French language DRS parsing

Ngoc Luyen Le

UNIVERSITE BRETAGNE LOIRE MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

Ecole Doctorale N°601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Ngoc Luyen LE

French Language DRS Parsing

Thèse présentée et soutenue à PLOUZANÉ, le 15 septembre 2020
Unité de recherche : Lab-STICC UMR 6285 - CNRS
Thèse N° : 2020IMTA0202

Rapporteurs avant soutenance :

Panayota Tita Kyriacopoulou	Professeure des universités, Université Gustave Eiffel
Benoît Crabbé	Professeur, Université Paris Diderot & Institut Universitaire de France

Composition du jury :

Président :	Ismaïl Biskri	Professeur, Université du Québec à Trois-Rivières
Examinateur :	Panayota Tita Kyriacopoulou	Professeure des universités, Université Gustave Eiffel
	Benoît Crabbé	Professeur, Université Paris Diderot & Institut Universitaire de France
	Annie Forêt	Maître de Conféfrences HDR, IRISA-ISTIC
Dir. de thèse :	Philippe Lenca	Professeur, IMT Atlantique
Co-dir. de thèse :	Yannis Haralambous	Directeur d'études, IMT Atlantique

IMT ATLANTIQUE

DOCTORAL THESIS

French Language DRS Parsing

Author:
Ngoc Luyen LE

Supervisors:
Yannis HARALAMBOUS
Philippe LENCA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis prepared at

Department of Computer Science
DECIDE - Lab-STICC - CNRS, UMR 6285
IMT Atlantique

This research was sponsored by Crédit Mutuel Arkéa Bank

Plouzané, Autumn 2020

Acknowledgements

First and foremost I express my deep sense of gratitude and profound respect to my thesis advisors, professor Yannis Haralambous and professor Philippe Lenca who have helped and encouraged me at all stages of my thesis work with great patience and immense care. I am specially indebted to Yannis Haralambous who gave me the golden opportunity to do this wonderful thesis and gave me the opportunity to acquire a great amount of knowledge and experience in the fields of natural language processing as well as deep learning, and devoted his time to help me both in resolving scientific matters as well as correcting the manuscripts that I wrote during these years.

I gratefully acknowledge the members of my Individual Thesis Monitoring Committee for their serving time and valuable feedback on my report in the academic years. I would particularly like to acknowledge Dr. Éric Saux and Professor Serge Garlatti, who gave me their extremely useful help, advice and encouragements.

I also would like to thank all my colleagues from the Computer Science department of IMT Atlantique and the CNRS Lab-STICC DECIDE research team for making these past years be unforgettable memories. Among others, a very special thank you to Lina Fahed and Kafcah Emani Cheikh for their invaluable advice on my research and for always being supportive of my work. I am also very grateful to Armelle Lannuzel who all helped me in numerous ways during various stages of my PhD.

I took part in many other activities with countless people teaching and helping me every step. Thank you to my friends from Plouzané, Brest and other cities, especially tonton Riquet, Francette, Denise, Daniel, Nanette, Landy, Fabrice, Gildas and Polla. I am really thankful to Dr. Pierre Larmande who have gave a chance to study in France and guided me during my master internship. I would like to thank to my teachers and colleagues at information technology faculty of Da Lat University.

As always it is impossible to mention everybody who had an impact to this work however there are those whose spiritual support is even more important. I feel a deep sense of gratitude for my grand parents, father, mother, my brothers, who formed part of my vision and taught me myriad good things that really matter in life. I owe my deepest gratitude towards my beloved and supportive wife, Hoai Phuong who is always by my side when times I needed her most and helped a lot in making this study, and my lovable son An Lap, who served as my inspiration to pursue this undertaking. Their infallible love and support has always been my strength. Their patience and sacrifice will remain my inspiration throughout my life.

This research is financed in a collaboration between Crédit Mutuel Arkéa Bank and IMT Atlantique. I would especially like to thank to Dr. Riwal Lefort, Maxime Havez and Bertrand Mignon from the Innovation IA Department of Arkéa who gave me permission to work in their industrial laboratory. I also appreciate very much the discussions that we had.

Last, I dedicate this thesis to the memory of my grandfather and my grandmother, whose role in my life was, and remains, immense.

Abstract

IMT Atlantique
Department of Computer Science

French Language DRS Parsing

by Ngoc Luyen Le

The rise of the internet, of personal computers and of mobile devices has been changing various communication forms from one-way communication, such as the press or television, to two-way flows of information or interactive communications. In particular, the advent of social networking platforms makes this communication trend ever more prevalent. User-generated contents from the social networking services become a giant source of information which can be useful for organizations or businesses in the sense that users are regarded as clients or potential clients for businesses or members of organizations. The exploitation of user-generated texts can help to identify their sentiments or intentions, or reduce the effort of agents in businesses or organizations who are responsible for gathering or receiving information on social networking services. In this thesis, we realized a study about semantic analysis and representation for natural language texts in various formats such discourses, utterances, and conversations from interactive communication on the social networking platforms.

With the purpose of finding an effective way to analyze and represent semantics of natural language utterances, we examine and discuss various approaches ranging from the using rule-based methods to current deep neural network approaches. Deep learning approaches require massive amounts of data, in our case: natural language utterance and their meaning representations—to leverage this requirement we employ an empirical approach and propose a general architecture for a meaning representation framework for the French language.

First of all, for each sequence of input texts, we analyze each word morphologically and syntactically using the formalism of dependency syntax, and this constitutes the first module of our architecture. During this step, we explore lemmas, part-of-speech tags and dependencies as features of words.

Then, a bridge between syntax and semantic is built based on the formalism of Combinatory Categorial Grammars (CCG), which provides a transparent syntax-semantic interface. This constitutes the second module of our architecture. The morphological and syntactic data obtained from the previous module are employed as input in the process of extraction of a CCG derivation tree. More precisely, this process consists of two stages: the first one is the task of the assignment of lexical categories to each word depending on its position and its relationship with other words in the sentence; the second one focuses on the binarization of dependency trees. The parsing of CCG derivation trees is realized on binary trees by applying the combinatory rules defined in CCG theory.

Finally, we construct a meaning representation for utterances based on the Discourse Representation Theory (DRT) which is built from Discourse Representation Structure (DRS) and the Boxer tool by Johan Bos. This constitutes the last module of our architecture. Data such as CCG derivation trees obtained by the previous module are used as input for this module, together with additional information such as chunks and entities. The transformation of input CCG derivation trees into the DRS format is able to process linguistic phenomena such as anaphoras, coreferences and others. As output, we obtain data either in FOL or in the DRS boxing format.

By implementing our architecture we have built a French CCG corpus based on the French Tree Bank corpus (FTB). Furthermore, we have proven efficiency of the use of embedding

features from lemmas, POS tags and dependency relations in order to improve the accuracy of the CCG supertagging task using deep neural networks.

Résumé

IMT Atlantique
Département Informatique

Analyse de la structure de représentation du discours pour le français

par Ngoc Luyen LE

L'essor d'Internet, des ordinateurs personnels, des appareils numériques et mobiles changent diverses formes de communication, passant à sens unique comme les articles, les livres, les télévisions au flux de deux sens d'informations ou aux communications interactives. Plus particulièrement, l'avènement des plateformes de réseaux sociaux rend cette communication tendance de plus en plus populaire. Les contenus générés par les utilisateurs à partir des services de réseaux sociaux deviennent une source géante d'informations qui peuvent être utile aux organisations ou aux entreprises sur l'aspect où les utilisateurs sont considérés comme des clients ou des clients potentiels pour les entreprises ou les membres d'organisations. L'exploitation des textes générés par les utilisateurs peut aider à identifier leurs sentiments ou leurs intentions, ou réduire l'effort des agents dans les entreprises ou les organisations qui sont responsables de recueillir ou de recevoir des informations sur les services de réseaux sociaux. Dans la cadre de cette thèse, nous réalisons une étude sur l'analyse sémantique et la représentation de textes en langage naturel qui ont été créés sous différents formats tels que discours, énoncés, conversations issues de la communication interactive sur les plateformes de réseaux sociaux.

Dans le but de trouver un moyen efficace d'analyser et de représenter la sémantique pour des énoncés de langage naturel donnés, nous examinons et discutons des divers travaux importants, allant de l'utilisation de méthodes basées sur des règles aux approches actuelles des réseaux de neurones profonds. Avec la limitation d'une quantité massive de données sur les paires d'énoncés du langage naturel et sa représentation de sens qui sont devenues une exigence obligatoire pour les approches d'apprentissage en profondeur, nous utilisons l'approche empirique et avons proposé une architecture générale pour un cadre de représentation de sens utilisant pour le français saisie en langage naturel.

Tout d'abord, avec chaque séquence de texte donnée, nous réalisons un processus d'analyse des informations morphologiques de chaque mot dans le premier module de l'architecture. À partir de là, nous explorons la lemma, l'étiquette et les caractéristiques du mot dans le texte. Ces informations cruciales sont utilisées comme entrée pour extraire la relation entre les mots et les constituants en utilisant la grammaire des dépendances. En conséquence, nous obtenons les informations syntaxiques et de dépendance de chaque mot des textes d'entrée.

Ensuite, le pont entre la syntaxe et la sémantique est construit sur la base du formalisme grammatical avec la grammaire catégorielle combinatoire (CCG) qui nous aide à posséder une interface transparente syntaxique-sémantique dans le deuxième module. Le résultat d'analyse obtenue du module précédent est utilisé comme entrée du processus d'extraction d'un arbre de dérivation CCG. Plus particulièrement, ce processus comprend deux étapes dont la première consiste à attribuer des catégories lexicales à chaque mot en fonction de sa position et de sa relation avec les autres mots de la phrase. Le second se concentre sur la binarisation de l'arbre de dépendance en un arbre binaire. L'analyse de l'arbre de dérivation CCG est réalisée sur l'arbre binaire en appliquant les règles combinatoires définies dans la théorie CCG.

Enfin, nous construisons une représentation de sens pour un énoncé donné basée sur la théorie de la représentation du discours (DRT) qui est construite à partir de la structure de représentation du discours (DRS) et du travail Boxer de Johan Bos dans le dernier module. Par conséquent, le résultat de l'analyse de l'arbre de dérivation CCG du module précédent est considéré comme l'entrée pour ce module, à côté des informations supplémentaires sur les morceaux

et les entités dans la phrase. La transformation de l'arbre de dérivation CCG d'entrée en un format DRS accompagne la résolution des phénomènes linguistiques telles que l'anaphorique, la co-référence, etc. En conséquence, nous obtenons la forme logique ou les formats de boxe de DRS pour les énoncés d'entrée.

La mise en œuvre de l'architecture proposée permet d'obtenir des résultats importants tels que la proposition d'une méthode pour obtenir un arbre de dérivation CCG à partir de la structure de dépendance d'une phrase donnée. À partir de cela, nous construisons un corpus français de CCG basé sur un corpus de français. En outre, nous prouvons l'efficacité des intégrations de l'utilisation du lemme, de l'étiquette pos et des informations de dépendance afin d'améliorer la précision de la tâche de super-étiquetage CCG avec un modèle de réseau de neurones profond. Dans l'ensemble, nous créons un prototype pour transformer des énoncés de langage naturel donnés en DRS ou en représentation de forme logique.

Contents

List of Figures

List of Tables

List of Abbreviations

SNS	**S**ocial **N**etworking **S**ites
SN	**S**ocial **N**etwork
NLP	**N**atural **L**anguage **P**rocessing
POS	**P**art-**O**f-**S**peech
CCG	**C**ombinatory **C**ategorial **G**ammar
DRS	**D**iscourse **R**epresentation **S**tructure
DRT	**D**iscourse **R**epresentation **T**heory
FTB	**F**rench **C**orpus **T**reebank
DMR	**D**iscourse **M**eaning **R**epresentation
GMB	**G**roningen **M**eaning **B**ank
PMB	**P**arallel **M**eaning **B**ank
AMR	**A**bstract **M**eaning **R**epresentation
UCCA	**U**niversal **C**onceptual **C**ognitive **A**nnotation
UDS	**U**niversal **D**ecompositional **S**emantics
PDTB	**P**enn **D**iscourse **T**ree**B**ank
SPARQL	**S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
CNN	**C**onvolutional **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etwork
TLG	**T**ype-**L**ogic **G**rammar
FOL	**F**irst-**O**rder **L**ogic
RDF	**R**esource **D**escription **F**ramework
OWL	**W**eb **O**ntology **L**anguage
MRL	**M**eaning **R**epresentation **L**anguage
PCCG	**P**robabilistic **C**ombinatory **C**ategorial **G**rammars
TAG	**T**ree-**A**djointing **G**rammar
LTAG	**L**exicalized **T**ree-**A**djointing **G**rammar

LFG	**L**exical **F**unctional **G**rammar
HPSG	**H**ead-driven **P**hrase **S**tructure **G**rammar
CFG	**C**ontext-**F**ree **G**rammar
PCFG	**P**robabilitic **C**ontext-**F**ree **G**rammar
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphics **P**rocessing **U**nit
TPU	**T**ensor **P**rocessing **U**nit
SGD	**S**tochastic **G**radient **D**escent
GOFAI	**G**ood **O**ld Fashioned **A**rtificial **I**ntelligence
CRF	**C**onditional **R**andom **F**ields
CMM	**C**onditional **M**arkov **M**odel
HMM	**H**idden **M**arkov **M**odel
WSJ	**W**all **S**treet **J**ournal
ME	**M**aximum **E**ntropy
CKY	**C**ocke **K**asami **Y**ounger
MST	**M**aximum **S**panning **T**ree

Résumé étendu

Analyse de la structure de représentation du discours pour le français

par Ngoc Luyen Le

1 Introduction

L'essor d'Internet, des ordinateurs personnels, des appareils numériques et des mobiles a eu comme effet une mutation des diverses formes de communication, qui étaient auparavant à sens unique, comme les articles de presse, les livres, la télévision, et qui sont aujourd'hui des flux d'informations allant dans les deux sens. Plus particulièrement, l'avènement des plate-formes de réseaux sociaux a rendu ce type de communication de plus en plus populaire. Les contenus générés par les utilisateurs à partir des services de réseaux sociaux deviennent une source massive d'informations qui peuvent être utiles aux organisations ou aux entreprises dans la mesure où elles considèrent les utilisateurs comme des clients ou des clients potentiels. À travers les productions textuelles des utilisateurs il est possible d'identifier leurs sentiments ou leurs intentions. L'automatisation de cette tâche permet de réduire celle des agents dans les entreprises ou organisations qui sont responsables de recueillir des informations sur les services de réseaux sociaux.

Le but général est le développement d'un cadre qui permet d'analyser des énoncés de langue française dans le contexte du discours, afin d'en obtenir une représentation du sens. Au cours des dernières décennies, de nombreuses théories fondamentales du langage ont été introduites et appliquées aux analyses syntaxique et sémantique, cependant il reste encore de nombreuses questions ouvertes comme, par exemple, le choix du cadre de représentation sémantique qui convienne à la représentation du discours ou la résolution d'ambiguïtés sémantiques, d'anaphore où de coréférence.

Voici les contributions que nous décrirons dans cet article :

- Une nouvelle méthode algorithmique qui permet l'analyse de la structure de dépendance d'une phrase et sa transformation en un arbre de dérivation de grammaire catégorielle combinatoire (GCC).
- Une nouvelle ressource de corpus GCC pour la langue française, basée sur le corpus *French Tree Bank* [1]
- Une architecture pour la représentation sémantique de textes en langue française.

Après un aperçu de l'état de l'art, nous présentons l'interface syntaxe/sémantique basée sur les GCC. Ensuite, nous décrivons notre architecture proposée, nos expérimentations et leur évaluation, à travers un exemple. Enfin, nous terminons l'article avec une conclusion.

[1] `https://github.com/lengocluyen/FrenchCCGBank`

2 État de l'art de la représentation sémantique du discours

Nous appellerons «discours» des unités de texte utilisées pour l'analyse des phénomènes linguistiques qui s'étendent sur plus d'une phrase, et générées dans le cadre d'une communication orale ou écrite naturelle entre personnes, sur un sujet particulier. Un discours peut être perçu de diverses manières, en fonction de facteurs tels que le contexte, l'ambiguïté langagière, les émotions ou les sentiments. De nombreuses approches ont été mises en œuvre afin de proposer un cadre de représentation du discours. Elles appartiennent généralement à trois approches principales : les approches basées sur des règles, l'approche empirique et les approche axées sur les corpus.

En guise d'exemple d'approches basées sur des règles, mentionnons des systèmes interprétant le langage naturel en langage de requête de base de données basés sur des règles spécifiques au domaine, comme le système SAVVY (1984) qui répond aux questions des humains, en utilisant des techniques basées sur les règles (Johnson, 1984) ; le système LUNAR (1973) qui permet aux gens de poser des questions en langage naturel et de demander au système d'effectuer des calculs dans le domaine de la géologie en utilisant les techniques basées sur des règles et la syntaxe (Woods, 1973) ; le système NLIDB (1983) qui génère des arbre syntaxiques en consultant un ensemble de règles de syntaxe (Templeton and Burger, 1983; Androutsopoulos, Ritchie, and Thanisch, 1995).

Nous qualifions d'«empiriques» les approches utilisant des techniques basées sur des règles, une analyse statistique basée sur des données ou une association de ces deux méthodes (Ge and Mooney, 2005; Raymond and Mooney, 2006; Wong and Mooney, 2006; Chiang, 2005). L'utilisation de formalismes grammaticaux tels que les grammaires d'arbres adjoints, les grammaires lexicales de fonctions, les grammaire catégorielles combinatoires ou les grammaire syntagmatique guidée par les têtes sont assez populaires dans le but de construire un cadre immédiat pour dériver des structures sémantiques à partir de structures syntaxiques (Forbes et al., 2003; Reddy et al., 2016). Dans le même contexte, l'application *Boxer* se sert d'un formalisme grammatical GCC lexicalisé, dans lequel les mots de la phrase sont affectés à des catégories lexicales pour produire la structure de représentation du discours dans un formalisme *ad hoc* (Curran, Clark, and Bos, 2007; Bos, 2008; Bos, 2015).

Enfin, dans la dernière décennie on a constaté l'émergence de plus en plus de méthodes basées sur des corpus tels que FrameNet (Baker, Fillmore, and Lowe, 1998), Propbank (Palmer, Gildea, and Kingsbury, 2005), PDTB (Prasad et al., 2005; Prasad et al., 2008), OntoNotes (Hovy et al., 2006), GMB (Basile et al., 2012), SemBanking (Banarescu et al., 2013), UDS (White et al., 2016), etc., qu'elles soient supervisées ou non-supervisées (Kate and Mooney, 2006; Allen et al., 2007; Litman et al., 2009; Young et al., 2010). Par exemple, à l'aide d'un apprentissage supervisé de correspondance de séquence à séquence, l'entrée en langage naturel $q = x_1 x_2 \dots x_n$ est associée à une représentation sous forme logique $a = y_1 y_2 \dots y_m$ obtenue comme le argmax d'une probabilité conditionnelle

$$p(a \mid q) = \prod_{t=1}^{n} p(y_t \mid y_{<t}, q) \tag{1}$$

Dans cette méthode le «codage» est la tâche qui consisté à convertir l'entrée en langage naturel q en une représentation vectorielle a, tandis que le «décodage» consiste à générer les formules $y_1 y_2 \dots y_m$ correspondant au vecteur de codage (Dong and Lapata, 2016; Liu, Cohen, and Lapata, 2018).

Dans le but d'analyser et de représenter la sémantique pour des énoncés de langage naturel donnés, nous avons parcouru diverses méthodes, allant de l'utilisation de règles aux approches actuelles basées sur des réseaux de neurones profonds. Néanmoins, comme ces dernières demandent au préalable une quantité massive de données d'apprentissage, exigence obligatoire

pour les approches d'apprentissage en profondeur, nous allons plutôt utiliser l'approche empirique et notre proposition d'architecture générale pour la représentation du discours en français va dans ce sens.

3 Interface entre syntaxe et sémantique

L'utilisation d'un formalisme grammatical pour effectuer l'analyse de phrases est cruciale pour passer de la syntaxe à la sémantique. Parmi les informations les plus utiles que nous allons utiliser, il y a les fonctions grammaticales. En général, cinq éléments jouent le rôle de fonctions grammaticales principales pour la construction d'une phrase, à savoir le sujet, le verbe, les objets direct et indirect, le complément et l'adverbe. La position de ces fonctions grammaticales est différente selon la langue. Par exemple, en français, une phrase utilise principalement l'ordre sujet-verbe-objet. Cependant, des phrases avec des ordres différents peuvent apparaître, comme par exemple dans le cas de l'interrogation directe. En utilisant la morphologie, les informations syntaxiques et les fonctions grammaticales, nous nous proposons de faire des déductions sur la sémantique. Différents formalismes grammaticaux ont été introduits dans ce but, nous avons choisi les grammaires combinatoires catégorielles en raison de leur capacité à gérer des phénomènes à longue distance et de l'utilisation des λ-expressions (Le and Haralambous, 2019).

3.1 Grammaires combinatoires catégorielles

Le concept de *grammaire combinatoire catégorielle* (GCC) a été introduit par Mark Steedman dans les années 2000 (Steedman, 1999; Steedman, 2000a; Steedman, 2000b). Les GCC ont été introduites comme extension des grammaires catégorielles (Ajdukiewicz, 1935; Bar-Hillel, 1953) avec l'ajout de règles d'inférence de catégorie afin d'obtenir une large couverture des langages naturels. Une GCC est essentiellement un formalisme grammatical lexicalisé dans lequel les mots sont associés à des catégories syntaxiques et lexicales spécifiques à une langue donnée.

De façon plus formelle, une GCC est un quintuplet $G = <\Sigma, \Delta, f, \varsigma, \Re>$ où:

- Σ est un ensemble fini de symboles appelés *terminaux*, ils correspondent aux mots de la phrase étudiée.
- Δ est un ensemble fini de symboles appelés *catégories d'axiomes*. Celles-ci sont formés à partir de symboles de base, par exemple S, NP, N, PP, etc., et leurs combinaisons à travers les opérateurs $\backslash$ et $/$. Ainsi, si X,Y $\in \Delta$, alors X/Y et $\mathrm{X}\backslash\mathrm{Y} \in \Delta$.
- f est la *fonction de catégorie lexicale* qui gère la correspondance entre terminaux et catégories.
- ς est le symbole d'*axiome de départ* de la grammaire, $\varsigma \in \Delta$.
- $\Re$ est un ensemble fini de règles de production de grammaire formelle qui, dans le contexte des GCC sont appelées *règles combinatoires* et que nous décrivons ci-dessous.

Nous désignons par X, Y, Z des méta-catégories (elles représentent n'importe quelle catégorie de la grammaire).

En premier lieu, l'ensemble $\Re$ comprend deux règles de base héritées des grammaires catégorielles AB:

$$\begin{array}{lllll} \text{Règle d'application avant:} & \mathrm{X}/\mathrm{Y}{:}f & \mathrm{Y} \quad :\mathrm{a} & \Rightarrow & \mathrm{X}{:}f(a) \\ \text{Règle d'application arrière:} & \mathrm{Y}{:}a & \mathrm{X}\backslash\mathrm{Y}{:}f & \Rightarrow & \mathrm{X}{:}f(a) \end{array} \tag{2}$$

Ces règles permettent de considérer aussi bien la catégorie du mot/syntagme qui suit (opérateur «/») un mot/syntagme donné que celle qui le précède (opérateur «\»).

Une contribution importante des GCC vis-à-vis des grammaires catégorielles AB est l'ajout d'un ensemble de règles inspirées des *combinateurs* de la logique combinatoire (Curry et al., 1958). Ces règles permettent le traitement des dépendances à longue distance et des constructions d'extraction/coordination. Ils peuvent être représentés à l'aide du λ-calcul, utilisé comme notation pour la représentation sémantique, comme suit:

$$\begin{array}{ll} \text{Règle de montée de type avant} & \text{X}:x \Rightarrow_T \text{T}/(\text{T}\backslash\text{X}):\lambda f.f(x) \\ \text{Règle de montée de type arrière} & \text{X}:x \Rightarrow_T \text{T}\backslash(\text{T}/\text{X}):\lambda f.f(x) \end{array} \tag{3}$$

$$\begin{array}{lccc} \text{Règle de composition avant} & \text{X}/\text{Y} & \text{Y}/\text{Z} & \Rightarrow_B \text{X}/\text{Z} \\ & \lambda y.f(y) & \lambda z.g(z) & \lambda z.f(g(z)) \\ \text{Règle de composition croisée avant} & \text{X}/\text{Y} & \text{Y}\backslash\text{Z} & \Rightarrow_B \text{X}\backslash\text{Z} \\ & \lambda y.f(y) & \lambda z.g(z) & \lambda z.f(g(z)) \\ \text{Règle de composition arrière} & \text{Y}\backslash\text{Z} & \text{X}\backslash\text{Y} & \Rightarrow_B \text{X}\backslash\text{Z} \\ & \lambda z.g(z) & \lambda y.f(y) & \lambda z.f(g(z)) \\ \text{Règle de composition croisée arrière} & \text{Y}/\text{Z} & \text{X}\backslash\text{Y} & \Rightarrow_B \text{X}/\text{Z} \\ & \lambda z.g(z) & \lambda y.f(y) & \lambda z.f(g(z)) \end{array} \tag{4}$$

$$\begin{array}{lccc} \text{Règle de substitution avant} & (\text{X}/\text{Y})/\text{Z} & \text{Y}/\text{Z} & \Rightarrow_S \text{X}/\text{Z} \\ & \lambda zy.f(z,y) & \lambda z.g(z) & \lambda z.f(z,g(z)) \\ \text{Règle de substitution croisée avant} & (\text{X}/\text{Y})\backslash\text{Z} & \text{Y}\backslash\text{Z} & \Rightarrow_S \text{X}\backslash\text{Z} \\ & \lambda zy.f(z,y) & \lambda z.g(z) & \lambda z.f(z,g(z)) \\ \text{Règle de substitution arrière} & \text{Y}\backslash\text{Z} & (\text{X}\backslash\text{Y})\backslash\text{Z} & \Rightarrow_S \text{X}\backslash\text{Z} \\ & \lambda z.g(z) & \lambda zy.f(z,y) & \lambda z.f(z,g(z)) \\ \text{Règle de substitution croisée arrière} & \text{Y}/\text{Z} & (\text{X}\backslash\text{Y})/\text{Z} & \Rightarrow_S \text{X}/\text{Z} \\ & \lambda z.g(z) & \lambda zy.f(z,y) & \lambda z.f(z,g(z)) \end{array} \tag{5}$$

Dans le cas des constructions de coordination, la catégorie lexicale $(X/X)\backslash X$ est utilisée pour valider la combinaison de composants similaires dans les règles de formule (6.2). La règle correspondante se présente comme suit :

$$\text{Coordination :} \quad \text{X} : g \quad \text{X} : f \quad \Rightarrow_{\Phi^n} \quad \text{X}:\lambda x.f(x) \wedge g(x) \tag{6}$$

Donnons un exemple simple de GCC: $G =< \Sigma, \Delta, f, \varsigma, \Re >$, où:

- Σ := {Henri, regarde, la, télévision}.
- Δ := {S, NP}
- $\mathcal{C}(\Delta)$:= {S, NP, S\NP}

- Function f: f(Henri) := {NP, NP/NP}, f(regarde) := {S\NP, (S\NP)/NP}, f(la_télévision) := {NP}
- ς := S (la phrase)
- $\Re$ comportant les règles décrites dans (6.1), (6.2), (6.5), (6.6), (6.7).

Afin d'obtenir une dérivation GCC, les catégories lexicales appropriées doivent d'abord être affectés à chaque mot de la phrase. Cette affectation est loin d'être unique, en effet un mot peut avoir différentes catégories lexicales selon sa position ou sa fonction dans la phrase. Voici celle que nous avons choisi pour notre exemple :

(1)

Henri	←	NP	:	$henri'$
regarde	←	(S\NP)/NP	:	$\lambda x.\lambda y.regarde'xy$
la	←	NP	:	la'
la	←	NP/NP	:	$\lambda x.la'(x)$
bien	←	(S\NP)/(S\NP)	:	$\lambda f.\lambda x.bien'(fx)$
dors	←	(S\NP)	:	$\lambda x.dors'x$

Dans les règles combinatoires ci-dessus, chaque catégorie est accompagnée d'une λ-expression représentant sa sémantique. Dans la fig. 5.2 nous illustrons l'arbre de dérivation CCG obtenu en appliquant les règles combinatoires et aboutissant à la sémantique de la phrase.

$$\dfrac{\dfrac{\dfrac{\text{Henri}}{NP} \quad \dfrac{\text{regarde}}{(S\backslash NP)/NP} \quad \dfrac{\dfrac{\text{la}}{NP/NP} \quad \dfrac{\text{télévision}}{NP}}{NP}\,{>}}{:x \quad :\lambda x\lambda y.regarde'xy \quad\quad :y}}{S\backslash NP : \lambda x.regarde'\,la\_télévision'\,x}\,{>}}{S : regarde'\,la\_télévision'\,henri'}\,{<}$$

FIGURE 1: Une dérivation GCC de la phrase: «Henri regarde la télévision»

3.2 Extraction d'une dérivation GCC à partir des dépendances syntaxiques

Dans cette section, nous décrivons un processus en trois étapes permettant d'obtenir des arbres de dérivation GCC. Dans la première étape, nous avons choisi parmi les nombreuses catégories lexicales qui ont été utilisées dans le corpus pour chaque mot, celles qui «s'emboîtent» dans la phrase donnée et les attachent en tant qu'étiquettes à l'arbre de dépendances. Dans la deuxième étape, nous extrayons des blocs de l'arbre des dépendances et ajoutons de nouveaux nœuds et arêtes afin de binariser l'arbre. Dans la dernière étape, nous déduisons des catégories lexicales pour les nouveaux nœuds et vérifions que nous pouvons monter à partir des feuilles jusqu'à la racine de l'arbre en appliquant des règles combinatoires – la racine de l'arbre doit alors nécessairement être munie de la catégorie lexicale S.

Affectation de catégories lexicales aux nœuds de l'arbre de dépendances

Les arbres de dépendances sont obtenus par *analyse des dépendances* (cf. figure 5.7). Pour ce faire, il existe différentes méthodes, certaines basées sur des algorithmes d'apprentissage automatique entraînés sur de grands ensembles de phrases annotées syntaxiquement, d'autres

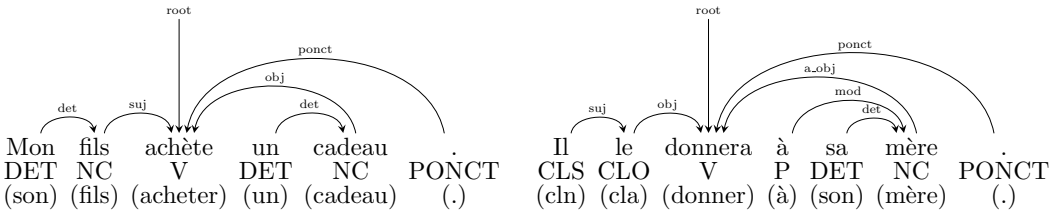


FIGURE 2: Exemples d'arbres de dépendances avec étiquettes de parties du discours

basées sur des approches empiriques utilisant des grammaires formelles. Dans notre cas, nous avons utilisé la version française de MaltParser (Candito et al., 2010).

La théorie de GCC assigne deux types de catégories lexicales aux mots: les *catégories de base* (par exemple, S, NP, PP), et les *catégories complexes* obtenues par combinaison de catégories de base en utilisant les fonctions d'application $\backslash$ et $/$. Par exemple, S$\backslash$NP est une catégorie complexe. Elle peut être attribuée à un mot/syntagme susceptible de se trouver adjacent à un mot/snytagme de catégorie lexicale NP à sa gauche, afin de produire un S; S$/$NP signifie que le mot/syntagme NP est attendu à droite. Afin d'affecter des catégories lexicales aux nœuds de l'arbre de dépendances, nous traitons d'abord les mots qui dont les catégories lexicales observées dans le corpus sont uniques. Ainsi, par exemple, les noms ont une catégorie lexicale NP, les adjectifs ont une catégorie lexicale NP$/$NP ou NP$\backslash$NP selon qu'ils sont placés à gauche ou à droite du nom, etc. Une fois que nous avons affecté les catégories lexicales uniques (ou dépendantes de la position, comme dans les adjectifs), nous passons aux verbes.

