).

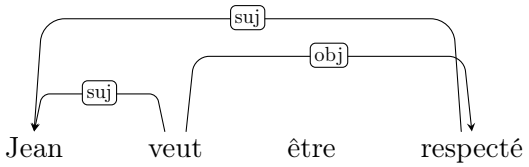


Figure 8.7: Graphe pour *Jean veut être respecté.*

Transitions	Configurations		
	σ	β	A
	[]	[Jean, veut, être, respecté]	$\emptyset$
SH	[Jean]	[veut, être, respecté]	$\emptyset$
SH	[Jean, veut]	[être, respecté]	$\emptyset$
LA_{SUJ}	[Jean, veut]	[être, respecté]	$A_1 = \{(\text{veut}, \text{suj}, \text{Jean})\}$
SH	[Jean, veut, être]	[respecté]	A_1
P0	[Jean, veut]	[respecté]	A_1
SH	[Jean, veut, respecté]	[]	A_1
RA_{OBJ}	[Jean, veut, respecté]	[]	$A_2 = A_1 \cup \{(\text{veut}, \text{obj}, \text{respecté})\}$
P1	[Jean, respecté]	[]	A_2
LA_{SUJ}	[Jean, respecté]	[]	$A_3 = A_2 \cup \{(\text{respecté}, \text{suj}, \text{Jean})\}$

Table 8.9: Exemple d'analyse avec le jeu de transition de DSR pour la phrase *Jean veut être respecté.*
SH = shift, LA = left-attach, RA = right-attach, P0 = pop0, P1 = pop1.

Cependant, les graphes linguistiques que nous traitons ont un taux d'arcs croisés assez faible (de l'ordre de 3% environ), ce qui ne rend pas nécessaire l'utilisation d'un ensemble d'actions plus générales permettant d'analyser tous les types de graphes. En effet, le Swap est une action non-déterministe qui multiplie déjà grandement le nombre de possibilités pour arriver au résultat final. Il arrive que son utilisation puisse pénaliser l'analyseur car l'apprentissage automatique de la meilleure séquence est plus complexe. Nivre et al. (2009) ont notamment montré que réduire le nombre de swap pouvait largement améliorer les scores d'analyse dans le cas de langues à forte non-projectivité comme le tchèque.

Nous illustrons l'analyse de *Jean veut être respecté* (Figure 8.7) à la Table 8.9. L'exemple montre l'importance de l'utilisation des deux actions pop pour supprimer les deux éléments au sommet de pile.

6. L'expansion (ou subdivision) d'un graphe est le résultat de l'ajout d'un ou plusieurs sommets sur une ou plusieurs arêtes (par exemple, transformation de l'arête ·—· en ·—·—·).

Traits par défaut de l'analyseur. DyALog-SR utilise des traits similaires à l'extension de l'analyseur de Sagae et Tsujii (2008) (Table 8.7). Il ajoute néanmoins quelques informations sur les co-parents et tire, entre autres, partie de la valence des gouverneurs (c'est-à-dire le nombre d'arcs sortant, en prenant en compte la relation dépendances ou non) et de celle des gouverneurs.

Une prédiction via une recherche par faisceaux. Enfin, l'analyseur utilise une recherche par faisceaux et la programmation dynamique à la manière de (Huang et Sagae, 2010). À l'entraînement, DyALog-SR est proche de la méthode d'une mise à jour à la première erreur (*early update*, (Collins et Roark, 2004)), mais au lieu de s'arrêter et de passer à la phrase suivante, l'analyse de la phrase continue. Par ailleurs, la mise à jour du vecteur de poids n'est pas faite en pénalisant l'action au score le plus élevé dans le faisceau, mais en déterminant quelle action pénalise le plus l'analyse. Ce faisant, DyALog-SR utilise une méthode mixte entre la méthode de mise à jour dès la première erreur et de violation maximale.

8.2 Prédiction de la syntaxe profonde : un manque de contexte évident

Une fois les deux analyseurs présentés, nous souhaitons tester leur efficacité sur la prédiction de la syntaxe profonde. Avant de discuter les premiers résultats, nous présentons le protocole expérimental et les métriques d'évaluations.

8.2.1 Protocole expérimental

Comme nous l'avons précisé en introduction de ce chapitre. Nous analysons trois corpus dont deux pour l'anglais (DM et PAS dans leur forme bilexicale, alignés au niveau des tokens par Oepen et al. (2014) lors de la campagne d'évaluation (*shared task*) SemEval 2014 [7]). Nous utilisons la répartition suivante : sections 00-19 pour l'entraînement, 20 pour l'ensemble de développement et 21 pour le test. [8] Les corpus contiennent les lemmes et les catégories morpho-syntaxiques prédits par les analyseurs sous-jacents à la représentation de chaque corpus (English Ressource Grammar pour DM (Flickinger, 2000), Enju Parser pour PAS (Miyao et al., 2004)). Enfin, toutes les analyses prédites sont évaluées par rapport au corpus de référence en utilisant les métriques standards de précision, rappel et F_1-score (étiquetés et non étiquetés), en utilisant l'algorithme de scorage de la campagne d'évaluation.

Pour le français, nous utilisons le corpus DeepFTB (cf. section 7.2.2, p. 120) [9] qui est une annotation automatique via notre système de réécriture (p. 105) de la version SPMRL

7. Cet alignement supprime les phrases qui n'ont pas pu être analysées par les deux corpus.

8. Nous utilisons le même découpage que Oepen et al. (2014) pour pouvoir être comparable avec les travaux antérieurs.

9. À noter que nous avons pris le partie d'analyser automatiquement le corpus en l'état et donc de conserver les cycles. En effet, une étape préliminaire du travail a montré que la différence entre le corpus

2014 (Seddah et al., 2014) du French Treebank en dépendances (Candito et al., 2010). Nous réutilisons le découpage SPRML 2014 pour les ensembles d'entraînement, de développement et de test. Cette version comporte les lemmes et les catégories morpho-syntaxiques prédits. Contrairement à l'anglais, la morphologie est fournie et a été prédite. De plus, ces données ont la particularité de considérer les expressions polylexicales (*multi-word expression*) directement au niveau de la structure syntaxique via une étiquette DEP_CPD (cf. Figure 8.8). La prédiction de ce genre d'expressions est un défi. Elle correspond à des difficultés qui peuvent mener à une dégradation des performances de l'analyseur utilisé (Candito et Constant, 2014). Pour pallier en partie ce problème, les données SPMRL 2014 mettent à disposition deux traits morphologiques supplémentaires appelés *mwehead* et *pred*. Le premier donne la tête de l'expression polylexicale et le second indique si un token est issu d'une expression polylexicale. Ces traits sont prédits. À noter aussi que DM marque les expressions polylexicales directement dans la structure de dépendances via une étiquette COMPOUND, mais que nous n'avons pas d'indices de prédiction à notre disposition.

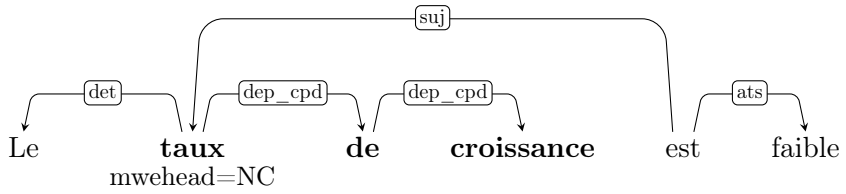


Figure 8.8: Exemple d'annotations d'expressions polylexicales dans les données SPMRL 2014 pour le français.

8.2.2 Des modèles baseline assez faibles

Nous évaluons alors les performances de nos deux analyseurs (S&T parser et DSR) sur les trois corpus (Table 8.10). D'emblée, la différence entre l'approche gloutonne et l'approche par faisceaux est très nette : de 2 points à 2.5 points de différence de F_1-score en moyenne. De plus, la différence est surtout importante au niveau du rappel, nous laissant penser que l'approche gloutonne n'est pas à même de récupérer correctement l'intégralité des dépendances qui composent le graphe au niveau profond. L'exploration de l'espace de recherche n'est sans doute pas suffisante.

Par ailleurs, en comparant les résultats à ceux présentés durant la campagne d'évaluation SemEval 2014 (Oepen et al., 2014) (Table 8.11), nous remarquons que les scores de nos analyseurs sont relativement faibles : -8 points pour S&T parser sur DM par rapport au meilleur système et -6.50 pour DSR sur le même corpus. En effet, toutes les approches qui présentent les meilleurs scores sont des approches globales (utilisant soit des modèles par factorisation d'arcs (McDonald et al., 2005)), soit des méthodes d'ensembles de modèles. Ainsi, Kanerva et al. (2014) utilise une cascade de classifieurs par séparateurs à vaste

cyclique et sa version sans cycle était faible : environ +0.5 pour la version avec cycles. Cela s'explique par le fait que les cycles directs (mod/suj, cf. section 6.3.1.1, p. 87) sont présents en grand nombre et présentent des constructions faciles à identifier pour tous nos analyseurs.

DM (dév.)	LP	LR	LF
S&T parser, baseline	83.66	80.33	81.97
DSR, baseline	84.92	82.80	83.85
PAS (dév.)			
S&T parser, baseline	86.95	83.45	85.17
DSR, baseline	88.80	87.20	87.99
DeepFTB (dév.)			
S&T parser, baseline	76.31	73.08	74.66
DSR, baseline	77.29	76.95	77.12

Table 8.10: Résultats pour TParser et DSR sur les trois corpus avec leurs traits baseline respectifs (dév.).

marge (SVM, (Vapnik, 1995; Scholkopf et Smola, 2001)) qui prédisent successivement les arcs, leurs étiquettes et la racine du graphe. Chaque classifieur utilise un ensemble de traits riches composés entre autres de traits syntaxiques en dépendances. Le classifieur qui prédit les étiquettes sur les arcs utilise aussi des informations issues de plongements lexicaux extraits via word2vec (*word embeddings*, (Mikolov et al., 2013)). Thomson et al. (2014) utilise un système par factorisation d'arcs à base de SVM qui impose certaines contraintes sur le graphe résultant : par exemple, certaines étiquettes ne peuvent être présentes qu'une et une seule fois pour chaque prédicat. Ensuite, Martins et Almeida (2014) utilise un analyseur par factorisation d'arcs mettant en œuvre un décodage par décomposition duale (Martins et al., 2011; Rush et Collins, 2012) et des traits d'ordre supérieur, appelé *TurboSemantic-Parser* ou TParser. La modélisation globale du problème qu'utilise l'analyseur lui permet d'atteindre des scores proche de ceux de Du et al. (2014) qui utilise un ensemble d'analyseurs d'arbres avec des stratégies différentes pour le passage d'un graphe à un arbre. L'utilisation de ces différentes stratégies couplées à des analyseurs par transitions et par factorisation d'arcs ainsi qu'un système simple de vote pour générer le graphe final leur permettent d'établir l'état de l'art durant la *campagne d'évaluation*.

Au vue des différentes stratégies utilisées par les meilleurs modèles, nous pouvons conclure qu'avoir une vision globale du problème peut aider la prédiction de la structure argumentale. De fait, étant donné la plus faible densité d'information de ces corpus, comparés à leurs version surfacique, ajouter du contexte à nos analyseurs via des information décrivant différentes topologies syntaxiques est l'une des techniques qui permettrait de pallier leurs faibles performances. Pour ce faire, nous décidons d'injecter de la connaissance linguistique issue de traits syntaxiques extraits de divers analyseurs surfaciques : un analyseur en constituants, le Berkeley Parser (Petrov et al., 2006) et deux analyseurs

test	PAS	DM
Du et al. (2014)	92.04	89.40
Martins et Almeida (2014)	91.76	89.16
Thomson et al. (2014)	89.63	83.97
DyALog-SR, baseline	87.02	83.91
Kanerva et al. (2014)	87.54	81.53
S&T parser, baseline	84.18	81.17

Table 8.11: Comparaison avec l'état de l'art.

en dépendances, Mate (utilisé pour l'anglais, (Bohnet, 2010)) et FrMG (utilisé pour le français, (Villemonte de La Clergerie, 2010)).

8.3 Prédiction de la syntaxe profonde au moyen de traits syntaxiques : approche à l'état de l'art

Nous décrivons les expériences que nous avons menées afin de mettre en avant l'utilité de connaissances linguistiques riches pour l'analyse de la structure argumentale de la phrase. En effet, l'utilisation d'informations syntaxiques surfaciques issues d'analyseurs divers (en constituants et en dépendances) est une des clés qui permet de faire atteindre des performances élevées aux algorithmes mis en œuvre dans les analyseurs présentés ci-dessus.

Nous décrivons d'abord les traits syntaxiques utilisés avant de conduire nos expériences, d'abord en utilisant le S&T parser et en faisant une analyse détaillée des résultats, avant de regarder l'impact sur DSR. À la suite de ces analyses, nous verrons si l'utilité des traits syntaxiques se confirme sur des modèles globaux tels que le TurboSemanticParser (TParser) de Martins et Almeida (2014).

8.3.1 Description des traits syntaxiques

Nous combinons les traits de nos analyseurs avec différents types de traits syntaxiques (issus d'analyseurs en constituants et en dépendances). Intuitivement, comme nous savons que la syntaxe et la sémantique ne sont pas indépendantes l'une de l'autre, nous pensons que des informations syntaxiques peuvent aider l'analyse de structures en syntaxe profonde. Nous utilisons les traits syntaxiques suivants :

— **Fragments d'arbres en constituants.** Ce sont des fragments d'arbres syntaxiques prédits par l'analyseur de Petrov et al. (2006) (le *Berkeley paser* ou BKY) entraîné par *jackniffing* sur 10 plis (*10-fold jackniffing*) et extraits via l'algorithme de linéarisation proposé lors de la campagne d'évaluation CoNLL 2009 (Hajič et al., 2009). Les fragments peuvent être considérés comme des parties du discours améliorées, car leur nombre est relativement restreint.

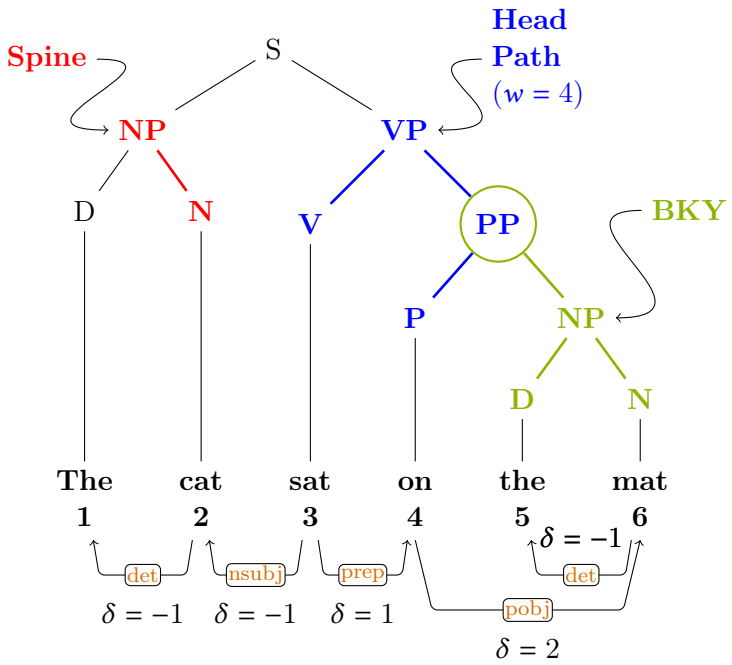


Figure 8.9: Schéma des traits syntaxiques. Le cercle autour de PP marque son appartenance à deux types de traits: BKY (fragment d'arbres) et Head Path.

— **Épines dorsales élémentaires (*spines*).** Nous avons extrait deux grammaires dorsales (*spinal grammar*) (Seddah, 2010), une pour l'anglais et une pour le français, en utilisant une table de percolation des têtes (Magerman, 1995) : celle issue de l'analyseur Bikel (Bikel, 2002) est utilisée pour l'anglais [10] et celle de Dybro-Johansen (2004) est utilisée pour le français. Les épines dorsales (*spines*) sont assignées de manière déterministe et représentées en rouge sur le schéma 8.9. Elles peuvent être considérées comme le chemin entre la catégorie morpho-syntaxiques d'un token de la phrase et sa projection maximale.

— **Dépendances prédites.** Ce sont les dépendances prédites par l'analyseur Mate (Bohnet, 2010) entraîné sur la version en dépendances surfaciques de type Stanford (de Marneffe et Manning, 2008) du Penn Treebank (Marcus et al., 1993). Nous utilisons les étiquettes de dépendances ainsi qu'une notation représentant la distance δ entre deux nœuds liés par un arc (et donc la taille de l'arc), le signe de l'entier généré représente la direction de l'arc. δ est calculé par $t - h$ où t est l'indice du token gouverné et h l'indice du token tête. Des exemples sont donnés en brun sur la Figure 8.9. Pour l'anglais, nous avons aussi exploité ces traits adjoints à la catégorie morpho-syntaxique du token tête (HPOS). L'idée était d'évaluer le pouvoir informationnel de représentations syntaxiques un peu plus absraite, puisque une

10. Nous avons utilisé une version légèrement modifiée pour éviter que certains déterminants ne soient considérés comme têtes dans des configurations particulières.

paire ⟨dépendance, cat. morph.⟩ peut être vue comme généralisant un ensemble de sous-arbres en constituants. Pour le français, nous utilisons aussi les dépendances bilexicales syntaxiques issues d'un analyseur TAG (Joshi et al., 1975) à large couverture, reposant sur une méta-grammaire écrite manuellement (FrMG (Villemonte de La Clergerie, 2010)) avec désambiguïsation apprise sur le French Treebank. Les prédictions TAG de l'analyseur sont automatiquement converties en dépendances bilexicales. L'idée est d'avoir un analyseur qui sache exploiter des sources diverses et notamment les cadres de sous-catégorisation présents dans le Lefff (Sagot, 2010). Par ailleurs, FrMG supporte le traitement de certains types de coordinations elliptiques, ce qui lui confère une prédiction des dépendances longue distance meilleure que certains analyseurs stochastiques.

— **Chemins dans l'arbre en constituants.** Inspirés de Björkelund et al. (2013), nous avons utilisé les dépendances de Mate pour extraire le chemin non dirigé le plus court entre un token et sa tête. Si deux tokens sont liés par une relation de dépendances, alors nous cherchons le chemin le plus court dans les constituants pour l'extraire. Nous exploitons aussi une information numérique w représentant le nombre de nœuds qui composent ce chemin (PATHS) (en bleu sur la Figure 8.9).

L'idée globale derrière les traits syntaxiques est la suivante : nous souhaitons utiliser des informations issues de divers types de ressources (arbres en constituants et arbres en dépendances) pour donner à nos analyseurs sémantiques un pouvoir de généralisation en réutilisant des informations syntaxiques. Les épines dorsales sont alors vues comme des *supertags* déterministes (à la manière des catégories CCG, par exemple) et apportent un contexte vertical puisqu'elles représentent le chemin entre un token et sa projection maximale dans la phrase. Les chemins en constituants, quant à eux, donnent un contexte horizontal et généralisent les dépendances bilexicales syntaxiques. Ils permettent aussi de donner des indices en cas de dépendances longue distance, en décomposant une dépendance longue en une série de nœuds non-terminaux dans l'arbre en constituants, ce qui est utile pour les analyseurs dont la propagation est forte ou l'exploration de l'espace de recherche insuffisante, comme les analyseurs gloutons par transition.

Les dépendances bilexicales syntaxiques, enfin, modélisent l'interface entre la syntaxe et la sémantique et apportent des indices à l'analyseur sur des constructions qui ont une transformation régulière de la syntaxe vers la sémantique. Par exemple, dans le cas du schéma d'annotation DM, la dépendance entre le nom et le déterminant est toujours inversée par rapport à la représentation syntaxique, les dépendances bilexicales syntaxiques *à la* Mate capturent cette transformation. Nous donnons aux Tables 8.12 et 8.13, les comptes pour chaque type de traits syntaxiques et ce sur chaque ensemble (entraînement, développement et test).

8.3.2 Expériences et résultats

Nos expériences se fondent sur l'évaluation de la combinaison de quatre types de traits syntaxiques décrits à la section 8.3.1 : (i) fragments d'arbres (BKY) (ii) dépendances Mate

	Fragments	Mate labels+δ	Dorsales	Chemins
Train	648	1305	637	27 670
Dev	272	742	265	3 320
Test	273	731	268	2 389

Table 8.12: Statistiques sur les types de traits syntaxiques sur l'anglais.

	Fragments	FrMG labels+δ	Dorsales	Chemins
Train	339	1550	635	19 240
Dev	189	691	266	3 507
Test	229	807	352	5 746

Table 8.13: Statistiques sur les types de traits syntaxiques sur le français.

prédites (bn) remplacées par les prédictions FrMG pour le français (frmg) (iii) épines dorsales élémentaires (spines) (iv) chemin dans les constituants du gouverneur au gouverné (paths).