Le verbe principal de la phrase, qui est normalement la racine de l'arbre de dépendances, peut avoir des *dépendances d'argument*, étiquetées *suj*, *obj*, *a_obj*, *de_obj*, *p_obj*, c'est-à-dire les correspondances avec le sujet, l'objet direct et indirect, et / ou des *dépendances complémentaires*, étiquetées *mod*, *ats*, etc., représentant des informations complémentaires telles que le nombre, l'heure, le lieu, etc. On affecte la catégorie lexicale S$\backslash$NP à un verbe principal ayant un sujet à sa gauche, puis on ajoute $/$NP (ou $\backslash$NP, selon sa position par rapport au verbe) *pour chaque objet direct ou indirect* (dans l'ordre des mots dans la phrase).

Les verbes auxiliaires suivis d'autres verbes reçoivent la catégorie lexicale (S$\backslash$NP)$/$(S$\backslash$NP). Par exemple dans la phrase «Je voudrais demander un rendez-vous pour demain», le verbe principal est «demander». Il a un sujet, donc sa catégorie lexicale doit contenir S$\backslash$NP. De plus, il a une dépendance d'objet directe pointant vers «rendez-vous». Par conséquent, nous lui attribuons également une catégorie lexicale (S$\backslash$NP)$/$NP. Le verbe «voudrais» étant auxiliaire, il obtient la catégorie lexicale (S$\backslash$NP)$/$(S$\backslash$NP).

Binarisation des arbres de dépendance

La binarisation de l'arbre de dépendance est effectuée sur la base d'informations sur la structure de phrase dominante pour la langue spécifique. En français, la plupart des phrases sont de type SVO, comme dans «Mon fils (S) achète (V) un cadeau (O)» ou SOV comme dans «Il (S) le (O) donne (V) à sa mère (indirect O)» (figure 5.7). En utilisant cette propriété générale, spécifique à la langue française (et aux langues romanes, en général), nous pouvons extraire et classer les composants de la phrase en sujets, objets directs, objets indirects, verbes et phrases complémentaires.

L'algorithme proposé pour transformer un arbre de dépendances en arbre binaire se compose de deux étapes. Tout d'abord, nous extrayons des groupes de mots dans l'arbre de dépendance qui est basé sur des informations syntaxiques et des étiquettes de dépendance entre les

Algorithm 1: Binarisation de l'arbre de dépendances

Binarisation_arbre_dépendant $(groupe\_de\_mots)$
Input: La liste des groupes de mots de la phrase
Output: L'arbre binaire)

$arbre\_binaire \leftarrow$ vide; $sous\_arbres\_binaires \leftarrow$ vide;
$i \leftarrow$ longueur de $group\_de\_mots$;
while $i \geq 0$ **do**
  if $group\_de\_mots[i]$ *est une liste de groupes de mots* **then**
    $sous\_arbres\_binaires \leftarrow$
     $construction\_arbre\_binaire\_avec\_mots(groupe\_de\_mots)$;
  else
    $sous\_arbres\_binaires \leftarrow$ récursion sur $binarisation\_arbre\_dpendant$
     avec $groupe\_de\_mots[i]$;
  if $arbre\_binaire$ *est vide* **then**
    $arbre\_binaire \leftarrow sous\_arbres\_binaires$;
  else
    créer un arbre temporaire $arbre\_binaire\_temporaire$;
    $racine$ est le nœud racine de $arbre\_binaire\_temporaire$;
    mettre $arbre\_binaire$ à la droite de $arbre\_binaire\_temporaire$;
    mettre $sous\_arbres\_binaires$ à la gauche de $arbre\_binaire\_temporaire$;
    $arbre\_binaire \leftarrow arbre\_binaire\_temporaire$;
    $sous\_arbres\_binaires \leftarrow$ *vide*;
  $i \leftarrow i - 1$;
return *arbre_binaire*

mots. Par exemple, le groupe de mot de sujet est obtenu en trouvant un mot qui a une dépendance étiquetée *suj*, le groupe de mot des verbes correspond à la racine de la structure de dépendance, les groupes d'objet directs ou indirects sont obtenus sous forme de mots avec des liens dirigés vers la racine (le verbe) et ayant des étiquettes *obj* ou *p_obj*, etc. Ensuite, nous construisons un arbre binaire pour chaque groupe de mots, comme décrit dans l'algorithme 6, puis combinons les arbres binaires dans l'ordre inverse de la structure de la phrase dominante. Par exemple si SVO est la structure dominante, nous commençons par construire l'arbre binaire du groupe de mots d'*objet*, puis le combinons avec l'arbre binaire du groupe *verb*, et finalement nous obtenons l'arbre binaire du groupe *sujet*. Sur la figure 5.9, le lecteur peut voir quatre groupes de blocs dans l'arborescence de dépendances, affichés sous forme de régions de l'arbre binarisée.

Construction d'une dérivation GCC complète et validation

La dernière étape est la vérification de l'affectation des catégories lexicales aux mots par la construction d'un arbre GCC complet. Cette dernière opération est effectuée de manière itérative en appliquant des règles combinatoires. Nous partons des feuilles de l'arbre binarisé (cf. figure 5.9) – pour lequel nous avons déjà des catégories lexicales de l'étape 1 – et remontons en appliquant des règles combinatoires.

Les règles combinatoires nécessitent généralement deux paramètres d'entrée pour former une nouvelle catégorie lexicale, à l'exception de la règle de montée de type qui nécessite un seul paramètre d'entrée. Dans l'arbre binaire, chaque fois que deux nœuds ont des informations de catégorie lexicale qui leur sont affectées, nous pouvons inférer la catégorie lexicale du troisième nœud, comme dans la figure 5.10.

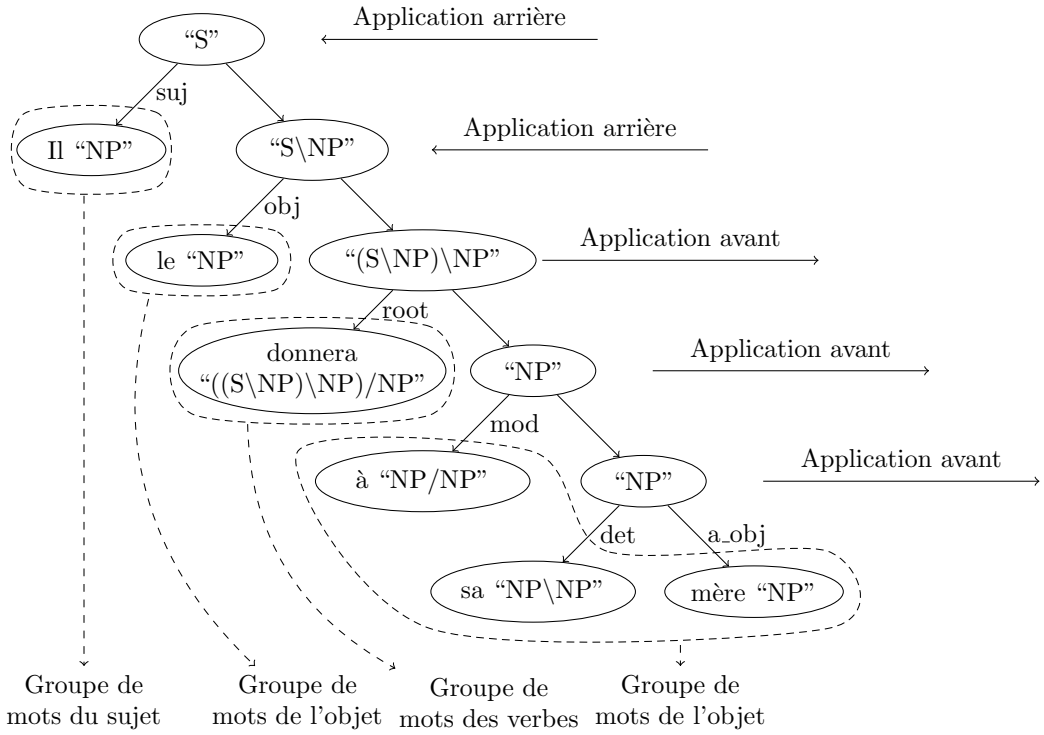


FIGURE 3: Les groupes de mots (en pointillés) et l’arbre binaire de la phrase

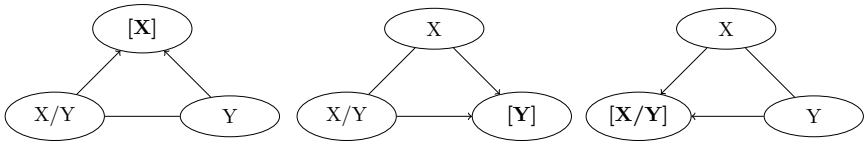


FIGURE 4: Les règles d’inférence des catégories lexicales

3.3 Expérimentation et évaluation sur le corpus *French TreeBank*

Nous avons utilisé le corpus *French TreeBank* (FTB –) qui contient environ un million de mots extraits d’articles du quotidien *Le Monde* pour la période 1991–1993. Nous avons utilisé la version basée sur les dépendances, décrite dans (Candito, Crabbé, and Denis, 2010; Candito et al., 2009). En appliquant notre méthode à l’ensemble complet de 21 550 arbres de dépendances du corpus, nous avons obtenu des arbres de dérivation GCC pour 94,02% des phrases.

Au total, nous obtenons un ensemble de 73 catégories lexicales distinctes pour le corpus français complet. Plus précisément, on voit que le nombre de catégories lexicales augmente rapidement au cours des 10 000 premières phrases (figure 5.15) et de moins en moins par la suite. De plus, les catégories lexicales NP/NP et NP, qui correspondent aux articles, adjectifs et noms, sont attribuées à plus de la moitié des mots (figure 5.16).

Le taux d’échec de notre approche est élevé par rapport aux résultats dans d’autres langues (99,44% de phrases analysées avec succès en anglais (Hockenmaier and Steedman, 2007) ou 96% en hindi (Ambati, Deoskar, and Steedman, 2018)). Parmi les causes d’erreur, les trois sont principales sont : erreurs d’étiquetage de partie du discours, erreurs dans les dépendances ou leur étiquetage et, enfin, erreurs résultant de problèmes linguistiques complexes tels que lacunes lexicales, des lacunes parasitaires, etc.

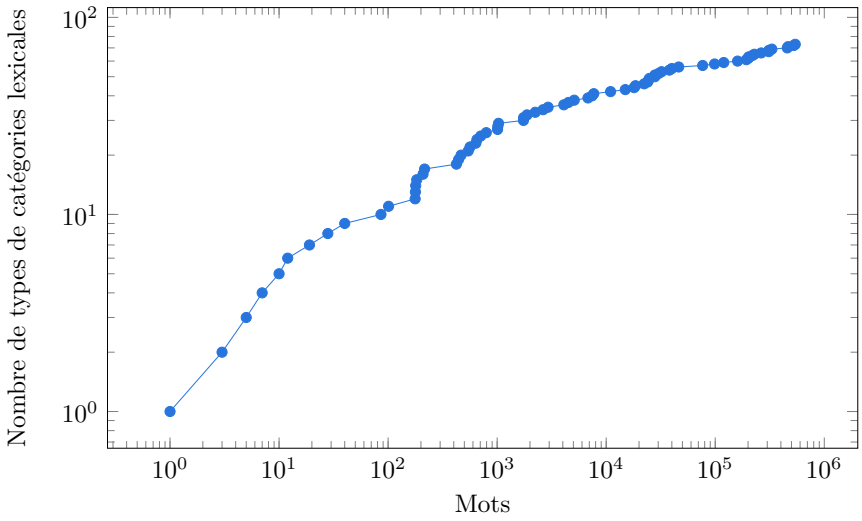


FIGURE 5: Croissance des types de catégories lexicales (échelle logarithmique pour les deux axes) dans le corpus FTB

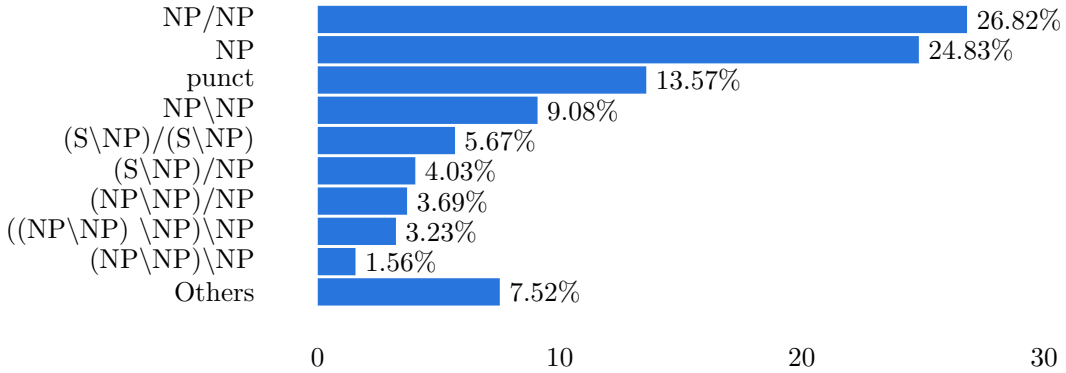


FIGURE 6: Distribution des catégories lexicales GCC dans le corpus FTB

4 Architecture proposée d'analyseur de structure de représentation du discours pour le français

Les GCC fournissent une interface transparente entre la syntaxe de surface et la représentation sémantique sous-jacente. De plus, elles possèdent le même pouvoir expressif que le λ-calcul puisque leur fondement est la logique combinatoire. Dans la section précédente, nous avons réalisé un arbre d'analyse de dérivation GCC pour une phrase donnée en utilisant des informations de syntaxe morphologique et de dépendance. Ces résultats nous permettent de constituer le premier pas vers l'interprétation sémantique des entrées en langage naturel. Dans cette section, nous nous concentrerons sur la théorie de la représentation du discours qui permet de gérer le sens au-delà des limites des phrases. En nous basant sur les travaux de Johan Bos sur le système *Boxer Framework* pour l'anglais, nous proposons une approche de la tâche d'analyse sémantique pour la langue française.

4.1 Introduction de la structure de représentation du discours

Au début des années 80, un cadre théorique pour la sémantique dynamique a été introduit par Hans Kamp sous le nom de *théorie de représentation du discours* (TRD). Le but était de traiter des phénomènes linguistiques tels que les pronoms anaphoriques, le temps, la présupposition

et les attitudes propositionnelles (citekamp1981theory,kamp2013discourse). L'émergence de la TRD a rendu possible une approche dynamique de la sémantique du langage naturel. Dans cette approche, le sens d'une phrase donnée est identifié dans une relation avec son contexte. En particulier, l'interaction entre une phrase et son contexte est réciproque. Ainsi l'analyse d'un énoncé dépendra de son contexte et le contexte peut être mis à jour pour donner lieu à un nouveau contexte lorsqu'on ajoute les informations de l'énoncé à celui-ci.

Le noyau de la TRD est la SRD, qui est le composant principal pour la construction de représentations sémantiques pour les textes. L'objectif de la SRD n'est pas seulement de réaliser des interprétations de phrases uniques, mais aussi de représenter des unités linguistiques, des paragraphes, des discours ou des textes plus larges. En général, la représentation du sens à travers la SRD se déroule phrase par phrase. Chaque phrase traitée fournit ses informations à la SRD qui contient les informations des phrases précédentes.

La représentation des SRD utilisée dans cette section repose sur des expressions de base de la théorie des types compatibles avec les représentations sémantiques formelles (Bos et al., 2017). Dans ce formalisme, le type $\langle e \rangle$ est utilisé pour les expressions de référents de discours ou de variables, tandis que le type $\langle t \rangle$ est utilisé pour les expressions SRD de base, comme suit :

$$\begin{aligned} \langle exp_e \rangle &::= \langle ref \rangle | \langle var_e \rangle \\ \langle exp_t \rangle &::= \langle srd \rangle \end{aligned} \tag{7}$$

Une expression de SRD de base $\langle srd \rangle$ est une paire d'ensembles : un ensemble de référents du discours $\langle ref \rangle$ qui représente les objets en discussion, et un ensemble de conditions $\langle condition \rangle$ qui sont des propriétés des référents du discours et des relations entre eux. Nous utilisons la notation suivante:

$$\langle srd \rangle ::= \begin{array}{|c|}\hline \langle ref \rangle * \\ \hline \langle condition \rangle * \\ \hline \end{array} \tag{8}$$

En général, les conditions de SRD sont de trois types: $\langle élémentaire \rangle$, $\langle lien \rangle$, et $\langle complexe \rangle$:

$$\langle condition \rangle ::= \langle élémentaire \rangle \mid \langle lien \rangle \mid \langle complexe \rangle \tag{9}$$

Les conditions de base sont des propriétés des référents de discours ou des relations entre eux:

$$\begin{aligned} \langle élémentaire \rangle ::= \; & \langle sym_1 \rangle (\langle exp_e \rangle) \mid \langle sym_2 \rangle (\langle exp_e \rangle, \langle exp_e \rangle) \\ & \mid temps(\langle exp_e \rangle, \langle sym_0 \rangle) \\ & \mid nommé(\langle exp_e \rangle, \langle sym_0 \rangle, classe), \end{aligned} \tag{10}$$

où $\langle exp_e \rangle$ désigne des expressions de type, $\langle sym_n \rangle$ désigne des prédicats n-aires, $\langle nombre \rangle$ désigne des nombres cardinaux, $temps$ exprime des informations temporelles et $classe$ désigne des classes d'entités nommées.

Les conditions de lien sont des marqueurs ou constantes référents de discours qui sont utilisés pour des références ou des inégalités entre les marqueurs ou les constantes du discours:

$$\begin{aligned} \langle lien \rangle ::= \; & \langle exp_e \rangle = \langle exp_e \rangle \mid \langle exp_e \rangle = \langle nombre \rangle \\ & \mid \langle exp_e \rangle \neq \langle exp_e \rangle \mid \langle exp_e \rangle \neq \langle nombre \rangle. \end{aligned} \tag{11}$$

Les conditions complexes représentent les SRD embarqués: implication ($\rightarrow$), négation ($\neg$), disjonction ($\vee$), ainsi que des opérateurs modaux exprimant la nécessité ($\Box$) et la possibilité

($\Diamond$). Les types de conditions complexes sont unaires et binaires:

$$\begin{aligned} \langle complexe \rangle &::= \langle unaire \rangle \mid \langle binaire \rangle \\ \langle unaire \rangle &::= \neg\langle exp_t \rangle \mid \Box\langle exp_t \rangle \mid \Diamond\langle exp_t \rangle \mid \langle ref \rangle : \langle exp_t \rangle \\ \langle binaire \rangle &::= \langle exp_t \rangle \rightarrow \langle exp_t \rangle \mid \langle exp_t \rangle \vee \langle exp_t \rangle \mid \langle exp_t \rangle ? \langle exp_t \rangle, \end{aligned} \tag{12}$$

où la condition $\langle ref \rangle : \langle exp_t \rangle$ désigne des verbes à contenu propositionnel.

Illustrons la SRD à travers l'exemple de la phrase suivante :

(2) a. La femme achète des poissons.

b. Elle les donne à son mari.

La phrase 13.a peut être analysée et réécrite dans le formalisme de la SRD comme suit:

$$[x, y : \text{femme}(x), \text{poisson}(y), \text{acheter}(x, y)].$$

Plus spécifiquement, l'expression SRD contient deux référents de discours $\langle ref \rangle = \{x, y\}$, alors que l'ensemble de conditions inclut $\langle condition \rangle = \{\text{femme}(x), \text{poisson}(y), \text{acheter}(x, y)\}$. Supposons maintenant que la phrase de l'exemple 13.b soit suivie de la phrase 13.a. L'expression SRD pour la deuxième phrase inclut les référents de discours $\langle ref \rangle = \{u, v, w\}$, alors que l'ensemble de conditions est $\langle condition \rangle = \{\text{donner}(u, v, w), \text{mari}(w), \text{personne}_1(v), \text{chose}_1(w)\}$. Ainsi, la SRD de la deuxième phrase sera réécrite comme suit: $[u, v, w : \text{donner}(u, v, w), \text{mari}(w), \text{personne}_1(v), \text{chose}_1(w)]$. Enfin, nous obtenons l'expression SRD finale après avoir intégré la SRD de la deuxième phrase dans la SRD de la première phrase, en résolvant l'anaphore comme suit :

$$\begin{aligned} [x, y, u, v, w \ : &\text{femme}(x),\ \text{poisson}(y),\ \text{acheter}(x, y), \\ &\text{donner}(u, v, w),\ \text{mari}(u),\ v = x,\ w = y]. \end{aligned}$$

Afin d'illustrer les différentes expressions de SRD, nous disposons de trois formalismes que nous illustrons à l'aide des phrases ci-dessus:

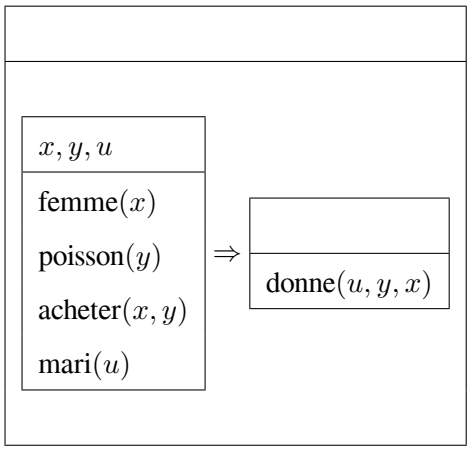
1. La notation SRD «officielle»:

$$<\{\emptyset\}, <\{x, y, u\}>, \{\text{femme}(x), \text{poisson}(y), \text{acheter}(x, y), \text{mari}(u)\}> \\ \Rightarrow <\{\emptyset\}, \{\text{donne}(u, y, x)\}>.$$

2. La notation linéaire:

$$[\emptyset : [x, y, u : \text{femme}(x), \text{poisson}(y), \text{acheter}(x, y), \text{mari}(u)] \Rightarrow [\emptyset : donne(u, y, x)]].$$

3. La notation en boîtes imbriquées :



4.2 Relation entre la GCC et la SRD

En général, nous pouvons transformer un arbre d'analyse de dérivation GCC en expression SRD en définissant une correspondance entre les catégories lexicales et les types sémantiques. Par exemple, si les catégories lexicales primitives employées dans le corpus français GCC sont NP (groupe nominal), S (La phrase), la catégorie S est associée à une expression SRD de type $\langle t \rangle$, alors que les catégories NP correspondent à une expression de type $\langle e,\ t \rangle$. Avec des catégories complexes où la direction de la barre oblique indique si l'argument est placé à la gauche si une barre oblique est utilisée, ou à la droite en cas de barre oblique, la catégorie (S\NP)/NP, qui correspond à un groupe verbale transitif, nécessite un NP comme argument à sa gauche et se traduit par une expression SRD de type $\langle \langle e \rangle, \langle e,\ t \rangle \rangle$. La catégorie lexicale NP/NP qui indique un article ou un adjectif, nécessite une catégorie NP comme argument à sa droite, et possède le type sémantique $\langle \langle e,\ t \rangle,\ \langle e \rangle \rangle$. La figure 7.1 montre quelques exemples de telles correspondances.

Les expressions λ-typées peuvent également être considérées comme une interface entre les catégories lexicales et les expressions SRD. Si $\lambda x.\varphi$ est une expression λ, alors x est une variable de type $\langle e \rangle$ et φ est une formule de type $\langle t \rangle$.

4.3 Construction d'un analyseur SRD pour le français

Notre objectif dans ce travail a été le développement d'un *framework* qui permette de réaliser des analyses sémantiques pour la langue française. Le manque de disponibilité de données d'entraînement massives nous a empêché d'utiliser des modèles de réseaux de neurones profonds. En outre, l'utilisation de l'approche traditionnelle a fait ses preuves dans de nombreuses applications. Ainsi, *Boxer* (programmé en Prolog) est un outil efficace pour obtenir une représentation sémantique de phrases en langue anglaise et il est devenu le composant le plus important dans la génération des corpus de représentation sémantique GMB et PMP (**abzianidz**; Bos, 2008). En suivant l'approche empirique, nous pouvons éviter la contrainte sur les données disponibles et nous servir de la propriété de compositionnalité dans l'analyse de la construction de phrases.

Nous présenterons une architecture complète pour obtenir une représentation du sens à partir d'un énoncé français fourni en entrée (figure 4.1). Cette architecture comprend différentes étapes de traitement que nous pouvons regrouper dans trois tâches principales: (a) pré-traitement et analyse de la syntaxe et des dépendances de l'énoncé, (b) utilisation du formalisme grammatical des GCC, (c) analyse de la représentation sémantique de discours.

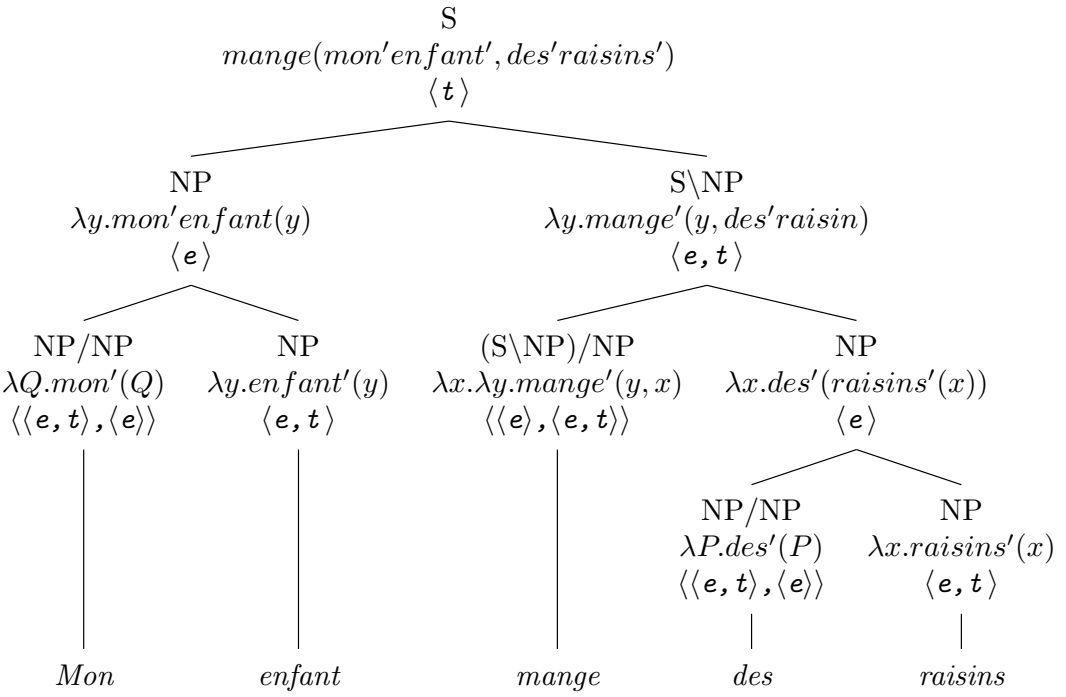


FIGURE 7: Correspondance entre des différentes expressions pour la phrase "Mon enfant mange des raisins»

Analyses syntaxique et grammaticale

Nous utilisons des grammaires de dépendances pour analyser la structure syntaxique de phrases données. La plupart du temps, la racine de l'arbre de dépendances correspond au verbe principal et tous les autres mots sont directement ou indirectement reliés à ce verbe par des arêtes dirigées. Chaque arête a une étiquette qui décrit la nature de la relation entre les deux mots. Ces étiquettes appartiennent à un ensemble de fonctions syntaxiques, par exemple, sujet, objet, oblique, déterminant, attribut, etc. Les fonctions syntaxiques sont des relations grammaticales jouant un rôle important dans la reconnaissance des composants de la phrase.

Pour le discours fourni en entrée, les informations syntaxiques sur les mots et leurs interrelations peuvent être obtenues via un analyseur de dépendances. Il existe de nos jours plusieurs analyseurs de dépendances pour le français tels que MaltParser Candito, Crabbé, and Denis, 2010, Stanford Parser (Green et al., 2011), MSTParser (McDonald, Lerman, and Pereira, 2006), SpaCy (Honnibal, Goldberg, and Johnson, 2013; Honnibal and Johnson, 2015), Grew Parser (Guillaume and Perrier, 2015). Nous avons choisi d'utiliser MaltParser afin d'obtenir des informations morphosyntaxiques sur les mots dans les phrases. Nous conservons les informations suivantes pour chaque mot: lemme, étiquettes POS et relations de dépendance.

Nous avons utilisé MElt (Denis and Sagot, 2009; Denis and Sagot, 2012; Sagot, 2016) pour effectuer la tokenisation et l'analyse morphologique (radical, préfixe, suffixe, etc.) de chaque mot des phrases d'entrée. La sortie de cette étape est utilisée comme entrée pour MaltParser afin d'analyser les structures de dépendance de la phrase donnée.

Extraction de l'arbre de dérivations GCC

Afin d'obtenir un arbre de dérivations CCG pour chaque phrase française d'entrée, nous avons utilisé l'approche empirique introduite à la section précédente 3. En utilisant les informations de syntaxe et de dépendance obtenues à l'étape précédente, nous traitons d'abord les mots qui ont des catégories lexicales uniques. Ainsi, par exemple, les noms ont une catégorie lexicale NP, les

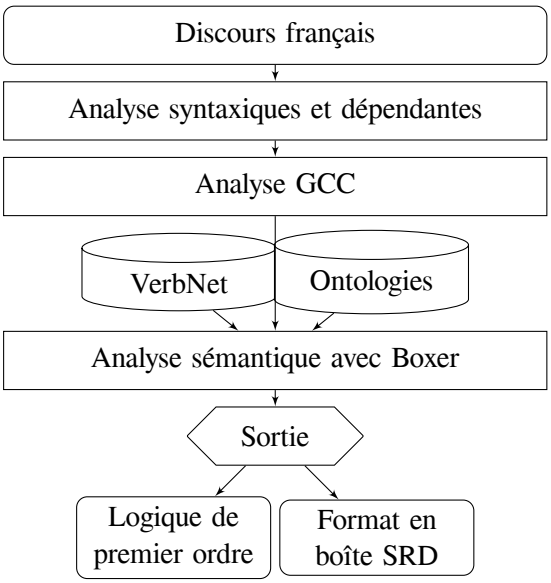


FIGURE 8: Architecture proposée pour l'analyse sémantique française

adjectifs ont une catégorie lexicale NP/NP ou NP\NP selon qu'ils placés à gauche ou à droite du nom, etc. Une fois ces catégories lexicales uniques (mais dépendantes de la position, puisque, par exemple, les adjectifs en français peuvent être situés des deux côtés du nom) attribuées, nous passons aux verbes. La catégorie lexicale S\ NP est affectée à un verbe principal ayant un sujet à sa gauche, puis on ajoute un /NP (ou un \NP, selon sa position par rapport au verbe) pour chaque objet direct ou indirect (dans l'ordre des mots dans la phrase).

L'étape suivante consiste à binariser l'arbre de dépendances sur la base d'informations sur la structure de la phrase dominante: en français, la plupart des phrases sont de types SVO ou SOV. En utilisant cette propriété linguistique générale, nous utilisons un algorithme pour extraire et classer les composants de la phrase en sujet, objet direct, objet indirect, verbes et phrases complémentaires. Cet algorithme vise à transformer un arbre de dépendances en un arbre binaire. En appliquant les règles combinatoires, nous obtenons l'arbre binaire correspondant à une dérivation GCC donnée pour la phrase d'entrée.

Pour chaque phrase d'entrée, nous obtenons un seul arbre de dérivation GCC, correspondant à son entrée d'arbre de dépendances. L'arbre de dérivation CCG de sortie est re-écrit de manière à être compatible avec le format d'entrée de Boxer. En même temps, la phrase est analysée afin d'extraire des composants d'entités nommées (par exemple, *lieu*, *personne*, *date*, *heure*, *organisation*, etc.) en utilisant l'application SpaCy.

Représentation sémantique à travers *Boxer*

Implémentée dans le langage Prolog avec un code source accessible au public, l'application *Boxer* est conçue pour fournir une analyse sémantique de discours pour l'anglais avec des arbres de dérivation GCC en entrée et une représentation du sens sous la forme de SRD en sortie. Pour faire de même en français, nous avons dû adapter le code source aux spécificités de la langue française.

Les verbes sont la composante centrale de la plupart des phrases. Une fois qu'un verbe est donné, nous sommes en mesure de connaître les composants qui peuvent lui être attachés. Par exemple, le verbe «acheter» doit être suivi d'un objet direct, et le verbe «dormir» n'en dispose pas de par son intransitivité. Les relations entre un verbe et ses arguments nominaux sont illustrées par des rôles thématiques (par exemple, agent, expérience, thème, objectif, source, etc.).

FIGURE 9: L’arbre de dérivation CCG de la phrase «Tous les soirs, mon voisin met sa voiture au garage».

Dans *Boxer*, les verbes et leurs rôles thématiques sont extraits du corpus de ressources lexicales VerbNet (Schuler, 2005). Pour le français, nous avons utilisé le corpus VerbNet français (Pradet, Danlos, and De Chalendar, 2014) (voir un exemple sur la figure 10), tandis que un ensemble d’ontologies nous a fourni des relations hiérarchiques ou d’équivalence entre entités, concepts.