Résultats sur l'anglais. Ils sont donnés aux Tables 8.14 et 8.15. Toutes les améliorations à partir de la baseline sont statistiquement significatives avec un seuil observé (*p-value*) $p < 0.05$. Il n'existe cependant pas de différence significative entre nos deux meilleurs modèles pour chaque treebank. [11]

L'utilisation de chaque type de traits indépendamment augmente les performances de l'analyseur par rapport à la baseline de 0.5 points pour les traits bn sur dm, jusqu'à 1.44 pour les paths et de 1.10 pour les spines à 1.85 pour les paths sur pas. Lorsque sont considérés la conjugaison de deux types de traits dans la Table de dm, il semble que les traits extraits des dépendances bénéficient du contexte supplémentaire ajouté par les traits extraits des constituants, ce qui donne une augmentation de 2.29 points pour bky+spines. Plus intéressant encore, le gain maximal est obtenu par l'addition de traits de topologie différentes : avec les spines (+2.80) qui représentent des structures verticales et bky (+2.76) qui représentent des fragments d'arbres souvent larges (c'est-à-dire qui couvrent plusieurs tokens). En ce qui concerne pas, nous observons des tendances similaires même si les gains sont plus distribués. Contrairement à dm où la conjugaison de plusieurs types de traits donnent des résultats inférieurs, une tendance inverse sur pas est observée, puisque l'utilisation de quatre types de traits différents conduit au deuxième meilleur modèle (all(hpos) = bky+bn(hpos)+spines+paths, +2.82) alors qu'en enlevant les spines, une amélioration légère est observée à +2.92. En effet, ajouter trop de traits dans le modèle dégrade quelque peu les scores, surtout en ce qui concerne dm qui a un jeu d'étiquettes plus large à

11. Nous avons testé la significativité statistique entre nos modèles et la baseline avec un *paired bootstrap test* (Berg-Kirkpatrick et al., 2012).

DM (dév.)	LP	LR	LF	
BASELINE	83.66	80.33	81.97	
BN	84.12	80.91	82.48	+0.51
BKY	85.10	81.70	83.36	+1.39
SPINES	84.72	81.31	82.98	+1.01
PATHS	85.15	81.74	83.41	+1.44
BN(HPOS)	**85.63**	**82.19**	**83.88**	**+1.91**
BKY+SPINES	85.41	81.88	83.61	+1.64
SPINES+PATHS	85.49	82.01	83.71	+1.74
BKY+BN	85.47	82.08	83.74	+1.77
BKY+PATHS	85.70	82.22	83.92	+1.95
BN(HPOS)+SPINES	85.94	82.48	84.17	+2.20
BKY+BN(HPOS)	85.96	82.46	84.18	+2.21
BN(HPOS)+PATHS	85.97	82.59	84.25	+2.28
BN+SPINES	86.05	82.55	84.26	+2.29
BN+PATHS	**86.05**	**82.64**	**84.31**	**+2.34**
BKY+SPINES+PATHS	85.64	82.23	83.90	+1.93
BKY+BN+SPINES	85.88	82.50	84.16	+2.19
BKY+BN(HPOS)+SPINES	86.38	82.81	84.56	+2.59
BN(HPOS)+SPINES+PATHS	86.28	82.91	84.56	+2.59
BKY+BN(HPOS)+PATHS	86.49	82.94	84.68	+2.71
BKY+BN+PATHS	**86.55**	**82.98**	**84.73**	**+2.76**
BN+SPINES+PATHS	**86.59**	**83.02**	**84.77**	+2.80
ALL	85.73	82.27	83.96	+1.99
ALL(HPOS)	**86.13**	**82.64**	**84.35**	**+2.38**

Table 8.14: Résultats et gains sur DM pour S&T parser (dév.).

l'entraînement que PAS. L'explosion du nombre de traits dans le modèle est alors la cause principale de cette dégradation due au fléau de la dimension (*curse of dimensionality*). Globalement, les résultats montrent que les informations syntaxiques sont bénéfiques pour nos analyseurs. Comme chaque type de traits représente un fragment unique d'information, les systèmes bénéficient de ces informations qui apportent plus de structures à un problème complexe d'analyse sémantique.

Résultats sur le français. Ils sont donnés à la Table 8.16. Toutes les améliorations à partir de la baseline sont statistiquement significatives avec un seuil observé (*p-value*) $p < 0.05$.

Chaque type de traits syntaxiques améliore les performances de notre analyseur. Nous notons notamment des améliorations importantes avec les PATHS [12] et les dépendances FRMG. Notre analyseur étant glouton par essence, l'utilisation des PATHS lui donne le contexte manquant pour améliorer ses performances. En outre, l'utilisation de FRMG apporte des améliorations complémentaires, car cet analyseur exploite certaines informations

12. Les PATHS sont calculés avec les dépendances prédites par Mate et par Berkeley. En effet, les performances des PATHS générés avec FrMG sont légèrement inférieurs.

PAS (dév.)	LP	LR	LF	
BASELINE	86.95	83.45	85.17	
SPINES	88.15	84.47	86.27	+1.10
BN	88.21	84.77	86.46	+1.29
BN(HPOS)	88.55	85.00	86.74	+1.57
BKY	88.63	84.97	86.76	+1.59
PATHS	**88.85**	**85.24**	**87.01**	**+1.84**
BKY+SPINES	88.84	85.20	86.98	+1.81
SPINES+PATHS	89.04	85.45	87.21	+2.04
BN(HPOS)+SPINES	89.18	85.49	87.30	+2.13
BN(HPOS)+PATHS	89.17	85.62	87.36	+2.19
BN+PATHS	89.32	85.74	87.49	+2.32
BKY+PATHS	89.44	85.72	87.54	+2.37
BKY+BN	89.30	85.87	87.55	+2.38
BN+SPINES	89.48	85.81	87.60	+2.43
BKY+BN(HPOS)	**89.49**	**85.80**	**87.61**	**+2.44**
BKY+SPINES+PATHS	89.35	85.54	87.40	+2.23
BKY+BN+SPINES	89.56	86.02	87.75	+2.58
BN(HPOS)+SPINES+PATHS	89.76	86.15	87.92	+2.75
BN+SPINES+PATHS	89.88	86.13	87.96	+2.79
BKY+BN+PATHS	89.82	86.20	87.97	+2.80
BKY+BN(HPOS)+PATHS	**89.93**	**86.32**	**88.09**	**+2.92**
ALL	89.70	86.11	87.87	+2.70
ALL(HPOS)	**89.91**	**86.14**	**87.99**	**+2.82**

Table 8.15: Résultats et gains sur PAS pour S&T parser (dév.).

de sous-catégorisation du Lefff (Sagot, 2010) : même si elles ne sont pas directement accessibles dans les étiquettes de dépendances, nous savons d'après les expériences de transformations effectuées à la section 7.3 (p. 123), que certaines prédictions sont meilleures que pour les analyseurs stochastiques, notamment pour les sujets. FrMG exploite des informations en internes sur les sujets impersonnels ainsi qu'un outil permettant de repérer ce genre de cas (Danlos, 2005). D'autre part, FrMG exploite mieux les informations sur les expressions polylexicales ce qui peut faire une différence lors de la prédiction sémantique puisque ces expressions sont directement intégrées à la structure sémantique (comme pour DM).

Ensuite, la combinaison de différents types de traits améliore encore les performances de l'analyseur. Il est intéressant de remarquer que combiner des informations issues des constituants seulement n'est pas d'une grande utilité (seulement +0.67 pour BKY+SPINES). À l'inverse, l'utilisation de traits issus des constituants et des dépendances apporte un gain important allant de +1.60 à +3.92. Sans surprise, la combinaison de nos deux meilleurs types de traits (FRMG et PATHS) donne les meilleurs scores. De plus, il est intéressant de voir que l'augmentation est presque déductible via la première partie de la Table : seules, les dépendances FRMG améliorent de +2.55 et PATHS de +1.71. L'addition de ces deux scores donne +4.26 seulement 0.24 au-dessus des scores obtenus à partir de FRMG+PATHS

DeepFTB (dév.)	LP	LR	LF	
BASELINE	76.31	73.08	74.66	
SPINES	76.99	73.66	75.29	+0.63
BKY	77.26	73.79	75.49	+0.83
BN	77.15	74.00	75.54	+0.88
PATHS	78.06	74.74	76.37	+1.71
FRMG	**78.97**	**75.52**	**77.21**	**+2.55**
BKY+SPINES	77.07	73.66	75.33	+0.67
BKY+PATHS	77.83	74.48	76.12	+1.46
SPINES+PATHS	77.90	74.54	76.18	+1.52
BKY+BN	78.00	74.60	76.26	+1.60
BN+SPINES	77.96	74.66	76.28	+1.62
BN+PATHS	78.24	74.93	76.55	+1.89
BKY+FRMG	78.66	75.12	76.85	+2.19
FRMG+SPINES	79.74	76.17	77.91	+3.25
FRMG+PATHS	**80.41**	**76.84**	**78.58**	**+3.92**
BKY+BN+SPINES	78.06	74.65	76.32	+1.66
BKY+SPINES+PATHS	78.07	74.77	76.39	+1.73
BKY+BN+PATHS	78.56	75.25	76.87	+2.21
BN+SPINES+PATHS	78.57	75.31	76.91	+2.25
BKY+FRMG+SPINES	78.81	75.41	77.07	+2.41
FRMG+SPINES+PATHS	**80.57**	**76.95**	**78.72**	**+4.06**
BKY+FRMG+PATHS	**80.57**	**77.02**	**78.76**	**+4.10**
ALL(BN)	78.29	74.99	76.60	+1.94
ALL(FRMG)	79.32	75.91	77.58	+2.92

Table 8.16: Résultats et gains sur le DeepFTB pour S&T parser (dév.).

suggérant alors que les informations apportées par ces types de traits sont complémentaires et non de même sorte. Ces résultats sont confirmés puisque les deux meilleurs modèles sont la combinaison de trois types de traits incluant toujours les dépendances FRMG et les PATHS.

Au vu des diverses expériences menées sur l'anglais et sur le français, nous pouvons affirmer l'utilité des traits syntaxiques pour la prédiction de structures prédicat-argument. Les améliorations sur les deux langues montrent des tendances similaires : les meilleurs modèles sont globalement les mêmes (surtout pour DM et DeepFTB), et les tendances observées sont généralisables aux deux langues. En effet, il est à noter que la combinaison de quatre types de traits syntaxiques génère une dégradation des performances pour DM et DeepFTB. C'est très clair pour le DeepFTB qui ne bénéficie pas d'un corpus d'entraînement aussi important que pour les deux autres ressources. Nous conjecturons que cette perte de performances est due au fléau de la dimension (*curse of dimensionality*) puisque, la combinatoire augmentant rapidement, l'éparpillement des traits dans le modèle rend plus difficile la construction d'une bonne généralisation. Enfin, l'utilisation d'analyseurs différents pour le calcul des PATHS et des dépendances prédites paraît être une des voies à

explorer, puisque les prédictions semblent être complémentaires et non de même nature.

L'analyse des résultats dessinent un panorama des améliorations provoquées par les traits syntaxiques. Néanmoins, nous souhaitons caractériser davantage le type d'amélioration en dressant une analyse détaillée des améliorations apportées par nos traits par rapport à différents critères facilement quantifiables.

8.3.3 Analyses détaillées des résultats

Pour avoir une vue plus précise des améliorations apportées par les traits, nous utilisons les critères suivants : l'amélioration par étiquette, l'impact des traits sur la longueur des dépendances [13] et enfin nous considérons aussi des constructions connues pour être difficiles à analyser automatiquement.

8.3.3.1 Analyse par étiquette

La première analyse est une analyse des améliorations constatées sur chaque étiquette. Les Figures 8.10a, 8.10b et 8.10c détaillent les scores sur les cinq étiquettes les plus fréquentes dans le corpus. Les scores sont meilleurs sur les étiquettes les plus fréquentes, notamment pour les arguments verbaux pour l'anglais. Dans le cas du français, trois constats sont à faire :

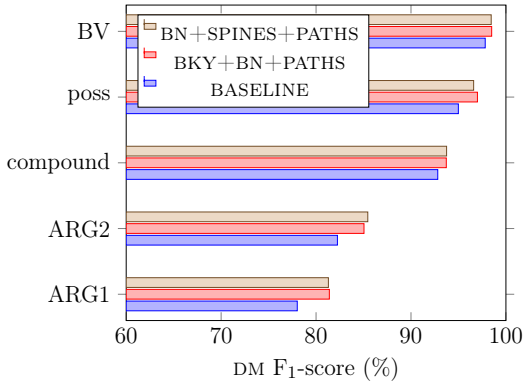
— L'amélioration la plus conséquente se trouve sur la prédiction des expressions polylexicales, comme nous l'avions montré à la section 7.3 (p. 123), l'amélioration est due aux traits syntaxiques issus de FrMG qui prédit ces arcs avec une grande précision.

— Comme sur l'anglais, les arguments verbaux sont mieux prédits, surtout dans le cas des sujets.

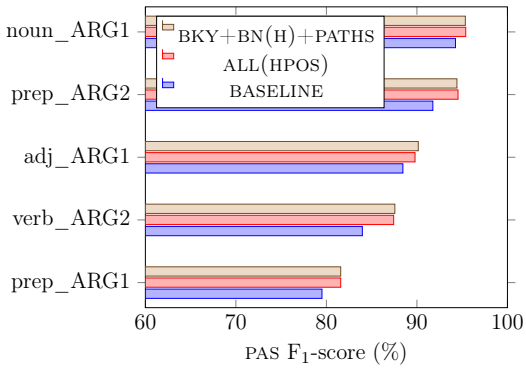
— Les modifieurs, quant à eux, bénéficient aussi des traits syntaxiques, c'est notamment le cas lorsque sont mixées les informations de constituants et de dépendances.

Nous remarquons aussi deux cas intéressants pour DM : les prédictions de *_and_c* et de *ARG3* sont améliorées d'au moins 5 points (Figures 8.11a & 8.11b), montrant que les coordinations sont mieux traitées et que la désambiguïsation des arguments peu fréquents (ou à plus longue distance) est améliorée grâce à l'ajout de traits syntaxiques modélisant un contexte non-local. Ce constat peut être fait aussi sur le DeepFTB (Figures 8.11c et 8.10c) : les coordinations sont mieux traitées mais surtout les modèles sont capables de prédire des arcs *argc* (qui marquent l'argument causateur dans une construction causative). Nous savons que cette construction est peu fréquente dans le corpus (un peu moins de 200 occurrences dans le corpus d'entraînement pour une petite dizaine dans le corpus de développement). Par ailleurs, la différence de scores avec l'utilisation des SPINES (+5 points,

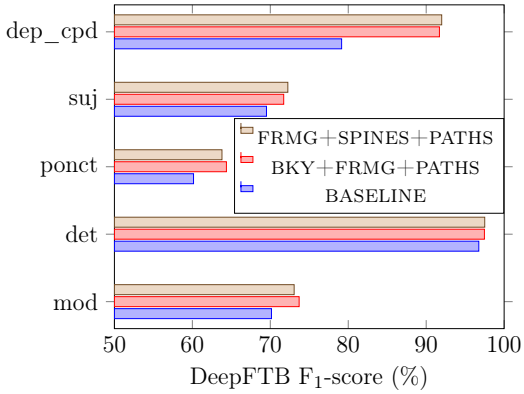
13. La longueur des dépendances l_d est calculée en fonction de la taille des arcs, c'est-à-dire la valeur absolue de la différence entre l'indice de la tête et celui du gouverné : $l_d = |h-t|$ avec h l'indice de la tête et t celui du gouverné. Nous utiliserons le terme de **dépendances longue** (*long dependency*, LD) pour désigner ces dépendances, pour ne pas confondre avec les dépendances longue distance (*long-distance dependency*) linguistiques qui peuvent être de longueur relativement faible. Néanmoins, nombre de dépendances longues vont correspondre à des dépendance longue distance.



(a) Étiquettes les plus fréquentes (DM, dév.).



(b) Étiquettes les plus fréquentes (PAS, dév.)



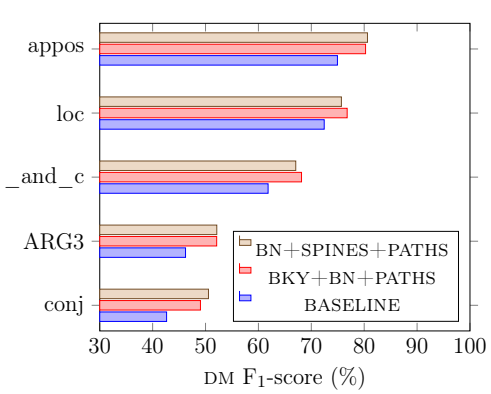
(c) Étiquettes les plus fréquentes (DeepFTB, dév.).

Figure 8.10: Étiquettes les plus fréquentes sur l'ensemble de développement des trois corpus pour le S&T parser.

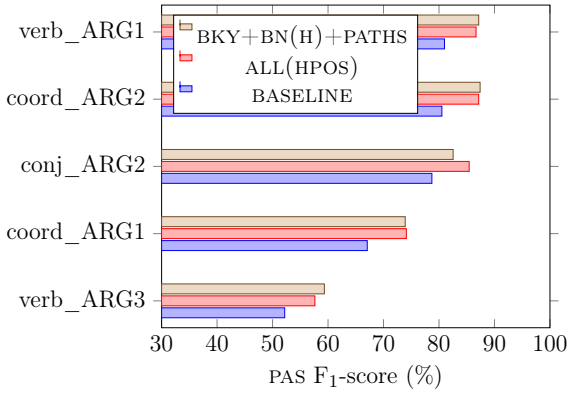
Figure 8.10c) est certainement due à la chance : la faible représentativité du phénomène dans le corpus de développement peut expliquer à lui seul cette différence.

8.3.3.2 Impact sur la longueur des phrases

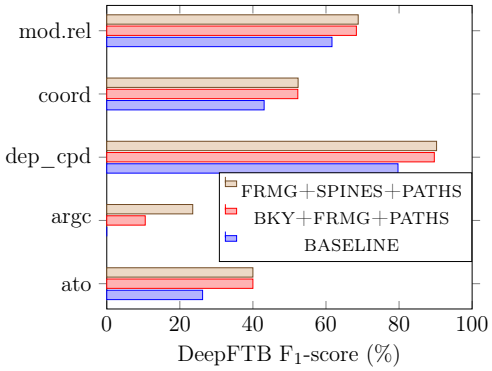
Les phrases les plus longues sont connues pour être difficiles à analyser automatiquement et ce pour nombre de modèles de *parsing*. Les Figures 8.12a, 8.12b et 8.12c montrent l'évolution du F_1-score de nos modèles en fonction de la longueur des phrases (regroupées par groupes de 10) pour les trois corpus. Il est intéressant de noter que l'incrément est plus fort pour les phrases les plus longues, surtout pour les phrases de plus de 50 mots (pour DM). L'utilisation des chemins syntaxiques favorise la capture de dépendances complexes et améliore la phase d'apprentissage sur ces constructions. C'est notamment le cas pour S&T parser qui décode la structure prédicat-argument de manière gloutonne, plus sujette



(a) Étiquettes les plus fortement améliorées (DM, dév.).



(b) Étiquettes les plus fortement améliorées (PAS, dév.).



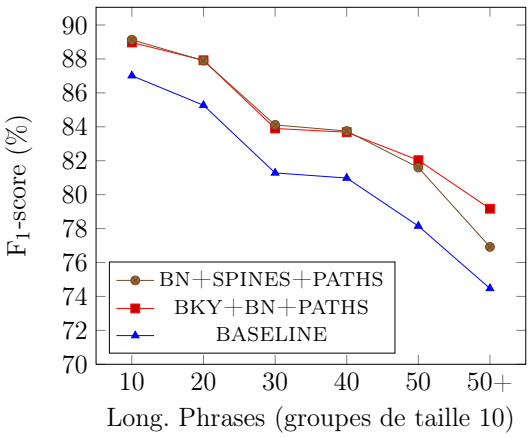
(c) Étiquettes les plus fortement améliorées (DeepFTB, dév.).

Figure 8.11: Étiquettes les plus fortement améliorées sur l'ensemble de développement des trois corpus pour le S&T parser.

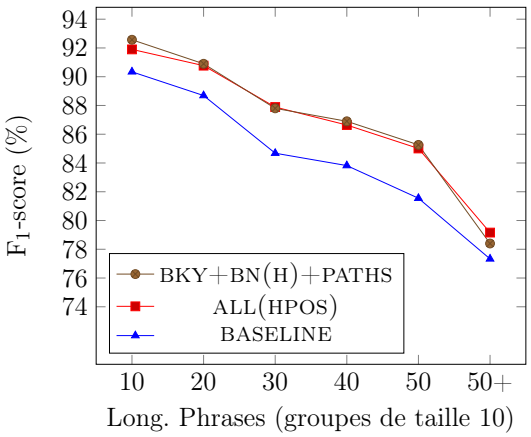
à la propagation d'erreurs. Ce phénomène est particulièrement visible pour le français (Figure 8.12c) où l'amélioration induite par les traits syntaxiques sur les phrases longues est très importante (entre 4 et 5 points). Néanmoins, nous observons tout de même que nos traits syntaxiques ne sont pas capables de pallier la perte de performance induite pour les phrases les plus longues. En effet, la courbe du F_1-score pour les meilleurs modèles suit exactement la même tendance que celle du modèle baseline : plus la phrase est longue, plus les scores sont bas.

8.3.3.3 Constructions linguistiques difficiles : dépendances longues et partage du sujet

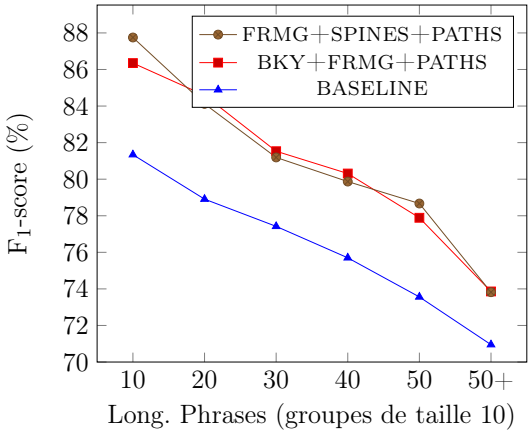
Ensuite, nous nous concentrons sur l'analyse des dépendances longues ou LDs. Nous étudierons d'abord ces dépendances en considérant la longueur des arcs du corpus, c'est-à-dire



(a) Longueur des phrases (DM, dév.).



(b) Longueur des phrases (PAS, dév.).



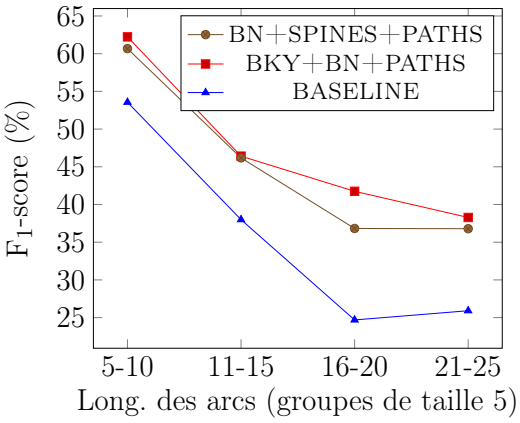
(c) Longueur des phrases (DeepFTB, dév.).

Figure 8.12: Longueur des phrases par groupe de 10 sur l'ensemble de développement des trois corpus pour le S&T parser.

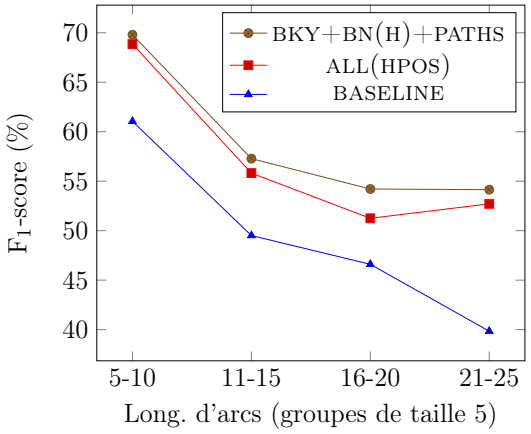
la distance entre deux mots liés par une relation de dépendances. Nous nous concentrerons enfin sur les partages du sujet, pour permettre un traitement des dépendances longues plus proche de la définition linguistique donnée par Cahill (2004) et Rimell et al. (2009) des dépendances longue distance. [14]

Dépendances longues (LDs) par longueur d'arcs. Pour beaucoup de systèmes, les LDs sont difficiles à récupérer parce qu'elles sont en général sous-représentées dans le corpus

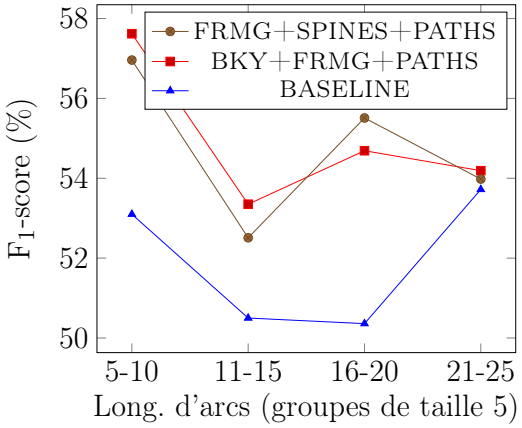
14. En effet, pour avoir une évaluation facilement quantifiable, l'utilisation de la longueur des arcs n'est pas toujours un critère susceptible de récupérer l'intégralité des phénomènes longue distance, car certaines extractions par exemple ne font pas l'objet de dépendances très longues et sont pourtant linguistiquement considérées comme des dépendances à longue-distance.



(a) Dépendances longues (DM, dév.).



(b) Dépendances longues (PAS, dév.).



(c) Dépendances longues (DeepFTB, dév.).

Figure 8.13: Dépendances longues sur l'ensemble de développement des trois corpus pour le S&T parser.

d'entraînement et que ces constructions requièrent souvent des connaissances linguistiques profondes. À la Table 8.17, nous reportons la distribution des dépendances longues par groupes de 5 et ce jusqu'à des longueurs de 40. Elles ne représentent que 15% de tous les dépendances dans les corpus. Les dépendances les plus longues sont les arguments verbaux correspondant aux proto-rôles d'agent et de patient ainsi que les liens marquant des coordinations avec partage du sujet. Par ailleurs, le français favorise les dépendances longues ce qui est reflété par la distribution calculée sur le DeepFTB.

Les résultats présentés aux Figures 8.13a, 8.13b et 8.13c nous indiquent que, sans informations structurelles apportées par les SPINES, les dépendances surfaciques ou les PATHS, la prédiction des dépendances les plus longues n'est pas précise. En revanche, avec l'utilisation de ces traits, nos modèles tendent à s'améliorer drastiquement, avec un gain de

Groupes	5-10	11-15	16-20	21-25	26-40
dm	2 907	734	329	141	91
pas	3 705	1 007	408	175	127
DeepFTB	4 734	1 431	826	479	703

Table 8.17: Nombre d'arcs longue distance (dév.).

près de 25 points pour les dépendances les plus longues sur les corpus de l'anglais (groupes entre 16-20 et 21-25 notamment). Concernant le français, les tendances sont les mêmes quoique moins spectaculaires (environ 6 points), par ailleurs nous n'observons qu'une très faible amélioration pour les longueurs entre 21 et 25. Il est probable que la distribution des dépendances longues dans le corpus favorise leur apprentissage : en effet, ces dépendances sont deux à trois fois mieux représentées en français qu'en anglais. Le système d'apprentissage automatique à la base de notre analyseur a donc plus de chances d'avoir appris une bonne généralisation contrairement à l'anglais où la masse de dépendances longues est extrêmement faible. De fait, des traits syntaxiques complexes apportent toujours une amélioration, mais elle est moindre par rapport aux corpus où ces dépendances sont largement sous-représentées.

Table 8.18, nous donnons les améliorations globales lorsque les arcs de longueurs 5 à 40 sont considérés. Pour les deux corpus dm et pas, l'amélioration est la même (environ 9 points), montrant de nouveau que les informations structurelles apportées par les traits syntaxiques sont la clé de meilleures prédictions. Le DeepFTB ne bénéficie pas autant des informations apportées par les traits syntaxiques, cela est de nouveau corrélé à la distribution des arcs longue distance dans les corpus et au fait que le chevauchement entre la syntaxe de surface et la syntaxe profonde est forte (environ 69% des arcs dans le corpus). Nous constatons tout de même que les tendances entre les deux meilleurs modèles sont les mêmes pour chacun des corpus : les paths combinés aux dépendances surfaciques sont importants alors que les spines tendent à pénaliser les modèles. En effet, il faut souligner que les spines ne sont que des *projections partielles* qui ne reflètent pas d'informations d'attachement. De fait, utilisés seuls, ils ne sont pas capables d'apporter le contexte nécessaire pour prédire les LDs.

Coordinations avec partage du sujet. Nous nous concentrons maintenant sur les partages du sujet. Ils sont extraits au moyen d'un simple motif de graphe, c'est-à-dire deux verbes qui partagent le même rôle agentif (ARG1 ou suj) au sein d'une coordination. La différence entre le nombre de dépendances évaluées sur chaque corpus est due à la manière dont les schémas d'annotation annotent le phénomène. dm opte pour des structures coordonnées avec une chaîne de dépendances prenant racine au niveau du premier conjoint, les

DM (dév.)	LP	LR	LF	
BASELINE	54.95	42.53	47.95	
BN+SPINES+PATHS	64.23	50.55	56.57	+8.62
BKY+BN+PATHS	64.88	50.90	57.05	+9.10

(a) DM Corpus (dév).

PAS (dév.)	LP	LR	LF	
BASELINE	66.62	50.17	57.23	
ALL(HPOS)	74.03	57.58	64.78	+7.55
BKY+BN(HPOS)+PATHS	74.62	58.95	65.86	+8.73

(b) PAS Corpus (dév).

DeepFTB (dév.)	LP	LR	LF	
BASELINE	57.60	47.71	52.19	
FRMG+SPINES+PATHS	60.99	50.92	55.50	+3.31
BKY+FRMG+PATHS	61.58	51.57	56.13	+3.94

(c) DeepFTB Corpus (dév).

Table 8.18: Évaluation globale des dépendances longue distance pour S&T parser (dév).

DM (dév.)	LP	LR	LF	
BASELINE	90.00	48.57	63.09	
BN+SPINES+PATHS	96.02	53.65	68.84	+5.85
BKY+BN+PATHS	96.07	54.29	69.37	+6.28

(a) sur DM (dév., 315 dépendances).

PAS (dév.)	LP	LR	LF	
BASELINE	97.51	61.48	75.41	
ALL(HPOS)	97.86	64.78	77.96	+2.55
BKY+BN(HPOS)+PATHS	98.57	65.09	78.41	+3.00

(b) sur PAS (dév., 636 dépendances).

DeepFTB (dév.)	LP	LR	LF	
BASELINE	95.97	53.72	68.89	
FRMG+SPINES+PATHS	96.90	56.43	71.33	+2.44
BKY+FRMG+PATHS	96.90	56.43	71.33	+2.44

(c) sur DeepFTB (dév., 443 dépendances).

Table 8.19: Évaluation sur les partages du sujet pour S&T parser (dév).

conjonctions de coordination étant considérées comme sémantiquement vides. Pour PAS, la dernière conjonction de coordination et chaque conjonction est un prédicat bivalent prenant le conjoint gauche et droit comme argument. Le DeepFTB adopte la même annotation que le French Treebank surfacique : le premier conjoint est la tête de la coordination et la conjonction de coordination est conservée.

Les scores de nos meilleurs modèles sont reportés aux Tables 8.19. Une fois encore, les modèles améliorent le F_1-score. Cependant, les améliorations ne sont pas similaires. En effet, DM considère que la conjonction est sémantiquement vide et attache un arc $\_and\_c$ entre les deux verbes coordonnés. Par conséquent, cet arc est bien plus difficile à prédire, puisqu'il n'est pas particulièrement informatif pour nos traits par défaut qui s'appuient majoritairement sur les tokens, les lemmes et les catégories morpho-syntaxiques. Le gain de 6.30 points pour DM (Table 8.19a, resp. +3 pour PAS) indique que l'utilisation d'informations syntaxiques complémentaires apportées par les traits est essentiel lorsque le schéma d'annotation favorise un nombre important de mots sémantiquement vides. Concernant le français, l'amélioration est similaire à PAS, car l'annotation est plus proche de PAS (cf. section 3.2.2, p. 49) que de DM (cf. section 3.2.1, p. 47), en effet le DeepFTB ne considère par la conjonction comme sémantiquement vides, ce qui donne plus d'indices à l'analyseur pour repérer ce genre de constructions. Cela donne une bonne idée du type d'informations requises pour analyser des graphes sémantiques (ou à défaut des structures prédicat-argument) : plus la distance entre le prédicat et son argument est grande, plus le contexte à utiliser pour désambigüiser l'attachement doit être large.

8.3.3.4 Impact de la surface sur les prédictions profondes : une idée préconçue ?

	PAS	DM	DeepFTB
Chevauchement ($\in$ surface)	+2.87	+2.67	+1.96
Partie restante ($\notin$ surface)	+2.70	+2.74	+3.70

Table 8.20: Évaluation de l'amélioration entre la baseline et le meilleur modèle sur le chevauchement surface / profond et sur le reste des arcs.

Il est possible de se demander si les améliorations observées sur nos meilleurs modèles proviennent d'un chevauchement potentiellement fort entre les arbres de surfaces et les structures prédicat-argument. Après tout, la syntaxe profonde reste à mi-chemin entre une structure sémantique pleine et entière et un arbre syntaxique de surface. Il n'est donc pas impossible que les améliorations constatées soient dues à l'amélioration de la prédiction du chevauchement entre surface et profond uniquement. Par exemple, le passage du surfacique vers le profond requiert une large part de changement d'étiquettes (surtout pour les corpus de l'anglais) : *nsubj* devient souvent *ARG1*. Nous testons l'hypothèse en calculant le nombre d'arcs non dirigés, non étiquetés en commun entre les prédictions de notre analyseur de surface en dépendances (Mate ou FrMG) et nos corpus. L'intuition derrière le fait que

l'isomorphisme est calculé sur des arcs non dirigés et non étiquetés est la suivante : prendre en compte la direction de l'arc revient à considérer comme non isomorphe les dépendances qui auraient été inversées, c'est notamment le cas de BV pour DM qui inverse la relation entre le déterminant et le nom en profond. La régularité de ce changement peut être considérée comme une forme d'isomorphisme. En ce qui concerne les étiquettes, les prendre en compte pour le calcul des chevauchements structurels reviendrait à considérer que le chevauchement sur l'anglais est de 0%, car les schémas d'annotations de DM et PAS n'ont aucune étiquette commune avec les prédictions de Mate issues du schéma Stanford (de Marneffe et Manning, 2008) (utilisé pour la prédictions des traits BN), contrairement au schéma d'annotation du DeepFTB qui conserve les fonctions grammaticales.

Le chevauchement représente 30% du nombre d'arcs total pour DM, 27% pour PAS et 69% pour le DeepFTB. Il faut ici distinguer les corpus de l'anglais de ceux du français. Quoiqu'important pour l'anglais, le chevauchement ne représente pas la majorité des arcs. Pour le français, c'est l'inverse, mais nous l'avions déjà considéré, puisque les expériences à la section 7.3 (p. 123) démontraient l'influence de la prédiction surfacique sur la réécriture via les règles d'OGRE. Néanmoins, les résultats à la Table 8.20 montrent que nos meilleurs modèles fonctionnent globalement de la même manière sur les deux sous-ensembles pour l'anglais. Un autre constat intéressant est que, contrairement au modèle pour PAS, le modèle pour DM est meilleur sur la partie profonde seule. Cela nous laisse penser que l'utilisation de traits issus du Penn Treebank est d'une certaine manière moins optimale quand ils sont appliqués sur un treebank qui n'est pas dérivé du Penn Treebank (DM est issu d'une grammaire HPSG dont les règles ont été écrites à la main). Enfin, en ce qui concerne le français, les résultats nous montrent que l'approche est valide, puisque les traits syntaxiques ont un impact plus favorable plus sur la partie qui n'est pas issue du niveau surfacique. Cela s'explique, car les décisions profondes sont plus difficiles à prendre que les décisions surfaciques et l'analyseur a besoin d'être aidé par des traits informatifs.

L'analyse détaillée converge vers une même conclusion : l'utilisation de traits syntaxiques au sein d'un analyseur par transitions glouton améliore la prédiction de structures profondes, même dans une perspective multilingue et sur des corpus qui, bien que présentant des caractéristiques communes, ne sont pas issus des mêmes théories. Cependant, le S&T parser est un analyseur glouton : chaque décision prise est indépendante et répond à un optimum local (la meilleure décision possible au temps t). Nous savons par ailleurs que d'après les baselines de la section précédente qu'il est préférable d'utiliser des méthodes de décodage plus puissantes telles que la recherche par faisceaux (*beam search*). L'idée est de voir l'impact des traits syntaxiques lorsque l'analyseur présente d'emblée de meilleures performances.

8.3.4 Des résultats similaires pour l'approche par faisceaux

Nous reportons les scores pour DSR sur les trois corpus aux Tables 8.21. L'examen des résultats montrant une amélioration constante des performances donne l'observation suivante : le contexte apporté par les traits syntaxiques se retrouve sur un autre analyseur. Bien entendu, les améliorations ne sont pas aussi importantes surtout en ce qui concerne DM (+2.80 pour l'analyse gloutonne et +1.80 pour l'analyse par faisceaux). En revanche, même si nous pouvons dire que les traits syntaxiques sont utiles pour améliorer des approches plus performantes, il n'est pas facile de conclure qu'une combinaison de types de traits syntaxiques est généralisable : l'analyseur par faisceaux utilisé ici bénéficie de l'utilisation des quatre types de traits contrairement aux scores que nous présentions à la Table 8.14. Concernant le cas du français, l'analyse est différente dans la mesure où l'analyse gloutonne et l'analyse par faisceaux bénéficie toutes deux des traits syntaxiques et c'est encore plus vrai pour l'analyse par faisceaux (+6.40) sur le meilleur modèle, même si la baseline de DSR est bien au-dessus de celle du S&T parser (+2.46). Ces résultats semblent indiquer que l'apport de la syntaxe de surface est crucial sur un corpus comme le DeepFTB, ce qui n'est pas étonnant. En effet, comme nous le disions lors de la présentation des corpus et de leurs caractéristiques, DM est sans doute le corpus le plus sémantique parmi ceux que nous avons considérés, alors que le DeepFTB est à mi-chemin entre la syntaxe et la sémantique, il est donc normal qu'il bénéficie de l'apport de l'information syntaxique de surface.

DM (dév.) | beam 4	LP	LR	LF	
BASELINE	84.92	82.80	83.85	
ALL	86.83	84.51	85.66	+1.81

(a) Évaluation sur DM (DSR).
ALL = BKY+FRMG+SPINES+PATHS.

PAS (dév.) | beam 4	LP	LR	LF	
BASELINE	88.80	87.20	87.99	
ALL	91.64	90.17	90.90	+2.91

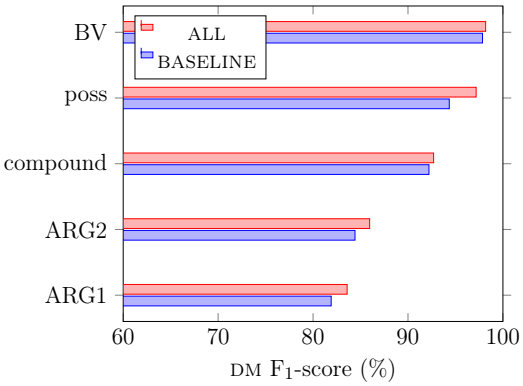
(b) Évaluation sur PAS (DSR).
ALL = BKY+FRMG+SPINES+PATHS.

DeepFTB (dév.) | beam 6	LP	LR	LF	
BASELINE	77.29	76.95	77.12	
BKY+FRMG+PATHS	83.75	83.28	83.52	+6.40

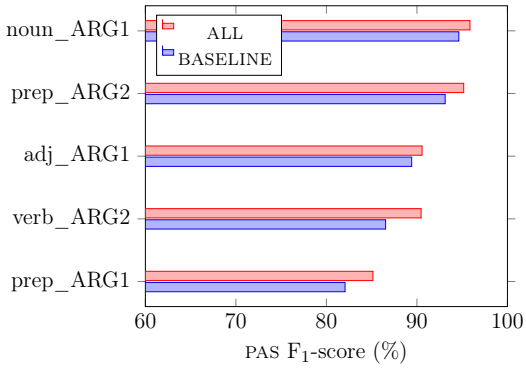
(c) Évaluation sur DeepFTB (DSR).

Table 8.21: Scores pour DyALog-SR sur le corpus de développement (baseline et meilleur modèle).

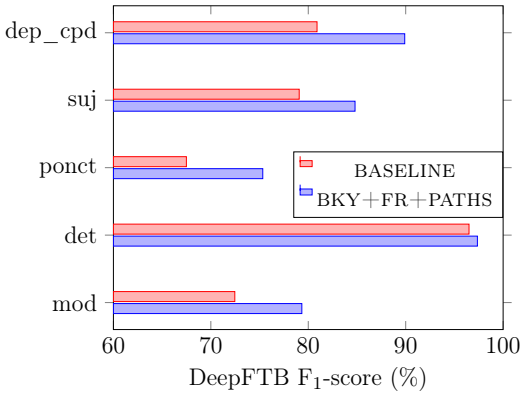
Cependant, une analyse globale n'indique pas en détails les améliorations apportées à l'analyseur. Nous détaillons les résultats en suivant le schéma des améliorations que nous avons pu construire à la section précédente : amélioration sur les étiquettes les plus fré-



(a) Évaluation détaillée sur DM.



(b) Évaluation détaillée sur PAS.



(c) Évaluation détaillée sur le DeepFTB.
FR = FRMG

Figure 8.14: Évaluation sur les étiquettes les plus fréquentes pour DSR (dév).

DM (dév.)	LP	LR	LF	
BASELINE	53.59	42.74	47.00	
ALL	66.23	52.91	58.15	+11.15

(a) DM Corpus (dév).

PAS (dév.)	LP	LR	LF	
BASELINE	70.27	60.35	64.13	
ALL	81.33	73.97	77.09	+12.96

(b) PAS Corpus (dév).

DeepFTB (dév.)	LP	LR	LF	
BASELINE	59.49	55.63	57.50	
BKY+FRMG+PATHS	72.73	68.67	70.51	+13.01

(c) DeepFTB Corpus (dév).

Table 8.22: Évaluation globale des dépendances longues (de longueur 5 à 40) pour DSR (dév.)

quentes et sur les dépendances longues. Les résultats sur les étiquettes Figure 8.14 révèlent que globalement les améliorations constatées sur S&T parser sont reportés sur DSR, laissant penser que les améliorations dues aux traits syntaxiques se répartissent de la même manière. Cela signifierait que l'apport des traits syntaxiques n'est pas aléatoire en fonction des analyseurs, mais donne en général le même genre d'indices pour la prédiction de la structure argumentale. La différence des scores de précision, rappel et F_1-score globaux entre S&T parser et DSR peut donc être expliqué par le fait que cet analyseur présente une baseline assez faible, comparativement à des systèmes présentant de meilleures stratégies de décodage. Ensuite, les résultats sur les dépendances longues (Table 8.22) indiquent que la syntaxe bénéficie plus à l'analyseur par faisceaux qu'à l'analyseur glouton, les performances sont meilleures. Plus intéressant encore, pour PAS, la baseline du DSR présente des scores bien supérieurs à l'analyseur glouton, mais les améliorations sont constantes. Cela nous conforte encore dans l'idée que des traits bien choisis bénéficient tout autant à des analyseurs utilisant des présupposés différents.