Les problèmes concernant l’anaphore et les déclencheurs de présupposition introduits par des phrases nominales, des pronoms personnels, des pronoms possessifs, des pronoms réflexifs, des pronoms démonstratifs, etc., sont traités au cas par cas, sur la base de l’algorithme de résolution proposé dans (Bos, 2003). Enfin, la représentation sémantique de l’analyse du discours est produite sous deux formalismes différents: la logique de premier ordre et le format de boîtes imbriquées de la SRD.

% Primary: ***'NP V NP'*** ('allow-64')

% Syntaxe: [np:***'Agent'***,v,np:***'Thème'***]

% GCC: (s:dcl\np)/np

% Rôles: [***'Thème'***,***'Agent'***]

% Exemple: ***'Luc approuve l'attitude de Léa'***

VerbNet:

(approuver, (s:dcl\np)/np, [***'Thème'***,***'Agent'***]).

(autoriser, (s:dcl\np)/np, [***'Thème'***,***'Agent'***]).

(supporter, (s:dcl\np)/np, [***'Thème'***,***'Agent'***]).

(tolérer, (s:dcl\np)/np, [***'Thème'***,***'Agent'***]).

FIGURE 10: Un extrait de la ressource lexicale Verbnet pour le français

5 Expérimentation et évaluation

Nous illustrons les capacités de l'analyse du discours français via notre architecture par l'exemple suivant: «Tous les soirs, mon voisin met sa voiture au garage. Il arrose ses rosiers avec son fils.». Nous avons deux phrases dans ce texte, contenant des pronoms possessifs, des pronoms personnels et des groupes nominaux.

$\exists\ z3\ z7\ z8\ z9\ z10\ z11\ z12\ z13\ z14\ z5\ z6.(np\_fils(z6) \wedge de(z6,z5) \wedge np\_male(z5) \wedge np\_rosier(z14) \wedge de(z14,z13) \wedge np\_male(z13) \wedge np\_male(z12) \wedge au(z10,z11) \wedge np\_garage(z11) \wedge np\_voiture(z10) \wedge de(z10,z9) \wedge np\_male(z9) \wedge np\_voisin(z8) \wedge de(z8,z7) \wedge np\_male(z7) \wedge \forall\ z4.(np\_soir(z4) \rightarrow \exists\ z1.(a\_topic(z4) \wedge \exists\ z2.(Recipient(z2,z10) \wedge Theme(z2,z8) \wedge a\_mettre(z2)) \wedge alors(z4,z1))) \wedge Theme(z3,z14) \wedge Actor(z3,z12) \wedge a\_arroser(z3) \wedge avec(z14,z6))$

FIGURE 11: La sortie en logique du premier ordre de l'exemple

Nous appliquons d'abord un analyseur pour obtenir des relations de dépendances. Nous obtenons alors un arbre de dérivation GCC en sortie de l'étape d'extraction de dérivation GCC. Les résultats sont représentés (figure 9) dans un format compatible avec le format d'entrée de *Boxer*. Chaque mot est traité comme un terme (*t*) avec les informations suivantes: *catégorie lexicale GCC*, *mot original*, *lemme*, *étiquette POS*, *chunks* et *informations d'entité nommée*.

Le lecteur peut voir la sortie de l'exemple dans deux formats: logique de premier ordre (la figure 11) et format en boîte de SRD (la figure 12). Dans le cas de la langue française, *Boxer* arrive à analyser correctement les phénomènes linguistiques tels que les pronoms possessifs (*ses, mon, sa, son*), les quantificateurs propositionnels (*tout*) et les groupes nominaux (*sa voiture au garage*). Cependant, il y a encore place à l'amélioration, par exemple, on n'obtient pas l'ordre chronologique des actions dans l'exemple.

D'autre part, nous avons expérimenté notre système avec 4 525 phrases du corpus français FTB () afin d'avoir une vue d'ensemble sur un corpus à large couverture. La longueur des phrases

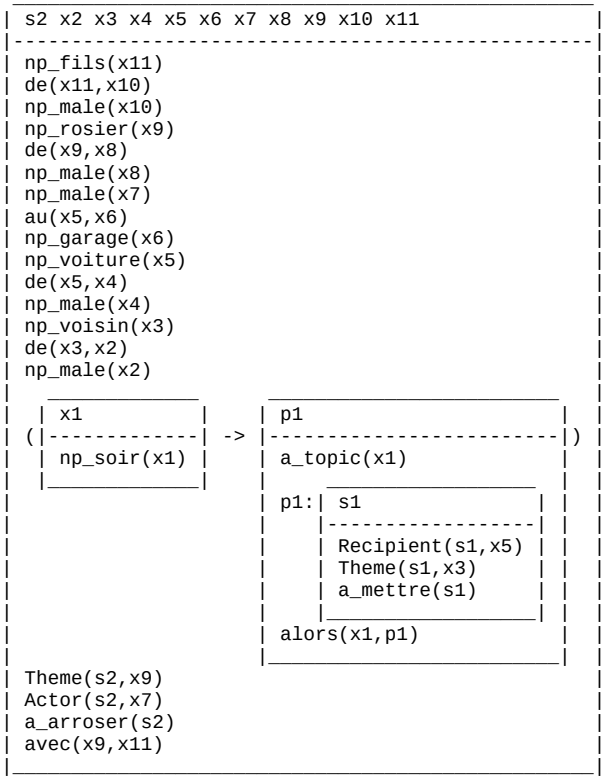


FIGURE 12: La sortie en SRD de l'exemple

dans notre expérimentation est limitée à 20 mots car le corpus FTB a été extrait de journaux français, les phrases sont donc assez longues et complexes. Nous avons obtenu que 61,94% des phrases peuvent être analysées avec succès par notre système. En analysant les erreurs survenues dans nos résultats, nous identifions deux causes principales : la première découle d'erreurs dans l'analyse des dépendances ou dans l'étape d'analyse GCC ; la seconde provient de manques dans la définition de la représentation sémantique des catégories lexicales GCC pour le français.

L'analyse sémantique est une tâche difficile dans le domaine du traitement du langage naturel. Nous obtenons une analyse du discours français étape par étape, et pour obtenir une représentation sémantique fidèle, nous devons nous assurer de l'exactitude des étapes d'analyse précédentes. S'il y a une erreur dans celles-ci, elle entraînera fatalement des erreurs dans les résultats. Par exemple, les erreurs des étiquettes de partie de discours sont la principale cause de résultats erronés. Ce type d'erreur va des erreurs de bas niveau des analyseurs de dépendances aux phrases qui sont intrinsèquement ambiguës et peuvent avoir plusieurs arbres de syntaxe, tels que *la belle porte le voile* où *belle*/*porte*/*voile* peuvent être aussi bien nom/verbe/nom que adjectif/nom/verbe.

Enfin, des problèmes linguistiques complexes surviennent lors du traitement d'énoncés dans lesquele-s l'omission d'un mot ou d'un groupe de mots – qui autrement sont nécessaires à l'exhaustivité grammaticale d'une phrase – est tolérée. Ces problèmes entraînent souvent une identification incorrecte des arguments verbaux. Par exemple, dans *Henri veut aller au parc et sa mère à la bibliothèque*, l'absence d'un verbe entre les mots *mère* et *à la bibliothèque* aboutit à l'obtention de catégories lexicales incorrectes pour les mots restants.

6 Conclusion

Nous avons proposé une approche empirique pour construire une application de représentation sémantique pour la langue française, basée sur le cadre GCC (pour analyser dans le cadre de

la phrase) et sur SRD (pour traiter les relations sémantiques entre les phrases d'un discours). Les informations syntaxiques et les relations de dépendances entre les mots sont analysées et extraites à l'aide d'un analyseur de dépendances. Après cela, les informations sont utilisées pour construire un arbre de dérivation GCC pour chaque phrase. Enfin, les phrases sous forme d'arbre de dérivation GCC sont traitées par Boxer, que nous avons adapté à la langue française par intervention directe sur le code source, et nous obtenons une représentation sémantique du discours en logique du premier ordre ou en format en boîtes imbriquées SRD. Dans les recherches futures, nous prévoyons de construire un corpus avec des discours et leur représentation de signification sous forme de SRD, en utilisant cette application. Nous prévoyons également d'utiliser des modèles de réseaux de neurones profonds pour améliorer la robustesse des résultats obtenus.

Chapter 1

Introduction

The emergence and popularization of social networks in this century have opened up new communication channels between individuals as well as between organizations and individuals, by replacing traditional interaction channels such as post, telephone and email. Along with the explosion of the number of users of social networks, organizations have realized the importance of being adequately represented on social networks, in order to increase their presence and establish a communication channel with their customers. Therefore, information flows received and processed by the appropriate agents in organizations have encountered an increasing growth. Much of the information is created in the form of texts such as ordered lists of comments or messages around a given theme. The automatic or semi-automatic processing of these texts is an essential requirement for any organization. However, the analysis and understanding of texts by the machine is still confronted with many challenges. On social networks, conversations between two or more people involve the exchange of news, ideas, evaluations, requirements, questions or complaints, in the form of a sequence of messages or comments. In general, these texts are written in natural language and cannot be directly understood by machines. Thanks to the advancements in the domain of natural language processing, machines can assist people in performing various tasks involving reading and understanding language, from simple tasks such as spelling correction or spam detection to complex tasks such as question answering, text summarizing, and machine translation. More specifically, the classification of conversations can be realized by the use of supervised learning algorithms, and this can be realized with a high level of precision. However, in order to understand a conversation, the analysis of discourse must go through different complex and interactive levels of text study such as morphology, syntax, semantics and pragmatics. In other words, the meaning of a sentence or a discourse in context needs to be captured and synthesized through a general language representation that can be interpreted both by the machine and by humans.

It is the role of a semantic representation framework to introduce a formalism through which machines become capable of capturing the meaning of expressions in natural language. In fact, semantic representations define lexical categories and rules that map meaning to syntax. Thus, texts in natural language are transformed into a new representation form where semantic relationships are linked to each other and implicit information can be explored by reasoning rules. Ontologies are of the most popular semantic representation frameworks used for knowledge description. More specifically, ontologies are appropriate for the construction of knowledge bases in particular domains. We can build information warehouses and access their knowledge with requests through a special query language. In particular, by organizing knowledge through ontologies, we can provide feedback for information requests or questions in conversations.

A conversation in the frame of communication between an individual and an organization on social networks will normally involve a set of topics of interest for the individual or for the organization. Such a conversation can be considered as a dialog, where texts are exchanged between the agents. The analysis of these texts must be placed in the context of the given conversation, and information can be spread out in different sentences—we call a sequence of sentences, a *discourse*. Synthesizing and representing information contained in a discourse

allows fast capture of the individuals' intentions, and this gives us a means to classify more precisely the type of discourse.

In the context of this thesis, we concentrate on studying discourses between individuals and organizations on social networks. We look for approaches to analyze these discourses at the semantic level for French language. More specifically, our research involves analysis of discourses and their representation in a given semantic representation form. Besides that, we create knowledge bases through ontologies in order to semantically annotate the discourse and to obtain relationships between concepts that occur in it.

1.1 The context

In the last decade, social networks have become an important communication channel for companies and organizations with customers. Through this channel, customers can easily communicate with a counselor or a consultant at any time of the day. However, the number of users attempting to communicate through social media channels is increasing. Therefore, companies are required to build chatbot systems to help them receive and process messages from customers over social network applications. Nevertheless, the application scope of these chatbots is quite limited with simple functions based on defined templates or existing training datasets. Currently, most chatbots do not access the content of texts in a robust manner and therefore are unable to take decisions in a conversation with users. Essentially, all processes and decisions are still made by company agents. Therefore, understanding the content of user messages can help improve the chatbot system in classifying messages, in responding or in providing requested information, as well as in reducing the workload of agents.

In general, data involved in daily exchanges between customers and supporters, not only on social networks but also via emails, messenger, and forums are mainly in textual form. These communications begin with the enumeration of customer issues. Thereafter, exchanges pursue in order to clarify and resolve these issues. Instead of processing message threads and redirect them to counselors, having the machine analyze messages and capture meaning can assist companies in providing a better solution for understanding user intentions and for providing answers to customers.

Messages in a conversation are created in chronological order, and information can be spread in more than one sentences, produced at different times. Therefore, messages should be placed into the context of the conversation to be understood. This is also necessary, in particular, to detect the intentions of users. We normally begin with syntactic analysis in order to analyze and determinate the structure of a text considered as a sequence of tokens with respect to a given formal grammar. After terminological analysis we provide the meaning of terms, their roles, and the relationship between individual terms that we use in order to detect lexical hierarchies, such as hyponyms, synonyms, antonyms. At a higher level, and thanks to the principle of compositionality, semantic analysis allows us to access meaning of syntactic structures such as phrases, clauses, sentences or paragraphs. In order to accomplish this, we construct a semantic parser mapping natural language utterances into a semantic representation. These representations are regularly based on grammars or other underlying formalisms, used to derive valid logical forms.

With nearly 4,000 languages having an elaborate writing system among the 7,000 living languages on the world[1], every language cab be described through graphemes, morphemes, and grammatical rules allowing to create complete sentences. Therefore, syntactic analysis in each language is very different in terms of graphemes, and morphemes, as well as the relationship between elements in the sentence. We need to adopt a linguistic theory in order to analyze the

[1] Metrics from https://www.ethnologue.com/enterprise-faq/how-many-languages-world-are-unwritten-0

fundamental grammar structure of a language, for example, *Phrase Structure Grammar* (Chomsky, 1959) where the constituency relation concept involves subject-predicate division and relies on phrasal nodes such as noun phrases for subjects and verb phrases for predicates, or *Dependency Grammar* (Tesnière, 1934, Hays, 1960; Hays, 1964a; Hays, 1967, Osborne, 2019) in which there is a root for every sentence and other words are directly or indirectly connected to it, *Montague Grammar* (Montague, 1970a; Montague, 1970b; Montague, 1970c) where grammatical relationships are expressed through FOL and lambda calculus, or *Categorial grammar* (Bar-Hillel, 1953; Bar-Hillel et al., 1960; Lambek, 1988) that is a development branch of phrase structure grammar, in which syntactic constituents are combined as functions or according to a function-argument relationship.

Besides obtaining shallow structures such as grammatical constituents and syntactic relations, and identifying the parts of speech, our grammar theory explores the meaning of sentences or phrases through deep structure analysis. In this case the meaning of utterances and phrases is determined by combining the meaning of their subphrases using rules that are driven by the syntactic structure. Thanks to compositional semantics we obtain the complete meaning of a given utterance. Furthermore, meaning representation of an utterance is crucial in tasks such as linking linguistic elements to non-linguistic elements, representing unambiguous canonical forms at the syntax level, reasoning on what is true in the world as well as inferring new knowledge from semantic representations. We have different approaches at our disposal for representing meaning: first-order logic, frames, rule-based architectures, and conceptual graphs, to name a few.

Discourse processing in the context of this thesis focuses on conversations or communications in which coherent sequences of sentences/phrases are created by using social network applications. We analyze and investigate features that are part of communicative acts such as context of utterance, relationship, and mode of discourse. As one of the goals of this analysis, we identify the topic of a conversation or the intention of the individual leading it. Therefore, the determination of formal semantic framework helps in constructing mathematical models that are used to define the relations between expressions in discourse.

The purpose of this thesis is to propose an approach for analyzing and capturing the meaning of an utterance in the context of discourse in social networks, as well as the development of a framework that allows parsing utterances in order to obtain meaning representation with input utterances in French language. In previous decades many fundamental language theories have been introduced and applied to syntactic and semantic analysis; however there are still many problems to solve, corresponding to the following questions:

1. What is a discourse in social networks and how can we extract discourses from social networks?

2. How do we analyze texts from a discourse at the syntactic and semantic level?

3. What semantic representation framework is suitable for analyzing and representing discourses?

4. How do we extract meaning from a sentence or from a discourse ?

1.2 Challenges

In natural language, words, phrases or sentences can sometimes have more than one meaning. The ambiguity of language is always a complex problem that NLP algorithms have to solve by taking disambiguation decisions. In fact, there are different kinds of ambiguity: ambiguity of word sense, of word category, of syntactic structure, of semantic scope. Each kind of ambiguity requires a specific approach. For example, a knowledge-based approach will map meanings to

words in context by using the WordNet corpus (Chaplot and Salakhutdinov, 2018); a supervised approach to the same problem will require a large training data (Huang et al., 2019). Nevertheless, despite the variety of approaches, the results we obtain are still limited and depend on the particular knowledge domain.

Interpretation of an entity, occurrence or concept in a given utterance may depend on entities, occurrences or concepts located in other utterances. This linguistic phenomenon, called anaphora, raises the problem of how to identify relations between entities, occurrences or concepts and their referents in one or many utterances. There are generally many types of anaphora such as pronominal anaphora, propositional anaphora, adjectival anaphora, modal anaphora, etc. Since more than one sentences are potentially involved, the anaphora resolution problem should also be coined in the context of discourse.

Discourse on social network has its own characteristics, that singularizes it from other types of discourse. First, utterances in social networks inherit both from the spoken and from the modality. Secondly, emoticons can be used to accompany words to express sentiments in the utterance. In general, emoticons are considered to be handy and reliable indicators of sentiments and are used as expressive, non-verbal components in utterances playing a role similar to the one of facial expression in speech (Novak et al., 2015). The use of emoticons may or may not obey to grammatical rules of a given natural language. Thus, using emoticons for extracting meaning from utterance remains a challenge.

Converting a natural language utterance to a machine-understandable representation can be operated using semantic parsing. There are numerous fundamental formal semantics theories for representing meaning such as Abstract Meaning Representation (Banarescu et al., 2013), Discourse Representation Theory (Kamp and Reyle, 1993), or Conceptual Meaning Representations (Abelson and Schank, 1977; Jackendoff, 1992). Each framework has its own advantages and disadvantages and therefore the choice of a framework for analyzing discourse on social networks can play an important role with respect to the quality of obtained results.

Moving from natural language processing to natural language understanding remains a big challenge for researchers in computational linguistics. With fundamental theories about syntactic, semantic, or discourse analysis, we can construct parsing systems in order to capture and represent utterance meanings. However, there are still many drawbacks because of the diversity and complexity of natural language used to represent the real world. It is no exaggeration to say that solving this problem will open a new era for human-machine interaction.

1.3 Contributions

Heading towards the ultimate goals of achieving semantic representation of French discourses on social networks, here are the contributions we will describe in the frame of this thesis:

- Based-on the fundamental theory of Combinatory Categorial Grammar and Dependency Grammar, a novel algorithmic method that allows analysis of the dependency structure of a sentence and its transformation into a CCG derivation tree (Le and Haralambous, 2019). This is an important step to obtain information on syntax and semantics of the sentence.

- Applying the above method to the French Tree bank corpus, we created a new corpus resource for the French language. In general, a CCG corpus (such as CCGBank[2] for English), can be used as training data for machine learning or deep learning algorithms in order to create a CCG parser for a given language (Le and Haralambous, 2019).

[2] http://groups.inf.ed.ac.uk/ccg/ccgbank.html

- In order to obtain a CCG parser for French sentences, we proposed a supervised architecture model for the CCG Suppertagging task based-on using the morphological and dependency information in the sentence (Le and Haralambous, 2019). The experimentation and evaluation have been realized on two corpora: *French Tree Bank*[3] for French and *Groningen* M∃∀NİNG *Bank*[4] for English.

- Finally, by combining the above French language CCG analysis results with Discourse Representation Theory, we propose an architecture for achieving a semantic representation from French discourse input. The output is obtained in two formats: First-Order Logic and Discourse Representation Structure (Le, Haralambous, and Lenca, 2019).

[3] `http://ftb.linguist.univ-paris-diderot.fr`
[4] `https://gmb.let.rug.nl`

Part I

Discourse Analysis and Social Networking Platforms

Chapter 2

Discourse on Social Networks: Challenges and Opportunities

The emergence of social networking platforms provides a new communication channel of our times, in which people can create and publish their thoughts, information, statements about any topic, freely and without limitation. In general, each platform has its own special features and characteristics as well as its own target audience. User-generated contents therefore will be different in terms of format, content and purpose. In this chapter, we will introduce an overview about social networking services, their features and their user objects in order to unravel differences between these platforms and user-generated contents which we aim to capture and explore. Furthermore, we also synthesize challenges and opportunities on these platforms, which will be geared towards serving businesses or organizations with their clients or potential clients.

2.1 Social Networks Overview

2.1.1 Social Networks Definitions

The emergence of the Internet in the 60s of the last century has established a new era for the world of computers and communications. The Internet provides a new communication channel for the transmission of information, a new mechanism for information dissemination, and a new medium for collaboration and interaction between individuals and computers without depending on geographic location (Leiner et al., 2009). A notorious application of the Internet is the World-Wide Web[1] which is an information system where resources such as Web pages and multimedia contents are identified by unique links and are accessible over the Internet. There are currently over 1.7 billion Web sites on the World-Wide Web and this number steadily increases. A Web site can be have many functions and can be used in various fashions depending on its purpose, such as personal Web sites, corporate Web sites for organizations, governments, or companies, searching portals, commercial Web sites, or social networking Web sites.

Another invention in the domain of communications, namely smartphones, has brought important changes in the way people interact on Internet-based applications as well as on Web-based services. According to statistics of the gsma.com Web site based on real-time intelligence data, there are currently over 5.13 billion mobile device users in the world. Mobile devices are currently used not only for making and receiving calls or messages over a radio frequency but also for a wide variety of tasks ranging from playing games, sharing files such as pictures and videos, accessing the Internet through the use of integrated Web browsers. Using social

[1] Introduced by Tim Berners-Lee, a British computer scientist, in 1989. The first Web site—info.cern.ch—has been developed on August 1991 while he was working at CERN, in Geneva, Switzerland. He is also the author of the *Semantic Web* the central idea of which is the availability of a common framework for data sharing and re-usability across application, enterprise, and community boundaries (Berners-Lee, Hendler, Lassila, et al., 2001).

networks through dedicated applications on mobile device has become an inevitable trend of modern times. Thereby, people can access their social network at any time, and at any place.

Social Networking Sites or *Social Networking Platforms* are currently popular social media terms, representing a multitude of different online platforms allowing people to build social relationships with other people, to communicate with each other, to discuss opinions about news, events or favorite topics, to share knowledge about similar interests, to rate and review products or company services. More specifically, these platforms are built upon Web-based services or Internet-based applications that are designed to enable users to construct public or semi-public profiles within a bounded scope of the platform, to search and establish a connection to other individuals or groups in order to or articulate or view their sharing contents, or to interact with them through messages (Boyd and Ellison, 2007; Obar and Wildman, 2015; Quan, Wu, and Shi, 2011). Social Networks provide a medium for creating and publishing contents for all users. User-generated contents can be read by other community members, and can be continuously modified by authors.

The above description of social networks is based on their most common characteristics. Besides, each social network platform can have its own specificity depending on its purposes and audiences. Along with allowing the participation of individuals to social networks, *connectivity features* enable them not only to make their presence visible but also to meet with each other to form virtual communities. Users can create content by *interaction features* such as comments or messages. In general, user profiles, connectivity and interaction will be required on most current social network platforms. In addition, each platform has specific features depending on its purpose. For example, linkedin.com is aimed for professionals, companies and businesses, and its users can create profiles for spreading their professional achievements and career. Another example, instagram, is a platform mostly accessed via mobile devices and designed for the young users—it allows sharing of visual creative content.

Every social network is distinct in its own way, reaches out to different audiences and serves for a different purpose. Classification of social networking sites can be based on criteria such as user interests or shifting technology features. Thereby, social networking sites can be classified into three general categories: *the generic social networking*, *the professional* and *the teenager social networking sites* (Wink, 2010). There have been other classifications, such as (White, 2016) using seven categories such as *social connections*, *multimedia sharing*, *professional*, *informational*, *educational*, *hobbies* and *academic networking sites* or (Aichner and Jacob, 2015) using thirteen categories such as *blogs*, *business networks*, *collaborative projects*, *enterprise social networks*, *forums*, *microblogs*, *photo sharing*, *products/service reviews*, *social bookmarking*, *social gaming*, *social networks*, *video sharing* and *virtual worlds*. Based on user expectation (Foreman, 2018), we propose the division of social networks into the following categories:

- **Social Connecting Networks**. They currently have the largest number of users in the world. These social networking sites were constructed for Internet users wanting to keep in touch with friends or family members and they are used for sharing information as well as keeping up-to-date activities of other people whether across town or around the world. Typical representatives for this category are Facebook.com, Twitter.com, Tumblr.com.

- **Multimedia Sharing Networks**. They have been designed for users wishing to post and share music, videos, photos or other kinds of media. The main representative of this category is Youtube.com, the largest user-driven content provider in the world. Its success comes from the user-to-user social experience through multimedia content (Wattenhofer, Wattenhofer, and Zhu, 2012). Besides, we have many other platforms such as: Instagram.com, Vimeo.com, Flickr.com, TikTok.com.

- **Social Messaging Networks**. They are accessed through applications on mobile devices or platforms that allow users to send messages to, or receive messages from, other

users. Social networking services have created messaging services as a basic component of their overall platform. The most widely used mobile applications of this kind are Whatsapp.com, Messenger.com, WeChat.com, SnapChat.com, Viber.com.

- **Professional Networks**. They target professional users to share their professional experience, abilities or qualifications, to build relationships with new contacts, to stay in touch with business partners and to create business opportunities. The pioneer of this category is Linkedin.com for business professionals, ResearchGate.com and Academia.edu in the Academic domain. Beside these we also have ViaDeo.com, Xing.com, MeetUp.com.

- **Publishing and Discussion Networks**. They connect users through content, news, information or opinions. Online tools are provided to find, discuss or publish content. In this category, we have the emergence of social networks sites such as Quora.com, Reddit.com, or Medium.com.

- **Consumer & Business Review Networks**. They give users a place to find, review or share information about products, services, brands, travel destinations or anything else. These platforms are also a place to learn from the experiences of others at any time without depending on one-way information from businesses. We have some names that lead the way on this competition such as Google My Business, Yelp.com, TripAdvisor.com, FourSquare.com.

- **Social Commerce**[2] **Networks**. They are designed to integrate social experience with online transaction experience. Buying or selling is done by users for who online tools are provided to find items, to follow brands, to share their experiences and to make their transactions. Users can be individuals or organizations. Common networks in this group are GroupOn.com, AirBnb.com, Fancy.com, OpenSky.com.

- **Interest-Based Networks**. They are constructed to connect audiences based on common interests, hobbies, ideas or passions as reading, music, home design, online game, etc., by removing barriers of geography, time zones, cultural habits or societal restraints. Popular platforms are GoodRead.com, Friendster.com, Last.fm, Houzz.com, MegPlay.com.

- **Anonymous social networks**. They create a space where real-life information on users is not required to fill-in profiles and where users can post and share content anonymously. These social networking sites types are often considered as 'dark sides' of social media, thus they may not be suitable for some audiences. Among platforms of this class we have AferSchool.com, Whisper.com, Ask.fm.

2.1.2 A Brief History of Social Networks

In the early days of the Web, when the concept of social networks was still vague, a large number of Web sites have been built for various purposes. Some of them appeared in the form of generalized virtual communities with some characteristics of social networks such as Geocities (1994), TheGlobe.com, Tripod.com Classmates.com (1995) and PlanetAll.com (1996). These platforms have been designed to bring users together in order to interact with each other through chat rooms. Online publishing tools were built for users to create and share their personal information or ideas with others. Nevertheless, the first real social network site is considered to be SixDegrees.com, developed by Andrew Weinreich in 1997. This first platform allowed

[2]There are different definitions around the "Social Commerce" term. In our context, we used the definition: "Social Commerce is a subset of electronic commerce that involves using social media, online media that supports social interaction and user contributions, to assist in the online buying and selling of products and services." (Marsden, 2011).

users to create their profiles, connect with others and display the list of their friends. Making friends, expanding virtual groups, sending and receiving messages are important features of this site which is generally considered as being the first social network in history.

In the period between 1997 and 2001, many other social networks have were launched to bring some improvements in functionality. The amelioration of the profile function allowed users to customize their information into personal, professional, and dating profiles. Users could also receive updates from others by marking them as friends and manage relationships on their personal profiles. Typical examples are AsianAvenue.com (1997), OpenDiary.com(1998), Makeoutclub.com, CyWorld.com(1999), LunarStorm.se, BlackPlanet.com (2001). In 2001, Ryze.com was released as a novel type of social network for professional and business communities.

The next wave of social networks began when Frienster.com was launched in 2002 with improvements such as connectivity functions, share of content such as videos, photos, messages and comments with other members via profiles and networks. Users could contact other members and expand their networks through friend-of-friend relationships. Thereby, this platform gained over 300,000 users before May 2002 (Boyd and Ellison, 2007). The success of Frienster.com ushered in a new period of explosion of SNS. The big names of current SNS were born in the period between 2003 and 2008 such as LinkedIn.com (2003), FaceBook.com[3] (2004), Youtube.com, Reddit.com (2005), Twiter.com, QQ.com, VK.com (2006), and Tumblr.com (2007). Besides, we are also familiar with other cases likes MySpace.com, Flickr.com, Hi5.com (2003), Yahoo!360 (2005). Based on the web technology development and the large amount of users around the world, a multitude of different SNS were released with the basic architecture of a SNS to which special functions were added, aiming to connect different audiences from personal users to professional users, and creating and sharing multimedia content, blogs, microblogs, discussions and reviews.

In the period between 2009 and 2019, the concurrence between SNS became more intense than ever by witnessing the collapse or transformation of SNS of previous stages in other forms. MySpace.com, Frienster.com, Yahoo!360 and many others were the typical losers of this competition. Many causes lead to the failure of these platforms such as the lack of loyalty to its users, or the lack of technological innovation and of enhancement of the user experience. Besides, a major branch of social messaging platforms emerged during this period: WhatsApp (2009), Viber(2010), Messenger, WeChat, Line (2011), SnapChat, Telegram, Google Hangout (2013). Messaging or chat platforms became widely popular communication channel across the globe. These communication channels, some of which originated via SNS, have developed into independent platforms such as Google Hangout derived from Google+ and Messenger.com separated from Facebook.com.

2.1.3 User Characteristics and Motivations

Table 2.1 represents the mapping between active behaviors by SNS users and available functionalities provided by SNS platforms. In general, user behavior on SNs can be classified in four categories, namely: content creation, content transmission, relationship building and relationship maintenance (Chen et al., 2014). Content creation is focused on creating or generating content by microblogs, blogs, photos or videos in the social network site. These user-generated contents can possibly contain valuable information that may influence the behavior of others, for example, user-generated reviews can change consumers' purchase intention (Park, Lee, and Han, 2007). Content transmission behavior is identified by sharing information on SNS among friends, colleagues or family members. With this behavior, information and knowledge

[3] Facebook was launched on February 4, 2004 with a registration limitation to students in the US and Canada. It opened to the general public in 2006.

Active behavior	**Functionality**
Content Creation	• Posting blog/articles • Changing/posting current status • Posting photos/videos • Updating/editing profile • Tagging photos, hashtagging
Content Transmission	• Sharing resources form other sites • Sharing friends' shared material • Sharing friends' blogs/microblogs • Sharing friends' photos/videos • Sharing or transmitting friends' statuses
Relationship Building	• Creating groups or public profiles • Visiting public profiles or discussion boards • Joining groups or public profiles • Creating events and sending invitations • Searching friends and sending applications for adding friends, following new profiles • Accepting applications for adding friends • Sending private messages to non-friends
Relationship Maintenance	• Responding to invitations to participate in events from groups • Interacting with groups • Participating in friends' topics • Commenting on photos/videos • Chatting with friends • Looking at friends' personal information • Visiting friends' profiles • Looking at news about friends • Looking at friends' photos

TABLE 2.1: Mapping active user behavior and functionalities

is rapidly being diffused in virtual communities. User content transmission is nowadays an important channel for online marketing (Garg, Smith, and Telang, 2011). Building relationship on SNS allows users to enhance their online and social network by connect with each other based their shared interests, political views, or geographical locations. In addition, the relationship extension also contributes to the acceleration of the transmission of information. For example,

relationship building can help reinforcing attendance of a vendor to their customers (Wang and Head, 2007). Finally, similar to offline social relationships, an online relationship on SNS requires active maintenance to survive because relationships tend to decay over time (Roberts and Dunbar, 2011).