Cependant, bien que le décodage de S&T parser et DSR ne soit pas le même, les deux analyseurs reposent sur un même principe d'analyse par transitions. De fait, il est probable que les traits syntaxiques favorisent ce genre d'analyseurs car ils ne sont pas à même d'avoir une vue globale de la structure à décoder, contrairement à des analyseurs par factorisation d'arcs, tels que TParser (Martins et Almeida, 2014). Ainsi, est-il intéressant de regarder le comportement de notre jeu de traits sur cet analyseur.

8.3.5 Analyse par factorisation d'arcs : le modèle de Martins et Almeida (2014)

8.3.5.1 Présentation de la factorisation d'arcs

Contrairement à l'analyse par transitions, l'analyse par factorisation d'arcs (*arc-factored parsing*) vient de la tradition de la théorie des graphes et a été popularisée pour l'analyse syntaxique via des méthodes d'*arbres couvrants de poids maximum* (McDonald et al., 2005), où l'idée principale est de trouver, dans un graphe complet, l'arbre contenant tous les nœuds du graphe (c'est-à-dire couvrant) de poids maximal, c'est-à-dire que l'arbre dont le score est le plus élevé correspond à l'analyse correcte. En effet, les graphes sont parmi les structures les plus étudiées en informatique théorique et nombre d'algorithmes efficaces sont connus. De fait, certains chercheurs (McDonald et al., 2005; McDonald, 2006; Koo et al., 2007; Hall et al., 2011; Lluís et al., 2013; Lluís et al., 2014) se sont concentrés sur la réutilisation d'algorithmes de graphes pour l'analyse de structures syntaxiques. Pour ce faire, le score d'un arbre ou d'un graphe est décomposé en la somme des scores de ces sous-graphes et (pour le modèle le plus simple) en le score de chacun de ses arcs, d'où le nom de *factorisation d'arcs*. Pour le cas non étiqueté, la formalisation est la suivante :

$$\text{score}(G) = \sum_{\psi \in \Psi_G} \text{score}(\psi)$$

où Ψ_G dénote l'ensemble des sous-graphes de $G = \langle V, E \rangle$ avec $E = V \times V$. Ce score représente la probabilité qu'un arbre/graphe particulier soit l'analyse correcte pour une phrase S. Comme nous le disions, dans le cadre le plus simple pour un graphe orienté non étiqueté (définition 7.1, p. 93), les facteurs ψ sont les arcs $(u, v) \in E$ et l'analyse la plus probable revient à trouver (avec $\Phi(\mathbf{x})$ l'ensemble de tous les graphes $\mathbf{y}$ possibles pour une phrase $\mathbf{x}$ donnée) :

$$\arg\max_{\mathbf{y} \in \Phi(\mathbf{x})} \text{score}(G) = \arg\max_{\mathbf{y} \in \Phi(\mathbf{x})} \sum_{(u,v) \in E} \text{score}(u, v)$$

8.3.5.2 Parsing par factorisation d'arcs : vers des méthodes par décomposition duale

Pour les arbres, nombre d'algorithmes sont connus pour obtenir l'arbre de poids couvrant maximal dans un graphe en temps polynomial, parmi lesquels l'algorithme de Chu et Liu (1965); Edmonds (1967) ou encore l'algorithme de Kruskal (1956) qui permettent de trouver une solution au problème en temps polynomial. L'avantage d'une telle approche réside dans le fait que l'analyse peut être vue globalement et les décisions ne sont plus locales comme pour l'analyse par transitions où une décision à l'étape i est conditionnée par la configuration à l'étape $i - 1$. Néanmoins, la généralisation de ces approches aux graphes n'est pas triviale, puisqu'il ne peut exister un algorithme de graphe couvrant de poids maximal. Quand bien même des modèles par factorisation d'arcs simples peuvent exister, ils ne donnent pas de scores comparables aux algorithmes utilisés sur les arbres (Kuhlmann, 2014; Thomson et al., 2014). Ainsi, généraliser cette approche pour l'analyse de graphes est un problème complexe qui nécessite de passer par des méthodes bien plus efficaces comme par exemple la décomposition duale largement popularisée en TAL par Rush et al. (2010), Koo et al. (2010) ou encore Martins et al. (2011).

Pour les graphes, bien souvent, l'explosion combinatoire de tels décodages demandent d'avoir recours à des techniques approximatives telles que l'échantillonnage (Andrieu et al., 2003), l'utilisation d'heuristiques (Zhang et al., 2014) ou encore l'inférence variationelle (Martins et al., 2010). Toutes ces approches posent de nombreux problèmes en terme de performances. L'idéal serait de pouvoir utiliser des techniques de décodage efficaces (algorithme de Chu, Liu et Edmond, décodage en prenant en compte des hypothèses fortes d'indépendances, etc.) en les combinant. C'est à cela que sert la décomposition duale qui permet de combiner un ensemble fini de sous-systèmes en les forçant à trouver un accord sur la prédiction des sous-structures qu'ils ont en commun. Une source classique d'explosion combinatoire réside dans la combinaison de plusieurs modèles (Tromble et Eisner, 2006). Dans le cas limité de la composition de deux modèles (Rush et Collins, 2012), le problème devient le suivant ($\Phi(\mathbf{x})$ représente le nombre de (sous)-graphes possibles pour une phrase $\mathbf{x}$ donnée) :

$$\hat{y} = \arg\max_{y \in \Phi(\mathbf{x})} f_1(z_1) + f_2(z_2)$$

où z_1 et z_2 se chevauchent et sont des vues différentes d'un même problème, le problème d'optimisation est le suivant :

$$\begin{array}{ll} \max & f_1(z_1) + f_2(z_2) \\ \text{avec} & z_1 \in \Phi(\mathbf{x})_1, z_2 \in \Phi(\mathbf{x})_2 \\ \text{s.c.} & z_1 \sim z_2 \end{array}$$

La notation $\sim$ est une relation d'équivalence sur $(\mathcal{Y}_1, \mathcal{Y}_2)$ indiquant que z_1 et z_2 sont en accord sur leurs chevauchements (c'est-à-dire ce sont des contraintes d'accord). Le problème peut être étendu à n modèles en considérant des modèles où les composants sont très simples (un arc, un petit ensemble d'arcs, etc.), appelés parties basiques. Martins et al. (2011) montrent comment étendre le problème à n modèles et comment le problème d'optimisation ainsi posé ne peut pas être facilement décomposé en sous-problèmes. Cela est dû aux contraintes d'accords entre les sous-problèmes. C'est ici que la décomposition duale entre en jeu puisqu'elle permet une approximation en dualisant ces contraintes via l'utilisation de multiplicateurs lagrangiens (cf. (Martins et al., 2010) et (Rush et Collins, 2012) pour plus de détails).

Une fois ces méthodes décrites, il est alors possible de présenter les parties basiques utilisées par l'analyseur de Martins et Almeida (2014) (Figure 8.15). Le décodage de chaque type de parties est donc opéré par des sous-systèmes en utilisant l'algorithme AD^3 décrit dans Martins et al. (2015).

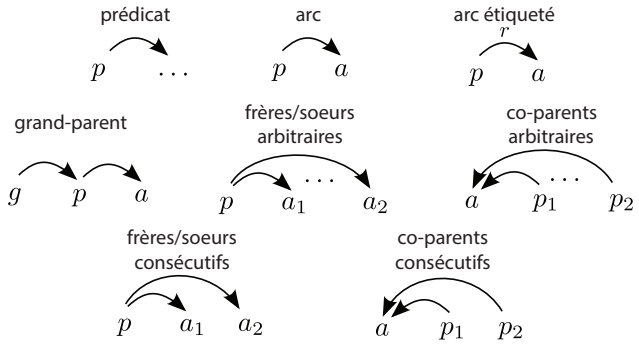


Figure 8.15: Parties utilisées pour l'analyse de graphe par factorisation d'arcs selon Martins et Almeida (2014).

Prédicats et arcs. Chaque partie basique fait l'objet d'un composant unique. L'analyseur utilise un algorithme présentant une complexité en $O(n^2|\mathcal{R}|)$ où $|\mathcal{R}|$ est le nombre d'étiquettes dans le corpus et n le nombre de mots d'une phrase.

Grand-parents, Co-parents et Frères et sœurs arbitraires. Ces parties requièrent l'inclusion d'une paire d'arcs seulement. Chacune de ces types de parties sont gérés en utilisant un facteur par paires (*pairwise factor* appelé PAIR dans AD^3). La complexité est en $O(L^3)$.

Automates de prédicats. C'est une adaptation du modèle d'automate de tête décrit par Alshawi (1996) et réutilisé par Koo et al. (2010). Martins et Almeida (2014) l'adaptent pour l'analyse de structures prédicat-argument de la manière suivante : un automate est introduit par chaque prédicat p et la direction d'attachement (droite ou gauche). Pour un automate droit (la définition d'un automate gauche est analogue), $\langle a_0, a_1, \ldots, a_{k+1}\rangle$ définit la séquence des arguments à droite de p avec a_0 = start et a_{k+1} = end. Le composant ci-dessous qui capture les frères et sœurs consécutifs dans le graphe est alors le suivant :

$$f^{\text{csib}}_{p,\rightarrow}(p \rightarrow a_1, \ldots, p \rightarrow a_k) = \sum_{j=1}^{k+1} \sigma_{\text{csib}}(p, a_{j-1}, a_j).$$

Maximiser $f^{\text{csib}}_{p,\rightarrow}$ a une complexité de $O(L^3)$.

Automates d'arguments. Ils servent à la maximisation des co-parents consécutifs. L'automate est analogue à celui ci-dessus mais les flèches sont inversées. Soit $\langle p_0, p_1, \ldots, p_{k+1}\rangle$ la séquence de prédicats droits qui ont a comme argument (description analogue pour la séquence de prédicats gauches) avec p_0 = start et p_{k+1} = end. On définit alors la formule suivante qui se maximise en $O(L^3)$:

$$f^{\text{ccp}}_{a,\leftarrow}(a \leftarrow p_1, \ldots, a \leftarrow p_k) = \sum_{j=1}^{k+1} \sigma_{\text{ccp}}(a, p_{j-1}, p_j).$$

La grande force d'un système comme celui de Martins et Almeida (2014) réside dans l'intégration de parties d'ordre supérieur (co-parents, grand-parents, frères et sœurs consécutifs, etc.) qui donnent au modèle d'analyse un nombre important d'informations. Contrairement à un modèle ensembliste par votes, la décomposition duale rend possible l'accord des sous-systèmes entre eux par itérations successives. De plus, l'analyseur peut toujours utiliser un jeu de traits particulièrement riche ce qui lui procure une grande flexibilité, puisqu'il est possible d'y intégrer une forte connaissance linguistique.

En effet, nous allons voir maintenant que l'intégration de connaissances linguistiques riches au niveau de l'analyse est la clé d'une bonne prédiction de la structure argumentale.

8.3.5.3 Analyse des résultats

Table 8.23, deux informations apparaissent d'emblée : d'une part, l'analyseur voit ses scores augmentés par les traits syntaxiques, mais d'autre part, l'augmentation n'est pas aussi importante que pour nos analyseurs par transitions. En effet, les traits d'ordre supérieur déjà intégrés au parser lui assure une bonne prédiction, comme le démontre la Table 8.24 où est reproduite une étude (sur les corpus de l'anglais) de l'impact de la syntaxe lorsque les traits d'ordre supérieur sont pris en compte ou progressivement enlevés

DM (dév.)	LP	LR	LF	
BASELINE	90.16	87.15	88.63	
ALL	91.13	89.37	90.24	+1.51

(a) Évaluation sur DM.

PAS (dév.)	LP	LR	LF	
BASELINE	91.69	90.41	91.05	
ALL	92.64	92.01	92.32	+1.27

(b) Évaluation sur PAS.

DeepFTB (dév.)	LP	LR	LF	
BASELINE	83.04	78.80	80.86	
FRMG+SPINES+PATHS	86.32	83.99	85.14	+4.30

(c) Évaluation sur DeepFTB.

Table 8.23: Scores pour TParser sur le corpus de développement (baseline et meilleur modèle).

de l'analyseur. Nous remarquons que sans ces traits, l'apport de la syntaxe est à peu près le même que pour DSR, nous laissant penser que l'utilisation des grand-parents et des co-parents sont une abstraction partielle de nos traits syntaxiques.

	-Traits Synt.	+Traits Synt.	δ
DM, baseline	86.99	89.13	+2.14
+grand-parents	87.66	89.43	+1.77
+co-parents	88.08	89.7	**+1.62**
PAS, baseline	89.73	91.68	+1.95
+grand-parents	90.15	91.92	+1.77
+co-parents	90.93	92.11	**+1.18**

Table 8.24: Étude de l'apport des traits syntaxiques mêlés aux traits d'ordre supérieur ou non. F_1-score étiqueté pour TParser (test). *Baseline = arc-factored + siblings.*

Quoiqu'il en soit, l'utilité des traits syntaxiques en entrée d'un analyseur sémantique est démontré au travers des différentes analyses que nous avons conduites et ce sur plusieurs analyseurs avec des approches différentes. L'idée n'étant pas nouvelle et déjà exploitée en partie, il est important de se comparer aux travaux existants, ce que nous faisons ci-dessous.

8.3.6 Synthèse et comparaison avec les travaux antérieurs

Nous avons montré que, tout comme l'analyse syntaxique, deux axes pouvaient être exploités pour prédire des structures prédicat-argument de qualité :

1. L'utilisation d'une phase de décodage intelligente.
2. L'utilisation d'un jeu riche de traits et notamment des traits extraits à partir d'informations syntaxiques diverses pour amener du contexte aux analyseurs mais aussi

des informations complémentaires.

Pour le moment, nous savons que l'approche est reproductible sur plusieurs analyseurs, schémas d'annotation et langues et qu'elle améliore la prédiction baseline de chacun. Il convient alors de se comparer aux approches existantes, car les corpus de l'anglais ont été exploités dans le cadre d'une campagne d'évaluation SemEval 2014 (Oepen et al., 2014).

8.3.6.1 Réévaluation sur les systèmes de la campagne d'évaluation SemEval 2014

Nous reprenons les scores de la Table 8.11 que nous étendons en ajoutant les nouveaux résultats produits par nos modèles à base de traits syntaxiques. (Table 8.25).

test	PAS	DM
TParser+syntaxe	**92.11**	**89.70**
Du et al. (2014)	92.04	89.40
Martins et Almeida (2014)	91.76	89.16
DyALog-SR+syntaxe	90.13	85.66
Thomson et al. (2014)	89.63	83.97
S&T parser+syntaxe	87.50	83.84
Kanerva et al. (2014)	87.54	81.53
TParser, baseline	**90.93**	**88.08**
DyALog-SR, baseline	87.02	83.91
S&T parser, baseline	84.18	81.17

Table 8.25: Comparaison avec l'état de l'art (SemEval 2014, (Oepen et al., 2014)).

Au vu des résultats [15], nous constatons que l'utilisation d'informations syntaxiques issues des arbres syntagmatiques et des dépendances est une aide précieuse pour la prédiction de structures prédicat-argument. En effet, notre analyseur glouton (S&T parser) rivalise avec des analyseurs globaux comme celui de Kanerva et al. (2014) qui font usage de plongements lexicaux pour améliorer la détection des étiquettes. De part ces résultats, il est clair que l'apport de la syntaxe pour la prédiction sémantique est importante à considérer. Par ailleurs, les analyseurs par transitions réputés moins performants que les analyseurs plus globaux à base de factorisation d'arcs bénéficient grandement de cette approche. DyALog-SR, par exemple, dépasse Thomson et al. (2014) de près de 2 points, alors que leur modèle est optimisé pour avoir une vue sur l'intégralité du graphe : la prédiction n'est pas faite de manière incrémentale à la façon d'un analyseur par transitions. Enfin, comme nous l'avons déjà suggéré à plusieurs reprises, l'utilisation des mêmes traits syntaxiques dans un analyseur exploitant une phase de prédiction intelligente, TParser, nous permet de dépasser l'état de l'art pré-établi. En somme, l'utilisation de traits syntaxiques est tout à fait utile dans le cas d'une baseline présentant déjà des bonnes performances.

15. La campagne d'évaluation pour ces corpus a connu une deuxième version en 2015. Cependant, il n'est pas possible de se comparer aux modèles produits durant la deuxième version, car, comme l'indique Oepen et al. (2015), le nombre de phrases par corpus a été porté à 35 657 pour 802 717 soit 8% de plus que la version 2014. Par ailleurs, le corpus DM utilisé en 2015 est une version améliorée car extraite sur un DeepBank (Flickinger et al., 2012) plus récent.

8.3.6.2 L'utilisation de méthodes à base d'informations syntaxiques : une approche qui a fait ses preuves

Cependant, l'approche consistant à utiliser des informations syntaxiques pour améliorer l'analyse sémantique n'est pas nouvelle et a déjà fait l'objet de nombreux travaux. C'est notamment le cas de Gildea et Palmer (2002) qui utilisent des chemins extraits des arbres en constituants pour l'analyse automatique de PropBank. Nous pouvons encore citer Chen et Rambow (2003) qui utilisent des informations issues d'une grammaire TAG automatiquement extraite sur le Penn Treebank. Ils montrent que l'apport de traits syntaxiques plus profonds que les seules informations de surface font une différence sur la reconnaissance et l'étiquetage des arguments. Du point de vue de la prédiction syntaxique, l'idée d'utiliser la syntaxe en dépendances pour aider la prédiction des arbres syntagmatiques a été explorée par Farkas et al. (2011) qui proposent un modèle à base de traits riches.

Ensuite, des approches à base de méthodes de noyau (*kernel methods*) ont aussi été largement exploitées pour modéliser automatiquement au niveau d'un SVM des informations issues d'arbres en constituants. Nous citons notamment Moschitti et al. (2008) qui exploitent des fonctions noyau à base d'arbres (*tree kernels*) pour intégrer de l'information linguistique au sein même de l'algorithme d'apprentissage automatique. La méthode est appliquée à l'analyse de PropBank. Cette méthode, quoiqu'intéressante, pose un problème majeur lorsque l'on cherche à incorporer des informations syntaxiques issues de sources diverses : il faut redéfinir la fonction noyau afin de garantir que les propriétés satisfassent la condition de Mercer (cf. (Hofmann et al., 2008) pour plus de détails), ce qui enlève de la flexibilité à la méthode.

Enfin, dans un cadre plus général, certaines approches modélisent conjointement l'information syntaxique et l'information sémantique, c'est le cas notamment de nombre de méthodes utilisées pour la prédiction de la syntaxe et de la sémantique de type PropBank exploitant les données des campagnes d'évaluation CoNLL 2008 (Surdeanu et al., 2008) puis 2009 (Hajič et al., 2009). Parmi ces méthodes, nous pouvons citer une méthode par transitions à base de variables latentes et de deux piles (une pour la syntaxe et une pour la sémantique) faiblement synchronisées (Gesmundo et al., 2009; Henderson et al., 2013). L'avantage de cette approche est de proposer une analyse synchronisée de la syntaxe et de la sémantique tout en permettant à l'une et l'autre des représentations d'apporter de l'information en simultané. Nous pouvons aussi citer des approches par factorisation d'arcs comme celles de Lluís et al. (2013) ou Lluís et al. (2014). La première utilise une méthode d'analyse via des graphes bipartites couplée à un algorithme de décomposition duale pour permettre de résoudre le problème d'optimisation posée par la modélisation jointe de la syntaxe et de la sémantique. La seconde emploie une modélisation du plus court chemin dans les dépendances syntaxiques pour favoriser l'analyse sémantique de type PropBank.

8.4 Conclusion

L'utilité d'une approche jointe est cruciale dans le cadre de la prédiction conjointe syntaxe et sémantique ou quand il est intéressant de tirer pleinement partie de l'une ou l'autre des représentations. Cependant, une approche jointe qui mêlerait arbres en constituants, arbres en dépendances et graphes sémantiques demande un travail important en terme de complexité du décodage pour qu'elle soit praticable. C'est pourquoi, nous avons préféré utiliser directement des traits extraits d'analyseurs surfaciques, une approche similaire à une approche de type *pipeline* où chaque sortie d'analyseurs est donné en entrée du suivant.

Néanmoins, nous pensons que modéliser l'interaction de la syntaxe et la sémantique, comme le font les approches jointes représente une méthode qui évite l'utilisation d'un ensemble complexe d'analyseurs. En effet, notre approche améliore certes l'état de l'art, mais fait appel à trois voire quatre analyseurs différents à la manière d'une méthode d'ensemble [16]. Il faut en effet se rappeler que la prédiction de la structure argumentale n'est pas nécessairement un but en soi. En TAL, les applications sont larges et souvent la structure argumentale peut servir à des tâches d'analyse de sentiment ou encore à des fins d'extraction information. À l'heure où les flux textuels sont légions et ininterrompus, il est probable que chercher des approches conceptuellement plus simples n'est sans doute pas une mauvaise chose. Ainsi, avons-nous chercher des techniques d'induction de règles de transformation entre la représentation syntaxique de surface (un arbre) et la représentation de la structure argumentale (un graphe). Pour ce faire, nous avons exploré une méthode tirant partie des travaux dans le domaine de la fouille de sous-graphes fréquents (cf. (Jiang et al., 2013) pour un panorama des techniques), méthode que nous présentons dans le chapitre suivant.

16. Notons que tout est relatif puisque le modèle de Du et al. (2014) présente un ensemble de 19 modèles calculés avec deux analyseurs différents, un pour les graphes (Titov et al., 2009) et un pour les arbres et grâce à l'utilisation de plus de 5 stratégies de transformation d'arbres en graphes, là où notre approche n'utilise « que » 3 analyseurs différents et une phase d'extraction des traits utiles.

Chapitre

9 Induction automatique de règles : de la surface vers le profond

Sommaire

Introduction

Dans ce chapitre, nous présentons une approche pour l'induction automatique de règles de transformation. La méthode que nous proposons utilise des idées issues des grammaires synchrones (Aho et Ullman, 1969) et de la fouille de sous-graphes fréquents (Yan et Han, 2002; Jiang et al., 2013) pour induire le passage d'une représentation surfacique vers une représentation en syntaxe profonde. Pour tester la validité de l'approche, nous réalisons une série d'expériences sur deux corpus de l'anglais et un corpus du français. Nous comparons aussi nos résultats à ceux obtenus par Ballesteros et al. (2014), car leur approche est comparable à la nôtre. Enfin, il faut noter que le développement que nous faisons dans ce chapitre se veut exploratoire. En effet, ce chapitre peut être vu comme une ouverture qui propose une solution préliminaire complètement automatisable de notre système de réécriture décrit au chapitre 7.

9.1 Induction de règles de transformation : un problème complexe

De nombreux travaux ont été menés en TAL pour apprendre le passage d'une représentation vers une autre. D'emblée, l'idée présente un intérêt dans des domaines comme la traduction automatique (*machine translation*) où une langue source doit être transformée en une cible. Par exemple, des approches utilisant des grammaires hors contexte synchrones Chiang (2007) ou des grammaires TAG synchrones (Shieber, 2007; DeNeefe et Knight, 2009) ont été proposées. Néanmoins, la traduction automatique n'est pas le seul domaine pour lequel de telles approches existent. En effet, pour l'interface entre la syntaxe et la sémantique, des méthodes d'induction ont été aussi discutées, c'est le cas avec les TAG synchrones où Shieber et Schabes (1990) proposent une mise en correspondance d'une représentation syntaxique vers sa formule logique ou encore de Nesson et Shieber (2008) qui en proposent une amélioration. Nous pouvons aussi citer les travaux de Wong et Mooney (2006, 2007) qui proposent d'apprendre directement cette mise en correspondance mais se restreignent au corpus de requête sur la base de données GeoQuery (Zelle et Mooney, 1996). Plus récemment, les travaux de Peng et al. (2015) mettent en œuvre une méthode permettant de générer des graphes sémantiques (issus de l'Abstract Meaning Representaion (Banarescu et al., 2013)) à partir d'une phrase et ce au moyen de grammaires par remplacement d'hyperarcs synchrones et de techniques d'échantillonnage de Monte-Carlo à base de chaînes de Markov (Andrieu et al., 2003).

Néanmoins, à notre connaissance, il n'existe que très peu de travaux consacrés à l'interface entre la syntaxe et la sémantique qui se proposent d'apprendre automatiquement le passage d'un arbre syntaxique vers un graphe sémantique. L'approche la plus similaire est sans doute celle de Ballesteros et al. (2014) qui proposent d'induire automatiquement le passage d'une représentation de surface vers une représentation profonde dérivée de la Théorie Sens-Texte (Žolkovshij et Mel'čuk, 1967). Leur approche est testée sur l'AnCora-UPF (cf. section 3.1.2, p. 45). Le problème n'est pas trivial car cela revient à trouver automatiquement le morphisme entre un arbre $T = \langle V, E_T, L_T \rangle$ et un graphe $G = \langle V, E_G, L_G \rangle$ (avec V l'ensemble des nœuds, E l'ensemble des arcs et L l'ensemble des relations de dépendances), à la manière de l'approche SPO. Le morphisme sera donc un sous-graphe H qui déterminera les différences entre T et G.

À partir de cette constatation simple, le problème de *l'induction d'étapes de transformation* revient à trouver la séquence de *décompositions canoniques en sous-graphes* la plus générale sur l'ensemble d'un corpus aligné entre les structures d'entrée et les structures de sortie. Dans le cas qui nous intéresse, l'alignement entre la structure syntaxique et la structure sémantique s'effectue au niveau des nœuds. Pour aboutir à une séquence de décompositions canoniques, nous réutilisons des idées issues du domaine de la fouille de sous-graphes fréquents (cf. Jiang et al. (2013) pour une vue d'ensemble de ce domaine), domaine qui, pour une collection de graphes, trouvent les sous-graphes les plus fréquents

dans les données.

Dans la suite de ce chapitre, nous décrivons d'abord succinctement le domaine de la fouille de sous-graphes fréquents et l'algorithme que nous avons intégré à notre approche, avant de discuter de l'approche elle-même et de conclure par une discussion des résultats et par une comparaison avec les travaux existants.

9.2 Fouille de sous-graphes fréquents : un début de solution ?

Le but premier de la fouille de données en général est d'extraire des informations utiles et statistiquement signifiantes à partir de données. Les données peuvent, bien sûr, être de toute forme : vecteurs, tableaux, textes, images, etc. Ces données peuvent aussi être représentées de nombreuses façons. Dans le cas qui nous intéresse, les données sont des graphes et la fouille de sous-graphes fréquents est en somme l'essence même de la fouille de graphes. L'objectif est d'extraire tous les sous-graphes les plus fréquents dans un jeu de données, sous-graphes dont la fréquence est au-dessus d'un certain seuil, appelé *support* et dénoté par $s_{\min}$.

Le problème peut être décomposé en deux sous-problèmes complexes : (i) isomorphisme de sous-graphes, (ii) énumération de tous les sous-graphes fréquents (génération de candidats). Le premier problème, reconnu NP-complet (cf. section 7.1.2.2, p. 95), peut être résolu en partie grâce à une représentation canonique de tous les graphes (section 9.2.1) et le second par des techniques de génération de candidats (section 9.2.2). Avant d'entrer dans les détails, il est possible de donner une méthode intuitive : l'idée est de faire « grossir » les sous-graphes via une recherche en largeur ou en profondeur dans les graphes composants le jeu de données, c'est le processus de *génération de sous-graphes candidats*. Ensuite, il faut regarder si ces sous-graphes apparaissent suffisamment fréquemment dans les données pour les conserver, étape que le nomme *comptage du support* (*support counting*). Formellement, la recherche de sous-graphes fréquents peut être décrite de la manière suivante : étant donné un graphe $G \in \mathcal{G}$ et un sous-graphe potentiellement fréquent g, soit :

$$\sigma(g, G) = \begin{cases} 1 & \text{si } g \text{ est isomorphe à un sous-graphe de } G \\ 0 & \text{si } g \text{ n'est isomorphe à aucun sous-graphe de } G \end{cases}$$

$\sigma(g, \mathcal{G}) = \sum_{G_i \in \mathcal{G}} \sigma(g, G_i)$, où $\sigma(g, \mathcal{G})$ dénote la fréquence d'occurrences de g dans $\mathcal{G}$, c'est-à-dire le support de g dans $\mathcal{G}$. La recherche de sous-graphes fréquents consiste à trouver chaque graphe g tel que $\sigma(g, \mathcal{G}) \geq s_{\min}$.

9.2.1 Utilisation de représentations canoniques

Le problème central de la fouille de sous-graphes fréquents c'est l'isomorphisme de sous-graphes, puisque pour trouver des sous-graphes fréquents, il faut pouvoir les comparer à tous les sous-graphes d'un certain ordre dans le corpus de graphes considérés. L'idée est de trouver un ordre canonique pour chaque graphe, ce qui facilite les comparaisons.

Le mécanisme le plus simple pour représenter un graphe est d'employer une matrice d'adjacence ou une liste d'adjacence. Cependant, ces représentations à elles seules ne permettent pas de détecter facilement l'isomorphisme de deux graphes, dans la mesure où un même graphe peut être représenté de manière différente. C'est la raison pour laquelle, il convient d'adopter une stratégie d'étiquetage cohérente qui garantit que deux graphes identiques sont étiquetés de la même manière, peu importe l'ordre dans lequel les nœuds et les arcs sont présentés : c'est une représentation dite *canonique*. Un moyen simple pour transformer une matrice d'adjacence en une forme canonique est d'aplatir la matrice d'adjacence en concaténant les lignes et les colonnes de sorte à obtenir un code comprenant une liste d'entiers avec un ordre lexicographique minimal (ou maximal) imposé. Il est ensuite possible de réduire les calculs résultant des permutations de la matrice en utilisant un schéma invariant sur les nœuds (Read et Corneil, 1977) qui permet au contenu d'une matrice d'adjacence d'être partitionné en fonction des étiquettes sur les nœuds.

Parmi toutes les décompositions canoniques, nous pouvons citer le **Code en profondeur minimal**, utilisant un parcours en profondeur (*depth first search*, DFS) sur le graphe (*Minimum DFS Code*) ou M-DFSC. Le M-DFSC est une technique selon laquelle chaque nœud possède un identifiant unique généré via un parcours en profondeur du graphe. Chaque arc du graphe est alors représenté comme un tuple : $\langle i, j, l_i, l_e, l_j \rangle$ où (i) i et j sont les identifiants des nœuds (ii) l_i et l_j sont les étiquettes des nœuds i et j respectivement (iii) l_e est l'étiquette de l'arc connectant i et j. En se fondant alors sur l'ordre lexicographique généré par un parcours en profondeur, le M-DFSC d'un graphe G peut être défini comme l'étiquetage canonique de G (Yan et Han, 2002).

Les codes DFS pour la branche la plus à gauche et la branche la plus à droite du graphe à la Figure 9.1 sont : $\{ (0,1,a,\emptyset,b), (1,2,b,\emptyset,e), (2,3,e,\emptyset,f), (3,4,f,\emptyset,c), (4,2,c,\emptyset,e) \}$ et $\{ (0,9,a,\emptyset,d), (9,10,d,\emptyset,f), (10,11,f,\emptyset,g), (11,9,g,\emptyset,d) \}$.

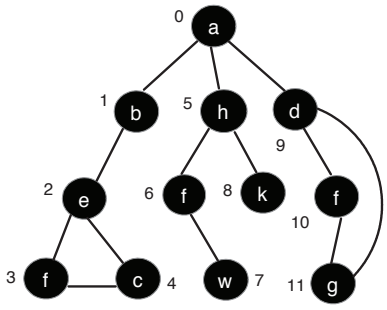


Figure 9.1: Graphe étiqueté d'un parcours en profondeur.

Une fois cet ordre canonique trouvé, il faut pouvoir générer les candidats potentiels avec la contrainte de ne pas générer deux fois le même candidat ou de ne pas générer des

candidats inutiles.

9.2.2 Génération de candidats

Pour ce faire, l'une des solutions les plus couramment admises est d'utiliser une *jointure par niveau* (*level-wise join*). Cette stratégie a été introduite par Kuramochi et Karypis (2001). Intuitivement, un $(k+1)$-sous-graphe [1] est généré en combinant deux k-sous-graphes fréquents qui partagent un même $(k-1)$-sous-graphe. Ce $(k-1)$-sous-graphe commun est appelé le *cœur* des deux k-sous-graphes. Le problème de cette stratégie réside dans le fait qu'un k-sous-graphe peut avoir k différents $(k-1)$-sous-graphes et l'opération de jointure peut générer un nombre important de doublons. Kuramochi et Karypis proposent de limiter les $(k-1)$-sous-graphes aux deux sous-graphes avec l'étiquetage canonique le plus petit, ce qui réduit significativement le nombre de doublons. Il existe bien entendu d'autres approches pour générer des candidats, mais c'est probablement la plus utilisée.

Cependant, des approches ont essayé de passer outre la phase de génération de candidats pour accélérer la découverte de sous-graphes. Nous présentons l'algorithme gSpan (Yan et Han, 2002), sans doute le plus utilisé dans ce domaine, que nous avons intégré à notre système d'induction.

9.2.3 gSpan : un algorithme efficace de fouilles de sous-graphes

gSpan (Yan et Han, 2002) est l'un des algorithmes les plus utilisés en recherche de sous-graphes fréquents. Cet algorithme utilise la représentation canonique du code en profondeur minimal, m-DFSC, décrite ci-dessus. L'algorithme crée un *demi-treillis* de tous les motifs possibles qui résulte en un espace de recherche hiérarchique appelé *arbre-code en profondeur* (*code tree DFS*).

Étant donné un ensemble de graphes $\mathcal{G}$, un *demi-treillis* (*lattice*) est, dans le cadre de la fouille de sous-graphes fréquents, une hiérarchie qui permet de modéliser l'espace de recherche afin de retrouver des sous-graphes fréquents. Chaque nœud représente un sous-graphe connecté d'un graphe dans $\mathcal{G}$. Le nœud le plus bas représente le sous-graphe vide et les nœuds au plus haut niveau représentent les graphes de $\mathcal{G}$. Un nœud p est un parent du nœud q dans le treillis, si q est un sous-graphe de p et que q diffère de p grâce à un arc seulement. Le nœud q est dit le fils de p. Tous les sous-graphes de chaque graphe $G_i \in \mathcal{G}$ sont présents dans le treillis et tous les sous-graphes ne sont représentés qu'une et une seule fois.

Exemple : Étant donné un ensemble de graphes $\mathcal{G} = \{G_1, G_2, G_3, G_4\}$, le treillis correspondant *Lattice*$(\mathcal{G})$ est donné à la Figure 9.2. Le nœud $\emptyset$ au niveau le plus bas (niveau 0) représente donc le sous-graphe vide et les nœuds au plus haut niveau (niveau 3 et 4) représentent les graphes G_1, G_2, G_3 et G_4. Pour exemple, les parents du sous-graphe B–D

1. k réfère à l'élément ajouté successivement (soit un arc, soit un nœud).

sont les sous-graphes A–B–D (on ajoute l'arc A–B) et B–D–G (on ajoute l'arc D–G). De même, les sous-graphes B–C et C–F sont les fils du sous-graphe B–C–F.

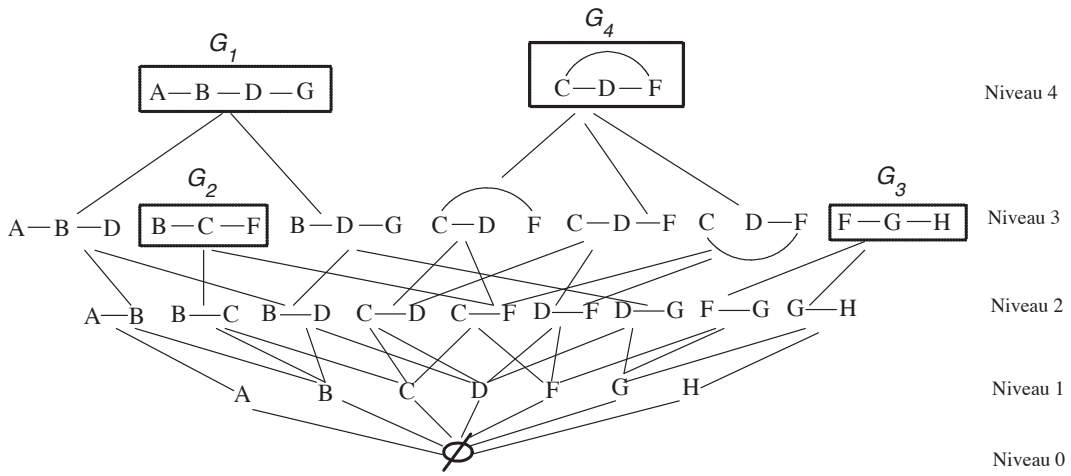


Figure 9.2: Exemple de treillis de sous-graphes pour l'ensemble $\mathcal{G}$.

Dans le cadre de gSpan, chaque nœud de cet arbre de recherche représente un code DFS unique. Le niveau $(k-1)$ de l'arbre possède des nœuds contenant les codes DFS pour k sous-graphes. La génération de candidats est ainsi aisée puisqu'il suffit de supprimer de l'arbre de recherche les nœuds qui n'ont pas un code DFS minimal évitant ainsi une surgénération de candidats. Nous avons utilisé cet algorithme pour sa faible empreinte mémoire, car, selon Yan et Han (2002), l'utilisation d'un parcours en profondeur permet d'utiliser moins de mémoire que l'utilisation (plus courante dans le domaine) d'une recherche en largeur.

Maintenant que nous avons esquissé le problème de l'induction de règles entre des arbres et des graphes et que les bases de la recherche de sous-graphes fréquentes sont posées, nous pouvons détailler l'approche que nous avons définie.

9.3 Apprentissage de règles de transformation : une méthode préliminaire

Avant d'aller plus avant dans les détails, nous donnons une vue globale du problème : la méthode que nous avons développée reprend des idées issues de (Jijkoun et de Rijke, 2007) en les adaptant à nos problèmes. Au départ, nous considérons un corpus aligné d'arbres [2] en entrée et de graphes en sortie. Le but est alors de créer un système capable de convertir l'entrée en sortie. La méthode apprend une séquence de règles de réécriture de graphe grâce à des itérations successives qui se déroulent ainsi :

2. Un arbre étant un cas particulier de graphe, nous utiliserons la terminologie arbre/graphe, sous-arbre/sous-graphe de façon interchangeable.

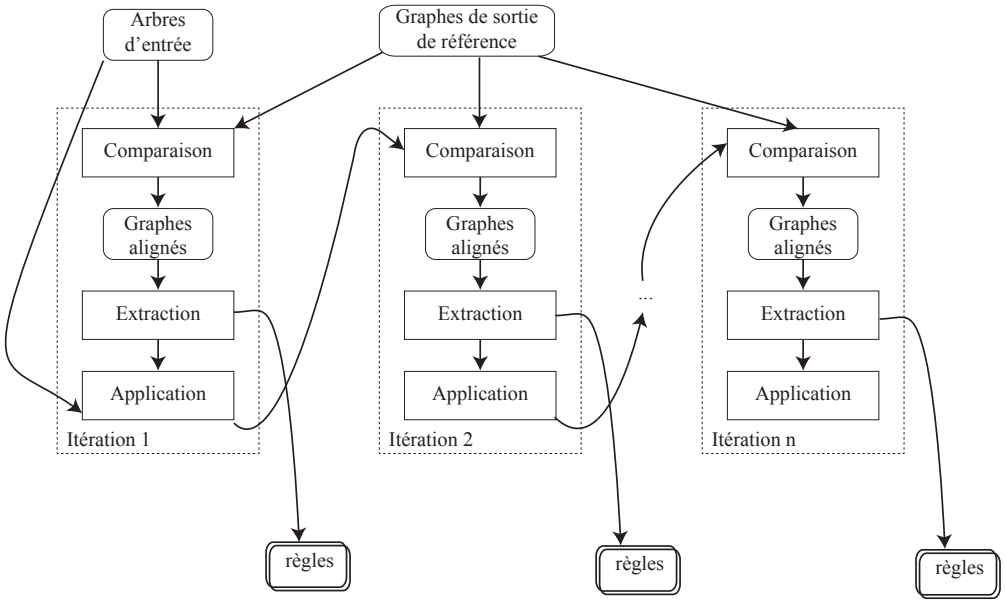


Figure 9.3: Vue d'ensemble de l'approche d'induction de règles de réécriture.

1. Comparaison de paires de structures (arbre / graphe) du corpus.
2. Identification des règles de réécriture possibles.
3. Apprentissage de classifieurs qui contraignent l'application des règles.
4. Application des règles sur le corpus d'entrée.
5. Redémarrer à l'étape 1.

La Figure 9.3 montre une vue d'ensemble de la méthode d'induction de règles de réécriture.

Formellement, nous définissons le corpus d'entraînement $C_{t,i} = \{(t_{i,1}, g_1), \ldots, (t_{i,n}, g_n)\}$ à l'itération i comme le corpus aligné de n arbres et n graphes. Nous faisons de même pour $C_{d,i}$, le corpus de développement à l'itération i, qui servira à tester le F_1-score entre l'arbre $t_{i,k}$ et le graphe g_k. Nous cherchons alors une séquence $\hat{\mathcal{R}}_{1:m}$ d'ensembles de règles $\mathcal{R}_i = \{r_1, r_2, \ldots, r_l\}$ induites à chaque itération i. La séquence $\hat{\mathcal{R}}_{1:m}$ maximise le F_1-score sur $C_{d,m}$ avec m le nombre d'itérations effectuées par l'algorithme, de telle sorte que, pour chaque $(t_m, g) \in C_{t,m}$, $t_m = g$, c'est-à-dire que le F_1-score $F_1(t_m, g) = 1$.

Pour ce faire, à chaque itération i, l'ensemble $\mathcal{R}_i$ est appliqué sur chaque arbre du corpus d'entraînement $C_{t,i}$ et du corpus de développement $C_{d,i}$, produisant ainsi deux nouveaux corpus, un d'entraînement et un de développement où les arbres t_i ont été réécrits par les règles de $\mathcal{R}_i$. Le F_1-score est alors calculé sur le corpus de développement. Si le F_1-score à l'itération i est supérieure au F_1-score de l'itération précédente $i-1$ et que l'amélioration est supérieur à un certain seuil τ (en général $\tau \leq 0.01$), l'algorithme passe à l'itération suivante, sinon il s'arrête.

La stratégie adoptée est une stratégie d'*ascension de colline* (*hill climbing*) où chaque itération cherche à s'approcher de $\hat{\mathcal{R}}$. Le F_1-score est propre à la tâche que nous souhaitons résoudre, c'est-à-dire transformer un arbre syntaxique en graphe sémantique. Trouver chaque ensemble de règles R_i revient à trouver un moyen d'apprendre la transformation d'un arbre vers un graphe. Nous détaillons dans la suite chaque étape permettant de mettre en œuvre cette approche.