Online Social Networking Site usage is one of the most popular online activities of Internet users worldwide. In 2019, about 2.82 billion persons have been using social network services on their connected devices. This number will certainly keep growing and a predicted number for 2021 is of 3.1 billion persons. Figure 2.1 shows the distribution of users on different platforms. With more than 2 billion users, `Facebook.com` and `Youtube` are currently leading when it comes to the number of users. We also notice the rapid rise of messaging platforms such as `Whatsapp`, `FacebookMessenger` and `WeChat`. We have 6 over 18 platforms in this ranking specialized in messaging between users. This shows the prominent trend of users using social networks for messaging with others and it is one of the few things that people do more than social networking.

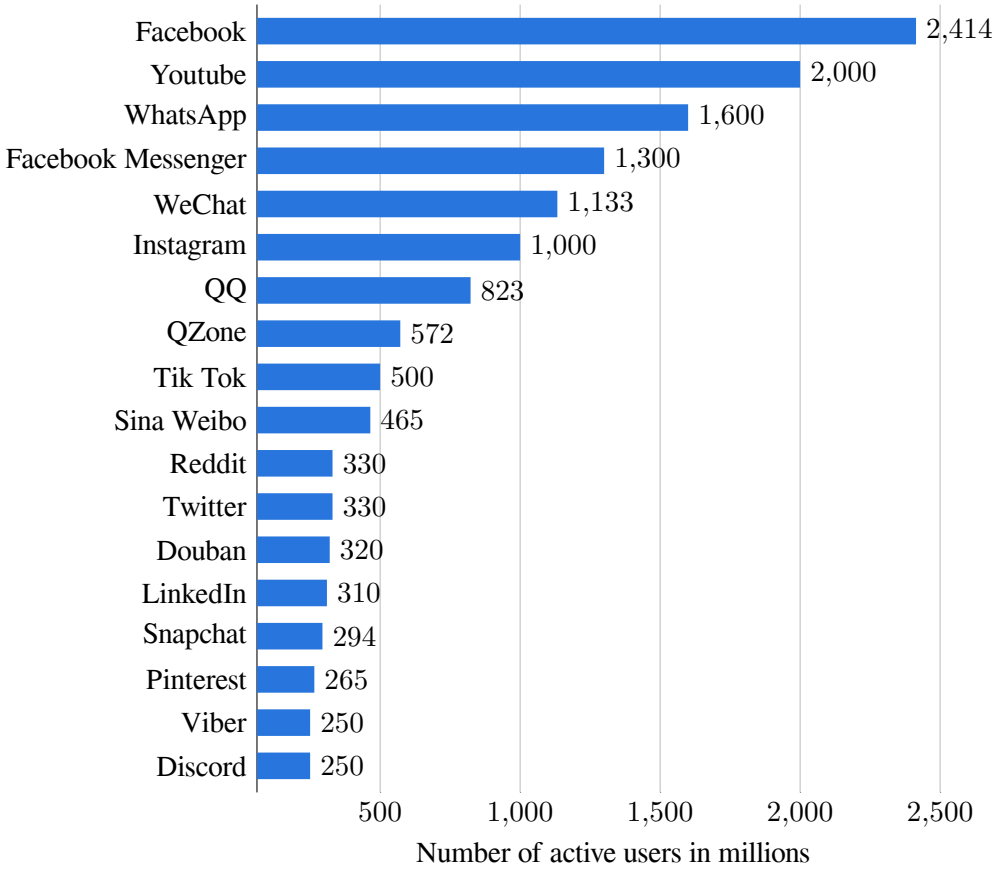


Figure 2.1: A statistic about amount of active users on the 18 biggest social networks (Source: wearesocial.com October 2019)

Similar to the annual increase in the number of persons using SNS, the amount of time people spend on SNS also keeps growing (Figure 2.2). In 2018, people spent, in average, 136 minutes per day on SN platforms, vs. 90 minutes, six years earlier. In more than two hours, various activities can be realized such as staying up-to-date with news and current events, to find funny or entertaining content, to share photos, videos with others, to meet new people, to search or find products to purchase, or to do messaging with others. These activities are the main motivations of people using social network platforms.

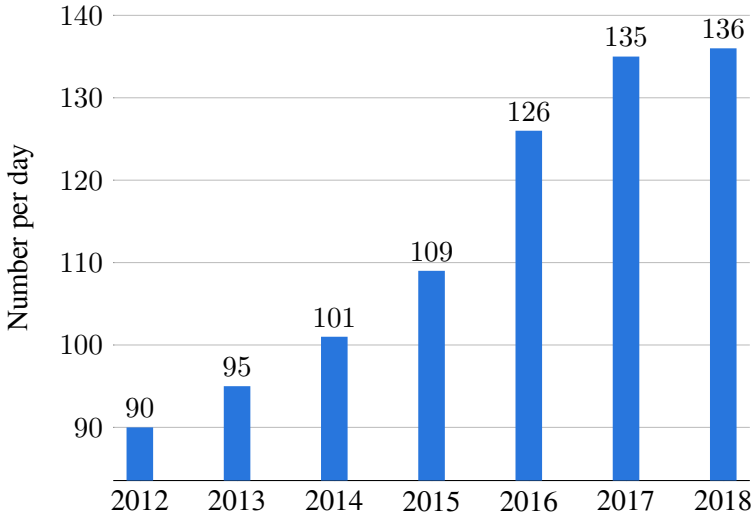


FIGURE 2.2: A statistic about daily time spent on social networks (Source: statista.com)

2.2 Opportunities and Challenges

2.2.1 Opportunities

SN services have become popular channels of information, media and communication for companies and their customers. Through this medium, customers have a platform to share reviews about specific services or products, to contact companies directly and to communicate with each other. The participation of a company in SN platforms gives it a change to maintain and make stronger relationships with its customers through its presence; it also allows it to improve its customer service, by receiving and handling reports on customer problems.

Traditionally a brand or company was widely known through word-of-mouth communication between individuals. Nowadays, SN services are offering enormous opportunities as a new communication channel for companies and their potential customers. A SN platform with millions of users interacting with each other is a great place for companies to ultimately reach their customers. By the use of search tools, companies can achieve benefits by exploring user-generated contents and user interactions, and by opening close relationships with their customers.

Instead of using traditional marketing channels such as radio, television, newspapers, SN platforms have become a modern approach to introduce or promote new services and products to potential customers. Customers can use these SN platforms to communicate with each other or to share their thoughts, ideas, comments, experiments, and evaluations about a company's services or products. In addition, SN platforms provide features allowing customers to follow information about products or services by other customers.

The message management feature, that exists on most SN platforms, allows users to converse with other users. Therefore, companies can receive various messages from their customers requesting information about services or products, or complaining about shortcomings. Conversation management helps companies to quickly provide required information or to resolve problems of their customers. Furthermore, using SN services for collecting feedback can be realized more quickly and more cost-efficiently than using traditional ways such as email, or phone surveys (Rana, Kumar, et al., 2016).

The use of SNS platforms as a media channel to provide company-specific content, services or products information is currently quite popular. Information is directly received by customers and, as a result, conversations about a particular topic are generated in real time in the network.

Discourses generated in this way contain useful information about opinions of customers with behalf to the company.

SNS platforms are commonly accessed by connected devices such as laptops, tablets, smartphones, and smart television. People can easily access SN platforms from anywhere and at any time. Therefore, they changed the way a company communicates with its customers. People can contact and converse with company representatives at any moment about products and services, be it for resolving problems or for obtaining information.

Marketing campaigns allow reaching customers in a low-cost, impact-full and effective way. SN platforms have become a crucial marketing channel for businesses. Indeed, a company can identify its target customers for new promotions, products or services based on customer attention on the company content or interaction activities on SNS.

2.2.2 Challenges

While there are numerous opportunities that SN platforms bring up for companies today, there are also key challenges that companies need to overcome to benefit from these opportunities. One of the challenges is diversity and abundance of SN platforms of different kinds such as social networking sites, online forums, instant messaging services. This requires companies or organizations to build common integrated social media strategies (Paliwal, 2015). Based on the characteristics of SN platforms, companies need a standard framework to determine on what they want to focus on and how to establish their target audience.

The spread of online fake news and misinformation can cause public panic and change the perception users have for a given brand. In particular, false contents can be received by any audience and can be shared with anyone on SN platforms. Therefore, misinformation can spread very fast in a cascaded manner and one of its negative effects is that it can be difficult to control (Tong, Du, and Wu, 2018). Misinformation detection has thus become a challenge and a hot research topic in recent years.

SNS platforms allow users to own personal virtual spaces. They can post any content in their space, and this content will be read by others. If some of the content is posted with negative sentiments about a given brand, it will directly affect the reputation of the concerned companies. There is little or no control on the type of content that gets shared on SN platforms. Accordingly, it becomes difficult for companies to protect themselves from destructive comments or rumors.

Information propagates through SN platforms via cascades. In a competitive context, if a new service or product is introduced to a target audience, it can grasp an advantage in terms of revenue and profits. Therefore, competing with other companies on SN platforms is a challenge that companies need to plan and execute in the frame of business strategies in order to grow.

Privacy, security, and safety issues in SN platforms play an important role in motivating people engaged in online social networking activities. Indeed, the availability of a large amount of personal information on SN services has attracted the attention of criminals and hackers who can engage in undesired or fraudulent activities such as identity theft, phishing, spamming, clickjacking or cross-site scripting (Deliri and Albanese, 2015). Malicious users can start attacks against SN providers in order to appropriate personal information and access information shared by users. Therefore, SN service users can be at risk.

2.3 Discourse on Social Networks

2.3.1 Basic Definitions

Definition 2.3.1. Discourse is a unit of text used by linguists for the analysis of linguistic phenomena that ranges over more than one sentences. Discourse is created based on a natural spoken or written communication between people on a particular subject.

The term 'conversation' denotes talk, especially of the informal type, between two or more people, in which news or ideas are exchanged. In ordinary usage, conversation usually implies a spoken expression rather than written language. However, interactions of people via SNS occurs regularly in written language. Therefore, we can use various terms—discourse, conversation, speech, talk—for what might appear to be very much the same thing.

Definition 2.3.2. Discourse Theory expresses broadly the study of aspects of language and communication distinct from linguistic structure. In computational linguistics, discourse theory attempts to provide a new theoretical apparatus and frameworks to deal with language phenomena.

Discourse theory intersects with various disciplines such as linguistics, communication studies, anthropology, literary studies, political science, social psychology, translation studies, to sociology or cultural studies (Karlberg, 2011). In our scope, discourse theory will be associated with language structure and phenomena, and with meaning representations through texts.

Definition 2.3.3. Discourse Analysis is a natural language processing task that focuses on extracting linguistic structures and phenomena using with different units and ranging from sentence and paragraph to conversation at different levels of syntax, semantics, or pragmatics. In addition, it also concerns studies of contextual meaning representation of texts.

In order to understand thoroughly a text or a conversation, we need to put it into context. Therefore, instead of focusing on smaller units of texts, such as words or phrases, discourse analysis tends to study larger units of language such as paragraphs, entire conversations or texts. Discourse analysis also provides an analytical framework and tool for the study of texts (Moser, Groenewegen, and Huysman, 2013).

2.3.2 Discourse Forms

Along with the explosion of mobile and new digital technologies (smartphones, tablets, Internet-connected devices), both the number of users and the amount of information sharing on SN platforms have increased. Users can easily converse with others. Therefore, the number of discourses keeps growing on all platforms. Discourses are regularly produced during written communication on a user-generated content, during user-sharing content, or during messaging between people about a particular topic.

	Text	Speech
Monologue	◇ Articles ◇ Reviews	◇ Broadcast-shared ◇ Radio-shared
Conversation	◇ Text messaging ◇ Blogs, Microblogs ◇ Discussions ◇ Emails	◇ Phone conversation ◇ Video conversation ◇ Live stream shows

Table 2.2: The different forms of discourse on SNS

A discourse can be a monologue or a conversation, depending on the number of participants. Various forms of discourses on SNS are enumerated in Table 2.2. Conversational discourses on social networking platforms encapsulate three main components: the topic (as textual or

visual content), the process involved in producing the texts, and the participants. When a user-originated topic appears on a SN it is read and attracts attention by other persons among their relationships. A discourse is then formed when the topic results in interactions between users of the SNS.

Chapter 3

Discourse Analysis and Applications

This chapter provides an overview of discourse processing and analysis based on extraction of meaning from language units such as words, terms, sentences, discourse. The relationship between language structures involves identifying the topic structure, the coherence structure, the coreference structure and the conversation structure for conversational discourse.

3.1 Intrasentential Linguistic Analysis

At the level beneath the sentence, three kinds of tokens are considered by linguistics (morphemes, words, terms) as well as their combinations in order to build sentences (syntax).

In oral language, units of sound used for communication between humans are called phones. Classes of phones that are discriminating in a given language are called phonemes. In a similar way, units of written communication between humans are called graphs and their discriminating classes in a given language are called graphemes. Phonemes and graphemes are concatenated (the former in a linear way, because of the characteristics of the human speech organ, the latter in a way that is not always linear but has a main direction and develops in a 2-dimensional space).

By a phenomenon called "first articulation," when emes (phonemes of graphemes) are concatenated, at some point meaning emerges, for example: in English, if we take the graphemes <c>, <a>, <t>, the grapheme <c> per se carries no meaning (other than the fact of being the third member of a system called the alphabet), the concatenation <ca> of <c> and <a> carries no meaning, but the concatenation <cat> carries a meaning.

Minimal units of meaning emerging from the concatenation of emes are called morphemes (and the discipline studying them, is called "morphology"). There are two kinds of morphemes: lexical morphemes (such as "cat") and grammatical morphemes (such as suffixes, prefixes, etc.), there are thousands of the former (and new morphemes appear frequently) and only a very small list of the latter (which are considered the "stable" part of a language).

"Words" are concatenations of morphemes (the discipline studying them is "lexicography," from the Greek "lexis" meaning word). They exist only in writing systems using intermorphemic spaces, such as alphabetic of abjad languages. Writing systems such as Thai, Chinese, Korean or Japanese use no intermorphemic spaces and hence the notion of "word" cannot be trivially defined. Nevertheless the term "word" is very widespread outside linguistics and therefore we cannot avoid it. Also word boundaries are often useful for disambiguation: for example in English, the graphemic string <the rapist> represents clearly three morphemes (the article <the>, the lexical morpheme <rap>, the suffix <ist>) while <therapist> represents two morphemes (the lexical morpheme <therap> and the suffix <ist>).

The third level of elementary tokens is the one of "term" (and the corresponding discipline, "terminology"). A term is a single word or a syntagm (a sequence of words according to a well-defined pattern) carrying a specific meaning in a given knowledge domain. Single-word

terms (also called "simple terms") have a monosemy constraint in a given knowledge domain. Multi-word terms (called "complex terms"), besides the monosemy constraint, must follow a specific pattern of part-of-speech properties and often carry more information than each one of their components. For example, "mobile phone" is a term in the domain of telecommunications, since it describes a very popular category of communication devices, and English speakers of our time period recognize the specific device, rather than just a phone which happens to be mobile. On the other hand, "pink phone" is just a phone which happens to be pink and does not correspond to a term in the domain of telecommunications.

In our study we took these three token levels into account. A morphological analysis of words provided the meaning of individual words, and terminological analysis allowed us to combine specific words that were components of complex terms. Once combined, we processed these complex terms as individual entities belonging to a given part-of-speech (the part-of-speech of the kernel of the complex term: for example, if "mobile" is an adjective and "phone" a noun, we attached the POS "noun" to the complex term "mobile_phone").

The discipline studying the catenation of morphemes (or words or terms) in order to build sentences, is syntax. We used parsers to obtain descriptions of syntactic structures of sentences in various formalisms (constituency syntax, dependency syntax, combinatory categorical grammars, etc.).

By a phenomenon called "second articulation," out of the catenation of morphemes (and hence of words and terms) emerges meaning. According to the principle of compositionality, the semantics of a sentence can be obtained out of the semantics of its parts (morphemes, words, terms) and the way they are combined (syntax). We have used the combinatory categorical grammar formalism to calculate sentence meaning out of the meaning of morphemes (words, terms), their category and their position in the sentence.

3.2 Discourse Analysis Overview

Discourse analysis has been a fundamental problem in language analysis beyond the sentence boundary where information in a sentence will be analyzed and extracted (e.g., identifying referents for a pronoun from context) in association with information of other sentences in a context (e.g., inferring a coherence relationship concerned with more two sentences). In other words, we focus on building frameworks or tools to automatically model language phenomena that are being implicit beyond the language units.

3.3 Coherence and Cohesion

According to **stede2011** we can define *coherence* as follows:

Definition 3.3.1. A coherent text is designed around a common topic. In the reading process, individual units of information enter meaningful relationships to one another. The text *coheres* and is not just a sequence of sentences to be interpreted in isolation.

In other words, coherence refers to linking adjacent material on the level of semantic/pragmatic interpretation. The way of doing this is not specified and one can equally well expect coherence to emerge through the use of connectives ("My son does not go to school yet, *because* he is only two years old") or implicitly through pragmatic means ("My son does not go to school yet. He is only two years old"). Coherence is what is expected from text, and surrealist games (such as the "cadavre exquis") or Zen meditation techniques invite the reader/meditating individual to build coherence *ab nihilo*.

On the other hand, *cohesion* can be defined as follows:

Definition 3.3.2. Sentences are connected to one another not only underneath the surface (thus requiring understanding and interpretation) but also by more-readily identifiable linguistic signals.

These signals can be lexical, such as pronouns or connectives, or syntactic, such as the use of comparatives (“Paris is a nice city. Brest is even better”), or parallel sentence structures (“I love my son. He loves me”).

3.4 Discourse Parsing Task

The discourse parsing task will first apply intrasentential linguistic analysis to obtain morphology, syntax and semantics of each individual sentence, and then will build a network of references between sentences in order to build a semantic representation of the entire discourse. The formalism for representing a discourse is based on First-Order Logic and uses boxes to represent the scope of definitions and relations.

Chapter 4

A Proposed Architecture for a French Discourse Meaning Representation Framework

An important area in research fields such as information retrieval, human computer interaction, or computational linguistics, is *natural language understanding*, in which meaning representation is considered as a crucial objective. Meaning representation is generated as a bridge between roles and objects present in texts and common-sense non-linguistic knowledge present in the context. This chapter introduces the state-of-the-art on speculations for extracting meaning out of discourses and building meaning representations, both in English and in French. We will present various approaches based on logical reasoning or on machine learning, by using available corpora. Finally, we present our proposed architecture to obtain a semantic representation framework for French discourse based on an approach by John Bos (Curran, Clark, and Bos, 2007; Bos, 2008; Bos, 2015).

4.1 State-of-the-Art in Discourse Meaning Representation

Discourse meaning representation plays an a crucial role for building natural language automatic analysis systems using machine-interpretable meaning representations. However, this task still faces difficult obstacles because of the diversity of discourse forms and the complexity of human language expressions. A discourse can be perceived in various ways, depending on factors such as context, ambiguity of language, emotions or sentiments. Many approaches have been implemented in order to propose a discourse representation framework. In the following section, we will present typical works tackling directly this task.

4.1.1 Rule-Based Approaches

In the early temps of semantic representation research, interpretative systems from natural language to some query language for database management systems have been built on the basis of domain-specific rule-based theories. Building a natural language interface can improve communication between humans and computers. One of the pioneers in the field is SAVVY, which builds replies to questions by humans, using pattern match techniques (Johnson, 1984). The system requires a lexicon of the vocabulary of the domain as well as syntactic and semantic information about each word. The main advantage of the pattern matching technique is its simplicity. Nevertheless, this methodology is quite brittle because of pattern limitation and shallow semantic representation.

Other methodologies are based on syntax-rule-based techniques. For instance, the LUNAR system has proposed a prototype for building a natural English understanding system which allows people to ask questions and request computations out their natural utterances in

the domain of geology (Woods, 1973). In general, the LUNAR system processes input phrases in three stages. First of all, the syntactic analysis of the phrase is realized by using heuristic information. The semantic analysis is then performed in order to obtain a formal semantic interpretation of the query to the system. In the final stage, the formal expression is obtained by the retrieval component in order to generate the answer for the request. By providing answers for most of the expected questions intended to query data from the database, this system has demonstrated a serious advance in natural language understanding.

Using natural language for querying data rather than formal query languages is easier for newbies who are not required to learn a new artificial language. Based on both syntax-rule-based and semantic-grammar-rule-based techniques, the NLIDB system first processes syntactically the utterance input through a parser which generates a parse tree by consulting a set of syntax rules. The semantic analysis then transforms the parse tree into an intermediate logic query (Androutsopoulos, Ritchie, and Thanisch, 1995). With this approach, NLIDB can be used for wide knowledge domains by updating semantic grammar definitions. Indeed, new semantic grammars need to be rewritten or declared in order to adapt the system to new knowledge domains. Stricter grammar rules that allow linking of semantic categories instead of syntax rules have been used in semantic-grammar-rule-based systems such as Thompson et al., 1969; Hendrix et al., 1978; Templeton and Burger, 1983.

4.1.2 Empirical Approaches

In the rule-base approach, semantic meaning representation has been applied mainly to simple natural utterances in a specific domain. Natural language utterances are very diversified and can be expressed under numerous forms. In fact, semantic analysis is a complex problem and has diverse applications such as machine translation, question answering systems, automated reasoning, knowledge representation, and code generation. “Empirical approaches” is the general term for approaches using rule-based techniques, data-driven statistical analysis, or a combination of these methods.

Ge and Mooney, 2005; Raymond and Mooney, 2006 have constructed a statistical parser that aims to produce a *semantically augmented parse tree* (SAPT). Each internal node of the SAPT includes both a syntactic and a semantic annotation that are captured by semantic interpretation of individual words and basic predicate-argument structure of the sentence input. A recursive procedure is then used to form a meaning representation for each node, in order to annotate the node and obtain meaning representation of the node’s children. In the final steps, SAPT is translated into a formal meaning representation.

Word Alignment-based Semantic Parsing (WASP[1]) (Wong and Mooney, 2006) has inherited the motivation found in statistical machine translation techniques to build a semantic parser. With no prior knowledge of the natural language syntax, WASP proposes an algorithm that trains a semantic parser out of a set of natural language sentences annotated with their standard meaning representations. More specifically, the authors of the system have analyzed the syntactic structure of sentences using a semantic grammar (Allen, 1995). Sentence meaning has been subsequently obtained by compositionality out of subparts extracted from the semantic parse. This work can be considered as a syntax-based translation model (Chiang, 2005). Thereby, translation is an important part of semantic parser. It includes a set of natural language sentence-meaning representation pairs. Another approach, called Synchronous Context-Free Grammar (SCFG) has been used for generating the pairs in the translation. SCFG defines pair rules: $X \rightarrow <\alpha, \beta>$ where $X \rightarrow \alpha$ denotes a production of the natural language semantic grammar and $X \rightarrow \beta$ is a production of the meaning representation grammar. For each input utterance e the task of semantic parsing provides derivations $<e, f>$, where f is a translation

[1] http://www.cs.utexas.edu/~ml/wasp/

of e. Therefore, the semantic parsing model consists of an SCFG G and a probabilistic model with parameterized λ that has a possible derivation d, and returns its likelihood of being correct given an input e. The translation f is defined as:

$$f = m(\arg \max_{d \in D(G|e)} \Pr_\lambda(d \mid e)) \tag{4.1}$$

where $m(d)$ is the meaning representation expression, $D(G \mid e)$ denotes the set of all possible derivations of G that yield e. In general, employing statistical machine translation techniques can be viewed as a syntax-based translation approach (Wong and Mooney, 2006). We can achieve good performance and results comparable to the state-of-the-art on GeoQuery with (Andreas, Vlachos, and Clark, 2013).

The typed versions of grammar formalisms such as TAG, CCG, LFG or HPSG can be used to build an immediate framework for deriving syntactic structure into a semantic representation. Dependency structure analysis provides full annotation of words and their relations with each other in the sentence. By using dependency trees, we can obtain a logical form through a sequence of three steps: binarization, substitution, and composition (Reddy et al., 2016). First, a binarization is realized by mapping a dependency tree into an s-expression. For example, the sentence "Microsoft defeated Amazon" has the s-expression

$$(\text{nsubj}(\text{dobj defeated Amazon})\text{Microsoft}).$$

As a second step comes the process of substitution by assigning a word or label in the s-expression to a λ-expression. For example,

$$\begin{aligned}
\text{defeated} &\mapsto \lambda x.\text{defeated}(x_e),\\
\text{Microsoft} &\mapsto \lambda y.\text{Microsoft}(y_a),\\
\text{Amazon} &\mapsto \lambda z.\text{Amazon}(z_a),\\
\text{nsubj} &\mapsto \lambda f g z.\exists x.f(z) \wedge g(x) \wedge \text{arg}_1(z_e, x_a),\\
\text{dobj} &\mapsto \lambda f g z.\exists x.f(z) \wedge g(x) \wedge \text{arg}_2(z_e, x_a).
\end{aligned}$$

The composition step starts with a β-reduction used to compose the λ-expression terms in order to obtain the final semantics of the input utterance. Here is an example after composition: "$\lambda z.\exists x.\text{defeated}(z_e) \wedge \text{Microsoft}(x_a) \wedge \text{arg}_2(z_e, x_a)$". In addition, the authors provide some remarks and post-processing operations related to the handling of prepositions, coordination and control. For the learning task, they consider semantic parsing as a graph matching operation and use a linear model.

The D-LTAG System introduced an approach for discourse parsing based on the use of the lexicalized Tree-Adjoining Grammar (Forbes et al., 2003). Considering the compositional aspects of semantics at discourse level is similar to the sentence level by factoring away inferential semantics and coreference features of the discourse markers. In their system, sentences are disconnected and parsed independently from the discourse input. Then, the discourse constituent units and discourse connectives are extracted from the LTAG output derivations of the sentences. Finally, fully lexicalized trees of the discourse input are parsed anew using the same process. One of the main contributions of this work is a corpus of discourse connectives that aims to determine the semantic meanings and the elementary tree of types that are lexicalized in the scope of discourse grammar.

The approach based on the empirical modeling of language, along with the development of grammatical reasoning theory have led to improve robustness and efficiency of NLP applications (Kaplan et al., 2004). The Boxer application has demonstrated the efficiency of using the empirical approach for achieving a semantic meaning representation from natural language

utterances in English (Curran, Clark, and Bos, 2007; Bos, 2008; Bos, 2015). Designed for a wide-coverage domain of natural language, Boxer generates semantic meaning representations from discourse input under the CCG lexicalized grammar formalism where words in the sentence are assigned to lexical categories. In the output, one obtains meaning representation either in DRS formalism or represented as First-Order Logic expressions.

Based on the result of Boxer framework, the FRED system allows the generation of RDF/OWL ontologies—a popular formal knowledge representation form on the web—and provides linked data out of natural language utterances (Gangemi et al., 2017; Draicchio et al., 2013). Indeed, natural language input is analyzed and transformed into DRSs by using Boxer. In parallel to this process, information on semantic role labeling and named entities is added to the DRSs output. The final stage focused on transforming DRSs into a RDF/OWL representation by the use of patterns.

Concerning French language, the *Grail* tool allows parsing of French discourses in order to obtain meaning representations in DRS (Moot, 1998; Moot, 1999; Moot, 2010; Lefeuvre, Moot, and Retoré, 2012). Unlike Boxer, the input of *Grail* requires syntactically analyzed data based on the TLG (Type-Logical categorial Grammars) formalism (Moortgat, 1997; Morrill, 2012). TLG has been mostly applied in theoretical issues and relations to logic and theorem proving, while CCGs have been rather concerned with keeping expressive power and automata-theoretic complexity to a minimum. Therefore, CCGs are more relevant to the issues of linguistic explanation and practical computational linguistics (Steedman and Baldridge, 2011).

4.1.3 Semantic Annotation Corpora

FrameNet (Baker, Fillmore, and Lowe, 1998) is one of the first corpora that were created for the purpose of semantic annotations. FrameNet has been developed on the base of a meaning theory called Frame Semantics (Fillmore et al., 1976). A *frame semantic* is a schematic representation of a situation using different participants, propositions, and conceptual roles. In other words, the meaning of a syntax can be interpreted as a semantic frame description containing information related on a type of event, relation, entity, and participants.

The Propbank corpus (Palmer, Gildea, and Kingsbury, 2005) adds a layer to the syntactic structures of the Penn Treebank (Marcus et al., 1994). This layer contains information on predicate-argument, or semantic role labels. This is a shallow semantic annotation resource because it does not annotate co-reference, quantification, or some higher-order language phenomena. Propbank prioritizes annotation of verbs rather than the one of other word types.

The PDTB corpus(Prasad et al., 2005; Prasad et al., 2008) is a large-scale semantic annotation corpus with the rationale that discourse relations can be identified by a set of explicit words or syntagms (discourse connectives). PDTB aims to annotate the million words of the Penn TreeBank by adding a layer of information related to discourse structure and discourse semantics. More specifically, there is a large number of annotation types in this layer, such as explicit or implicit discourse connectives and their arguments, semantic sense of each discourse connective, semantic classification of each argument, and attribution of discourse connectives and their arguments.

The OntoNotes corpus (Hovy et al., 2006) has been created on the base of a methodology that can produce a corpus with 90% inter-annotator agreement. This corpus focuses on a wide-coverage domain of meaning representation which encompasses word sense, predicate-argument structure, ontology linking, and co-reference. This corpus covers three languages—English, Arabic and Chinese—and numerous text genres such as articles, blogs, newswires, broadcast news, etc., in order to obtain a independent domain resource (Table 4.1).

The GMB corpus (Basile et al., 2012) is a large semantically-annotated English text corpus with deep semantics represented in the DRS formalism. It is an optimal semantic resource since it annotates various language phenomena, such as rhetorical relations, presuppositions,

TABLE 4.1: Volumetry of the OntoNotes corpus

	English	Chinese	Arabic
Newswire	625,000	250,000	300,000
Broadcast News	200,000	250,000	N/A
Broadcast Conversation	200,000	150,000	N/A
Web Data	300,000	150,000	N/A
Telephone Conversation	120,000	100,000	N/A
Pivot Text	N/A	300	N/A

predicate-argument structure, thematic roles, etc., which can be reused or be enhanced by contributions from other researchers. The authors have first created a gold-standard semantic representation corpus by manual annotation. Then, PMB, a parallel corpus over four languages (English, German, Italian and Dutch) has been developed in recent times with the same objective as GMB (Abzianidze et al., 2017).

The Sembanking corpus (Banarescu et al., 2013) is a semantic annotation corpus of English texts with annotated named entities, co-reference, semantic relations, temporal entities and discourse connectives using the AMR representation format. Based on assigning similar AMR representations to sentences having the same basic meaning, this corpus aims to abstract away from syntactic characteristics. Nevertheless, a limitation of this corpus is the fact that it does not annotate inflectional morphology for tense and number, or universal quantifiers. This corpus has been created by manual annotation as well.

The UCCA corpus (Abend and Rappoport, 2013) has been created by analyzing and annotating English texts using purely semantic categories and structures. The semantic annotations are based on argument-structure and linkage language phenomena. A Basic Linguistic Theory (Dixon, 2010b; Dixon, 2010a; Dixon, 2012) is used for grammatical description based on the computation of semantic similarity as its main criterion for structuring and categorizing constructions. More specifically, this corpus is based on a semantic schema that generalizes specific syntactic structures and is not relative to a specific domain or language. There are currently 160,000 tokens from English Wikipedia, as well as 30,000 parallel English-French annotations.

The UDS corpus (White et al., 2016) is built with the purpose of strengthening universal dependencies for current data sets with the addition of robust, scalable semantic annotations. This corpus aims to provide two important contributions. The first one is standardization of syntactic dependency annotations, so that they can be reused across different languages. The other contribution is to provide semantic annotations that include numerous types of semantic information for different languages. In its latest version (White et al., 2019), the UDS corpus provides a semantic graph specification with graph structures defined by using predicative patterns. As a result, we can query UDS graphs using SPARQL.

The emergence and rapid development of corpora that annotate syntactic and semantic information brought a novel approach for building language analysis tools. For example, syntactic analysis with POS tagging can be achieved with high accuracy by using probability estimates from the Penn Treebank training corpus: 97.96% in (Bohnet et al., 2018), 97.85% in (Akbik, Blythe, and Vollgraf, 2018), and 97.78% in (Ling et al., 2015). For the constituency parsing task, extracting a constituency-based parse tree that represents the syntactic structure of a sentence according to a phrase structure grammar, can be achieved with 96.34% accuracy via a supervised learning approach using the Penn Treebank corpus (Mrini et al., 2019). Recent

work in the NLP community shows that empirical or corpus-based methods are currently the most promising approach to improve accuracy and efficiency in many tasks (Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019).

4.1.4 Corpus-driven approaches

Supervised Learning Approaches

A consistent current tendency is the increasing use of supervised learning techniques that consider learning models on feeding examples of available input output pairs. Along with the rise of numerous semantic annotation corpora, the supervised learning approach is considered as a modern approach to developing robust, efficient NLP applications (Church and Mercer, 1993). Several works attempt to tackle semantic parsing using semantic annotation corpus with sentence-meaning representation form pairs (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2012; Kwiatkowski et al., 2010).

In machine learning, SVMs are considered as an important maximum-margin separator learning method to prevent over-fitting for high dimensional data such as texts (Joachims, 1998). Along with this supervised learning approach, Kernel-based Robust Interpretation for Semantic Parsing (KRISP[2]) (Kate and Mooney, 2006) provides an approach for mapping natural language utterances to formal meaning representations, this method uses SVMs with string kernel-based classifiers (Lodhi et al., 2002). KRISP defines a semantic derivation D of an natural language utterence s as parse tree of a meaning representation. Each node of the parse tree contains a substring of the sentence and a production, denoted as a tuple $(\pi[i \dots j],)$, where π is the productions and $[i \dots j]$ the substring $s[i \dots j]$ of s. A semantic derivation of an natural language is assigned to a meaning representation. If a semantic derivation is corresponding with the correct meaning representation of the natural sentence, it is called a *correct semantic derivation* otherwise it is an incorrect semantic derivation. The probability model using string-kernel based SVM classifiers $P_\pi(u)$ is defined as production π of the MRL grammar G that covers the natural lanuage substring u:

$$P(D) = \prod_{\pi,[i \dots j] \in D} P_\pi(s[i \dots j]). \tag{4.2}$$

In a nutshell, once the training corpus of natural language utterances are paired with their meaning representation $(s_i, m_i)|i = 1 \dots N$, KRISP first parses the meaning representations using meaning representation language grammar G. KRISP then learns a semantic parser iteratively for every production π of G, and for each iteration, position and negative examples sets are collected. In the first iteration, the set of positive examples for production π contains all sentences the meaning representation parse tree of which use the production π. The negative example set includes other training utterances. By this, an SVM classifier is trained for each production π with a normalized string kernel. Besides, we have others approaches using SVM for classifier in order to learn semantic parsing (Nguyen, Shimazu, and Phan, 2006; Merlo and Musillo, 2008; Lim, Lee, and Ra, 2013).

Conversation text is a rich source for analyzing the meaning of an utterance in a interactive communication context. Many dialog systems has been developed with parsing semantic from user utterances (Allen et al., 2007; Litman et al., 2009; Young et al., 2010). For example, in a train booking system, the sentence "I want to travel to Paris on Saturday" can be transformed to lambda-calculus expression "$\lambda.to(x, Paris) \wedge date(x, Saturday)$". This expression is a representation form of the semantic meaning extracted from the sentence. In this scope, Artzi and Zettlemoyer, 2011 has presented an approach for learning the meaning representation of

[2] http://www.cs.utexas.edu/~ml/krisp/

a user's utterance. The authors define a conversation $C = (U, O)$ as a sequence of utterances $U = [u_0, \ldots, u_m]$ and a set of conversational objects O. Training data is a set of n examples: $(j_i, C_i) : i = 1, \ldots, n$. For each example, the goal is to learn to parse the user utterance at position j_i in C_i. The label meaning representation data paired with x are defined as latent variables (Singh-Miller and Collins, 2007). Learning for semantic parsing is defined by PCCGs, which contain a lexicon and a weighted linear model for parse selection.

Unsupervised Learning Approaches

Unlike supervised machine learning, unsupervised learning infers unknown patterns from a data without reference to labeled outcomes. It is regularly used to discover potential underlying structures implied in the data. The authors of Poon and Domingos, 2009 were pioneers in attempting to build Unsupervised Semantic Parsing (USP) based on Markov logic. Their parsing model consists of three key ideas. First, the target predicate and object constants can be considered as clusters of syntactic variations of the same meaning and can be directly learned from data. Secondly, the identification and clustering of candidate semantic representation forms are integrated with learning for meaning composition. Thirdly, this approach starts from syntactic analyses and focuses on their translation into semantic representations. The input data of the training process consists of dependency structures of utterances that contain more semantic information than constituent structure. The output is a probability distribution over logical form clusters and their compositions. It is worth mentioning that a knowledge base extracted from the GENIA biomedical data set (Kim et al., 2003) has been used for the experiment and evaluation based on the performance in answering a set of questions.

A problem with the generation of self-induced clusters for the target logical forms is absence of information matching with an existent knowledge base, ontology, or database. The USP approach we describe above is consequently unable to directly answer complex questions against an existing database without an knowledge base matching step. Grounded Unsupervised Semantic Parsing (GUSP) (Poon, 2013) alternately employs the database as a form for indirect supervision. GUSP proposed an approach that combines unsupervised semantic parsing with grounded learning from a database, which does not require ambiguous annotations or oracle answers. With a set of natural language questions and a database, GUSP learns a probabilistic semantic distribution by using the Expectation-Maximization algorithm (Dempster, Laird, and Rubin, 1977). Dealing with the lack of direct supervision, GUSP constrains the search space through the database schema, and through bootstrap learning. The evaluation is based on the ATIS travel planning domain by experimenting directly the task of translating questions into database queries and by measuring question-answering accuracy.

Sequence-to-Sequence learning approaches

Along with the emergence of annotated corpora for semantic meaning representation, machine learning—especially through neural network models and their proven efficiency in a variety of NLP tasks—has motivated the investigation of the method of handling semantic meaning representation as a sequence transduction problem, where discourse is mapped into a meaning representation format. In what follows, we will examine some works based on this research orientation.

Transforming directly natural language into a logical formalism, a machine intepretable meaning representation, can be implemented through neural network models. Indeed, Dong and Lapata, 2016 introduces a method based on an attention-enhanced encoder-decoder model. This method transforms input sentences into vector representations, and then generates their logical forms by adjusting the sequence output on the encoding vectors. More specifically, it builds a learning model where natural language input $q = x_1 x_2 \ldots x_n$ is mapped to a logical

form representation $a = y_1 y_2 \dots y_m$. A conditional probability if calculated for each element in a versus q.

$$p(a \mid q) = \prod_{t=1}^{n} p(y_t \mid y_{<t}, q) \tag{4.3}$$

where $y_{<t} = y_1 y_2 \dots y_{t-1}$. Here, encoding is the task of converting natural language input q into the vector representation a, while decoding is the task of learning to generate $y_1 y_2 \dots y_m$ matched with the encoding vector.

Xiao, Dymetman, and Gardent, 2016 introduce an approach mapping natural language (restricted to questions) into logical form representations by using a RNN model associated with LSTM units (Hochreiter and Schmidhuber, 1997). Based on the use of a simple grammar to map logical forms paired with canonical utterances (Wang, Berant, and Liang, 2015), the authors have realized three different sequentialization approaches for a logical form: a normal linearization of the logical form, a canonical form, and a derivation sequence related to the underlying grammar (Table 4.2). Therefore, they have a large amount of choices for building a vector representation input using their encoder-decoder neural network model.

TABLE 4.2: An example of various cases of natural language input and logical form output

Natural language	article published in 1950
Logical Form	get[[lambda,s,[filter,s,pubDate,=,1950]],article]
Canonical Form	article whose publication date is 1950
Derivation Sequence	s0 np0 np1 typenp0 cp0 relnp0 entitynp0

The amount of annotated data needed for training is a recurrent problem in NLP task using the supervised approach, since acquiring data is expensive and sometimes even infeasible. To overcome this drawback, transfer learning can be used: a model trained on one task is repurposed for another, related task (Torrey and Shavlik, 2010). Fan et al., 2017 has proposed an approach to use multiple representations in a multi-task framework by modifying the parameters of the learning process. The repesentations have common structures that are implicit across different formalisms (SPARQL for WikiData (Vrandečić and Krötzsch, 2014), MQl for Freebase (Flanagan, 2008)) and tasks. They use encoder-decoder architectures for transfer learning in semantic parsing under the hypothesis that the sequence-to-sequence paradigm learns a canonicalized representation across all tasks. Based on a single task encoder-decoder baseline, the authors update the process to achieve an improvement in accuracy.

One of the latest work involving the generation of discourse meaning representation (Liu, Cohen, and Lapata, 2018) uses the GMB corpus. The authors propose an approach that begins with the conversion of DRSs data to tree forms and then builds a structure-aware model in which they divide the decoder task into three small decoding processes: basic DRS structure prediction, condition prediction, and referent prediction. Based on an encoder-decoder neural network model, natural language input X is encoded into vector representations and a Bidirectional RNN with LSTM units is used to obtain hidden state representation of the encoder layer. The decoder is initialized through the hidden state of the encoder layer, and then data are passed to a forward LSTM:

$$h_{d_j} = \text{LSTM}(e_{y_{j-1}}) \tag{4.4}$$

where h_{d_j} is the hidden state representation of the j-th token in the decoder, and e_{y_j} is a vector word embedding of output y_j. Besides, the decoder layer adds context information from the encoder layer by creating a embedding of the $(i-1)$-th predicted token to the output of the i-th

token. As a result, with experimental outcomes on the GMB corpus, this approach can recover DRS with an accuracy of 77.54%. On the other side, using the same GMB corpus, Noord et al., 2018; Noord, 2019 have demonstrated that using sequence-to-sequence neural network models allows obtaining well-formed DRSs with high accuracy.

4.2 Meaning Representation Parsing Through Deep Neural Network Models

Currently deep neural network models such as Convolutional Neural Networks(CNN), Recurrent Neural Network (RNN), Recursive Neural Networks, Attention Mechanisms, Parallelized Attention (Transformer), etc., are frequently applied to NLP tasks (Young et al., 2018). When compared to traditional machine learning algorithms, these models achieve impressive results. On the whole, it is not exaggerated to say that using deep neural network models is one the the leading research trends in the next decade for improving the outcome of NLP tasks such as machine translation, language modeling, question answering, natural language inference (Edunov et al., 2018; Melis, Kočiskỳ, and Blunsom, 2019; Šuster and Daelemans, 2018; Liu et al., 2019). Meaning Representation parsing may also fall inside the scope of this trend, more precisely the task of mapping natural language into an interpreted form that can be understood by the machine. As mentioned previously, we have recently observed the emergence of the use of the sequence-to-sequence model that is built upon RNNs or Attention Mechanisms. This proves that the use of deep neural network may only be in the early stages of its development and still has many prospects in the future.

Traditional machine learning algorithms can achieve a good outcome when data are of sufficient size. However, when data size is increased, the outcome is very difficult to improve. In other words, we observe a saturation on the result even in hte presence of a large data size. Therefore, the dramatic development of deep neural network models is based, among other factors, on their ability to outperform other machine learning methods in the presence of massive data amounts. Besides, deep neural network models carry other benefits, such as:

- The same neural network model can be applied to numerous applications and different data types. For example, the CNN model is frequently used in image and video recognition applications (Lawrence et al., 1997). It can also be used for recommendation system or NLP applications (Ying et al., 2018; Kim, 2014). The data input for deep learning methods can be in text, image, video, sound, or time series format.

- An important step in machine learning algorithms is feature engineering, where features are extracted from raw data in order to achieve a better data representation for the given problem. This work can improve the model's accuracy. Deep learning techniques can directly start with raw data. Features will be automatically searched and created by the neural network in learning process. Therefore, we can save time spent on searching feature representation of data.

- Neural network algorithms can uncover new, more complex features than those that humans can imagine. In other words, deep learning techniques are able to achieve an optimal representation of data by automatic learning.

- Hardware computation power advances are one of the foundation for deep learning innovations (Moshovos et al., 2018). Changing computation from using CPUs to parallel computation by GPUs improves performance and reduces time-consumption for the training phase. Besides, neural network model can be scaled for massive volumes of data.

- Deep neural network architecture is usually flexible enough to be reused for problems that can be encountered in other fields or in new problems in the future.

Although the importance of deep neural network models is increasing and several advances in its research are reaching great heights, deep learning still has a few drawbacks and challenges that we need to tackle in order to obtain the desired outcomes. One of the limitations of deep learning is the requirement of the amount of data using for training models. In general, deep neural network models has downsides as follows:

- Massive amount of available data collected over the recent years has contributed to the increase of the use of deep learning methods. Nevertheless, the deep neural networks model requires a large amount of data in order to obtain better performance versus traditional machine learning. Although there are approaches where neural networks manage to operate with small amounts of data, in most cases they require overwhelming amounts of data.

- Deep neural network architectures have a more complex structure than traditional machine learning algorithms. Therefore, they are expensive with respect to the time required to train models for computing complex data models. State-of-the-art neural network models can take several days to complete the training phase of a model. For example, BERT-large model (Devlin et al., 2018), which consists of 24 different layers and 1024 elements for each hidden state size, has in total 340 millions parameters. This model has taken 4 days to train completely from scratch.

- Deep learning approaches require a large number of computationally heavy operations to handle high-dimensional matrices that can be executed parallely on GPUs or TPUs. Therefore, it requires significant hardware GPU or TPU power. For example, by adjusting the batch size and adding more TPU power, time consumption for training of the BERT model was reduced from 4 days to 76 minutes (You et al., 2019). As a result, using more GPU or TPU power leads to an increase in the cost of implementing projects for users.

- The development of deep learning is based on applied research with important contributions coming form both the academic and the industrial side. The fundamental theory of deep learning is currently inherited from machine learning with concepts such as bias, variance, and overfitting that are also used in disciplines such as statistical learning and generalization bounds. In deep learning, concepts such as SGD, Batch Normalization, which are used for estimating the gradient descent with mini batches, have become more popular and are used as powerful regularizers. Indeed, their mathematical structure has not yet been clearly defined. We currently do not still have a standard theory for deep learning. As a result, it is difficult to select a right deep learning model or cost function for people with deep learning practical skills because this work requires knowledge of topology, training methods and other notions.

- Most of deep learning models behave as black boxes and are unable to provide explanations on their outcomes. In high-risk domains (e.g., health care, finance and banking), this issue influences the decision to use deep learning models because the trust in a model and the ability to to understand its behavior play the most crucial role.

When it comes to the representation parsing task, mapping natural language data sets to a meaning representation form plays an extremely important role in successfully using deep learning networks. It also becomes a crucial criterion for selecting the approach to be used. The generation of an open domain data set for this task requires a lot of time and effort because of the richness and diversity in structure of natural language. There exist some data sets for the

English language, in FOL, AMR, or DRS formalisms as listed in the previous section. For other languages such data sets are often lacking. In the case of French language, it is difficult to find a data set to be used for a semantic parsing task. This problem can be solved in some cases such as language translation or cross-language, using transfer learning. Nevertheless, the achieved results are rather limited (Mikolov, Le, and Sutskever, 2013; Singla, Bertino, and Verma, 2019).

Deep neural network models have limitations when it comes to *compositionality*, an important principle in linguistics in general and for the semantic parsing task in particular. Compositionality expresses the fact that the meaning of an expression can be determined from the meaning of its constituent expressions and the combination rules used to build them. For example, if a person knows meaning of a verb "to run" and an adverb "slowly", e can immediately understand the meaning of "to run slowly" even though e has never seen or heard this expression before (because of its semantic contradiction). This principle explains a part of reason how human can quickly create a large number of expressions from a limited vocabulary set (Loula, Baroni, and Lake, 2018). In general, deep neural network are not able to discover and store language skills like humans in order to reuse or recombine them in a hierarchical structure in order to face new problems or challenges. Many approaches have been proposed to tackle this problem such as using composition of primitives, which is based on the classic idea that new representations can be built by the combination of primitive elements (Lake et al., 2017), or by using the GOFAI infrastructure, which provides a valid and useful set of primitives (Garnelo and Shanahan, 2019). Besides, there is the *Recursive Model Network* architecture, one of the first models used to learn *distributed representations* of linguistic structures (Goller and Kuchler, 1996). This architecture has been proposed as an approach of compositional learning sequence out of natural language input (Socher et al., 2011; Lewis, 2019). Finally, the inability of deep neural network models to effect a computation on compositional learning is one of the main reasons for deep learning most critical limitations, besides of the requirement of feeding models with massive amount of data.

4.3 An Architecture for a French Semantic Parsing Framework

In the previous section we gave an overview on the state-of-the-art of research in semantic parsing, with approaches based on reasoning in syntax-based, grammar-based or rules-based, statistical models and deep neural model networks. The popularity of the sequence-to-sequence model in the recent period reveals the advantages of deep learning networks versus other traditional machine learning models. However semantic parsing is a challenging and important task in NLP domain that aims to transform natural language utterance into a meaning representation form.

Our objective in this work is the proposal and development of a framework in order to perform semantic parsing for French language. The lack of available data sets and the limitation in computation with compositional learning—i.e., learning concepts and combining them together in different ways—are major obstacles when engaging in deep neural network models. Furthermore, using the traditional approach has proven its success in numerous applications. For example, Boxer is an efficient tool for obtaining a semantic representation of English sentences and it became the most important factor in generating the GMB and PMP meaning representation corpora (Bos, 2008; Abzianidze et al., 2017). Following the empirical approach, we can overcome the constraint on available data and keep compositionality in analyzing linguistics through natural manner in building sentences.

In this section, we will introduce a complete architecture for achieving meaning representation from the French utterance input (Figure 4.1). This architecture includes different processing steps which we can encapsulate in four main tasks:

1. preprocessing with French discourse input,

2. analyzing syntax and dependency of the utterance input,
3. analyzing of grammar formalism with CCG, and
4. semantic meaning representation analysis.

These task will be sequentially performed with the output of one task being the input of the next. We will dive into the presentation each task with the next sections.

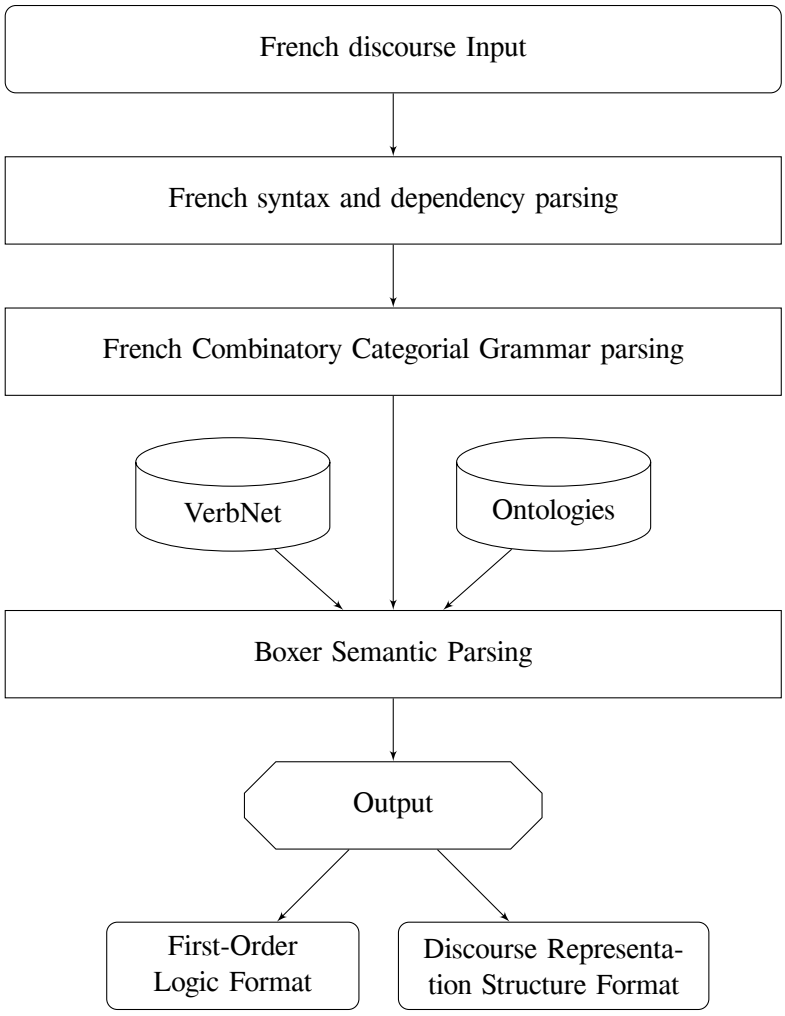


FIGURE 4.1: Proposed Architecture for French Semantic Parsing

4.3.1 Text Input Preprocessing

The diversity of social network services allows users to choose platforms that best suit their needs. Each platform provides a different framework to create and present content-generated input by their users. Depending on available tools of SN platform, we have different expression discourses which are created in conversation form or discussion in a thread about a subject (Figure 4.2). Discourses on SN platforms are a great source for classifying and analyzing information from various audiences about various topics. However, we need to realize some preprocessing steps before using these discourses into an analysis process within our architecture.

Input data preprocessing is a set of operations ranging from data extraction to data reorganization. This step aims to increase the quality of the input in order to extract important

information. There are many steps such as elimination of unnecessary characters, analysis of emoticons, or spell checking. In general, this task is divided into two small tasks: data collection and data preparation. Data collection focuses on extracting texts concerning communications or discussions about a interested topic on SN platforms. Data preparation is the process of cleaning up the raw data.

CLIENT: *Bonjour, J'aimerais avoir des informations concernant vos services. Pourriez-vous m'aider? (Hello, I would like to have information about your services. Could you help me?)*
MAX24: *Bonjour, Très bien je peux vous assister dans vos recherches. Quel service attire votre attention? (Hello, Very well I can assist you in your research. Which service attracts your attention?)*
CLIENT: *J'ai consulté la fiche de compte d'épargne de Livret A sur votre site. Aussi je souhaiterais connaître le taux maintenu de ce service? (I have consulted the Livret A savings account sheet on your site. Also I would like to know the maintained rate of this service?)*
MAX24: ...

FIGURE 4.2: A conversation on a message platform

The analysis of a sentence begins from lexical units or groups of lexical units. At the lowest level, text input is a character sequence and lexical units or tokens need to be identified and extracted from it. This process is called tokenization. In most cases, the extraction of tokens consists simply in splitting by the space character. However, a space-separated token may be created from different words as in contractions of particles in front of verbs or nouns starting with a vowel. Tokenization is best done in parallel with lemmatization, an important operation for French, in which many words may change form depending on some factor such as the gender or number for nouns and adjectives, tense or mode for verbs, etc. (for example, see Figure 4.3).

```
<sentence>
<word><token> je </token> <lemma> je </lemma></word>
<word><token> ne </token> <lemma> ne </lemma></word>
<word><token> arriverai </token> <lemma> arriver </lemma></word>
<word><token> pas </token> <lemma> pas </lemma></word>
<word><token> à </token> <lemma> à </lemma></word>
<word><token> la </token> <lemma> le </lemma></word>
<word><token> heure </token> <lemma> heure </lemma></word>
</sentene>
```

FIGURE 4.3: Tokenization and lemmatization of the sentence "je n'arriverai pas à l'heure" (I won't arrive on time)

4.3.2 Syntax and Dependency Analysis

Syntax is an essential component of every language, it is a set of specific rules, constraints, conventions, and principles defined in order to govern the way words are combined into phrases, phrases are linked into clauses, and clauses form sentences. Like in the case of English language, words in French language are combine in order to form constituent units. In general, these constituent components include words, phrases, clauses and sentences. A sentence is regularly built up following a hierarchical structure of the components: $sentence \leftarrow clauses \leftarrow phrases$

$\leftarrow words$. The sense of an utterance can be investigated on the basis of these constituent components.

Analysis of the language's syntax brings helpful information into many NLP tasks such as text processing, text classification, spell checking, annotation or other parsing tasks. While syntax analysis is the process of analyzing the input utterances with words conforming to the grammar rules of the language, we call *structural analysis* the task of extracting dependency relations in a sentence, represent its grammatical structure and the relationships between the words or constituent components. We currently have several typical parsing techniques for syntax and structure analysis as follows:

- POS Tagging
- Syntax Chunking
- Constituency Parsing
- Dependency Parsing

Parts of speech tags

La	réforme	des	retraites	sera	menée	à	son	terme	.
DET	NC	P	NC	V	VPP	P	DET	N	PUNC

TABLE 4.3: POS tagging of the example sentence "Pension reform will be completed." by using Stanford Tagger

Definition 4.3.1. Parts of speech tagging is the process of classifying words in a sentence to a corresponding part-of-speech label, depending on its context and role.

Parts-of-speech tags are among the lower analysis levels, together with morphological and lexical analysis. Their analysis provides information related to each word in the sentence (Table 4.3). In French language, words can belong to the following categories:

- **N**oun: the words in this category has the largest quantity in most languages, including French. This word type refers to a thing, a person, an animal, a place, an event, a substance, a quality, an ideal or an action. We have two small groups of this category: common nouns and proper nouns. In French, a noun may take several forms depending on gender (masculine or feminine) or number (singular or plural). For example, the word "chat" (cat) in French can take four forms depending on the context: un chat (M/S), une chatte (F/S), des chats (M/P or M+F/P), des chattes (F/P). In our POS tag list, N is used for labeling nouns.
- **Det**erminer: a class of words used in front of a noun or of a noun phrase. Determiners can be definite articles (le, la, les: the), indefinite articles (e.g. une, un, des: a, an), demonstratives (e.g., ce, cette, ces: this, that, these, those), possessive pronouns (e.g., ma, mon, mes: my; ta, ton, tes, votre, vos: your; sa, son, ses - its, her, his ; notre, nos - our ; leur, leurs - their). Besides, there are other determiner types such as quantity, relative or interrogative determiners. The DET label is used for this type of POS tag.
- **V**erb: a group of words used to describe an action, state, experience or occurrence. Verbs play an important role in order to constitute a predicate in a clause. Besides that, verbs

are governed by grammatical categories such as tense (i.e., past, present, or future), mood (i.e., indicative, imperative, subjunctive or conditional), voice (i.e., active, passive or reflexive), and aspect (i.e., perfective or imperfective). In general, verbs are mostly the main sentence part along with nouns. Verbs can be divided into auxiliary verbs which are used to indicate the tense or mood of other verbs (e.g., être, avoir: to be, to have), and main verbs (the major part). Modal verb can be auxiliary verbs or main verbs depending on their role in the sentence. V and AUX labels are used to present for main verbs and auxiliary verbs in POS tag list.

- **Adj**ective: a class of words used to modify or describe nouns or pronouns. The role of adjectives in a sentence is to provide more detail information about the word to which it refers. In English an adjective is usually placed in front of the noun it modifies, while in French the majority of adjectives occurs after the nouns it's describing in French, the exception being some adjectives with basic meanings (e.g., bon, beau, bref, grand, joli, petit: fine, good, brief, large, pretty, small etc.), functional adjectives (e.g., autre, même, nombreux, divers: other, same, numerous, various, etc.), emphatic adjectives (e.g., vaste, horrible, excellent: huge, terrible, exellent, etc.), adjectives with different meaning depending on their position (e.g. ,"cher" adjective is used before the noun with meaning "dear" and after the noun with meaning "expensive"). The ADJ label is assigned to adjectives in the POS tag set.

- **Adv**erb: a class of words or phrases that changes, modifies or qualifies several types of words including verbs, adjectives, clauses, other adverbs. Adverbs can be categorized into some typical categories depending on activities denoted by verbs in sentences: adverbs of manner, place, time, quantity, affirmation, and negation. First of all, adverbs of manner denote how an action can be carried out (e.g., rapidement, malheureusement, facilement: rapidly, sadly, easily). Secondly, adverbs of place focus on explaining where an action occurs in order to provide information of direction, distance or position (e.g., ici, dehors, autour: here, outside, around). Thirdly, adverbs of time express when or how often something occurs (e.g., toujours, aujourd'hui, hier: always, today, yesterday). Fourthly, adverbs of quantity are used to indicate the quantity of an action (e.g., moins, très, environ: less, very, about). Finally, adverbs of affirmation and negation are used to declare that some statement is true or false (e.g., certes, oui, vraiment, nullement, pas: certainly, yes, really, not, not). The ADV label is used for this class of words in the POS tag set.

- **P**reposition: a small set of words placed in front of nouns, pronouns, or phrases to express a relationship between words within a sentence. Like adverbs, prepositions do not change form. In French, a preposition can be combined with an article to create a single word (see Table 4.4). The PRP label is used for prepositions in the POS tag set.

TABLE 4.4: Contraction of prepositions and articles

Preposition	Article	Combined Word	Meaning
à	le	au	at, in ,to
	les	aux	
de	le	du	from, of, by
	les	des	

- In addition to the above categories of parts-of-speech that account for the majority of words in most languages, other categories can be listed here such as pronouns (**PRO**), interjections (**INTJ**), conjunctions (**CONJ**), punctuation (**PUNCT**), foreign words (**FW**) and others. Furthermore, a POS tag can be combined with features into subtags. For example, **N**oun POS tag together with the number feature or the proper noun feature gives a Singular Noun (**NN**), a Plural Noun (**NNS**) and a Proper Noun (**NNP**) tag.

The definition of the POS tag set can vary among languages and also among corpora in the same language. For example, the POS tag of the Penn Treebank corpus includes 36 labels plus 12 others for punctuation and special symbols (Taylor, Marcus, and Santorini, 2003), while the TreeTagger tool uses 58 tags[3], the amount of POS tags in the Lancaster-Oslo/Bergen Corpus of British English uses 134 distinct labels (Johansson, Leech, and Goodluck, 1978; Garside, Leech, and Atwell, 1984), while the French Tree Bank corpus uses 29 POS tag labels (see Table 4.5). The difference between POS tag sets causes difficulties for computational tasks that need to combine various corpora. A universal POS tag set with 12 categories, common to all languages, has been proposed to alleviate the problem (Petrov, Das, and McDonald, 2011).

Table 4.5: The POS tag list used in the French Tree Bank Corpus

#	**Category**	**Description**	**Example**
1	DET	Déterminant (Determiner)	le, la, les (the)
2	NC	Nom commun (Common noun)	président (president)
3	NPP	Nom propre (Proper noun)	Parisien (Parisian)
4	V	Verbe conjugué (Indicative or conditional verb form)	gonflent (inflate)
5	VINF	Verbe à l'infinitif (Infinitive verb form)	manger (to eat)
6	VIMP	Verbe à l'impératif (Imperative verb form)	imaginons! (imagine!)
7	VS	Verbe subjonctif (Subjunctive verb form)	sache (know), soient (be)
8	VPP	Participe passé (Past participle)	écrit (written)
9	VPR	Participe présent (Present participle)	concernant (concerning), tendant (tending)
10	ADJ	Adjectif (Adjective)	décisif (decisive)
11	ADV	Adverbe (Adverb)	rapidement (rapidly)
12	P	Préposition (Preposition)	dans (in), sur (on)
13	P+D	Préposition et déterminant (Preposition & determiner amalgam)	au (at, to), du (from)
14	P+PRO	Préposition et Pronoun (Preposition & pronoun amalgam)	à laquelle, auquel, auxquels (to which)
15	CLS	Clitique sujet (Subject clitic pronoun)	il (it, he)

[3] https://courses.washington.edu/hypertxt/csar-v02/penntable.html

16	PROREL	Pronom relatif (Relative pronoun)	dont, qui (of which, who)
17	CLR	Clitique réflexif (Reflexive clitic pronoun)	se, s' (themself, himself, herself, oneself)
18	CLO	Clitique objet (Object clitic pronoun)	le, lui, en (it)
19	CS	Conjonction de subordination (Subordination conjunction)	si, que (whether)
20	PRO	Pronom (Full pronoun)	elle (she), je (I)
21	ADVWH	Adverbe interrogatif (Interrogative adverb)	quand (when), pourquoi (why), comment (how)
22	DETWH	Déterminant interrogatif (Interrogative determiner)	quel, quelle (what, which)
23	PROWH	Pronom interrogatif (Interrogative pronoun)	que, quoi (what)
24	ADJWH	Adjectif interrogatif (Interrogative adjective)	quel, quelle, quels, quelles (what, which)
25	PREF	Préfix (Prefix)	micro-, ex-, quasi-
26	I	Interjection (Interjection)	Eh (Hey), Oh, Ah
27	CC	Conjonction de coordination (Coordination conjunction)	et (and), ou (or)
28	ET	Mot étranger (Foreign word)	show, banking
29	PONCT	Ponctuation (Punctuation mark)	',', ':', '.', ';', , '!'

POS tagging is one of the basic tasks in NLP, it assigns POS tag to words in the sentence by using local information from the word's definition and from the global context of relationships with precedent or following words or phrases. Different approaches are used. The currently typical approach is based on the sequence-to-sequence model with a RNN architecture as core (Ling et al., 2015; Yasunaga, Kasai, and Radev, 2017; Bohnet et al., 2018; Akbik, Blythe, and Vollgraf, 2018). Nevertheless, statistical methods such HMM, CMM, MEM have dominated the mainstream approaches before the era of deep neural network models (Ratnaparkhi, 1996; Màrquez and Rodríguez, 1998; Ekbal, Mondal, and Bandyopadhyay, 2007; Amri and Zenkouar, 2018). There have been numerous for works on the POS tagging task for French (; Clément, Lang, and Sagot, 2004; Blache et al., 2016) resulting in various applications (see Table 4.6).