9.3.1 Comparaison des structures d'entrée et de sortie

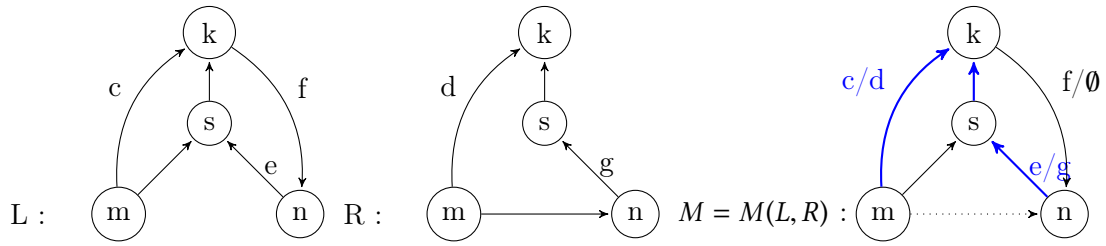


Figure 9.4: Deux graphes L et R (à gauche) et une fusion possible M (à droite). Sur le diagramme M, les arcs en commun sont en bleu, ceux à ajouter sont en pointillés et ceux à supprimer sont en noir.

Pour la comparaison entre structures d'entrée et de sortie, l'intuition revient à générer les différences entre deux représentations : un arbre en entrée et un graphe en sortie. Chaque structure d'entrée est synchronisée à une structure de sortie à la manière des grammaires synchrones (Chiang, 2007). L'alignement est fixé en utilisant les tokens entre la structure d'entrée et celle de sortie. En effet, pour les applications qui nous intéressent, les tokens ne sont jamais modifiés. Une fois cette synchronisation effectuée, nous pouvons générer les différences entre l'arbre et le graphe (arcs à ajouter, arcs à enlever et arcs à renommer), à la manière de la Figure 9.4. Nous décomposons ensuite en sous-graphes induits sur les arcs, en attachant les nœuds vides si nécessaire. L'algorithme de calcul des différences est un algorithme en temps linéaire sur les arcs qui fonctionne de la manière suivante : étant donné un arbre en entrée $A = \langle V, E, L_V, L_E, \phi \rangle$ et un graphe en sortie $G = \langle V', E', L'_E, L'_V, \phi' \rangle$, nous calculons les différences comme la différence des deux ensembles E et E' avec comme contrainte supplémentaire que deux arcs possédant la même étiquette sont considérés identiques. Deux sous-graphes de différences sont alors produits pour chaque phrase : le sous-graphe à ajouter et le sous-graphe à enlever. Il faut alors trouver les règles les plus saillantes parmi ces sous-graphes pour permettre une bonne réécriture.

9.3.2 Identification des règles de réécriture possibles

Les sous-graphes générés à l'étape précédente sont en général assez complexes et ne sont pas généralisables à un ensemble de phrase. Nous conjecturons donc que l'apprentissage sera impossible dans la mesure où aucun algorithme ne sera capable de généraliser une transformation. Pour pallier ce problème, il est utile d'avoir recours à une fouille de sous-graphes fréquents au sein de ce corpus de sous-graphes afin d'en tirer des règles de taille

plus petite mais plus générale (et donc plus faciles à apprendre). De fait, une fois la phase d'extraction de graphes de différences effectuée, nous utilisons *gSpan* (Yan et Han, 2002) pour trouver les sous-graphes les plus fréquents dans ce corpus. L'approche requiert de donner le support [3] comme hyperparamètre. Il est possible de voir le support comme une forme de pas d'apprentissage (*learning rate*) dans la mesure où un support trop grand risque de ne donner qu'un nombre limité de sous-graphes fréquents et un support trop petit risque d'en donner un nombre tellement important que les sous-graphes ne seront d'aucune utilité pour la réécriture.

Une fois l'algorithme appliqué, un ensemble H de sous-graphes est créé sur le corpus d'entrée (les arbres). Le corpus est alors reparcouru pour trouver les endroits où il existe un isomorphisme entre un sous-ensemble $H' \subseteq H$ de sous-graphes et les sous-arbres du corpus d'entrée. Il est alors possible de calculer les sous-graphes de parties droites pour chaque sous-graphe de H'. Une fois cette tâche effectuée, nous obtenons un corpus $\mathcal{W} = \{(\text{LHS}_i, \text{RHS}_i)\}$ où LHS est le sous-graphe de partie gauche et RHS celui de partie droite. Le nombre de règles est souvent important (plusieurs milliers).

9.3.3 Apprentissage de la contrainte d'application

À cette étape et pour chaque itération, nous avons extrait du corpus d'entraînement un certain nombre de règles de réécriture fréquentes. Dans cette section, nous allons décrire comment nous contraignons l'application de ces règles via des techniques d'apprentissage artificiel. Pour chaque règle, nous entraînons un classifieur qui détermine s'il faut appliquer la règle de réécriture à un endroit donné du corpus lorsqu'une image (*match*) est trouvée dans l'arbre d'entrée. L'entraînement est fait de la manière suivante : pour chaque règle $r = (\text{LHS}, \text{RHS})$, nous extrayons l'ensemble des occurrences de LHS dans tous les arbres du corpus d'entraînement. Pour chacune occurrence, nous encodons de l'information à propos de cette occurrence et son voisinage immédiat comme un vecteur de traits. De plus, pour chaque occurrence, nous vérifions aussi s'il s'agit bien d'une instance de r, c'est-à-dire si r a bien été extraite à cet endroit du corpus. Dans ce cas, l'occurrence est un exemple positif pour le classifieur, sinon l'exemple est négatif. Le classifieur est donc entraîné à séparer les exemples positifs (la règle r doit être appliquée) des exemples négatifs (la règle r ne doit pas être appliquée).

Les traits extraits sur les LHS et leur voisinage immédiat comprennent : la forme du token, du lemme, de la catégorie morpho-syntaxique, l'étiquette sur les arcs, les relations entre tokens, le degré entrant et sortant, etc.. Pour l'entraînement du classifieur, nous utilisons un Séparateur à Vastes Marges (SVM, *Support Vector Machines*, (Vapnik, 1995; Scholkopf et Smola, 2001)). Comme nous l'expliquions au début de cette section, une fois l'entraînement effectué, nous pouvons appliquer les règles sur le corpus d'entraînement à l'itération i pour obtenir l'itération $i + 1$ et continuer le processus d'apprentissage. Nous optimisons par ailleurs sur un corpus de développement tout en vérifiant que le F_1-score

3. Nous rappelons que le support est le nombre d'occurrences minimal pour qu'un sous-graphe soit considéré comme fréquent par l'algorithme.

augmente à chaque itération. Dès que le F_1 baisse ou n'augmente plus de manière significative ($\tau \leq 0.01$) nous arrêtons le processus d'entraînement. Nous obtenons alors un ensemble de règles $\hat{\mathcal{R}}$ qui est composé des meilleures séquences de règles à chaque itération.

9.3.4 Expériences et premiers résultats

Nous avons testé notre approche sur les mêmes corpus que ceux utilisés à la section 8.3 (p. 148), à savoir le corpus DM, et le corpus PAS pour l'anglais ; le DeepFTB pour le français. De plus, pour induire le passage de représentations surfaciques vers une représentation sémantique, nous utilisons les prédictions de Mate (Bohnet, 2010) sur l'anglais comme entrée de notre système pour l'anglais et les prédictions de FrMG (Villemonte de La Clergerie, 2010) comme entrée pour le français. Nous nous plaçons donc dans un cadre similaire aux expériences déjà réalisées au cours de cette thèse.

9.3.4.1 Un système privilégiant la précision

Nous donnons les résultats à la Table 9.1. Chaque expérience est lancée avec un maximum de 10 itérations et un ensemble maximal de 150 parties gauches pour chaque itération. Premier constat, les performances sont largement moins élevés que pour les approches d'analyse sémantique que nous présentions au chapitre précédent. Pour DM, le meilleur F_1-score est atteint après seulement 4 itérations, là où il en faut 8 pour PAS. Nous remarquons que le système à tendance à privilégier une forte précision, au détriment du rappel. Par exemple, dans le cas de PAS, la précision est largement supérieure à la baseline du S&T parser (p. 153, +3.70) et même supérieure à celle de la baseline de DSR (p. 164, +1.85). Pour le français néanmoins, les résultats en l'état sont préliminaires et peu concluants. Cela semble être dû au fort chevauchement entre la surface et le niveau profond (69%), mais il faudra investiguer davantage pour comprendre exactement la raison sous-jacente.

dév.	LP	LR	LF
DM (4 itér.)	80.44	59.75	68.57
PAS (8 itér.)	90.65	69.20	78.49
DeepFTB (2 itér.)	63.29	66.20	64.71

Table 9.1: Résultats pour l'induction de règles de transformation entre la surface (Mate pour l'anglais, FrMG pour le français) et le niveau profond.

Ce déséquilibre entre rappel et précision est dû au fait qu'après plusieurs itérations successives, il devient difficile de trouver des sous-graphes vraiment fréquents, puisque toutes les règles fréquentes ont été appliquées aux itérations précédentes. De fait, la classification a tendance à favoriser les exemples négatifs sur les exemples positifs et à en produire un très grand nombre. Cela entraîne un déséquilibre entre les classes (*imbalance learning*) (He et Ma, 2013) pour lequel il existe un nombre important de méthodes (He et Garcia, 2009) mais dont la présentation dépasse le cadre de cette thèse.

9.3.4.2 Discussion et comparaison avec les travaux antérieurs

Malgré le déséquilibre rappel/précision, l'intérêt non négligeable d'une telle approche est sa simplicité. Elle est assez proche de l'approche SPO dans la mesure où l'on cherche itérativement à créer des règles de réécriture (partie gauche et partie droite) ainsi qu'à générer par apprentissage automatique une contrainte d'application. La méthode a aussi le mérite d'être facilement interprétable puisque chaque règle générée est directement exploitable. Elle peut, par exemple, être facilement utilisée pour amorcer la création manuelle de règles de réécriture via un *framework* comme OGRE. Enfin, l'utilisation d'apprentissage automatique pour modéliser la contrainte d'application permet à l'utilisateur d'injecter au niveau des règles un nombre important de connaissances, notamment linguistiques. Par exemple, il est possible d'user de différents types de représentations syntaxiques synchronisées à notre corpus d'entrée (information en constituants, etc.) pour donner des informations complémentaires et réussir à mieux apprendre cette contrainte.

L'induction de règles de transformation pour l'analyse de structures prédicat-argument est moins répandue que les travaux en *parsing*. Les travaux les plus proches sont ceux de Ballesteros et al. (2014) qui développent une approche à base de transducteurs d'arbres pour le passage du niveau surfacique au niveau profond tel que définit par l'AnCoRa-UPF (Mille et al., 2013). [4] L'approche est similaire à la nôtre, dans la mesure où Ballesteros et al. (2014) font usage d'un étiquetage morpho-syntaxique prédit et d'une analyse en dépendances de surface utilisant l'approche jointe de Bohnet et Nivre (2012). Ensuite, la transduction est effectuée en trois temps : (i) identification des hypernœuds (la représentation peut regrouper des nœuds de surface en un seul nœud profond) (ii) création des arcs profonds (iii) étiquetage des arcs profonds. Le modèle est chaîné (*pipeline*) : chaque sous-sytème s'appliquant sur la sortie du précédent. Une fois les systèmes appliqués, Ballesteros et al. (2014) rapporte un F_1-score de 81.45 sur le corpus de test de l'AnCora-UPF. Leur score est du même ordre que celui que nous obtenons pour PAS (78.49), ce qui montre la validité de notre approche.

Néanmoins, ces recherches n'en sont qu'à un stade préliminaire et de nombreuses améliorations doivent être entreprises pour avoir une vue exacte du potentiel d'une telle approche. Enfin, comme nous le disions, la méthode produit des analyses avec une précision élevée, plus élevée que certaines des baselines de nos analyseurs présentés au chapitre 8. Il est donc probable que les règles extraites puissent être utilisées comme traits d'un analyseur de graphes. En effet, d'après les scores obtenus, nous pouvons estimer que les deux approches sont complémentaires et non similaires. De fait, la possibilité d'une hybridation de ces deux approches constitue une perspective possible et intéressante de notre travail. C'est pourquoi nous mettons à disposition de la communauté tout le nécessaire pour reproduire ces expériences.

4. cf. section 3.1.2, p. 45 pour une description du corpus et section 6.3.2, p. 89 pour une comparaison entre le schéma d'annotation profond utilisé dans cette thèse et celui de Mille et al. (2013).

Chapitre

10 Conclusion et perspectives

10.1 Résumé des contributions

Au cours de cette thèse, nous avons proposé différentes méthodes et techniques, toutes centrées autour d'un même but : produire des structures à l'interface syntaxe-sémantique et ce de manière robuste et automatique. Au niveau linguistique, nous avons aidé à la définition d'un schéma d'annotation en syntaxe profonde (Candito et al., 2014a; Perrier et al., 2014) adapté au français et avons créé un jeu de règles de réécritures permettant une pré-annotation de qualité en s'appuyant sur l'annotation surfacique existante. De fait, deux treebanks en syntaxe profonde pour le français ont vu le jour : le DeepSequoia [1] issu du corpus Séquoia (Candito et Seddah, 2012b), treebank hors domaine librement disponible et le Deep French Treebank dérivé du French Treebank (Abeillé et al., 2003) en dépendances (Candito et al., 2010) [2]. Les règles permettant la pré-annotation [3], ainsi que toute l'infrastructure nécessaire à leurs applications sont aussi mis à la disposition de la communauté via une licence libre.

Au niveau computationnel, nous avons proposé trois types de méthodes :

1. Un système de réécriture, appelé OGRE, fondé sur les approches algébriques (Rozenberg, 1997) et sur une étape de propagation d'informations novatrice garantissant de bonnes propriétés de terminaison et de confluence. Le système est par ailleurs parfaitement adapté à des applications linguistiques et librement disponible [4]. Comme nous l'avons démontré, il est capable de produire une pré-annotation et une annotation de qualité tout en proposant une alternative probante aux approches directes d'analyse automatique. Par ailleurs, il est robuste au changement de domaines et le même jeu de règles peut être appliqué sur des corpus annotés à partir de textes de provenances

1. Librement disponible à l'adresse suivante : `http://deep-sequoia.inria.fr`
2. Disponible pour tout détenteur de la licence du French Treebank.
3. `http://www.corentinribeyre.fr/projects/view/DeepRules`
4. `http://www.corentinribeyre.fr/projects/view/OGRE`

variées.

2. Un ensemble de méthodes réutilisant les concepts de l'analyse par transitions en dépendances (Nivre, 2005) et de l'analyse par factorisation d'arcs au moyen de décomposition duale. Pour l'analyse par transitions, nous avons proposé d'étendre les méthodes existantes pour le *parsing* d'arbres au *parsing* de graphes reprenant des idées issues des travaux de Sagae et Tsujii (2008) et Gesmundo et al. (2009). Pour l'analyse par décomposition duale, nous avons implémenté un jeu de traits plus riches augmentant ainsi les performances de l'analyseur. En effet, les méthodes de *parsing* ont été couplées à un ensemble de traits syntaxiques issus de prédictions en constituants et en dépendances surfaciques pour aider l'analyse sémantique. Une des contributions majeures est la fusion des constituants et des dépendances au sein d'analyseurs de graphes qui produisent des résultats à l'état de l'art sur les treebanks exploités. [5].
3. Une méthode préliminaire d'induction de règles de transformation qui utilise des techniques d'ascension de colline (*hill climbing*) et de recherche de sous-graphes fréquents (Jiang et al., 2013).

10.2 Perspectives

Chaque thèse amène son lot de contributions et souvent ces contributions ouvrent la voie à une myriade de perspectives, nous ne détaillons que celles qui nous paraissent les plus importantes.

10.2.1 Perspectives axées sur les corpus

Du point de vue linguistique, trois axes de recherche peuvent être des perspectives intéressantes à cette thèse : l'utilisation du MTF-LFG (Schluter et van Genabith, 2007) pour améliorer la qualité du DeepFTB, la création de structure de type Abstract Meaning Representation ou AMR (Banarescu et al., 2013) et la fusion des diverses ressources sémantico-discursives du français pour la création d'un corpus multi-strate *à la* Groningen Meaning Bank (Basile et al., 2012).

Amélioration de la qualité du DeepFTB. Le DeepFTB est un corpus issu d'une annotation automatique au moyen des règles de pré-annotation du DeepSequoia. Nous avons montré que la qualité de l'annotation automatique était très bonne (environ 94% de F_1-score étiqueté) mais que certains types de constructions n'étaient pas bien gérées. Pour améliorer la qualité de ce corpus acquis automatiquement, il serait possible d'utiliser le MTF-LFG, car l'ensemble des 4 700 ont été annotées automatiquement puis corrigées au moyen d'outils de fouille d'erreurs (Dickinson, 2006). Par ailleurs les coordinations ont été manuellement annotées. L'idée serait de pouvoir fusionner les annotations LFG et syntaxe profonde sur les phrases communes au deux corpus dans le but de corriger des erreurs éventuelles. Cependant, il faut garder à l'esprit que les modifications effectuées sur

5. Les *parsers*, modèles, et traits syntaxiques sont librement disponibles, `http://www.corentinribeyre.fr/projects/view/DeepFrenchParsing` et `http://www.corentinribeyre.fr/projects/view/DAGParser`.

le schéma d'annotation du French Treebank par Schluter et van Genabith (2007) (pour permettre une annotation LFG efficace) rendent la tâche difficile. En effet, l'isomorphisme entre les structures syntaxiques en constituants du MFT et du FTB n'est plus garanti, ce qui pose un vrai problème lors de la mise en correspondance des deux ressources.

Vers une représentation plus sémantique. Avec les développements récents de ressources sémantiques manuellement annotées et à large couverture pour le français, que ce soit le lexique Verbɜnet (Pradet et al., 2014; Danlos et al., 2014) ou le projet de FrameNet du français Asfalda (Candito et al., 2014a), il est désormais possible de coupler ces ressources avec le DeepSequoia et le DeepFTB pour se rapprocher d'une représentation qui pourrait ressembler aux représentations AMR (Abstract Meaning Representation) (Banarescu et al., 2013). En effet, l'AMR annote des graphes sémantiques qui s'abstraient de l'ordre des mots et font un usage important de la notion de paraphrase : plusieurs phrases avec le même sens auront une représentation similaire. Par exemple, des phrases telles que *Il la décrit comme un génie* ou *C'était un génie selon la description qu'il en faisait* ou encore *Elle était décrite comme un génie* auront le même graphe AMR. Pour la phase d'annotation, l'AMR reprend et synthétise les travaux de PropBank (Palmer et al., 2005), NomBank (Meyers et al., 2004) mais aussi VerbNet (Kipper-Schuler, 2006). L'idée est intéressante dans la mesure où la syntaxe profonde reste encore assez proche de la structure syntaxique, alors qu'il serait utile à terme de s'abstraire au maximum des idiosyncrasies syntaxiques pour avoir une représentation uniforme du sens d'un énoncé et de ses paraphrases ce qui permettrait d'exploiter des approches comme celles de Berant et Liang (2014).

Intégration d'informations sémantiques et discursives *à la Groningen Meaning Bank* (Basile et al., 2012). Il pourrait aussi être intéressant de coupler les travaux effectués sur le TLGBank (Moot, 2015) (utilisant un formalisme proche des CCG), sur Verbɜnet, sur Asfalda, sur le Wolf (le WordNet libre du français) (Sagot et Fišer, 2008; Hanoka et Sagot, 2012) et sur le French Discourse Treebank (Steinlin et al., 2015) aux nôtres pour obtenir une ressource multi-strate riche à la manière du Groningen Meaning Bank (Basile et al., 2012) qui donne accès à une sémantique dite profonde et à une représentation discursive pleine et entière. Bien entendu, le travail d'uniformisation en jeu dans une telle ressource est colossale, mais la qualité des ressources en sémantique pour le français est, à notre avis, aujourd'hui suffisante pour entreprendre une telle tâche.

10.2.2 Perspectives axées sur les méthodes

Du point de vue des méthodes, il est possible de proposer plusieurs axes d'amélioration :

Amélioration de la rapidité d'écriture des règles de réécriture. Nous avons montré l'intérêt d'utiliser un système de réécriture avec des règles écrites manuellement pour l'annotation automatique (ou la pré-annotation) de treebanks. Cependant, nous savons que le travail d'écriture de ces règles est long et présente un degré d'expertise parfois important sur les deux schémas d'annotation considérés : celui en entrée et celui en sortie. L'idée ici est d'utiliser une approche par l'exemple pour faciliter la création de ce genre de grammaires : étant donné un ensemble d'exemples présentant des constructions communes

(verbes à contrôle et à montée, coordinations, passif, causatif, etc.), l'utilisateur sélectionne quelques phrases représentatives de ce genre de constructions et annote les arcs à conserver et les arcs à modifier. Le système de réécriture peut alors induire les règles au moyen de ces indices. L'idée est qu'il n'est plus nécessaire d'avoir une connaissance approfondie du système de règles, ni une connaissance approfondie du schéma d'annotation d'origine. Nous avions proposé un début de réflexion lors de notre mémoire de Master (Ribeyre, 2012), mais énormément de recherche reste à conduire sur la meilleure manière d'induire les règles via des indices déposés par les utilisateurs.