Among the difficult subtasks of POS tagging is disambiguation. Indeed a surface form can have more than one possible POS tags. For example, the word *still* can be an adjective (the *still* air), a noun (the *still* of the night), an adverb (*still* loving you) or a transitive verb (Cold december's winds was *stilled*.). In French, the word *une* can be a determiner (dans *une* ville: in a city) or a noun (à la *une*: on the front page). the word *content* can be a verb (ills *content* une histoire: They tell a story), a noun (Macron a son *content* de la réforme de retraite: Macron has his satisfaction about pension reform), or an adjective (Philippe a l'air *content*: Philippe looks happy). An additional POS tagging difficulty comes from multi-word terms. For instance, syntagms such as *'load up on'*, *'run out of'*, etc., can be considered as one word. Similarly in French, the verb *'mettre à jour'* (update) can be considered one word even though it is constituted from three different words.

TABLE 4.6: A List of POS Tagging Systems for English and French

POS Tag System	Description	Supported Languages	Main Publications
Morfette	Lemmatization and POS tagging are independently handled with logistic regression models. Learning models are dynamically organized to produce a globally plausible sequence of morphological tag-lemmas pairs for a sentence. Their outcomes achieved 97.68% of accuracy on FTP corpus.	Romanian, Spanish, Polish, French	Chrupała, Dinu, and Van Genabith, 2008; Seddah et al., 2010
MElt (Maximum-Entropy Lexicon-enriched Tagger)	A Pos Tagger based on the maximum entropy conditional sequence model in association with external linguistic resources: an annotated corpus and morphosyntactic lexicons. This system obtained a 97.7% accuracy on the FTB corpus.	French, English, Spanish	Denis and Sagot, 2009; Denis and Sagot, 2010; Denis and Sagot, 2012; Sagot, 2016
Stanford POS Tagger	Based on the use of context information of both preceding and following tags, based on a dependency network representation, the authors have built a tagger system with bidirectional dependency learning with a conditional Markov model. Experiments have been realized on Penn TreeBank with 97.32% accuracy.	English, French, Chinese, German, Arabic, Spanish	Toutanova and Manning, 2000; Toutanova et al., 2003
TnT (Trigrams 'n'Tags)	The system was implemented based on a Viterbi algorithm which is guaranteed to find highest probability on sequence states, and a HMM. Furthermore the author proposed a technique to deal with unknown words by suffix trie and successive abstraction. Experiments on both German NEGRA corpus and English Penn Treenbank have been evaluated with 96.7% accuracy on both corpora.	German, English	Brants, 2000
TreeTagger	Implemented in C language, this tagger achieves a high performance and can tag 8,000 tokens per second. Based on the Markov Model, TreeTagger uses a decision tree to get more reliable estimates for contextual parameters. This approach achieved 97.5% accuracy on a German newspaper corpus.	German, English, French, Italian, Danish, and 21 others[4]	Schmid, 1994; Schmid, 1999

[4]Include: Danish, Swedish, Norwegian, Dutch, Spanish, Bulgarian, Russian, Portuguese, Galician, Greek, Chinese, Swahili, Slovak, Slovenian, Latin, Estonian, Polish, Romanian, Czech, Coptic and old French.

SEM (Segmenteur-Étiqueteur Markovien)	Using a linear CRF model to annotate French texts, by exploiting external lexical information, this tool can tackle multiword unit issues in POS tagging. The accuracy for all tokens is 97.7% on the FTB corpus	French	Constant et al., 2011
SVMTool	Based on a SVM learning framework that includes three main components: the learner, the tagger and evaluator, this tagger reached a state-of-the-art performance when it was published. It achieved an accuracy of 97.2% for English on the WSJ corpus.	English, Spanish, Catalan	Giménez and Màrquez, 2004
TATOO (ISSCO TAgger TOOl)	This tagger is based on HMMs and included two phases: a training stage to estimate the parameters of the model, and a tagging stage to select the highest probability of tags according the model developed in the first stage by using the Viterbi algorithm.	Any Language (requires preparation of a corpus for training)	Armstrong, Bouillon, and Robert, 1995; Robert, 1998
LGTagger (Laboratoire d'Informatique Gaspard-Monge)	Based on a CRF model along with language-independent features and features extracted from external linguistic resources consisting of morphosyntactic dictionaries and lexicalized local grammars. Evaluated on the FTB corpus with 97.7% accuracy.	French	Constant and Sigogne, 2011
LIA_TAGG (Laboratoire Informatique d'Avignon)	This tagger was built by using HMM and Viterbi algorithm, integrated with external lexical resources. The ambiguity is controlled by ambiguous tags which denote subsets of the tag set. Experiments on Brown corpus show a recall of 98.2% which is to be compared with a baseline recall 97.8%.	French, English	Nasr, Béchet, and Volanschi, 2004; Béchet, 2013
RDRPOS-Tagger	This tagger employs a transformation-based error-driven methodology to automatically construct tagging rules in the form of a binary tree for POS and morphological tagging tasks. Thereby authors proposed an incremental knowledge acquisition method, in which rules are stored in a special structure and new rules can be added to correct the errors of existing rules. Achieved about 97.17% accuracy on FTB corpus.	English, French, German, Vietnamese, Thai, Hindi, and many others[5]	Nguyen et al., 2014; Nguyen et al., 2016

[5] About 80 languages in the official page: https://github.com/datquocnguyen/RDRPOSTagger/tree/master/Models

Talismane	Pos tagging is one of important tasks of this system along with phrase chunking and dependency analysis. The Lefff corpus is used to exploit lexical information (Sagot et al., 2006). FTB was used for experiments with an average of 97% accuracy on POS labels.	French	Urieli and Tanguy, 2013
Apache OpenNLP	This pos tagger is built by employing a ME model to predict word tags. A tag dictionary is used to limit the number of possible tags for words as well as to increase the performance of the system.	Danish, English, Spanish, Dutch, French[6].	Baldridge, Morton, and Bierner, 2002
Spacy[7]	Based on industrial-strength experiences, the statistical model for POS tagging provides an exceptional performance on both speed and accuracy. However, the current architecture which is based on a multi-task CNN-CRF model has not been officially published in any article. Experiments give 94.62% accuracy on French Sequoia and WikiNER corpus (Candito and Seddah, 2012; Nothman et al., 2013).	English, German, French, Spanish, Italian, Dutch, Portuguese, and many others	Honnibal and Montani, 2017
NLP4J	The system is built on a novel technique called dynamic feature induction that provides a linearly separable feature space by inducing high dimensional features. The POS tagging accuracy is 97.64% on the Penn Treebank Corpus.	English	Choi and Palmer, 2012; Zhai, Tan, and Choi, 2016; Choi, 2016
Flair Framework[8]	Various NLP tasks such as NER, POS tagging may be formulated as sequence-labeling problems. This framework proposes a deep neural network model based on a bidirectional LSTM-CRF architecture with contextual string embeddings. This work currently obtains a state-of-the-art performance for POS tagging with 97.85% accuracy on the Penn TreeBank corpus[9]	English, German	Akbik, Blythe, and Vollgraf, 2018; Akbik, Bergmann, and Vollgraf, 2019; Akbik et al., 2019

[6]French language is not officially supported by OpenNLP, but it is possible to train a French Model on FTB corpus (`https://sites.google.com/site/nicolashernandez/resources/opennlp`)

[7]Official site: `https://spacy.io/` and French POS tagging models: `https://spacy.io/models/fr`

[8]Official site: `https://github.com/flairNLP/flair`

[9]A more recent study on POS tagging of Bohnet et al., 2018 achieves 97.96% accuracy on this corpus, but this study has not yet officially published neither its source code, nor its system.

Stanza[10]	The latest python NLP library has been developing for various NLP tasks from tokenization, lemmatization to the constituency and dependency parser. With the POS tagging task, the authors use the robustness of the neural network architecture based on the BILSTM with fully-connected layer, and applying a biaffine score mechanism to predicting the output labels Qi et al., 2019. They achieve a state-of-the-art performance on the Universal Dependencies v2.5 datasets with more than 100 treebanks about different human languages in the world.	English, French and 64 other languages	Qi et al., 2020

Chunk analysis

Chunking, also referred to by the terms *shallow* or *light parsing* is realized after POS tagging in order to extract and attach more structure information to sentences using POS tagging results (Abney, 1991). For example, the sentence "La réforme de retraites sera menée à son terme" (Pension reform will be completed) can be broken down into three flat chunks corresponding to two noun phrase chunks and one verb phrase chunk (see Figure 4.4)).

Definition 4.3.2. Chunking is the process of analyzing and classifying the flat/non-overlapping structure of a sentence to identify the basic non-recursive constituent parts and to group them into higher-level components that underlie a grammatical meaning form.

A chunk is a group of words built around a head lexical item and considered as a phrase. There is no rule about the size of a chunk, however it must have at least one word. A sentence is basically composed of combinations of different phrases (or different chunks in other words). The function of a phrase is presented by the function of the head word contained in it (Tallerman, 2013). In general, we classify types of phrases into five major categories as follows:

- Noun Phrase (NP): a group of words playing the role of a noun. In a NP, the head word can be a noun or pronoun. In a sentence, an NP can function as a subject, as an object, or as a complement component.
- Verb Phrase (VP): a sequence of lexical units that contain at least one verb playing the role of head word. Verb phrases indicate what happened in the clause, or sentence.
- Adverbial Phrase (ADVP): a group of words with an adverb playing the role of head word. Adverbial phrases are employed to give more information about other components.
- Adjectival Phrase (ADJP): the head of such a phrase is an adjective, used to describe or qualify other components in the sentence, such as nouns or pronouns. They are usually placed after the noun in French, and before the noun in English.
- Prepositional Phrase (PP): a preposition will be the head word of this type of phrase. Other dependent words can be of any word type such as noun, adjective, verb, etc. The function of this category is to provide modifiers for other components.

[10]Official site: `https://stanfordnlp.github.io/stanza/`

The definition of chunk categories varies among languages and depends on their structural grammars. Syntactic annotations for French corpora employ six categories for syntactic chunks: Noun Group (GN), Prepositional group (GP), Adjectival group (GA), Adverbial group (GR), Verb group with a preposition (PV), and Other verb groups (NV) (Gendner et al., 2004). In the French Tree Bank corpus (), chunk categories are separated in 10 groups illustrated in Table 4.7.

TABLE 4.7: The tagset used in the French Tree Bank corpus for the shallow parsing task

#	Tag	Meaning	#	Tag	Meaning
1	AP	Adjectival phrases	6	AdP	Adverbial phrase
2	COORD	Coordinated phrases	7	NP	Noun phrases
3	PP	Prepositional phrases	8	VN	Verbal nucleus
4	VP_{inf}	Infinitive clauses	9	VP_{part}	Nonfinite clauses
5	SENT	Sentences	10	S_{int}, S_{rel}, S_{sub}	Finite clauses

In order to identify chunks in a sentence, a classical approach is to use curated regular expression rules based on POS tag information to extract chunk boundaries (Grover and Tobin, 2006; Mohammed and Omar, 2011). Another popular approach is based on machine learning methods that require available corpora to learning patterns such as transformation-based learning (Ramshaw and Marcus, 1999; Avinesh and Karthik, 2007), decision forest model (Pammi and Prahallad, 2007), HMM and CRF (Awasthi, Rao, and Ravindran, 2006; Sha and Pereira, 2003), MEM (Sun et al., 2005), or SVM (Kudo and Matsumoto, 2001). In recent trends, chunking is considered as a sequence labeling problem where each word in a sentence is assigned a label (a prefix is one of the following three characters: I for Inside, O for outside, B for beginning of each chunk type). This type of label indicates full information about a chunk and its boundaries in the sentence (Akhundov, Trautmann, and Groh, 2018). As a result, using deep neural network models with this approach achieves start-of-the-art performance on the CoNLL chunking dataset (Sang and Buchholz, 2000; Zhai et al., 2017). The chunking task for French Language has also been realized by using rule-based approaches such as a cascade of finite state transducers to produce tree-like representations (Antoine, Mokrane, and Friburger, 2008), logical grammar (Blanc et al., 2010), or by applying machine learning methods such as an CRF model (Tellier et al., 2012).

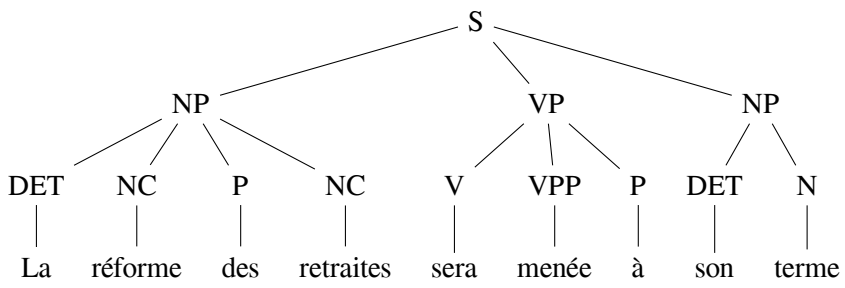


FIGURE 4.4: A visual example of the chunking task result for the sentence "Pension reform will be completed"

Constituency Structure and Parsing

A higher level of sentence analysis is the constituent-based grammar analysis in which the constituents and relations between them is identified in a sentence. Conventionally, a constituent is a lexical item or a group of lexical items organized in a hierarchical structure, and plays the role of a single unit in the sentence. A constituent can be a word, phrase, clause in the sentence. Constituency grammar is a theoretical foundation in order to analyze constituent structure. In fact, there are different terms used to denote this type of grammar like context-free grammar or phrase structure grammar, which was originally introduced by Noam Chomsky [11] in the 1950s (Chomsky, 1956; Chomsky, 1957; Chomsky, 1975). An example of constituent analysis of the sentence "La réforme des retraites sera menée à son terme" is given in Figure 4.5. The example shows that hierarchical constituents structures are usually nested. Constituency analysis provides deeper information about sentence structure than the shallow parser approach.

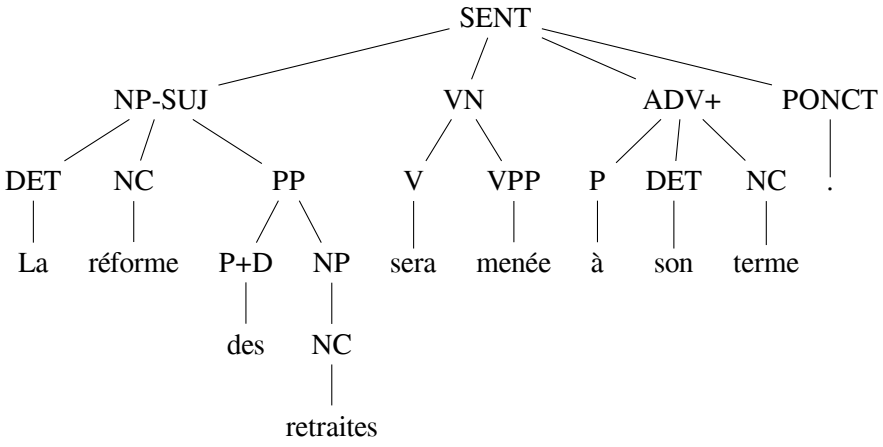


FIGURE 4.5: A visual tree of constituent analysis for the sentence "Pension reform will be completed." by using Berkeley Neural Parser[12]

Definition 4.3.3. A Context-Free Grammar G is conventionally defined as a quadruple (N, T, P, S), consisting of:

- N, a finite set of non-terminal symbols that corresponds to abstraction names or variables over terminals in T. For example, we use NP, VP, PP denoted for Noun Phrase, Verb Phrase, and Prepositional Phrase.
- T, a finite set of terminal symbols that correspond to lexical items in the natural language and build up the content of the sentence. This set is disjoints with the set of non-terminals.
- R, a finite set of rewriting rules or productions. Each element exists in the form of $A \rightarrow \beta$, where A is an element of N, β is a sequence of symbols constructed from $(N \cup T)$. For example in Table 4.8, some examples of grammar rules given.
- S, a designated start symbol used to represent the whole sentence, which is often noted S. It must be a member of N.

A derivation of the string of words is a sequence that generated by applying rules or productions. In other words, a derivation is the result of the process of inference based on the rules defined in R. Therefore, it can be organized as a parse tree with non-terminal symbols at the

[11] https://en.wikipedia.org/wiki/Noam_Chomsky
[12] https://github.com/nikitakit/self-attentive-parser

internal nodes and terminal at the leaves. For example, a parse tree or derivation in the figure 4.6 is the result of using grammar rules in the table 4.8

Grammar rules	Examples	
S → NP VP	[Il] [vend sa voiture à mon voisin]	[He] [sells his car to my neighbor]
NP → Determiner Noun	[sa] [voiture], [mon] [voisin]	[his] [car], [my] [neighbor]
NP → Pronoun	il	he
NP → Noun	voiture, voisin	car, neighbor
VP → VP NP PP	[vend] [sa voiture] [à mon voisin]	[sells] [his car] [to my neighbor]
VP → VP NP	[vend] [sa voiture]	[sells] [his car]
VP → Verb	vend	sells
PP → PP NP	[à] [mon voisin]	[to] [my neighbor]
PP → Preposition	à	to

TABLE 4.8: Grammar rule set in CFG and its examples

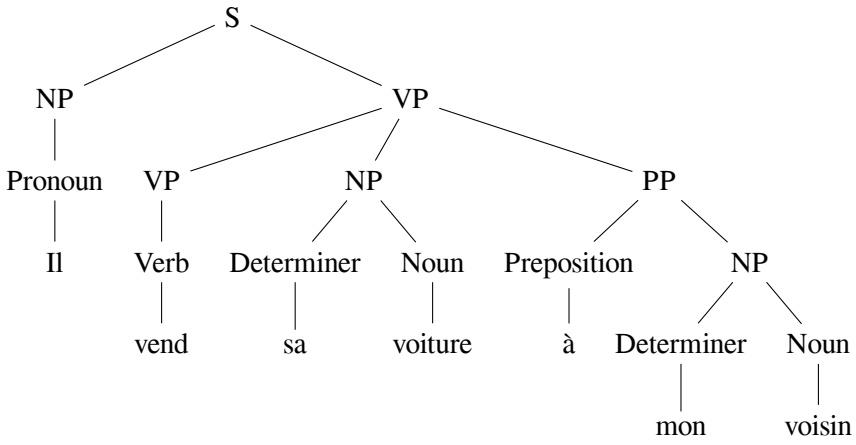


FIGURE 4.6: A formal parse tree by using CFG for the sentence "Il vend sa voiture à mon voisin" (He sells his car to my neighbor)

Definition 4.3.4. Consitutency Parsing is a process of analyzing and determining the constituent structures and their linking in order to extract a constituency-based parse tree from a sentence that must accord to the syntax and rules of the phrase structure grammar.

In order to build a constituency parser, we need to resolve some problems concerned to structural ambiguity. Two popular kinds of structural ambiguity are attachment ambiguity and coordination ambiguity.

As it happens often both in English and in French language, a particular constituent in a sentence can be attached at more than one places to the constituent parse tree. We have a

common pattern of this kind ambiguity: "*verb* 1st_noun_phrase preposition 2nd_noun_phrase" where the preposition and 2nd_noun_phrase are the prepositional phrases that can be attached to the 1st_noun_phrase or directly to the verb. Furthermore, the ambiguity still occurs even when the preposition and 2nd_noun_phrase are replaced by a relative or subordinate clauses. The sentence of the example (1) gives an illustration of this kind of ambiguity.

(3) Ma voisine [*a acheté*]$_{\text{VP}}$ [*un nouveau vélo*]$_{\text{1st NP}}$ [*pour*]$_{\text{PP}}$ [*son garçon*]$_{\text{2nd NP}}$.
My neighbor [*bought*]$_{\text{VP}}$ [*a new bike*]$_{\text{1st NP}}$ [*for*]$_{\text{PP}}$ [*her boy*]$_{\text{2nd NP}}$.

The second common kind of ambiguity occurs with conjunction words like *and*, *or*, *but* (in English), *et*, *ou* (in French), or similar words. Indeed, words, phrases, clauses and sentences can be coordinated with a modifier that can be placed either before or after the coordination (Okumura and Muraki, 1994). The examples (2) (3), below illustrate ambiguous cases relying on the usage of coordinated words.

(4) a. [*La maître observe la photo*] et [*la classe*]
[*The teacher observes the photo*] and [*the class*]

b. La maître observe [*la photo*] et [*la classe*]
The teacher observes [*the photo*] and [*the class*]

(5) a. [*Exigences de communication*] et [*performance*]
[*Communication*] and [*performance requirements*]

b. Exigences de [*communication*] et [*performance*]
[*Communication*] and [*performance*] requirements

Many approaches deal with ambiguous issues and achieve a full constituency parse tree. A dynamic programming approach developed by Richard Bellman [13] has become a powerful framework for tackling this task with popular algorithms such as Viterbi (Forney, 1973), or forward algorithm (Federgruen and Tzur, 1991). Based on the idea of solving a problem by breaking it into smaller problems (sub-problems), the optimal solution can recursively be found from results of the sub-problems (Bellman, 1966). Using this approach in constituency parsing task, we have three popular parsing algorithms: Cocke-Kasami-Younger (CKY) parsing algorithm (Kasami, 1966; Younger, 1967; Cocke, 1970), Early algorithm (Earley, 1970) and Chart parsing (Kaplan, 1973; Kay, 1980). These parsing algorithms still play a crucial role in current outstanding parsers. For instance, the state-of-the-art performance and accuracy on Penn Tree bank corpus used CKY-style algorithm to find the optimal constituency tree along with the modern approach based on sequence-to-sequence learning models (Kitaev and Klein, 2018; Zhou and Zhao, 2019; Mrini et al., 2019). A transition-based (shift-reduce) approach can also be also used to obtain a constituency parser that employs sequences of local transition actions to build up parsed tree-over-input sentences (Zhu et al., 2013; Mi and Huang, 2015; Liu and Zhang, 2017b; Liu and Zhang, 2017a).

Besides that, probability theory can be applied to solve the problem of disambiguation based on computing the probability of each derived interpretation. Probabilistic context-free grammars (PCFG) (Booth, 1969) use the probabilistic constituency grammar formalism. Based on this approach, a probabilistic CYK algorithm is proposed for building a constituency parser (Ney, 1991).

In order to achieve a constituency parse tree for French, we can used various applications such as Berkeley Neural Parser (Kitaev and Klein, 2018) or Stanford Parser (Zhu et al., 2013).

[13] `https://en.wikipedia.org/wiki/Richard_E._Bellman`

Furthermore, different constituency-parsed tree corpora for French are built by applying transformation algorithms to obtain a complete corpus. For example, the Sequoia Treebank contains 2,099 French sentences from different sources. It has been annotated with constituency trees (Candito and Seddah, 2012). Another example is the Modified French Treebank (MFT), extracted from the FTB corpus with added Lexical Functional Grammar annotations (Schluter and Van Genabith, 2008).

Dependency structure and parsing

The *dependency grammar* formalism focuses on the exploitation of dependency relations at the word-based level and explains more specifically the role of these relationships in the context of the sentence. Thereby lexical items are analyzed and connected to others by directed links representing binary asymmetric relations called *dependencies* that include the information about functional categories of the relation. In general, dependency grammar is a generic name for a class of contemporary grammatical theories based on the foundation of dependency relations analysis. Basic concepts and descriptions of this grammar theory has been introduced by the French linguist Lucien Tesnière[14] in his book "Élements de syntaxe structurale" (Element of Structural Syntax) published posthumously in 1959. Nowadays, he is honored as the pioneer in the field of dependency grammar theory by his crucial contributions (Tesnière, 1959). Various theoretical frameworks have been built up based on the grammatical theories of dependency structures such as Functional Generative Desscription (Sgall et al., 1986), Word Grammar (Hudson, 1991; Hudson and Hudson, 2007), Dependency Unification Grammar (Hellwig, 1986), Meaning-Text Theory (Mel′cuk et al., 1988), Functional Dependency Grammar (Järvinen and Tapanainen, 1997), Link Grammar (Sleator and Temperley, 1995), Operator Grammar (Harris, 1982), Extensible Dependency Grammar (Debusmann, 2006), Categorial Dependency Grammar (Dekhtyar, Dikovsky, and Karlov, 2015) or Universal Dependency Grammar (McDonald et al., 2013).

We will use notation $S = w_0, w_1, \ldots, w_n$ to denote a sentence, while $\mathcal{S}$ denotes a set of sentences. A dependency structure is expressed by G, while $\mathcal{G}$ indicates a set of dependency structures, that can be used in expressions of the next part.

Definition 4.3.5. A dependency structure can be defined as a directed graph. That is, a structure $G = (V, A)$ consisting of a set of vertices V and a set of directed edges A.

In general, a total order $<$ can be used on V to present the word order. The set of nodes, A, corresponds completely to the set of words and punctuation marks used to form a sentence. Furthermore, this set can occasionally contain subword morphemes such as stems or affixes used in the analysis of some languages. Dependency relation types are defined as a set $\mathcal{L} = \{l_1, \ldots, l_n\}$. Each dependency arc in A is a triple (w_i, l_k, w_j), representing a dependency relation type from the word w_i to the word w_j with label l_k. We have some notations representing relationships in A:

- $w_i \rightarrow w_j$ if and only if $(w_i, l_k, w_j) \in A$ for $l_k \in L$, used to indicate the dependency relation in a graph $G = (V, A)$.
- $w_i \rightarrow^* w_j$ if and only if $i = j$ or ($w_i \rightarrow^* w_{i'}$ and $w_{i'} \rightarrow w_j$) for $w_{i'} \in V$, expressed the reflexive transitive closure of the dependency relation in a graph $G = (V, A)$.
- $w_i \leftrightarrow w_j$ if and only if either $w_i \rightarrow w_j$ or $w_j \rightarrow w_i$, denoted the undirected depdendency relation in $G = (V, A)$.

[14] https://en.wikipedia.org/wiki/Lucien_Tesnière

- $w_i \leftrightarrow^* w_j$ if and only if $i = j$ or ($w_i \rightarrow^* w'_i$ and $w_{i'} \leftrightarrow w_j$) for $w_{i'} \in V$, indicated the reflexive transitive closure of the undirected dependency relations in G.

Besides, the dependency structure G need to obey some constraints or conditions as follows:

- Dependency structure is connected. G is connected, if $w_i, w_j \in V$, $w_i \leftrightarrow^* w_j$. The connectedness of a dependency structure can be obtained by inserting a special root node which is directly linked with a head word in the sentence.
- Dependency structure is hierarchical. G is acyclic, if $w_i \rightarrow w_j$, then not $w_i \rightarrow^* w_j$. The dependency structure is essentially a dependency tree.
- Every word has at most one syntactic head with the exception of the root node. G satisfies the single-head constraint, if $w_i \rightarrow w_j$, then not $w'_i \rightarrow w_j$, for any $w'_i \neq w_i$.
- The tree structure does not contain crossing dependency edges or projection lines. G is *projective*, if $w_i \rightarrow w_j$, then $w_i \rightarrow^* w'_i$, for any w'_i such that $w_i < w'_i < w_j$ or $w_j < w'_i < w_i$.

To illustrate the definition, we consider the dependency tree of Figure 4.7, which is represented by:

1. $G = (V, A)$
2. V = {root, La, réforme, des, retraites, sera, menée, à, son, terme, .}
3. A = {(root, pred, sera), (La, dep, réforme), (des, dep, réforme), (retraites, obj, des), (réforme, suj, sera), (menée, ats, sera), (à, mod, menée), (son, det, terme), (terme, obj, à), (., ponct, sera)}

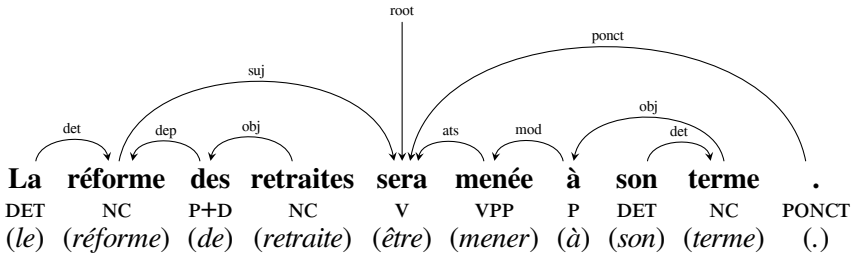
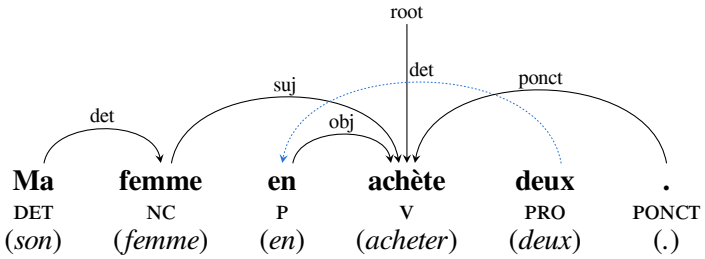


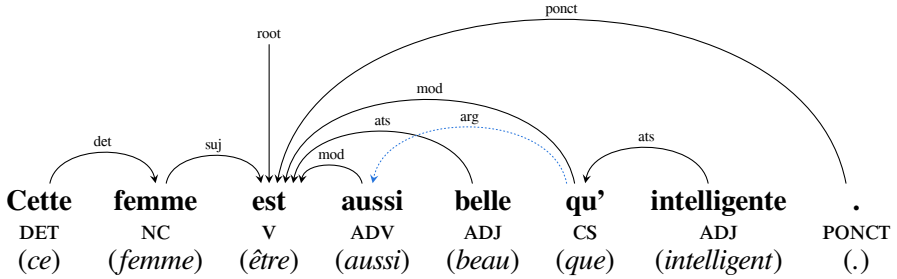
FIGURE 4.7: A visual tree of dependency parse tree of the sentence "Pension reform will be completed." by using BONSAI Malt Parser

Diving profound into projective and non-projective dependency tree situations, a dependency arc $(w_i,\ l,\ w_j)$ is said to be projective if there is a directed path from w_i to all words. We can draw a dependency arc between w_i and w_j without any other dependency arc crossing it (Covington, 2001). Accordingly, a dependency graph $G = (V, A)$ is viewed as a projective dependency tree if it is a dependency tree and if all dependency arcs are projective. By contrast. a dependency graph $G = (V, A)$ is considered as a non-projective dependency tree if it is a dependency tree and contains an non-projective arc. Most dependency structures in English and French are projective dependency trees. However, non-projective constructions of sentences still exist in some kind of expressions in these two languages even though they are infrequent. Some examples of French non-projective trees are given below: (6) using the clitic *en*, (7) using the comparative mode, (8) using relative pronouns.

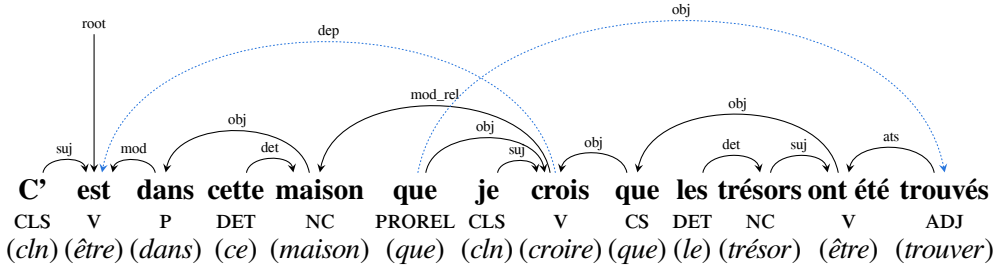
(6) *Ma femme en achète deux.* (*My wife buys two of them*)



(7) *Cette femme est aussi belle qu'intelligente.*
(*This woman is as smart as she is beautiful.*)



(8) *C'est dans cette maison que je crois que les trésors ont été trouvés .*
(*It is in this house that I believe that the treasures have been found.*)

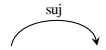
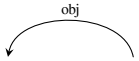
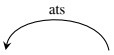
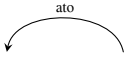


Definition 4.3.6. Dependency parsing is the process of determining a dependency structure for a sentence based on exploration of the grammatical relations of words with each other.

Consider a given sentence $S = w_0, w_1, \ldots, w_n$ with a hypothesis w_0 =root, the dependency parsing task is defined as the searching of dependency trees $G = (V, A)$ for the input sentence S where $V = 0, 1, \ldots, n$ denotes the vertex set, and A is the dependency arc set of tuples (w_i, l, w_j) which represents a dependency relation from w_i to w_j with label $l \in \mathcal{L}$ that is a set of dependency relation type. The set of dependency is defined based on the particular characteristic of the language used. For example, Figure 4.9 provides an overview about dependency relations used in the FTB corpus (Candito, Crabbé, and Falco, 2009).

Broadly speaking, there are currently different approaches for dependency parsing and it can be divided into two main types: grammar-based ones and data-driven ones. IN grammar-based approaches, dependency structures are mapped to phrase structures in order to benefit from parsing algorithms originally developed for constituency parsing. The earliest work of this approach is based on a modification of constituent-based grammar theory with building

new rules to adapt to dependency structures (Hays, 1964b; Gaifman, 1965). Therefore, dependency parsing methods are essentially similar to constituent parsing methods, based on dynamic programming algorithms. For instance, the parsing algorithm described in Hays, 1964b is a bottom-up dynamic programming algorithm that is not much different than the CKY algorithm used in context-free parsing. Another example is the link grammar parser, which uses a a form of top-down recursive algorithm with memorization to optimize the time performance of the parsing process (Sleator and Temperley, 1995). Also based on CFG theory, a bilexical dependency grammar is proposed by using the bilexical probability model associated a with CKY-style chart parser, this approach allows for more efficient parsing of dependency relations (Eisner, 1997; Eisner, 1996; Eisner and Satta, 1999; Eisner and Smith, 2005).

Name	**Description**	**Sentence and Arc Example**	
Dependency relations for verbal governors			
suj	Subject		The wind picks up
obj	Direct object		The cat stalks a mouse
de_obj	Argument introduced by de, non locative		He talks about his vacation
a_obj	Argument introduced by à, non locative		Henri thinks of Chloé
p_obj	Argument introduced by another preposition		I fight depression
mod	adjunct (non-argumental preposition, adverb)		Henri loudly shouts his anger
ats	Predicative adjective or nominal over the subject, following a copula		He's shorter than Michel
ato	Predicative adjective or nominal over the object		He finds it strange
aux.pass	Passive auxiliary verb		The car is repaired
aux.tps	Tense auxiliary verb		He won a million euros
aux.cause	Causative auxiliary verb		Who cleared the crowd ?

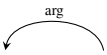
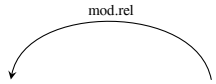
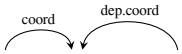
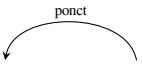
aff	Clitics in fixed expressions (including reflexive verbs)	 Il s' en souvient	He remembers it
Dependency relations for non-verbal governors			
det	Determiners	 Le contrat est terminé	The work is finished
arg	Used to linked prepositions	 De Brest à Paris	from Brest to Paris
mod.rel	Used to relative pronoun's antecedent to the verb governing the relative phrase	 La série dont Henri parle	The series that Henri talks about
coord	Links coordinators to the preceding conjunct	 Un professeur et un étudiant	A teacher and a student
dep.coord	Links a conjunct (except for the first) to the previous coordinator		
dep	Sub-speficied relation for prepositional dependents of non verbal governors	 L'utilisation des services	The use of services
ponct	Punctuations, except for commas playing the role of coordinators	 Henri mange du riz ?	Does Henri eat rice ?

TABLE 4.9: List of dependency relations extracted from the FTB corpus

Extensions to PCFGs have also been used for the dependency parsing task. Indeed, a PCFG can be lexicalized by assigning words and POS tag information to non-terminal nodes of the parse tree. Thus, we can apply the PCFG model to the lexicalized rules and trees. As a result, generative statistical parsers have been built with chart parsing algorithms for probabilistic or weighted grammars using the Penn Treebank corpus (Charniak, 2000; Collins, 2003). Furthermore, the combination of grammar-based models with each other has been studied in various works. For example, the PCFG structure model is combined with a dependency structure model to create a lexicalized phrase structure model. The factored model based on this novel structure uses the A* parsing algorithm allowing fast exact inference on probable parses ("Fast Exact Inference with a Factored Model for Natural Language Parsing"; Klein and Manning, 2003b).

Another working direction of the grammar-based approach is to view a dependency parsing task as a constraint satisfaction problem. Thereby, a grammar rule set is considered as a constraint set attached on dependency structures. The parsing task becomes a problem of searching a adapted dependency graph for a sentence that obeys all constraints defined. Based on this idea, Constraint Dependency Grammar (CDG) is created by mapping each grammatical rule to a constraint (e.g., word(pos(x)) $= D \Rightarrow$ (label(x) =DET, word(mod(x)) $= N$, pos(x) $<$ mod(x)), that mean, a determiner D modifies a noun N on the right with the label

DET) (Maruyama, 1990). Constraint propagation is one in many solutions for dependency parsing with CDG (Montanari, 1974; Waltz, 1975). About applications, PARSEC is a constraint-based dependency parser implemented based on constraint propagation, which is used for integration into a spoken recognition system (Harper and Helzerman, 1995; Harper et al., 1995). Furthermore, constraint propagation contributes to construct Topological Dependency Grammar (TDG) (Duchier, 1999; Duchier and Debusmann, 2001), where the grammatical constraint set consists of immediate dominance and linear precedence constrains.

The majority of dependency parsing systems is based on data-driven approaches. As its name indicates, data about sentences and theirs dependency structure annotations plays an important role when it comes to decide on using this approach. Nevertheless, the success of data-driven approaches remains dependent on the learning methods used. At the moment, supervised learning methods dominate in most approaches. A supervised dependency parsing method can be divided into two stages: learning the model and parsing. Learning the model is the task of building a model to learn from input sentence samples and their dependency structures in order to obtain the most optimal model. Parsing is the task of applying the model provided by the first task in order to obtain a dependency structure for a new sentence. In general, most data-driven dependency parsing approaches can be grouped in two main classes: transition-based approaches and graph-based approaches.

The idea of transition-based method has been proposed by Joakim Nivre (Nivre, 2004; Nivre, 2008). It employs a transition system and a supervised learning method to obtain dependency structures from sentence input. The learning model thus is constructed by predicting the next state transition and provides the history parse. The parsing problem focuses on predicting new parse tree using a greedy or a deterministic parsing algorithm and on inducing an optimal transition sequence model. A transition system is essentially an abstract machine that includes a set of states and a set of transitions between states. In the case of dependency parsing, the set of states is matched to a set $\mathcal{C}$ of configurations of internal dependency structures that includes initial and terminal configurations for the sentence. The set of transitions is correlated to a set $\mathcal{T}$ of steps in the derivation of a parse tree (Kübler, McDonald, and Nivre, 2009).

Definition 4.3.7. A configuration is defined as a triple $c = (\alpha, \beta, A)$, where:

- α is a stack of ordered words $w_i \in V$ that's is been processed,
- β is a buffer of ordered words $w_i \in V$ that's waiting to be processed,
- A is a set of dependency arcs (w_i, l, w_j) with $l \in L$, $w_i \; and \; w_j \in V$

A dependency parse tree needs to be derived through a transition sequence $\mathcal{C}_0^n = (c_0, c_1, \ldots, c_{n-1}, c_n)$, that begins in the initial configuration $c_0 = ([w_0]_\alpha, [w_1, \ldots, w_n]_\beta, \emptyset)$ for a input sentence and keeps on with a series of valid transitions of configurations $c_i = t(c_{i-1})$ with $i \in [1, n-1]$ and $\tau \in \mathcal{T}$, in order to reach a terminal configuration $c_n = (\alpha, \; [\emptyset]_\beta, \; A)$ for any α and A.

Definition 4.3.8. A transition is an action that adds a new dependency arc to A or modifies the stack α or the buffer β.

With the hypothesis that w_i denotes the word on the top of stack α, and w_j is the word at the bottom of the buffer β, a transition of an arc-standard parser will be chosen through three basic actions:

- LEFT-ARC (LA): add a new dependency arc $(w_i, \; l, \; w_j)$ to the arc set A and remove the word at the top of the stack α, with preconditions $\alpha, \; \beta \neq \emptyset$ and, $w_i \neq root \; node$.
- RIGHT-ARC (RA): add a new dependency arc $(w_j, \; l, \; w_i)$ to the arc set A, remove the word at the top of the stack α and replace w_j by w_i at the top of buffer β, with precondition $\alpha, \; \beta \neq \emptyset$.

- SHIFT (SH): remove the first word in the buffer α and push it onto the stack β, with precondition $\beta \neq \emptyset$.

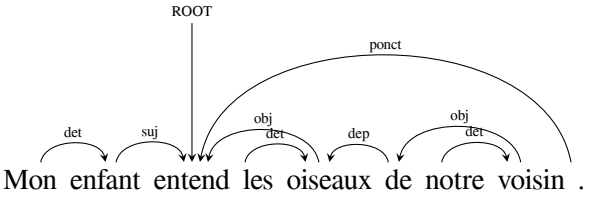


FIGURE 4.8: A visual dependency tree of the sentence "My child hears the birds of our neighbor."

TABLE 4.10: A transition sequence example for the sentence in the figure 4.8

C	**Transition sequence**	τ
c_0	([root]$_\alpha$, [Mon, enfant, …]$_\beta$, $A = \{\emptyset\}$)	SH
c_1	([root, Mon]$_\alpha$, [enfant, …]$_\beta$, $A = A \cup \{(\text{mon}, l_{\text{det}}, \text{enfant})\}$	LA
c_3	([root]$_\alpha$, [enfant, entend, …]$_\beta$, $A = A \cup \emptyset$)	SH
c_4	([root, enfant]$_\alpha$, [entend, …]$_\beta$, $A = A \cup \{(\text{enfant}, l_{\text{suj}}, \text{entend})\}$)	LA
c_5	([root]$_\alpha$, [entend, les, …]$_\beta$, $A = A \cup \{\emptyset\}$	SH
c_6	([root, entend]$_\alpha$, [les, …]$_\beta$, $A = A \cup \{\emptyset\}$	SH
c_7	([root, …, les]$_\alpha$, [oiseaux, …]$_\beta$, $A = A \cup \{(\text{les}, l_{\text{det}}, \text{oiseaux})\}$	LA
c_8	([root, entend]$_\alpha$, [oiseaux, …]$_\beta$, $A = A \cup \{\emptyset\}$	SH
c_9	([root, …, oiseaux]$_\alpha$, [de, …]$_\beta$, $A = A \cup \{\emptyset\}$	SH
c_{10}	([root, …, de]$_\alpha$, [notre, …]$_\beta$, $A = A \cup \{\emptyset\}$	SH
c_{11}	([root, …, notre]$_\alpha$, [voisin, …]$_\beta$, $A = A \cup \{(\text{notre}, l_{\text{det}}, \text{voisin})\}$	LA
c_{12}	([root, …, de]$_\alpha$, [voisin, …]$_\beta$, $A = A \cup \{(\text{voisin}, l_{\text{obj}}, \text{de})\}$	RA
c_{13}	([root, …, oiseaux]$_\alpha$, [de, …]$_\beta$, $A = A \cup \{(\text{de}, l_{\text{dep}}, \text{oiseaux})\}$	RA
c_{14}	([root, entend]$_\alpha$, [oiseaux, …]$_\beta$, $A = A \cup \{(\text{oiseaux}, l_{\text{obj}}, \text{entend})\}$	RA
c_{15}	([root]$_\alpha$, [entend, …]$_\beta$, $A = A \cup \{\emptyset\}$	SH
c_{16}	([root, entend]$_\alpha$, […]$_\beta$, $A = A \cup \{(., l_{\text{ponct}}, \text{entend})\}$	RA
c_{17}	([root]$_\alpha$, [entend]$_\beta$, $A = A \cup \{(., l_{\text{ponct}}, \text{entend})\}$	RA
c_{18}	([root]$_\alpha$, [entend]$_\beta$, $A = A \cup \{(\text{root}, l_{\text{root}}, \text{entend})\}$)	LA
c_{19}	([$\emptyset$]$_\alpha$, [root]$_\beta$, $A = A \cup \{(\text{root}, \_, \text{entend})\}$)	LA
c_{20}	([root]$_\alpha$, [$\emptyset$]$_\beta$, $A = A \cup \{\emptyset\}$)	

A dependency tree can be achieved from a transition sequence of a transition system. However, a transition sequence does not always produce a valid dependency tree because a dependency structure requires always conditions and constraints on connectedness, root node, and single head properties. In addition, all dependency trees generated by transition-based approaches satisfy the projective structure constraint.

The example in Table 4.10 shows a transition sequence by displaying steps from a configuration to another by using one of three standard transitions. We can see that the LA and RA transitions are used to reduce or replace words in the stack α and the buffer β. While the SH transition aims to shift a word into processing. The transition sequence in our example is not unique. In fact, different transition sequences can be created from an initial configuration. There are usually many choices for each state step corresponding to the change from a valid configuration to another valid configuration. Therefore, it is possible to obtain the same dependency parse tree from more than one transition sequences. Furthermore, different dependency parse trees can be derived from different transition sequences due to natural language ambiguity.

A problem of the transition-based approach is the occurrence of nondeterministic situations when changing states in non-terminal configurations. Accordingly, a deterministic first-best search or beam search can be applied to build up an oracle function $o(c)$ to find an optimal transition $\tau = o(c)$ that is used into a configuration chain to reach the next configuration $\tau(c)$ (Goldberg and Nivre, 2013). The use of machine learning techniques becomes more efficient by simply replacing the oracle function as a model parameter that must be learned from data. More generally, the learning model is based on predicting the correct transition $o(c)$ for any configuration c from corpus input, which need to be organized under the feature representation function $f(c) : \mathcal{C} \rightarrow \mathcal{T}$, where $\mathcal{C}$ is the set of possible configurations and $\mathcal{T}$ is the set of possible transitions. More specifically, sentences and their dependency parsing trees in the corpus will be used for extracting and validating the set of possible configurations $\mathcal{C}$ and the set of possible transitions $\mathcal{T}$ by using the transition system. Using the terminology of supervised learning, the training data is formed as $(f(c), \tau)$ or $f(c) \rightarrow \tau$, with $\tau = o(c)$. Various machine learning methods can be used such as memory-based learning (Nivre, Hall, and Nilsson, 2004), SVMs (Nivre et al., 2006), or the sequence-to-sequence deep neural network model, which is the most popular at the moment (Ballesteros et al., 2016; Ballesteros et al., 2017; Lhoneux, Ballesteros, and Nivre, 2019).

The most popular version of transition system is the arc-eager model which is developed by (Nivre, 2003) and used in dependency parsers for various languages such as Swedish and English (Nivre, Hall, and Nilsson, 2004; Nivre and Scholz, 2004). Instead of using three basic transition actions as in the arc-standard model, the arc-eager approach adds a REDUCE transition into the set of transitions. As a result, a dependency parse tree can be achieved in linear time by using a deterministic algorithm, and it is faster than results achieved by arc-standard approach.

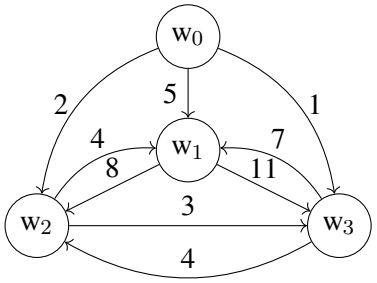
There are two directions in order to improve performance on proposed algorithms in the transition-based approach. The first one focuses on the transition system in which we can improve the system by defining novel transition actions or configurations, or by using other strategies of efficient searching for optimal transitions. For example, the author of (Gómez-Rodríguez and Nivre, 2013) has defined a new transition system with composition and restriction of the five transition actions SHIFT, UNSHIFT, REDUCE, LEFT-ARC, RIGHT-ARC. Similarly, the authors of (Zhang et al., 2016) have focused on building up a novel transition system that can generate arbitrary directed graphs in an incremental manner. The beam search using heuristic was modified to obtain higher efficiency (Johansson and Nugues, 2006; Zhang and Nivre, 2012). The latter dives into improvements on extracting new feature representation on the available corpus and using novel learning model (Zhang and Nivre, 2011; Chen and Manning, 2014; Weiss et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016).

Transition-based approaches also have limitations, such as being unable to produce non-projective dependency trees and to handle long-distance dependency problems. Therefore,

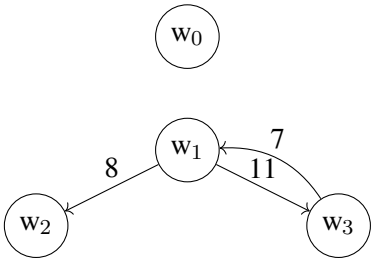
graph-based dependency parsing approaches are proposed to tackle directly with these problems. Graph-based methods can produce both projective and non-projective dependency trees, which occur in many languages. Also, graph-based models obtain better accuracy than the transition-based models when predicting long-distance arcs by scoring entire trees using global inference (McDonald and Nivre, 2011).

Based on graph theory, a discipline that has been studied for centuries, with numerous efficient algorithms available for processing directed and undirected graphs, the graph-based approach considers parsing as a search-based structure prediction problem in the space of all possible dependency trees for a given sentence (McDonald et al., 2005). More specifically, the notion of score is used on each dependency tree $G = (V, A)$ of a given input sentence. The overall score of each dependency tree achieved is computed on the scores of all edges in the tree G, i.e., $score(G) = score(V, A)$. Thus, the best dependency tree G_S in the space of all possible dependency trees $\mathcal{G}_S$ for the sentence S is defined as:

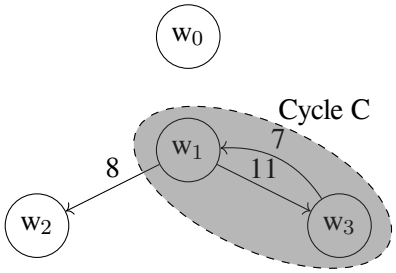
$$G_S = \underset{G=(V,A) \in \mathcal{G}_S}{\arg\max} \text{score}(G). \tag{4.5}$$



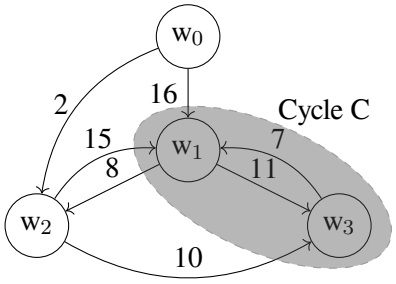
(A) The graph $G = (V, A)$ and its scores



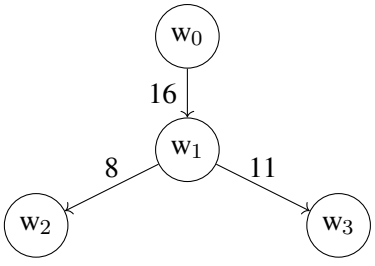
(B) The graph G' is not a MST



(C) A cycle found on G'



(D) The graph after contraction



(E) The dependency tree obtained at the final step from graph G

FIGURE 4.9: A visualization on a simple example with the Chu-Liu-Edmonds algorithms for a graph-based parsing

In order to explore this approach, let us consider the arc-factored parsing model, which is

one of the common graph-based parsing approaches (McDonald, 2006). For a given sentence, a standard directed graph $G_S = (V_S, A_S)$ is constructed, that is:

- A set of vertices $V_S = w_0, w_1, \ldots, w_n$, which are the ordered words of the sentence.
- A set of directed edges $A_S = \{(w_i, l, w_j) \mid \text{for } w_i,\ w_j \in V_S, l \in \mathcal{L} \text{ with } j \neq 0\}$ denotes all possible head-dependent relations.
- Weights on edges $\gamma_S = \{\gamma(i,j) \in \mathbb{R} \mid \text{for each } (i,j) \in (w_i, l, w_j) \text{ and } (w_i, l, w_j) \in A\}$ denote the score value of all possible edges in A.

The graph G is a directed multi-graph, that is: there exist possibly multiple arcs between a pair of vertices w_i and w_j. If node w_0 is eliminated, graph V_S, G_S becomes a complete graph. This is due to the fact that w_0 is a regular root node with outgoing edges from w_0 to all other words.

Definition 4.3.9. A maximum spanning tree (MST) of a directed graph G is the highest scoring of all edges of the directed graph G' that satisfies the spanning tree conditions $V' = V$.

Based on the graph-theoretic concept of spanning tree, arc-factored methods treat dependency parsing as searching the maximum spanning tree G'_S on the set of spanning trees of the directed graph G_S. To find the MST G'_S of a directed graph G_S, we can use Chu-Liu-Edmonds algorithm, that was independently developed by two different researchers: Chu, 1965 and Edmonds, 1967. This algorithm consists of a greedy edge-selection step, followed by a edge-weight updating step, and then possibly by a call of a recursive procedure on a new graph, derived from the original graph G_S. In general, the algorithm begins with each vertex in the graph, greedily finds the incoming edge with the highest score. If the result obtained is a tree, it is the MST tree. If not, there exists a cycle between at least one pair vertex in the graph. In the case of existence of a cycle, a procedure is used to identify the cycle and contract it into a single vertex and recalculate weights by adding the highest score into the outgoing and incoming edges of the cycle. Now, we have a new graph G_C with the new weights. A recursive procedure applies the algorithm to the new graph. This gives us either the resulting MST tree or a graph with a cycle. The call of the recursive procedure can continue as long as cycles occur. Parameters of vertices and edges are restored from the cycle when the recursion completes. The pseudo-code of the algorithm is Algorithm 2. As as result, the resulting spanning tree is the best non-projective dependency tree G_S for the input sentence.

An simple example of the use of the Chu-Liu-Edmonds algorithm is displayed in the Figure 4.9. We use the sentence $S = \{w_0, w_1, w_2, w_3, w_4\}$ where w_0 is a root vertex and the other vertices correspond to words, for example, “Elle est courageuse” (She is courageous) , w_1 is ‘elle’ (she), w_2 is ‘est’ (is) and w_3 is ‘courageuse’ (couragous). The initial weights of edges are random numbers.

Conventionally, the score of an edge is the dot product between a high dimensional feature representation f and a weight vector w:

$$\gamma(i,j) = w.f(i,j). \tag{4.6}$$

The score of a dependency tree $G_S = (V_S, A_S)$ for the input sentence S is subsequently defined as:

$$\text{score}(G_S) = \sum_{(i,j)\in\gamma_S} \gamma(i,j) = \sum_{(i,j)\in\gamma_S} w.f(i,j). \tag{4.7}$$

In general, we consider training data as $\mathcal{D} = \{(S_d, G_d)\}_{d=1}^{|D|}$, that is, pairs of input sentences S_t and their corresponding valid dependency trees G_d. By contrast with transition-based approach, we do not require a transformation on the training data, this is due to the fact that

Algorithm 2: Chu-Liu-Edmonds Algorithm

Data: A directed graph $G = (V, A)$ and scores γ
Result: A dependency tree out of MST G
begin
  $A' \longleftarrow \{(w_i, w_j) | w_j \in V, w_i = \arg\max_{w_i} \gamma(i, j)\}$;
  $G' \longleftarrow (V, A')$;
  if G' *has no cycles* **then**
    return G', it is a MST;
  else
    $C \longleftarrow$ a cycle in G', the result of a search procedure;
    $G_C \longleftarrow$ contract(G, C, γ), defined in algorithm 3;
    $G \longleftarrow$ a recursion on Chu-Liu-Edmonds algorithm's procedure with G_C and γ as input data;
    $w_j \longleftarrow$ find a vertex in C, such that, $(w_i, w_j) \in A$ and $(w_k, w_j) \in C$;
    $A \longleftarrow A \cup C \setminus \{(w_k, w_j)\}$;
    return G

Algorithm 3: Contracted Graph Algorithm

Data: A directed graph $G = (V, A)$, a cycle C and scores γ
Result: A contracted graph G_C
begin
  $G_C \longleftarrow G$ excluding nodes in C;
  Add a node w_c into G_C representing cycle C;
  for $w_j \in V \backslash C : \exists_{w_i \in C}(w_i, w_j) \in A$ **do**
    Add edge (w_i, w_j) to G_C with $\gamma(w_i, w_j) = \arg\max_{w_i \in C} \gamma(w_i, w_j)$;
  for $w_i \in V \backslash C : \exists_{w_j \in C}(w_i, w_j) \in A$ **do**
    Add edge (w_i, w_c) to G_C with $\gamma(w_i, w_c)$ = $\arg\max_{w_j \in C}[\gamma(w_i, w_j) - \gamma(a(w_j), w_j) + \gamma(C)]$, where $a(v)$ is the predecessor of v in C and $\gamma(C) = \sum_{v \in C} \gamma(a(v), v)$;
  return G_C

models are directly parsing over trees. Thus the training model focuses on assigning higher scores to valid trees rather than to invalid ones, instead of the classifying item into categories.

The extraction of features and the calculation of weights used for the score are the main problems that need to be solved, they correspond to variables w and f in formula (4.7). First, similar to the transition-based approach, extracting features to train edge-factored models is an important step that directly influences the accuracy of the result achieved when building the learning model. In general, we can used different features such as word embedding, original word form, lemma, POS tag of the head word and its dependent, dependency relation with itself, or distance from head to the dependent, etc. Secondly, we need to build a model to learn a set of weights corresponding to each feature. A popular learning framework for the problem is inference-based learning, also known as Perceptron Learning Algorithm (Rosenblatt, 1958; Collins, 2002). More specifically, an initial random set is assigned to initial weights. Each sentence in the training corpus $\mathcal{D}$ is parsed along with its initial weights. If the obtained parse tree matches the valid dependency tree in $\mathcal{D}$, we keep the weights unchanged. If not, we filter the features corresponding to the invalid parse and subtract a small number from their weights. Pseudo-code for the algorithm is illustrated in Algorithm 4.

Algorithm 4: Perceptron Learning Algorithm

Data: $\mathcal{D} = \{(S_d, G_d)\}_{d=1}^{|D|}$
Result: w: the weight vector
begin
  Initializaze w randomly;
  for $n \longleftarrow 1$ *to* N **do**
    for $d \longleftarrow 1$ *to* $|D|$ **do**
      $G' \longleftarrow \arg\max_{G' \in \mathcal{G'}_{S_d}} \sum_{(i,j)\in\gamma'} w.f(i,j)$;
      if $G' \neq G_d$ **then**
        $w \longleftarrow w + \sum_{(i,j)\in\gamma_d} w.f(i,j)$ - $\sum_{(i,j)\in\gamma'} w.f(i,j)$;
  return w;

Graph-based methods are frequently employed for building dependency parsing systems. The major part of ameliorations of novel approaches focuses on applying deep neural network models to achieve a new feature representation and weight vector. More specially, the score(G_S) in the equation (4.7) is calculated as the output layer of a multi-layer perceptron (Pei, Ge, and Chang, 2015). More specifically, the hidden layer h is calculated as:

$$h = g(W_h^e.a + b_h^e), \tag{4.8}$$

where g is an activation function, a is a vector that is concatenated out of word embeddings, and $e \in 0, 1$ indicates the direction between head and dependent. Then,

$$\text{score}(G_S) = W_o^e.h + b_o^e, \tag{4.9}$$

where W_h, W_o, b_h, b_o are the weights and bias of the hidden layer and of the output layer. Indeed, several neural network models are employed to graph-based systems, achieving significant results. For instance, experiments on the Penn Tree Bank corpus have proven the efficiency of the approach, obtaining the currently highest accuracy[15] for dependency parsing (Mrini et al., 2019; Zhou and Zhao, 2019).

Corpus data play an extremely important role for training and evaluating dependency parsing systems. In particuler, the Penn Tree Bank corpus has become quite popular in studies on dependency as well as constituency parsing. Similarly, the FTB corpus in French is the main resource contributing for builders of French dependency parsers. Each language has its own set of dependency relations. Therefore, a study on dependency can only apply to one language or to a group of closely related languages. Nevertheless, in recent time, work is in progress to define a general dependency set: *universal dependencies* are proposed as a consistent framework for building dependency tree bank annotations across languages (McDonald et al., 2013; Nivre et al., 2016). There are over 30 languages that have been annotated using universal dependency relations.

Dependency parsing for French language has been extensively studied on the basis of methods such as the above and accompanied by the development of French dependency corpora, such as the dependency structures on the French Tree Bank corpus that have been converted from the original format, namely constituency structures (Candito et al., 2009), or the development of the deep dependency structure on the Sequoia corpus ("Deep syntax annotation of the Sequoia French treebank"). Many dependency parsers have been completely implemented and

[15]Table `http://nlpprogress.com/english/dependency_parsing.html` synthesizes latest results with their accuracy.

deployed for different applications. In a nutshell, we describe below six powerful parsers for French, currently in use:

- The Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007) employs a statistical grammar-based approach with a latent-variable PCFG model. The core of its learning model is the Expectation-Maximization algorithm that is used to estimate probabilities of symbols by using latent annotations. Essentially, symbols are split according to phrasal or lexical properties. The original version of the Berkeley parser did not work for French. However, the development of a French version has been realized, with adaptations to grammar, and to unknown words suffixes.

- MaltParser (Nivre, Hall, and Nilsson, 2006) is built upon the transition-based approach. The arc-eager algorithm has been employed to build transitions between configurations in the transition system. Concerning to the feature representation of the learning model, the authors have used word forms, POS tags, dependency types of words. The original classifier of the parser is the first version of the library LIBSVM as of 2001 (Chang and Lin, 2011). However, the MaltParser implementation of the French version uses a linear classifier from the library LIBLINEAR (Fan et al., 2008). In fact, the French version has been customized and retrained using dependency structures from the French Tree Bank corpus.

- MSTParser (McDonald, Lerman, and Pereira, 2006) has been developed on the basis of the graph-based approach. Based on the computation of the score between pair words of the training corpus, this model processes candidate word pairs and gives as output either ”no link” or a link with a specified direction and type, by using the Greedy Prepend Algorithm (Yuret and Türe, 2006), which has been employed to obtain a decision list on dependency relation information. An adaptation of the MSTParser for French [16] has been customized by using the MElt tagger for extracting POS tag information of sentences.

- Grew Parser (Guillaume and Perrier, 2015) is based on a symbolic method defined in a graph rewriting framework for French language only. The parsing process is considered as a sequence of atomic transformations which begin from a list of lexical items and end up with the dependency tree. A set of rules is defined depending on linguistic contexts where a dependency relation can appear. Each atomic transformation is described by one of these rules. In the final step, a CKY-style algorithm is used in order to obtain valid dependency trees. Experiments obtain an 89.21% accuracy on the Sequoia corpus.

- Stanford Parser (Chen and Manning, 2014) is a large NLP framework that can do various tasks, such as POS tagging, constituency parsing, named entity recognition, or dependency parsing, for many languages. This framework has evolved significantly. In the latest version, a dependency parser has being developed, based on a transition-based approach, using an arc-standard system for treating transitions. A neural network model with a hidden layer is included, embeddings of features are added to the input layer, in fact the input layer is mapped to the hidden layer through a cube action function. The transition with the highest score is picked up and used for the next configuration. As as result, experiments have achieved an accuracy that is higher than the ones of MSTParser or MaltParser.

- SpaCy Parser (Honnibal, Goldberg, and Johnson, 2013; Honnibal and Johnson, 2015) also uses a transition-based approach. Based on the arc-eager system, a new non-monotonic transition “UNSHIFT” is added that aims to repair configurations in which the buffer

[16] Downloads of French versions for Berkeley Parser, MaltParser and MSTParser are available at: `http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html`.

is exhausted and the stack contains multi-words having no incoming arc. The training procedure employs the dynamic oracle-based search-and-learn training strategy which allows one or more transition sequences for a given tree and gives optimal predictions for all configurations (Goldberg and Nivre, 2012). An amelioration using a multi-task CNN training model for classification in the training procedure provides an 84.48% accuracy for labeled dependencies on the French Sequoia corpus (with universal dependency annotations) and the WikiNER corpus[17].

- Stanza Parser (Qi et al., 2019; Qi et al., 2020) uses a graph-based approach with an architecture described in (Dozat, Qi, and Manning, 2017). A neural network architecture with a multilayer BiLSTM network is used to obtain vector representations from input features. These vector representations are then used to calculate scores between input words and potential arcs. In the last step, inference on dependency trees uses a maximum spanning tree search based on the Chu-Liu-Edmonds algorithm. The system achieved a state-of-the-art performance on different languages by using Universal Dependency TreeBanks[18].

In addition, several recent studies focus on improving the performance of learning models in both transition-based and graph-based approaches by using complex deep neural network models. These studies achieve good accuracy results on various corpora created according to the universal dependencies (Nguyen, Dras, and Johnson, 2017; Fernández-González and Gómez-Rodríguez, 2019). Thanks to the availability of trained models and code sources, they open the road to the development of new dependency parsers for French.

4.3.3 A bridge between syntax and semantics

We already described the basic concepts and techniques needed to analyze a given sentence. On the lowest level, a sentence is considered as a sequence of characters. The tokenization step allows splitting the sentence into lexical items. Lemmatization refers to the analysis of radical tokens based on vocabulary resources nad morphological analysis, in order to remove inflectional endings and give the basic form of a word. Each word in the ordered word list of the sentence is classified into word-categories, and this task is called POS tagging task. At a higher level of syntactic analysis, words and their relationships are considered in a chunking task, in which groups of words in the sentence are identified based on the analysis of the head word and its adjacent words. We call such a group of words a chunk and its function depends on the function of its head word.