Amélioration de l'induction de règles de transformations. Le travail que nous avons présenté au chapitre 9 est un travail prospectif sur l'induction de règles de réécriture qu'il est bon d'améliorer. En particulier, nous pensons qu'il est possible de prendre en compte certaines des idées de Peng et al. (2015) sur l'apprentissage de grammaires par remplacements d'hyperarcs en utilisant des méthodes d'échantillonnage de Monte-Carlo par chaînes de Markov via une forêt partagée de fragments d'hypergraphe. L'apprentissage de la décomposition en fragments d'hypergraphe permettrait une meilleure gestion des sous-graphes fréquents, ce qui limiterait le phénomène observé après plusieurs itérations de notre modèle : l'extraction des règles n'est plus possible, car seule demeure une multitude de règles peu fréquentes que les algorithmes d'extractions de sous-graphes fréquents ne peuvent pas récupérer.

Modélisation conjointe du *parsing* et de la réécriture. Il est aussi possible de combiner l'analyse par transition à la réécriture. En effet, nous pensons que le problème de réécriture peut être reformuler comme un problème d'analyse par transition en insérant des actions spéciales qui permettent d'ajouter ou de supprimer un arc en fonction d'une configuration précise de la pile et du buffer. L'idée est d'imposer à chaque graphe une décomposition canonique en actions qui serait plus simplement apprise par des techniques d'apprentissage automatique supervisé. Par ailleurs, il est probablement possible de récupérer les séquences de transitions pour découvrir automatiquement (à l'aide d'algorithmes de recherche de sous-séquences fréquentes, par exemple) des recoupements intéressants. Ce faisant, nous pouvons conjecturer qu'un apprentissage automatique du phénomène de propagation mis en place dans notre système de réécriture est possible.

Prédiction jointe syntaxe-sémantique. L'idée n'est pas nouvelle puisqu'elle a déjà été exploitée en partie par Henderson et al. (2013). Néanmoins, il est possible d'utiliser les techniques de décomposition duale pour remplacer les deux piles proposées par Henderson et al. (2013) et produire un analyseur joint syntaxe et sémantique par factorisation d'arcs. Nous pensons notamment qu'il serait utile de réutiliser l'implémentation de Martins et Almeida (2014) en y introduisant de nouveaux sous-problèmes qui intégreraient les informations syntaxiques en constituants et en dépendances. Ce faisant, nous aurions une modélisation propre et élégante de l'interface syntaxe-sémantique en arrivant à prédire directement les constituants, les dépendances surfaciques et la structure prédicat-argument. Malgré l'intérêt de l'approche, une telle modélisation n'est pas sans poser des problèmes complexes de combinatoire et le système risque d'être particulièrement lent à entraîner et à utiliser.

Au cours des trois dernières années, nous avons travaillé sur plusieurs problèmes couvrant des ressources, des modèles et des méthodes. Nous espérons que ce travail pourra servir de point de départ à des travaux ambitieux en sémantique du français, mais aussi à améliorer les approches par réécriture de graphes qui nous semblent être prometteuses en TAL pour modéliser l'interaction évidente de la syntaxe et de la sémantique. Dans un soucis de reproductibilité, mais aussi pour permettre à d'autres de poursuivre dans cette voie, toutes nos données, modèles et logiciels sont disponibles sous licence libre.

Bibliographie

Anne Abeillé, Lionel Clément, et François Toussenel. 2003. Building a Treebank for French. In *Treebanks : Building and Using Parsed Corpora*, pages 165–188. Springer.

Anne Abeillé. 2007. *Les nouvelles syntaxes : grammaires d'unification et analyse du français*. Hermes Science Publications.

Vilmos Ágel. 2006. *Dependency and Valency : an International Handbook of Contemporary Research*. Dependenz und Valenz : ein internationales Handbuch der zeitgenössischen Forschung. Walter de Gruyter.

A. V. Aho et J. D. Ullman. 1969. Properties of syntax directed translations. *Journal of Computer and System Sciences*, 3(3) :319–334.

Kazimierz Ajdukiewicz. 1935. Die Syntaktische Konnexität. *Polish Logic*, pages 207–231.

Hiyan Alshawi. 1996. Head Automata and Bilingual Tiling : Translation with Minimal Representations. *CoRR*, cmp-lg/9607005.

Pascal Amsili et Myriam Bras. 1998. DRT et Compositionnalité. *Traitement Automatique des Langues*, 39(1) :131–160.

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, et MichaelI. Jordan. 2003. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2) :5–43.

Abhishek Arun et Frank Keller. 2005. Lexicalization in Crosslinguistic Probabilistic Parsing : The Case of French. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics 2005, pages 306–313. Association for Computational Linguistics, Ann Arbor, Michigan.

Collin F. Baker, Charles J. Fillmore, et John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 86–90. Association for Computational Linguistics, Montreal, Quebec, Canada.

Miguel Ballesteros, Bernd Bohnet, Simon Mille, et Leo Wanner. 2014. Deep-Syntactic Parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics : Technical Papers*, pages 1402–1413. Dublin City University and Association for Computational Linguistics, Dublin, Ireland.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, et Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Sofia, Bulgaria.

Yehoshua Bar-Hillel. 1953. A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29 :47–58.

Luciano Baresi et Reiko Heckel. 2004. Tutorial Introduction to Graph Transformation : A Software Engineering Perspective. In Hartmut Ehrig, Gregor Engels, Francesco Parisi-Presicce, et Grzegorz Rozenberg, éditeurs, *Graph Transformations*, numéro 3256 in Lecture Notes in Computer Science, pages 431–433. Springer Berlin Heidelberg.

Matthew Barros et Luis Vicente. 2011. Right node raising requires both ellipsis and multidomination. *University of Pennsylvania Working Papers in Linguistics*, 3(1) :15.

Karine Baschung. 1996. Une approche lexicalisée des phénomènes de contrôle. *Langages*, 30(122) :96–122.

Valerio Basile, Johan Bos, Kilian Evang, et Noortje Venhuizen. 2012. Developing a Large Semantically Annotated Corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3196–3200. Istanbul, Turkey.

Jonathan Berant et Percy Liang. 2014. Semantic Parsing via Paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1415–1425. Association for Computational Linguistics, Baltimore, Maryland.

Taylor Berg-Kirkpatrick, David Burkett, et Dan Klein. 2012. An Empirical Investigation of Statistical Significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005.

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the second international conference on Human Language Technology Research*, pages 178–182. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Sylvie Billot et Bernard Lang. 1989. The Structure of Shared Forests in Ambiguous Parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151. Association for Computational Linguistics, Vancouver, British Columbia, Canada.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, et Wolfgang Seeker. 2013. (Re)ranking Meets Morphosyntax : State-of-the-art Results from the SPMRL 2013 Shared Task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145.

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97.

Bernd Bohnet et Joakim Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Guillaume Bonfante, Bruno Guillaume, et Mathieu Morey. 2011a. Modular Graph Rewriting to Compute Semantics. In *International Workshop on Computional Semantics 2011*.

Guillaume Bonfante, Bruno Guillaume, Mathieu Morey, et Guy Perrier. 2010. Réécriture de graphes de dépendances pour l'interface syntaxe-sémantique. In *TALN 2010*.

Guillaume Bonfante, Bruno Guillaume, Mathieu Morey, et Guy Perrier. 2011b. Enrichissement de structures en dépendances par réécriture de graphes. In *TALN 2011*.

Joan Bresnan. 1982. *The Passive in Lexical Theory*. Massachusetts Institute of Technology.

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Wiley-Blackwell, Malden, Mass.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, et Manfred Pinkal. 2006. The SALSA Corpus : a German Corpus Resource for Lexical Semantics. In *Proceedings of LREC 2006*. Genoa, Italy.

Aoife Cahill. 2004. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. Thèse de Doctorat, Dublin City University.

Marie Candito. 1996. A Principle-based Hierarchical Representation of LTAGs. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 194–199. Association for Computational Linguistics, Copenhagen, Denmark.

Marie Candito. 1999. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées*. Thèse de Doctorat, Université Paris 7 Diderot.

Marie Candito. 2003. Guide d'annotation des verbes pronominaux. Rapport technique, Université Paris 7 Diderot. URL http://www.linguist.univ-paris-diderot.fr/~mcandito/Publications/Guide-annotation-se.pdf.

Marie Candito, Pascal Amsili, Lucie Barque, Farah Benamara, Gaël De Chalendar, Marianne Djemaa, Pauline Haas, Richard Huyghe, Yvette Yannick Mathieu, Philippe Muller, Benoît Sagot, et Laure Vieu. 2014a. Developing a french framenet : Methodology and first results. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, et Stelios Piperidis, éditeurs, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Marie Candito et Matthieu Constant. 2014. Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 743–753. Association for Computational Linguistics, Baltimore, Maryland.

Marie Candito, Benoît Crabbé, Pascal Denis, et François Guérin. 2009. Analyse syntaxique du français : des constituants aux dépendances. In *16e Conférence sur le Traitement Automatique des Langues Naturelles - TALN 2009*. Senlis, France.

Marie Candito, Benoît Crabbé, et Pascal Denis. 2010. Statistical French Dependency Parsing : Treebank Conversion and First Results. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, et Daniel Tapias, éditeurs, *Proceedings of the Seventh Edition of Language Resources and Evaluation Conference (LREC)*. European Language Resources Association.

Marie Candito et Guy Perrier. 2015. Guide d'annotation en dépendances profondes pour le français. Rapport technique, Université Paris 7 Diderot / Loria / Inria. URL http://passage.inria.fr/deepwiki/node/19, avec la participation de Karën Fort, Bruno Guillaume, Corentin Ribeyre, Djamé Seddah et Éric Villemonte de La Clergerie.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, et Éric de La Clergerie. 2014b. Deep Syntax Annotation of the Sequoia French Treebank. In *International Conference on Language Resources and Evaluation (LREC)*. Reykjavik, Islande.

Marie Candito et Djamé Seddah. 2012a. Effectively Long-distance Dependencies in French : Annotation and Parsing Evaluation. In *TLT 11 - The 11th International Workshop on Treebanks and Linguistic Theories*. Lisbon, Portugal.

Marie Candito et Djamé Seddah. 2012b. Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012 - 19e conférence sur le Traitement Automatique des Langues Naturelles*. Grenoble, France.

John Carroll, Ted Briscoe, et Antonio Sanfilippo. 1998. Parser Evaluation : a Survey and a New Proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*. Grenada, Spain.

John A. Carroll et Ted Briscoe. 1996. Apportioning Development Effort in a Probabilistic LR Parsing System through Evaluation. In *Proceedings of the ACL/SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 92–100. University of Pennsylvania.

John A. Carroll, Guido Minnen, et Ted Briscoe. 1999. Corpus Annotation for Parser Evaluation. In *Proceedings of the EACL-99 Post-Conference Workshop on Linguistically Interpreted Corpora*, pages 35–41. Bergen, Norway.

John Chen et Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 41–48. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2) :201–228.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, et Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 924–932. Association for Computational Linguistics, Sofia, Bulgaria.

Jinho D. Choi et Martha Palmer. 2011. Transition-based Semantic Role Labeling Using Predicate Argument Clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, RELMS '11, pages 37–45. Association for Computational Linguistics, Portland, Oregon.

Noam Chomsky. 1955. *The logical structure of linguistic theory*. MIT Library, Cambridge, Mass.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press.

Noam Chomsky. 1981. *Lectures on Government and Binding : The Pisa Lectures*. Studies in generative grammar. Mouton de Gruyter.

Grzegorz Chrupala et Josef van Genabith. 2007. Using Very Large Corpora to Detect Raising and Control Verbs. In Miriam Butt et Tracy Holloway King, éditeurs, *Proceedings of the LFG07 Conference*. CSLI Publications.

Y. J. Chu et T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Silvie Cinková, Josef Toman, Jan Hajič, Kristýna Čermáková, Václav Klimeš, Lucie Mladová, Jana Šindlerová, Kristýna Tomšů, et Zdeněk Žabokrtský. 2009. Tectogrammatical Annotation of the Wall Street Journal. *Prague Bulletin of Mathematical Linguistics*, 92 :85–104.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models : Theory and Experiments with Perceptron Algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8. Association for Computational Linguistics, Philadelphia, PA, USA.

Michael Collins et Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04. Association for Computational Linguistics, Barcelona, Spain.

Ann Copestake et Dan Flickinger. 2000. An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000)*. European Language Resources Association (ELRA), Athens, Greece.

Ann Copestake, Dan Flickinger, Carl Pollard, et Ivan A. Sag. 2005. Minimal Recursion Semantics : An Introduction. *Research on Language and Computation*, 3(2-3) :281–332.

Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, Francesco Tortorella, et Mario Vento. 1998. Graph Matching : a Fast Algorithm and its Evaluation. In *Proceedings on the Fourtheenth International Conference on Pattern Recognition*, volume 2, pages 1582–1584.

Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, et Mario Vento. 2001. An Improved ALgorithm for Matching Large Graphs. In *Proceedings on the Third IAPR-TC15, Workshop on Graph-based Representation in Pattern Recognition*, pages 149–159.

Benoît Crabbé, Denys Duchier, Claire Gardent, Joseph Le Roux, et Yannick Parmentier. 2013. XMG : eXtensible MetaGrammar. *Computational Linguistics*, 39(3) :591–629.

Denis Creissels. 2007. Réflexivisation, transitivité et agent affecté. In A. Rousseau, D. Bottineau, et D. Roulland, éditeurs, *L'énoncé réfléchi*, pages 83–106.

Nicolas Curien et Pierre-Alain Muet. 2004. *La société de l'information*. La documentation française édition.

James R. Curran, Stephen Clark, et Johan Bos. 2007. Linguistically Motivated Large-scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 33–36. Association for Computational Linguistics, Prague, Czech Republic.

Haskell B. Curry et Robert Feys. 1958. *Combinatory Logic*, volume 1. North-Holland.

Mary Dalrymple. 2001. *Lexical Functional Grammar*, volume 34 de *Syntax and Semantics*. Academic Press, New York.

Laurence Danlos. 2005. ILIMP : Outil pour repérer les occurrences du pronom impersonnel il. In *TALN 2005 - 12e conférence sur le Traitement Automatique des Langues Naturelles*. Dourdan, France.

Laurence Danlos, Takuya Nakamura, et Quentin Pradet. 2014. Vers la création d'un VerbeNet du français. In *Atelier FondamenTAL, TALN 2014*.

Marie-Catherine de Marneffe et Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *Coling 2008 : Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8. Association for Computational Linguistics, Manchester, United Kingdom.

Steve DeNeefe et Kevin Knight. 2009. Synchronous Tree Adjoining Machine Translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 2 - Volume 2*, EMNLP '09, pages 727–736. Association for Computational Linguistics, Singapore.

Marianne Desmets. 2008. Ellipses dans les constructions comparatives en comme. *Linx*, (58) :47–74.

Markus Dickinson. 2006. From Detecting Errors to Automatically Correcting Them. In *In Proceedings of the 11th Conference of the European Chapter of ACL*, pages 265–272.

Yu Ding, Yanqiu Shao, Wanxiang Che, et Ting Liu. 2014. Dependency Graph Based Chinese Semantic Parsing. In Maosong Sun, Yang Liu, et Jun Zhao, éditeurs, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, numéro 8801 in Lecture Notes in Computer Science, pages 58–69. Springer International Publishing.

Marianne Djemaa, Marie Candito, Philippe Muller, et Laure Vieu. 2016. Annotating french framenet frames on syntactic treebanks : methodology and results. Soumis.

David Dowty. 1991. Thematic Proto-Roles and Argument Selection. *Language*, 67(3) :547–619.

Frank Drewes, Hans-Joerg Kreowski, et Annegret Habel. 1997. Handbook of Graph Grammars and Computing by Graph Transformation. chapitre Hyperedge Replacement Graph Grammars, pages 95–162. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Yantao Du, Fan Zhang, Weiwei Sun, et Xiaojun Wan. 2014. Peking : Profiling Syntactic Tree Parsing Techniques for Semantic Graph Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 459–464. Association for Computational Linguistics and Dublin City University, Dublin, Ireland.

Ane Dybro-Johansen. 2004. *Extraction Automatique De Grammaires D'Arbres Adjoints à Partir D'Un Corpus Arboré Du Français.* Thèse de Master, Université Paris 7.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards - Mathematics and Mathematical Physics*, 71B(4) :233–240.

Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, et Gabriele Taentzer. 2006. *Fundamentals of Algebraic Graph Transformation.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Karel van den Eynde et Piet Mertens. 2006. *Le dictionnaire de valence DICOVALENCE : manuel d'utilisation.*

Richàrd Farkas, Bernd Bohnet, et Helmut Schmid. 2011. Features for Phrase-Structure Reranking from Dependency Parses. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 209–214. Association for Computational Linguistics, Dublin, Ireland.

Charles J. Fillmore. 1976. Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences : Conference on the Origin and Development of Language and Speech*, 280(1) :20–32.

Dan Flickinger. 2000. On Building a More Efficient Grammar by Exploiting Types. *Natural Language Engineering*, 6(1) :15–28.

Daniel Flickinger, Yi Zhang, et Valia Kordoni. 2012. DeepBank : A Dynamically Annotated Treebank of the Wall Street Journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pages 85–96. Edições Colibri.

Gottlob Frege. 1879. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.*

Gottlob Frege et Bertrand Russell. 1994. *Correspondance (juin 1902 - décembre 1904, mars - juin 1912).* E.P.E.L., L'Unebévue.

Yoav Freund et Robert E. Schapire. 1999. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3) :277–296.

Bertrand Gaiffe et Kamel Nebhi. 2009. Le corpus de l'Est Républicain. Rapport technique, Laboratoire ATILF.

Andrea Gesmundo, James Henderson, Paola Merlo, et Ivan Titov. 2009. A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009) : Shared Task*, pages 37–42. Association for Computational Linguistics, Boulder, Colorado.

Daniel Gildea. 2011. Grammar factorization by tree decomposition. *Computational Linguistics*, 37(1) :231–248.

Daniel Gildea et Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA.

Jane Grimshaw. 1990. *Argument Structure*. MIT Press, Cambridge, Mass.

Bruno Guillaume, Guillaume Bonfante, Paul Masson, Mathieu Morey, et Guy Perrier. 2012. Grew : un outil de réécriture de graphes pour le TAL. In Gilles Sérasset Georges Antoniadis, Hervé Blanchon, éditeur, *12ième Conférence annuelle sur le Traitement Automatique des Langues (TALN'12)*. ATALA, Grenoble, France.

Bruno Guillaume et Guy Perrier. 2009. Interaction Grammars. *Research on Language and Computation*, 7(2-4) :171–208.

Bruno Guillaume et Guy Perrier. 2012. Semantic Annotation of the French Treebank with Modular Graph Rewriting. In Jan Hajic, éditeur, *META-RESEARCH Workshop on Advanced Treebanking, LREC 2012 Workshop*. META-NET, Istanbul, Turkey.

Annegret Habel, Reiko Heckel, et Gabriele Taentzer. 1996. Graph Grammars with Negative Application Conditions. *Fundamenta Informaticae*, 26(3/4) :287–313.

Jan Hajič, Alena Böhmová, Eva Hajičová, et Barbora Vidová-Hladká. 2000. The Prague Dependency Treebank : A Three-Level Annotation Scenario. In Anne Abeillé, éditeur, *Treebanks : Building and Using Parsed Corpora*, pages 103–127. Amsterdam Kluwer.

Eva Hajičová et Petr Pajas. 2000. Evaluation of Tectogrammatical Annotation of PDT. In *Text, Speech and Dialogue*, pages 75–80. Springer.

Eva Hajičová, Barbara Partee, et Petr Sgall. 1998. *Topic-Focus Articulation, Tripartite Structures, and Semantic Content*. Springer, Dordrecht ; Boston, MA.

Eva Hajicová, Jirí Havelka, Petr Sgall, Katerina Vesela, et Daniel Zeman. 2004. Issues of Projectivity in the Prague Dependency Treebank. *The Prague Bulletin of Mathematical Linguistics*, 81 :5–22.

Jan Hajič. 1998. Building a Syntactically Annotated Corpus : The Prague Dependency Treebank. In Eva Hajičová, éditeur, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, et Yi Zhang. 2009. The CoNLL-2009 Shared Task : Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009) : Shared Task*, pages 1–18. Association for Computational Linguistics, Boulder, Colorado.

Johan Hall et Joakim Nivre. 2008. A Dependency-Driven Parser for German Dependency and Constituency Representations. In *Proceedings of the ACL Workshop on Parsing German*, pages 47–54.

Keith Hall, Ryan McDonald, Jason Katz-Brown, et Michael Ringgaard. 2011. Training Dependency Parsers by Jointly Optimizing Multiple Objectives. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1489–1499. Association for Computational Linguistics, Edinburgh, United Kingdom.

Valérie Hanoka et Benoît Sagot. 2012. Wordnet extension made simple : A multilingual lexicon-based approach using wiki resources. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Haibo He et Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21 :1263–1284.

Haibo He et Yunqian Ma. 2013. *Imbalanced Learning : Foundations, Algorithms, and Applications*. Wiley-IEEE Press.

James Henderson, Paola Merlo, Ivan Titov, et Gabriele Musillo. 2013. Multilingual Joint Parsing of Syntactic and Semantic Dependencies with a Latent Variable Model. *Computational Linguistics*, 39(4) :949–998.

Julia Hockenmaier et Mark Steedman. 2007. CCGbank : A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3) :355–396.

Thomas Hofmann, Bernhard Schölkopf, et Alexander J. Smola. 2008. Kernel methods in machine learning. *The Annals of Statistics*, 36(3) :1171–1220.

John E. Hopcroft et Robert E. Tarjan. 1972. Isomorphism of Planar Graphs. In Raymond E. Miller, James W. Thatcher, et Jean D. Bohlinger, éditeurs, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 131–152. Springer US.

Liang Huang, Suphan Fayong, et Yang Guo. 2012. Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 142–151. Association for Computational Linguistics, Montréal, Canada.

Liang Huang et Kenji Sagae. 2010. Dynamic Programming for Linear-Time Incremental Parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086. Association for Computational Linguistics, Uppsala, Sweden.

Richard Hudson. 1984. *Word Grammar*. Blackwell, Oxford, UK.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, et Dan Flickinger. 2012. Who Did What to Whom ? : A Contrastive Study of Syntacto-semantic Dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, LAW VI '12, pages 2–11. Association for Computational Linguistics, Jeju, Republic of Korea.

Ray Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*, volume 12. Cambridge, Mass.,Mit Press.

Chuntao Jiang, Frans Coenen, et Michele Zito. 2013. A Survey of Frequent Subgraph Mining Algorithms. *The Knowledge Engineering Review*, 28 :75–105.

Valentin Jijkoun et Maarten de Rijke. 2007. Learning to Transform Linguistic Graphs. In *Proceedings of the Second Workshop on TextGraphs : Graph-Based Algorithms for Natural Language Processing*, pages 53–60. Association for Computational Linguistics, Rochester, NY, USA.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, et Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of COLING 2012*, pages 1359–1376. The COLING 2012 Organizing Committee, Mumbai, India.

Aravind K. Joshi, Leon S. Levy, et Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1) :136–163.

Sylvain Kahane. 2000. Extractions dans une grammaire de dépendance lexicalisée à bulles. *TAL*, 41(1) :211–243.

Sylvain Kahane. 2001. A Fully Lexicalized Grammar for French Based on Meaning-Text Theory. In Alexander Gelbukh, éditeur, *Computational Linguistics and Intelligent Text Processing*, numéro 2004 in Lecture Notes in Computer Science, pages 18–31. Springer Berlin Heidelberg. DOI : 10.1007/3-540-44686-9_2.

Sylvain Kahane. 2003. The Meaning-Text Theory. *Dependency and Valency. An International Handbook of Contemporary Research*, 1 :546–570.

Sylvain Kahane et François Lareau. 2005. Grammaire d'Unification Sens-Texte : modularité et polarisation. pages 23–32. Grammaire d'Unification Sens-Texte : modularité et polarisation. Dourdan.

Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn, et Joseph Le Roux. 2015. Foreebank : Syntactic Analysis of Customer Support Forums. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisboa, Portugal.

Jenna Kanerva, Juhani Luotolahti, et Filip Ginter. 2014. Turku : Broad-Coverage Semantic Parsing with Rich Features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 678–682. Association for Computational Linguistics and Dublin City University, Dublin, Ireland.

Ronald M. Kaplan et Joan Bresnan. 1982. Lexical-functional grammar : A formal system for grammatical representation. In J. Bresnan, éditeur, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, et Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8.

Karin Kipper-Schuler. 2006. *VerbNet : A Broad-Coverage, Comprehensive Verb Lexicon.* Thèse de Doctorat, University of Pennsylvania.

Philipp Koehn. 2005. Europarl : A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings : the tenth Machine Translation Summit*, pages 79–86. AAMT, AAMT, Phuket, Thailand.

Terry Koo, Amir Globerson, Xavier Carreras, et Michael Collins. 2007. Structured Prediction Models via the Matrix-Tree Theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150. Association for Computational Linguistics, Prague, Czech Republic.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, et David Sontag. 2010. Dual Decomposition for Parsing with Non-projective Head Automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1288–1298. Association for Computational Linguistics, MIT, Massachusetts, USA.

Jean-Louis Krivine. 1997. *Lambda-Calcul. Types et modèles.* Dunod.

Joseph B. Kruskal. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1) :pp. 48–50.

Sandra Kübler, Ryan McDonald, et Joakim Nivre. 2009. *Dependency Parsing.* Morgan and Claypool Publishers, San Rafael, Californie.

Marco Kuhlmann. 2013. Mildly Non-projective Dependency Grammar. *Computational Linguistic*, 39(2) :355–387.

Marco Kuhlmann. 2014. Linköping : Cubic-Time Graph Parsing with a Simple Scoring Scheme. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 395–399. Dublin, Republic of Ireland.

Marco Kuhlmann et Giorgio Satta. 2009. Treebank Grammar Techniques for Non-Projective Dependency Parsing. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 478–486. Association for Computational Linguistics, Athens, Greece.

Michihiro Kuramochi et George Karypis. 2001. Frequent Subgraph Discovery. In *Proceedings of the First International Conference on Data Mining*, pages 313–320.

Clemens Lautemann. 1990. The complexity of graph languages generated by hyperedge replacement. *Acta Informatica*, 27(5) :399–421.

Beth Levin. 1993. *English Verb Classes and Alternations : A Preliminary Investigation.* University of Chicago Press.

Xavier Lluís, Xavier Carreras, et Lluís Màrquez. 2013. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association for Computational Linguistics*, 1 :219–230.

Xavier Lluís, Xavier Carreras, et Lluís Màrquez. 2014. A Shortest-path Method for Arc-factored Semantic Role Labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 430–435. Association for Computational Linguistics, Doha, Qatar.

David M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the33rd Annual Meeting of the Association for Computational Linguistics.*

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

André F. T. Martins et Mariana S. C. Almeida. 2014. Priberam : A Turbo Semantic Parser with Second Order Features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476. Association for Computational Linguistics and Dublin City University, Dublin, Ireland.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, et Mário A. T. Figueiredo. 2011. Dual Decomposition with Many Overlapping Components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 238–249. Association for Computational Linguistics, Edinburgh, United Kingdom.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, et Mário A. T. Figueiredo. 2010. Turbo Parsers : Dependency Parsing by Approximate Variational Inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 34–44. Association for Computational Linguistics, Cambridge, Massachusetts.

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, et Eric P. Xing. 2015. AD3 : Alternating Directions Dual Decomposition for MAP Inference in Graphical Models. *Journal of Machine Learning Research*, 16 :495–545.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing.* Thèse de Doctorat, University of Pennsylvania.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, et Jan Hajič. 2005. Non-projective Dependency Parsing Using Spanning Tree Algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530. Association for Computational Linguistics, Vancouver, British Columbia, Canada.

Brendan D. McKay. 1981. Practical Graph Isomorphism. *Congressus Numerantium*, 30 :45–87.

Igor A. Mel'čuk. 2012. *Semantics : From Meaning to Text*. Numéro vol. 1 in Semantics : From Meaning to Text. John Benjamins Publishing Company.

Igor Mel'čuk et Alain Polguère. 2008. Prédicats et quasi-prédicats sémantiques dans une perspective lexicographique. *Lidil. Revue de linguistique et de didactique des langues*, (37) :99–114.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, et Ralph Grishman. 2004. The NomBank Project : An Interim Report. In Adam Meyers, éditeur, *HLT-NAACL 2004 Workshop : Frontiers in Corpus Annotation*, pages 24–31. Association for Computational Linguistics, Boston, Massachusetts, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, et Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, et K.Q. Weinberger, éditeurs, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Simon Mille, Alicia Burga, et Leo Wanner. 2013. AnCora-UPF : A Multi-Level Annotation of Spanish. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 217–226. Charles University in Prague, Matfyzpress, Prague, Czech Republic.

Yusuke Miyao, Takashi Ninomiya, et Jun'ichi Tsujii. 2004. Corpus-Oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. In Keh-Yih Su, Jun'ichi Tsujii, Jong-Hyeok Lee, et Oi Yee Kwong, éditeurs, *Natural Language Processing – IJCNLP 2004*, numéro 3248 in Lecture Notes in Computer Science, pages 684–693. Springer Berlin Heidelberg.

Richard Montague. 1970. English as a Formal Language. In B. Visentini, éditeur, *Linguaggi Nella Societá e Nella Tecnica*, pages 188–221. Edizioni di Communita.

Richard Moot. 2015. A type-logical treebank for french. *Journal of Language Modelling*, 3(1) :229–264.

Alessandro Moschitti, Daniele Pighin, et Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2) :193–224.

François Mouret et Anne Abeillé. 2011. On the Rule of Right-Node-Raising in French. In *International conference on elliptical constructions*. Paris, Chicago Center, France.

Claude Muller. 2013. Le prédicat, entre (méta)catégorie et fonction. *Cahiers de lexicologie*, N° 102 :51–65.

Rebecca Nesson et Stuart Shieber. 2008. Synchronous Vector-TAG for Natural Language Syntax and Semantics. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 9)*. Tübingen, Germany.

Joakim Nivre. 2005. *Inductive Dependency Parsing of Natural Language Text*. Thèse de Doctorat, Växjö University.

Joakim Nivre. 2008. Sorting Out Dependency Parsing. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL)*, pages 16–27.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Joakim Nivre, Marco Kuhlmann, et Johan Hall. 2009. An Improved Oracle for Dependency Parsing with Online Reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 73–76. Association for Computational Linguistics, Paris, France.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, et Zdenka Uresova. 2015. SemEval 2015 Task 18 : Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926. Association for Computational Linguistics, Denver, Colorado.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, et Yi Zhang. 2014. SemEval 2014 Task 8 : Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72. Association for Computational Linguistics. URL `http://aclweb.org/anthology/S14-2008`.

Stephan Oepen et Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the 5th international conference on language resources and evaluation (lrec 2006)*.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, et Thorsten Brants. 2002. The LinGO Redwoods Treebank : Motivation and Preliminary Applications. In *COLING 2002 : The 17th International Conference on Computational Linguistics : Project Notes*.

Sebastian Padó. 2007. *Cross-lingual Annotation Projection Models for Role-Semantic Information*. Thèse de Doctorat, Saarland Universität.

Martha Palmer, Daniel Gildea, et Paul Kingsbury. 2005. The Proposition Bank : An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1) :71–106.

Yannick Parmentier et Joseph Le Roux. 2005. XMG : a Multi-formalism Metagrammatical Framework. In *17th European Summer School in Logic, Language and Information - ESSLLI 2005*, page _. Edinburgh/Scotland, United Kingdom. 12 pages. Poster session.

Patrick Paroubek, Eric Villemonte de La Clergerie, Sylvain Loiseau, Anne Vilnat, et Gil Francopoulo. 2009. The PASSAGE Syntactic Representation. In *7th International Workshop on Treebanks and Linguistic Theories (TLT7)*.

Barbara H. Partee. 1993. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrecht ; Boston, softcover reprint of the original 1st ed. 1993 édition.

Sébastien Paumier. 2006. Manuel d'utilisation de Unitex 1.2. Rapport technique, Université Paris-Est Marne-la-Vallée.

Sébastien Paumier et Claude Martineau. 2015. Manuel d'utilisation de Unitex 3.1 bêta. Rapport technique, Université Paris-Est Marne-la-Vallée.

Xiaochang Peng, Linfeng Song, et Daniel Gildea. 2015. A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41. Association for Computational Linguistics, Beijing, China.

Fernando C. N. Pereira et David H. D. Warren. 1980. Definite clause grammars for language analysis—A survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3) :231–278.

David M. Perlmutter et Paul M. Postal. 1984. *Toward a Universal Characterization of Passivization*. Studies in Relational Grammar 1. University of Chicago Press.

Guy Perrier, Marie Candito, Bruno Guillaume, Corentin Ribeyre, Karën Fort, et Djamé Seddah. 2014. Un schéma d'annotation en dépendances syntaxiques profondes pour le français. In *Traitement Automatique du Langage Natural (TALN)*. Marseille, France.

Slav Petrov, Leon Barrett, Romain Thibaux, et Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

John L Pfaltz et Azriel Rosenfeld. 1969. Web grammars. In *Proceedings of the 1st international joint conference on Artificial intelligence*, pages 609–619. Morgan Kaufmann Publishers Inc.

Carl Pollard et Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Martin Popel, David Mareček, Jan ŠtÄ›pánek, Daniel Zeman, et ZdÄ›nÄ›k Žabokrtský. 2013. Coordination Structures in Dependency Treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 517–527. Association for Computational Linguistics, Sofia, Bulgaria.

Quentin Pradet, Laurence Danlos, et Gaël de Chalendar. 2014. Adapting VerbNet to French using existing resources. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Ronald C. Read et Derek G. Corneil. 1977. The Graph Isomorphism Disease. *Journal of Graph Theory*, 1(4) :339–363.

Ines Rehbein et Josef van Genabith. 2009. Automatic Acquisition of LFG Resources for German - as Good as it Gets. In Miriam Butt et Tracy Holloway King, éditeurs, *Proceedings of the LFG09 Conference*. CSLI Publications.

Corentin Ribeyre. 2012. *Mise en place d'un système de réécriture de graphes appliqué à l'interface syntaxe-sémantique*. Master thesis, Univ. Paris Diderot 7.

Corentin Ribeyre. 2013. Vers un système générique de réécriture de graphes pour l'enrichissement de structures syntaxiques. In *RECITAL 2013 - 15ème Rencontre des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*, pages 178–191. Université de Nantes, Les Sables d'Olonne, France.

Corentin Ribeyre, Marie Candito, et Djamé Seddah. 2014a. Semi-Automatic Deep Syntactic Annotations of the French Treebank. In *The 13th International Workshop on Treebanks and Linguistic Theories*, Proceedings of the 13th International Workshop on Treebanks and Linguistic Theories. Tübingen Universität, Tübingen, Germany.

Corentin Ribeyre, Djamé Seddah, et Éric Villemonte de La Clergerie. 2012. A Linguistically-motivated 2-stage Tree to Graph Transformation. In Chung-Hye Han et Giorgio Satta, éditeurs, *TAG+11 - The 11th International Workshop on Tree Adjoining Grammars and Related Formalisms - 2012*. INRIA, Paris, France.

Corentin Ribeyre, Éric Villemonte de La Clergerie, et Djamé Seddah. 2014b. Alpage : Transition-based semantic graph parsing with syntactic features. In *International Workshop on Semantic Evaluation*. Dublin, Irlande.

Corentin Ribeyre, Éric Villemonte de La Clergerie, et Djamé Seddah. 2015. Because Syntax does Matter : Improving Predicate-Argument Structures Parsing Using Syntactic Features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*. Denver, USA.

Laura Rimell, Stephen Clark, et Mark Steedman. 2009. Unbounded Dependency Recovery for Parser Evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821. Association for Computational Linguistics, Singapore.

Brian Roark, Asaf Bachrach, Carlos Cardenas, et Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 324–333. Association for Computational Linguistics, Singapore.

Frank Rosenblatt. 1958. The Perceptron : a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6) :386–408.

John Robert Ross. 1967. *Constraints on variables in syntax.* Thèse de Doctorat, Massachusetts Institute of Technology.

Grzegorz Rozenberg, éditeur. 1997. *Handbook of Graph Grammars and Computing by Graph Transformation : Volume I. Foundations.* World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Alexander M. Rush et Michael Collins. 2012. A Tutorial on Dual Decomposition and Lagrangian Relaxation for Inference in Natural Language Processing. *Journal of Artificial Intelligence Research*, 45(1) :305–362.

Alexander M. Rush, David Sontag, Michael Collins, et Tommi Jaakkola. 2010. On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1–11. Association for Computational Linguistics, Cambridge, Massachusetts.

Bertrand Russell. 1969. *An inquiry into meaning and truth.* Penguin Books.

John I. Saeed. 2003. *Semantics.* Wiley.

Kenji Sagae et Jun'ichi Tsujii. 2008. Shift-reduce Dependency DAG Parsing. In *Proceedings of the 22Nd International Conference on Computational Linguistics*, COLING '08, pages 753–760. Association for Computational Linguistics, Manchester, United Kingdom.

Benoît Sagot. 2010. The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French. In *7th international conference on Language Resources and Evaluation (LREC 2010).* Valletta, Malta.

Benoît Sagot et Darja Fišer. 2008. Building a free French wordnet from multilingual resources. In *Ontolex 2008.* Marrakech, Maroc.

Geoffrey Sampson. 1995. *English for the Computer : The SUSANNE Corpus and Analytic Scheme.* Clarendon Press, Oxford : New York.

Noémie-Fleur Sandillon-Rezer. 2013. *Apprentissage de Grammaires Catégorielles.* Thèse de Doctorat, Université de Bordeaux.

Natalie Schluter et Josef van Genabith. 2007. Preparing, restructuring, and augmenting a French Treebank : Lexicalised parsers or coherent treebanks ? In *Proceedings of the Conference of the Pacific Association for Computational Linguistics (PACLING)*. Melbourne, Australia.

Natalie Schluter et Josef van Genabith. 2008. Treebank-Based Acquisition of LFG Parsing Resources for French. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco.

Douglas C. Schmidt et Larry E. Druffel. 1976. A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices. *Journal of the ACM*, 23(3) :433–445.

Bernhard Scholkopf et Alexander J. Smola. 2001. *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA.

Uwe Schöning. 1988. Graph Isomorphism is in the Low Hierarchy. *Journal of Computer and System Sciences*, 37(3) :312–323.

Djamé Seddah. 2004. *Synchronisation des connaissances syntaxiques et sémantiques pour l'analyse d'énoncés en langage naturel à partir du formalisme des Grammaires d'Arbres Adjoints.* Thèse de Doctorat, Université Henri Poincaré, Nancy I.

Djamé Seddah. 2010. Exploring the Spinal-STIG Model for Parsing French. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10).*

Djamé Seddah, Sandra Kübler, et Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109. Dublin City University, Dublin, Ireland.

Petr Sgall, Eva Hajicová, Jarmila Panevová, et Jacob Mey. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects.* Springer.

Libin Shen, Lucas Champollion, et Aravind K. Joshi. 2007. LTAG-spinal and the Treebank. *Language Resources and Evaluation*, 42(1) :1–19.

Stuart M. Shieber. 2007. Probabilistic Synchronous Tree-Adjoining Grammars for Machine Translation : The Argument from Bilingual Dictionaries. In Dekai Wu et David Chiang, éditeurs, *Proceedings of the Workshop on Syntax and Structure in Statistical Translation.* Rochester, New York.

Stuart M. Shieber et Yves Schabes. 1990. Synchronous Tree-Adjoining Grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, volume 3, pages 253–258. Helsinki, Finland.

Sidney Siegel et John Jr. Castellan. 1988. *Nonparametric Statistics for The Behavioral Sciences*. McGraw-Hill Humanities/Social Sciences/Languages, second édition.

Mark Steedman. 1996. *Surface Structure and Interpretation*. Numéro 30 in Linguistic Inquiry Monographs. MIT Press, Cambridge, MA, USA.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Jacques Steinlin, Margot Colinet, et Laurence Danlos. 2015. FDTB1 : Identification of discourse connectives in a French corpus. In *Traitement automatique du langage naturel*. Caen, France.

Timothy Angus Stowell. 1981. *Origins of Phrase Structure*. Thèse de Doctorat, Massachusetts Institute of Technology.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, et Joakim Nivre. 2008. The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *CoNLL 2008 : Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Coling 2008 Organizing Committee, Manchester, England.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Éditions Klinksieck.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, et Noah A. Smith. 2014. CMU : Arc-Factored, Discriminative Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 176–180. Association for Computational Linguistics and Dublin City University, Dublin, Ireland.

Jörg Tiedemann. 2009. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. *Recent advances in natural language processing V : selected papers from RANLP 2007*, 309 :237–248.

Ivan Titov, James Henderson, Paola Merlo, et Gabriele Musillo. 2009. Online Graph Planarisation for Synchronous Parsing of Semantic and Syntactic Dependencies. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, IJCAI'09, pages 1562–1567. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Roy W. Tromble et Jason Eisner. 2006. A Fast Finite-state Relaxation Method for Enforcing Global Constraints on Sequence Decoding. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 423–430. Association for Computational Linguistics, New York, New York.

Jeffrey R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1) :31–42.

Lonneke Van der Plas, Tanja Samardzic, et Paola Merlo. 2010. Cross-lingual validity of propbank in the manual annotation of french. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 113–117. Association for Computational Linguistics, Uppsala, Sweden.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Éric Villemonte de La Clergerie. 2010. Convertir des dérivations TAG en dépendances . In *Proceedings of TALN*.

Éric Villemonte de La Clergerie. 2013a. Exploring beam-based shift-reduce dependency parsing with DyALog : Results from the SPMRL 2013 shared task. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL'2013)*.

Éric Villemonte de La Clergerie. 2013b. Improving a symbolic parser through partially supervised learning. In *The 13th International Conference on Parsing Technologies (IWPT)*. Nara, Japon.

David P. Williamson et David B. Shmoys. 2011. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, premier édition.

Yuk W. Wong et Robert J. Mooney. 2006. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL-06)*.

Yuk Wah Wong et Raymond Mooney. 2007. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967. Association for Computational Linguistics, Prague, Czech Republic.

Xifeng Yan et Jiawei Han. 2002. gSpan : Graph-based Substructure Pattern Mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 721–724.

Szu-Ting Yi, Edward Loper, et Martha Palmer. 2007. Can Semantic Roles Generalize Across Genres ? In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics NAACL-HLT*, pages 548–555. Rochester, NY, USA.

John M. Zelle et Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press, Portland, Oregon.

Yuan Zhang, Tao Lei, Regina Barzilay, et Tommi Jaakkola. 2014. Greed is Good if Randomized : New Inference for Dependency Parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1024. Association for Computational Linguistics, Doha, Qatar.

Yue Zhang et Stephen Clark. 2011. Syntactic Processing Using the Generalized Perceptron and Beam Search. *Computational Linguistics*, 37(1) :105–151.

Yue Zhang et Joakim Nivre. 2011. Transition-Based Dependency Parsing with Rich Non-Local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 188–193.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, et Jingbo Zhu. 2013. Fast and Accurate Shift-Reduce Constituent Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 434–443. Association for Computational Linguistics, Sofia, Bulgaria.

Aleksandr Žolkovshij et Igor Mel'čuk. 1967. Essai d'une Theorie Semantique Applicable au traitement de langage. In *Second Conference Internationale Sur Le Traitement Automatique Des Langues, COLING 1967*.