Any sentence in a language can be analyzed with respect to a set of grammar rules in which reside characteristics of the language, along with a subset of the vocabulary. Shallow parsing partitions a sentence into chunks of words but it does not essentially comply with any specific grammar rule of the language. We have introduced two typical approaches for the sentence analysis task. First, constituency parsing, which uses context-free grammar as a theoretical foundation for forming a sentence structure; constituency parsing can be viewed as an evolution of shallow parsing by using recursive structures for constituents. The resulting constituency trees show how lexical items are grouped together in order to provide richer information about phrases in the sentence. Secondly, we have introduced dependency parsing which is based on dependency grammars and analyzes sentences using of dependency relations between the words. Dependency trees are used to illustrate how words are related to each other via dependency relations of specific types.

[17] Downloads of the model are available at: `https://github.com/explosion/spacy-models/releases/tag/fr_core_news_sm-2.2.5`.

[18] Official site: `https://universaldependencies.org/`.

Dependency-based or constituency-based analysis propose different approaches of sentence analysis. Each approach has its advantages and disadvantages. For example, constituency parsing provides the entire structure of a sentence with all substructures, while dependency parsing achieves better performance with non-projective and fragmented sentences and seems to have a closer connection to meaning representation. However, it is very difficult to assert that one approach is better than the other and vice versa due to the fact that each one is constructed on different grammatical theories.

The usage of a grammar formalism to perform sentence analysis is crucial to move from syntactic analysis to meaning analysis. Syntax and grammatical functions provide useful information for accessing the meaning of a given sentence. In general, five elements play the role of main grammatical functions for building a sentence, namely subject, verb, direct and indirect object, complement and adverb. The position of these grammatical functions is different depending on the language. For example, in French a sentence mostly uses the order subject-verb-object. However, sentence with different orders may occur, such as subject-object-verb. Using morphology, syntax information and grammatical functions, we can infer a higher level of linguistic representation, namely, semantics.

One of the main purposes of syntactic representation analysis is to constitute a bridgehead to semantic interpretation. On the syntactic side, we can take a dependency tree and represent it in predicate-like form, which includes a grammatical relation, a source, and a target term (e.g., sub(sleeps, Henri) is a semantic representation of the sentence "Henri sleeps"). Moving to the semantic side, we can use a formalism for semantic representation, called semantic frames, which define events and semantic roles declared by the participants. For example, a frame describing an eating event can include semantic roles like food, time, location, etc.

In general, grammar formalisms have been invented to describe constraints about syntax with grammatical rules. Otherwise, these formalisms also play an role in creating a mapping of syntax into semantic representation. Using a grammar formalism to create an interface between syntax and semantics has become a prevalent trend, fueled by the emergence of several different grammars. Beside of the head phrase structure and dependency grammars above, many other grammar formalisms have emerged, such as:

- Tree Adjoining Grammar (TAG, Joshi, Levy, and Takahashi, 1975), a tree-rewriting system where a grammar consists of a set of elementary trees divided in initial and auxiliary trees which are combined of two tree rewriting operations called adjunction and substitution. The set of grammar rules is somewhat similar to CFG, but TAG constitutes a tree-generating rather than a string-generating system such as CFG. Using these operations, elementary structures over local dependencies can be specified such as agreement, subcategorization or filler-graph relations. Lexicalized Tree-Adjoining Grammars (LTAG) are a variant of TAG that has been created using lexicons associating sets of elementary trees with lexical items.

- Lexical Functional Grammar (LFG, Bresnan, 1982; Kaplan and Bresnan, 1982), built upon the idea of using parallel formal representations corresponding to types of linguistic information. Accordingly, LFG analyzes sentences by two main kinds of syntactic structures. First, c-structures deal with the surface of syntactic information such as linear or hierarchical organization of words into constituencies. Secondly, f-structures (functional structure) are finite sets of attribute-value pairs where an attribute denotes a symbol and its value can be a symbol, a semantic form, a set, or another f-structure. At the f-structure representation, we can capture abstract syntactic relations such as agreement, control and raising, binding, or unbounded dependencies.

- Head-driven Phrase Structure Grammar (HPSG, Pollard and Sag, 1994), a constraint-based approach for representing rich contextual syntax and meaning representation for

natural language. Indeed, this grammar framework was heavily influenced by Generalized Phrase Structure Grammar (GPSG, Gazdar et al., 1985) that has been derived from Phrase Structure Grammar (PSG). In general, HPSG consists of two essential components: typed feature structures and descriptive constraints that play a role similar to grammatical rules. A typed feature structure is created from signs and grammatical rules, in which signs are basic types of HPSGs. Words and phrases are two different sign subtypes. A sign's feature structure contains its morphological, syntactic or semantic properties. A feature structure in HPSG can be represented by Attribute-Value Matrices (AVM).

- Categorial Grammar (CG) (Ajdukiewicz, 1935), a general term used for a number of related formalisms that create intermediate interfaces between syntax and semantics for natural languages. CG is classified in the same group with the three above grammar formalisms based on lexicalized theory for grammar. In its purest form, CG consists of two essential components: a lexicon and a set of category inference rules. The lexicon plays the role of a function assigning a set of categories to each basic symbol, while the category rules will determine how categories can be matched and what new categories can be reached by inference. Many variants of CG have been created, such as Type-Logical Grammar (TLG, Morrill, 1994; Moortgat, 1997) and Combinatory Categorial Grammar (CCG, Steedman and Baldridge, 2011)

In the remained of this thesis we will focus on the CCG theory and its applications. Essentially, we will introduce a method to apply CCGs to French sentences. The choice of CCG is based on the following rationale:

- CCGs provide a transparent interface between syntax and underlying semantic representation. They allows access to a deep semantic structure of the phrase using λ-expressions and facilitate recovering of non-local dependencies involved in the construction such as coordination, extraction, control, and raising.

- We map CCG derivation trees to λ-calculus expressions. The latter is a formal system in mathematical logic. Every CCG inference rule is mapped to an equivalent λ-calculus expression.

- CCG inference rules are based on the foundational theory of combinatory logic which has had many applications in natural language processing (Schönfinkel, 1924; Curry et al., 1958; Haralambous, 2019). Therefore, CCG is based on a strong mathematical theory as has a high potential for wide coverage applications in natural languages.

In the next chapters, we will dive into the description of a syntax-semantic interface with CCG. More specifically, we will study CCG and apply them to syntax analysis of natural language. The process of mapping syntax to semantic representations will be presented. We will also introduce our main contributions in obtaining a French CCG corpus based on using dependency structure of sentences.

Part II

Building French Discourse Meaning Representation Parsing Framework

Chapter 5

CCG Derivations out of Dependency Syntax Trees

Grammar formalisms act like a bridge linking syntax and semantics of natural languages. In the previous chapter we gave an overview on syntactic processing and on the importance of grammar formalisms for obtaining semantic representations. Among different grammar formalisms, we chose CCG for an interface between the syntax and semantics because of its ability to handle long-range phenomena and its mapping with λ-expressions. In this chapter, we first focus on the definition of CCG and problems arising. The lack of a CCG corpus for French has been one the motivations for obtaining a method to transform the French tree Bank corpus in order to achieve a CCG Corpus (Le and Haralambous, 2019). These issues will be addressed in next sections of the chapter, followed by an evaluation of the obtained CCG corpus.

5.1 Introduction

The concept of CCG has first been introduced by Mark Steedman[1] in the 2000s (Steedman, 1999; Steedman, 2000a; Steedman, 2000b). They were introduced as an extension of CGs (Ajdukiewicz, 1935; Bar-Hillel, 1953) with the addition of category inference rules in order to obtain a wide coverage of natural languages. CCG is essentially a lexicalized grammar formalism in which words are mapped to syntactic categories that capture language-specific information about basic words and their lexical categories. Moreover, the CCG formalism is regarded as a mildly context-sensitive grammar in Chomsky's hierarchy and thereby an efficient way to describe a natural language's syntax (Figure 5.1).

Similar to some grammar formalisms such as TAL or LFG, CCG is able to capture the non-local dependencies[2] related to a substantial number of linguistic phenomena such as coordination, raising and control construction, relativization or topicalisation. Accordingly, the processing of non-local dependencies has a important influence to the accurate and complete determination of semantic compositions that can be represented in the form of predicate-argument structures.

The analysis of sentences in natural languages in the CCG formalism provides a transparent interface between the syntactic representation and composition of the underlying meaning representation. In other words, rules of semantic composition are mapped one-to-one directly to rules of syntactic composition. Thus, the predicate-argument structure or λ-expressions can be use to rewrite any CCG derivation tree.

CCG is based on a strong foundation of combinatory logic, which bears a close connection to formal semantics (e.g., Montague Gammar, Partee, 1975 and Haralambous, 2019). In addition, CCGs can gain expressive power from λ-calculus which is broadly used in combinatory logic.

[1] Currently a professor of Cognitive Science at the university of Edinburgh. His personal page is available at the address: `https://homepages.inf.ed.ac.uk/steedman/`

[2] This term is a synonym of *long-distance dependency*, *long-range dependency* and *unbounded dependency*.

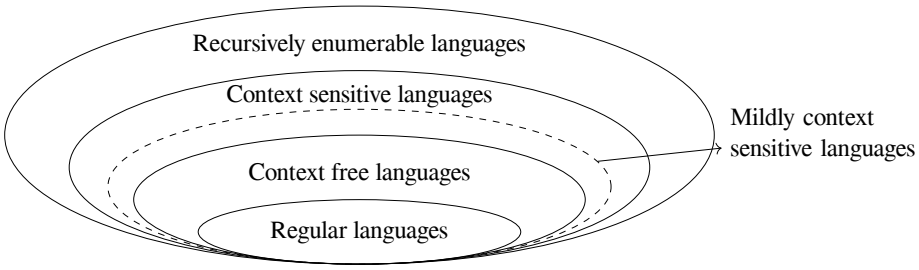


FIGURE 5.1: Chomsky's hierarchy of formal languages

With advantages on wide-coverage of natural languages, CCGs have been successfully applied to a broad range of different practical applications such as data-driven parsing (Vaswani et al., 2016; Lewis, Lee, and Zettlemoyer, 2016), wide-coverage semantic parsing (Bos, 2008), question answering (Clark, Steedman, and Curran, 2004; Ahn et al., 2005), and machine translation (Birch, Osborne, and Koehn, 2007; Nadejde et al., 2017).

5.2 Combinatory Categorial Grammars

5.2.1 Categories and Combinatory Rules

CCG is essentially a non-transformational grammatical theory relying on combinatory logic. Beyond that, CCG is also a lexicalized grammar formalism. Therefore, syntactic types or categories play an crucial role in the construction of a grammar. Categories are regularly used to identify constituents which are called as either axiom categories (primitive categories) or as complex categories. While axiom categories are used as exponents of basic units of sentences such as number, common noun, case, inflection and so on, complex categories are created from the association of different axiom categories or other complex categories. For example a complex category for verbs (e.g., X\Y) bears categories identifying the type of the results (X) and arguments/complements (Y). The order of arguments is fixed and cannot be changed in these categories.

Definition 5.2.1. Let Δ be a finite set of given axiom categories. A lexical category $\mathcal{C}(\Delta)$ over Δ is the smallest set such that:

- $\Delta \subseteq \mathcal{C}(\Delta)$;
- if $X,\ Y \in \mathcal{C}(\Delta)$, then $X/Y \in \mathcal{C}(\Delta)$;
- if $X,\ Y \in \mathcal{C}(\Delta)$, then $X\backslash Y \in \mathcal{C}(\Delta)$.

Definition 5.2.2. Let Σ be a set of terminals and $\mathcal{C}(\Delta)$ a finite set of lexical categories. A *lexicon* is a finite set of binary relations (σ, c) where $\sigma \in \Sigma$ and $c \in \mathcal{C}(\Delta)$.

Definition 5.2.3. A CCG is defined as a set $G = <\Sigma, \mathcal{C}(\Delta), f, \varsigma, \Re>$ where:

- Σ defines the finite set of *terminals*.
- $\mathcal{C}(\Delta)$ is defined as the definition 5.2.1
- f is the *lexical category function* mapping terminals into categories, which is also called a lexicon.
- ς is a unique *starting symbol*, $\varsigma \in \Delta$.
- $\Re$ is a finite set of *combinatory rules* which we describe below.

By X, Y, Z we denote meta-categories (they stand for any category of the grammar). The set $\Re$ includes two basic rules inherited from AB categorial grammars:

$$\begin{array}{l}\text{Forward Application:}\\ \text{X/Y:}f \qquad \text{Y} \quad \text{:a} \quad \Rightarrow \quad \text{X:}f(a),\\ \text{Backward Application:}\\ \text{Y:}a \qquad \text{X}\backslash\text{Y:}f \quad \Rightarrow \quad \text{X:}f(a).\end{array} \tag{5.1}$$

The first version of the rule set $\Re$ from pure CG (Ajdukiewicz, 1935; Bar-Hillel, 1953) contains only the application function with functors to the left $\backslash$ or to the right $/$, called Backward or Forward application. The restriction on the rule set aims to limit expressiveness to the level of CFGs. An important contribution of CCG theory is the extension of AB categorial grammars by a set of rules based on composition (**B**), substitution (**S**), and type-raising (**T**) *combinators* of combinatory logic (Curry et al., 1958). These rules allow processing of long-range dependencies, and extraction/coordination constructions. They can be represented via λ-calculus, as follows:

$$\begin{array}{l}\text{Forward Composition:}\\ \begin{array}{ccc}\text{X/Y} & \text{Y/Z} & \\ & & \Rightarrow_{\mathbf{B}} \\ \lambda y.f(y) & \lambda z.g(z) & \end{array} \qquad \begin{array}{c}\text{X/Z}\\ \\ \lambda z.f(g(z)),\end{array}\\ \text{Forward Crossing Composition:}\\ \begin{array}{ccc}\text{X/Y} & \text{Y}\backslash\text{Z} & \\ & & \Rightarrow_{\mathbf{B}} \\ \lambda y.f(y) & \lambda z.g(z) & \end{array} \qquad \begin{array}{c}\text{X}\backslash\text{Z}\\ \\ \lambda z.f(g(z)),\end{array}\\ \text{Backward Composition:}\\ \begin{array}{ccc}\text{Y}\backslash\text{Z} & \text{X}\backslash\text{Y} & \\ & & \Rightarrow_{\mathbf{B}} \\ \lambda z.g(z) & \lambda y.f(y) & \end{array} \qquad \begin{array}{c}\text{X}\backslash\text{Z}\\ \\ \lambda z.f(g(z)),\end{array}\\ \text{Backward Crossing Composition:}\\ \begin{array}{ccc}\text{Y/Z} & \text{X}\backslash\text{Y} & \\ & & \Rightarrow_{\mathbf{B}} \\ \lambda z.g(z) & \lambda y.f(y) & \end{array} \qquad \begin{array}{c}\text{X/Z}\\ \\ \lambda z.f(g(z)),\end{array}\end{array} \tag{5.2}$$

$$
\begin{array}{lll}
\text{Forward Substitution:} & & \\
(\mathrm{X/Y})/\mathrm{Z} & \mathrm{Y/Z} & \Rightarrow_{\mathbf{S}} \quad \mathrm{X/Z} \\
\lambda zy.f(z,y) & \lambda z.g(z) & \lambda z.f(z,g(z)), \\
\text{Forward Crossing Substitution:} & & \\
(\mathrm{X/Y})\backslash\mathrm{Z} & \mathrm{Y}\backslash\mathrm{Z} & \Rightarrow_{\mathbf{S}} \quad \mathrm{X}\backslash\mathrm{Z} \\
\lambda zy.f(z,y) & \lambda z.g(z) & \lambda z.f(z,g(z)), \\
\text{Backward Substitution:} & & \\
\mathrm{Y}\backslash\mathrm{Z} & (\mathrm{X}\backslash\mathrm{Y})\backslash\mathrm{Z} & \Rightarrow_{\mathbf{S}} \quad \mathrm{X}\backslash\mathrm{Z} \\
\lambda z.g(z) & \lambda zy.f(z,y) & \lambda z.f(z,g(z)), \\
\text{Backward Crossing Substitution:} & & \\
\mathrm{Y/Z} & (\mathrm{X}\backslash\mathrm{Y})/\mathrm{Z} & \Rightarrow_{\mathbf{S}} \quad \mathrm{X/Z} \\
\lambda z.g(z) & \lambda zy.f(z,y) & \lambda z.f(z,g(z)).
\end{array}
\tag{5.3}
$$

There are also type-raising rules in CCG:

$$
\begin{array}{l}
\text{Forward type-raising:} \\
\mathrm{X}{:}x \quad \Rightarrow_{\mathbf{T}} \quad \mathrm{T/(T\backslash X)}{:}\lambda f.f(x), \\
\text{Backward type-raising:} \\
\mathrm{X}{:}x \quad \Rightarrow_{\mathbf{T}} \quad \mathrm{T\backslash(T/X)}{:}\lambda f.f(x).
\end{array}
\tag{5.4}
$$

In the case of coordination constructions, the lexical category $(X/X)\backslash X$ is used to validate the combination of similar components in the rules of formula (5.2) and (5.3). It is formalized as follows:

$$
\begin{array}{l}
\text{Coordination:} \\
\mathrm{X}:g \quad \mathrm{X}:f \quad \Rightarrow_{\mathbf{\Phi}^n} \quad \mathrm{X}{:}\lambda x.f(x) \wedge g(x)
\end{array}
\tag{5.5}
$$

Let us give a simple example of CCG: $G =< \Sigma, \Delta, f, \varsigma, \Re >$, where:

- Σ := {Henri, regarde, la, télévision }.
- Δ := {S, NP }
- $\mathcal{C}(\Delta)$:= {S, NP, S\NP }
- Function f: f(Henri) := {NP, NP/NP}, f(regarde) := {S\NP, (S\NP)/NP}, f(la_télévision) := {NP}
- ς := S (sentence)
- $\Re$ as defined in formulas (5.1), (5.2), (5.3), (5.4), (5.5).

In order to achieve CCG derivation, each word in a sentence first needs to be assigned to the appropriate lexical categories. A word can have different lexical categories depending on its position or function in the sentence (e.g., word 'la' in the example 9). For instance, we have the following lexical categories:

(9)

Henri	←	NP
regarde	←	(S\NP)/NP
la	←	NP
la	←	NP/NP
bien	←	(S\NP)/(S\NP)
dort	←	(S\NP)

We see that transitive verbs (e.g., regarder (to watch)) are assigned to lexical categories that are different than those of intransitive verbs (e.g., dort (to sleep)); the latter take a single argument, which is of category NP. Modifier words (e.g., bien (well)) can be associated with a verb to become constituents taking the same argument from the verb accompanied with.

In the combinatory rules above, each category is accompanied by a corresponding λ-expression. Example 10 illustrates the process of obtaining a semantic interpretation out of a syntactic category. The transparency between syntactic interpretation and semantic representation is illustrated through each transformation from applying the combinatory rules in Figure 5.2.

(10)

Henri	←	NP	:	$henri''$
regarde	←	(S\NP)/NP	:	$\lambda x.\lambda y.regarde'xy$
la	←	NP	:	la'
la	←	NP/NP	:	$\lambda x.la'(x)$
bien	←	(S\NP)/(S\NP)	:	$\lambda f.\lambda x.bien'(fx)$
dors	←	(S\NP)	:	$\lambda x.dors'x$
Syntax	↔	CCG	↔	Semantics

By using the definition of CCG and the rule set $\Re$, we find a CCG derivation for the example illustrated in Figure 5.2.

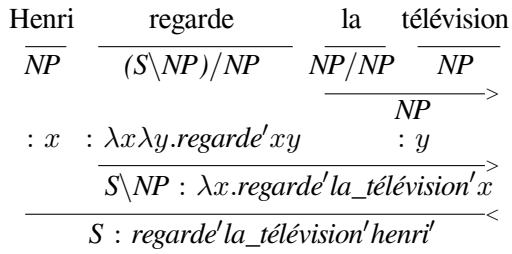


FIGURE 5.2: The 1st CCG derivation of the sentence: "Henri watches TV"

5.2.2 Some Principles Applied to CCG

Mapping between syntactic and semantic components must comply to the following three principles (Steedman, 2000b, p32) as follows:

- The Principle of Lexical Head Government: Both bounded and unbounded syntactic dependencies are specified by the lexical syntactic type of their head.
- The Principle of Head Categorial Uniqueness: A single nondisjunctive lexical cateogory for the head of a given construction specifies both the bounded dependencies that arise

when its complements are in canonical position and the unbounded dependencies that arise when those complements are displaced under relativization, coordination, and the like.

- The Principle of Categorial Type Transparency: For a given language, the semantic type of the interpretation together with a number of language-specific directional parameter settings uniquely determines the syntactic category of a word.

From the head government and head categorial uniqueness principles we can see that composition generalization may be required for sentences. In particular, generalization allows composition into functions with more than one argument. Generalized composition can be defined as follows:

Generailized Forward Composition:

$$\begin{array}{lll} \text{X/Y} & \text{(Y/Z)/\$}_1 & \\ f & \ldots\lambda z.g(z)\ldots & \Rightarrow_{\mathbf{B}^n} \\ & & \end{array} \begin{array}{l} \text{(X/Z)/\$}_1 \\ \ldots\lambda z.f(g(z\ldots)), \end{array}$$

Generalized Forward Crossing Composition:

$$\begin{array}{lll} \text{X/Y} & \text{(Y\textbackslash Z)\$}_1 & \\ f & \ldots\lambda z.g(z)\ldots & \Rightarrow_{\mathbf{B}^n} \\ & & \end{array} \begin{array}{l} \text{(X/Z)\$}_1 \\ \ldots\lambda z.f(g(z\ldots)), \end{array} \tag{5.6}$$

Generalized Backward Composition:

$$\begin{array}{lll} \text{(Y\textbackslash Z)\textbackslash\$}_1 & \text{X\textbackslash Y} & \\ \ldots\lambda z.g(z)\ldots & f & \Rightarrow_{\mathbf{B}^n} \\ & & \end{array} \begin{array}{l} \text{(X\textbackslash Z)\textbackslash\$}_1 \\ \ldots\lambda z.f(g(z\ldots)), \end{array}$$

Generalized Backward Crossing Composition:

$$\begin{array}{lll} \text{(Y/Z)/\$}_1 & \text{X\textbackslash Y} & \\ \ldots\lambda z.g(z)\ldots & f & \Rightarrow_{\mathbf{B}^n} \\ & & \end{array} \begin{array}{l} \text{(X\textbackslash Z)\$}_1 \\ \ldots\lambda z.f(g(z\ldots)), \end{array}$$

here the "\$ convention" is defined recursively: For a category α, notation $\alpha\$$ (respectively $\alpha/\$$, $\alpha\backslash\$$) denotes the set containing α and all functions (respectively leftward functions, rightward functions) mapping it to a category in $\alpha\$$ (respectively $\alpha/\$$, $\alpha\backslash\$$) (Steedman, 2000b, p. 42).

We can characterize all combinatory rules that conform to the directionality of their inputs by the three following principles (Steedman, 2000b, p. 54):

- The Principle of Adjacency: Combinatory rules may only apply to finitely many phonologically realized and string-adjacent entities.

- The Principle of Inheritance: All syntactic combinatory rules must be consistent with the directionality of the principal function. By applying this principle, we can exclude rules similar to the following instance of composition:

$$\text{X/Y} \quad \text{Y/Z} \quad \not\Rightarrow \quad \text{X\textbackslash Z} \tag{5.7}$$

- The Principle of Consistency: If the category that results from the application of a combinatory rule is a function category, then the slash functor defines the directionality of the corresponding arguments in the input function. With this principle, we can eliminate the following kind of rule:

$$\text{X\textbackslash Y} \quad \text{Y} \quad \not\Rightarrow \quad \text{X} \tag{5.8}$$

5.2.3 Spurious ambiguity

There is a second CCG derivation for the sentence “Henri regarde la télévision” (Henri watches TV) ans we illustrate it in Figure 5.3. Indeed, a sentence can be parsed into many different CCG structures. In other words, we encounter here a situation where a grammatical sentence may have many valid and equivalent semantic parses. This phenomenon is called *spurious ambiguity* and is the cause of a combinatory explosion in the search space when we parse. Nevertheless, this problem is not considered as a drawback of CCGs, on the contrary: its existence adds flexibility to CCGs when analyzing coordination and extraction constructions.

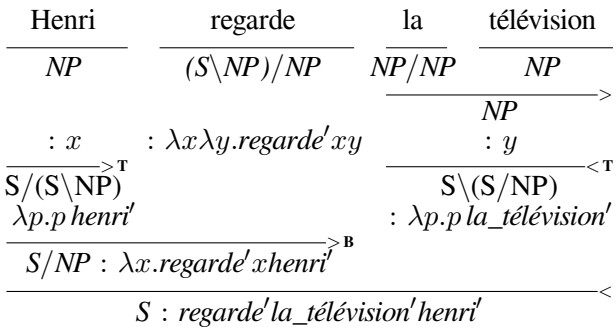


FIGURE 5.3: A 2nd CCG derivation for the sentence: “Henri watches TV”

5.3 The State-of-the-art of French CCG Corpora

Different approaches based on sentence structure can be used to build a CCG derivation tree. The two main ones are based more specifically on constituents and on dependencies. Each one has its own drawbacks and advantages—however, both require a pre-analysis step to obtain the necessary meta-information from the corpus.

Constituency trees have allowed the construction of a CCG Bank corpus for English (Hockenmaier and Steedman, 2007) out of the Penn Wall Street Journal Phrase Structure Treebank (Marcus, Marcinkiewicz, and Santorini, 1993), where a CCG derivation tree was created out of each sentence of the corpus. First, the authors used results from (Magerman, 1994) for determining the constituent type of each node, which was assigned to a head complement or to an adjust. Then, they transformed the constituency trees of the corpus into binarized trees. Finally, they assigned CCG categories based on the nodes of the binarized trees.

Similarly to (Hockenmaier and Steedman, 2007) and (Tse and Curran, 2010), (Xue et al., 2005) constructed a Chinese CCG Bank using the Penn Chinese Treebank. A CCG Bank for German has also been created, out of the Tiger Tree Bank (Hockenmaier, 2006). The authors used an algorithm based on the binarization of constituency trees. In particular, they built sentence digraphs, the nodes of which are labeled with syntactic categories and POS tags, and the edges of which are labeled with syntactic functions. These graphs are pre-processed and transformed into planar trees. Then a binarization step is applied to the planar trees in order to allow CCG derivation extraction. There exists also an Arabic CCG Bank built out of the Penn Arabic Tree Bank (El-Taher et al., 2014), using similar methods.

Concerning the approach based on dependency trees, (Bos, Bosco, and Mazzei, 2009) created an Italian CCG Bank out of the Turin University Tree Bank. The authors first converted dependency trees into phrase structure trees and then used an algorithm similar to the one that (Hockenmaier and Steedman, 2007). (Çakıcı, 2005) and (Ambati, Deoskar, and Steedman, 2018) used to extract CCGs for Turkish and Hindi. They parsed dependency trees and assigned

CCG lexical categories to nodes based on complement or adjunct labels. In the case of Hindi, a further parsing step was added, using the CKY algorithm.

As for French, Richard Moot created a corpus called TLGBank based not on CCGs, but on *Type-Logical Grammars* (TLG, Moot, 2015). TLGs and CCGs are derived from the same notion of Categorial Grammar, but TLGs have been used mainly for theoretical issues in relation to logic and theorem-proving, while CCGs are more relevant to computational linguistics since they keep expressive power and automata-theoretic complexity to a minimum (Steedman and Baldridge, 2011).

There also exists an extension of CCGs to French, introduced in (Biskri, Desclés, and Jouis, 1997) under the name *Application and Combinatory Categorial Grammar* (ACCG). This theory inherits the combinatory rules of CCGs and adds meta-rules to control type-raising.

In this chapter, we use the dependency-based approach because we consider that it has several advantages over constituency based structures (see Kahane, 2012 for an interesting discussion of this issue, as well as Osborne, 2019). First, dependency-based structures provide better connections between words and their neighbors in the sentence. Secondly, dependency-based structures give a closer look to the syntax-semantics interface. And finally, the dependency-base structures are more adapted to lexical function processing.

5.4 French Dependency Tree Bank

A tree bank is a linguistically annotated corpus made of large collections of manually annotated and verified syntactic analyses of sentences. Annotations are applied to words, components, phrases in the sentence, such as tokenization, part-of-speech tagging, constituency tree, dependency tree and semantic role labeling, named entity recognition, etc.

The French Tree Bank (FTB,) contains approx. a million words, mapped into 21,550 sentences taken from annotated news articles of the newspaper *Le Monde* in the period 1991–1993. Initially, the corpus was constituency-based. Then, a dependency-based version has been automatically created, as described in Candito, Crabbé, and Denis, 2010 and Candito et al., 2009. More specifically, each word is annotated with information on part-of-speech tagging, sub-categorization, inflection, lemma, and parts for compounds. Each sentence is annotated into two structures: dependency structure and constituency structure.

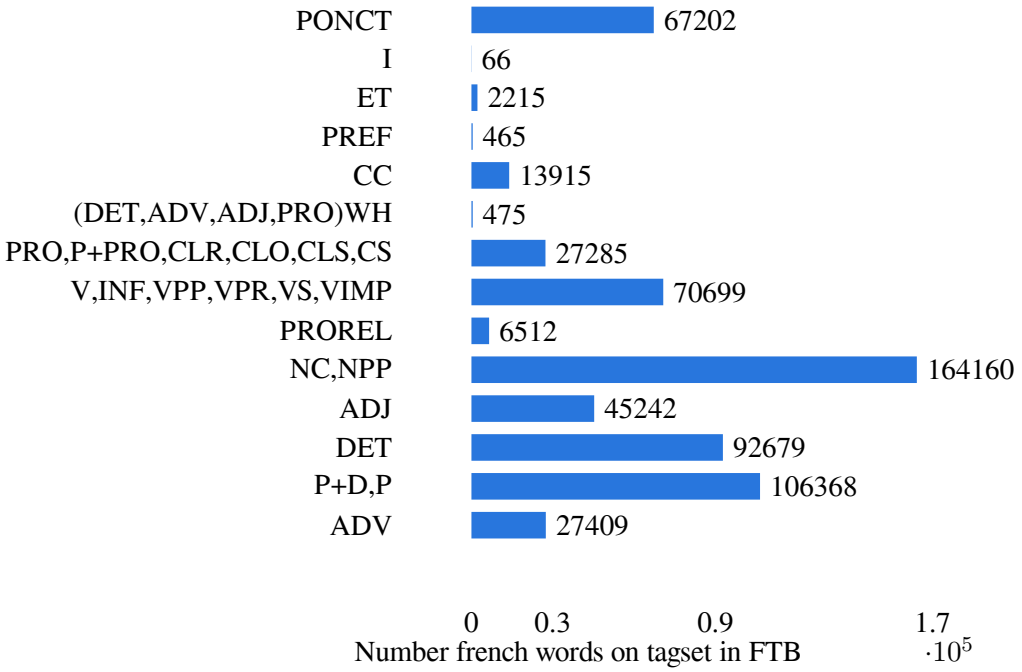


FIGURE 5.4: Distribution of French words by POS tags in the French Tree Bank corpus

Figures 5.4 and 5.5 display statistics about the distribution of POS tags and dependency relation types in the French Tree Bank corpus. Accordingly, we can see that words of noun type and their satellites, such as determinants, prepositions or adjectives, account for the largest number of types in the corpus. In a similar way, the mod, obj.p and det dependency relation types occupy the highest position in the distribution chart. These types are used for relations involving noun words.

5.5 CCG Derivation Tree Construction

In this section we describe our three-step process for obtaining CCG derivation trees. In the first step we chose among the many lexical categories that have been used in the corpus for each word those that fit together in the given sentence; we then attach them as labels to the dependency tree. In the second step we extract chunks from the dependency tree and add new nodes and edges in order to binarize the tree. In the final step we infer lexical categories for the new nodes and check whether we can move from the leaves to the root of the tree by applying combinatory rules—the root of the tree must necessarily have the lexical category S.

5.5.1 Assigning Lexical Categories to Dependency Tree Nodes

As already mentioned in § 5, dependency trees connect linguistic units (e.g., words) by directed edges representing syntactic dependencies. They are rooted digraphs, the root being the head of the sentence (most often the verb). Edges are labeled with labels belonging to a set of syntactic functions, e.g., subject, object, oblique, determiner, attribute, etc. Dependency trees are obtained by *dependency parsing*. There are different methods for building high precision parsers, some based on machine learning algorithms trained on large sets of syntactically annotated sentences, others based on an empirical approaches using formal grammars. For this task, we have used the French version of MaltParser (Candito et al., 2010).

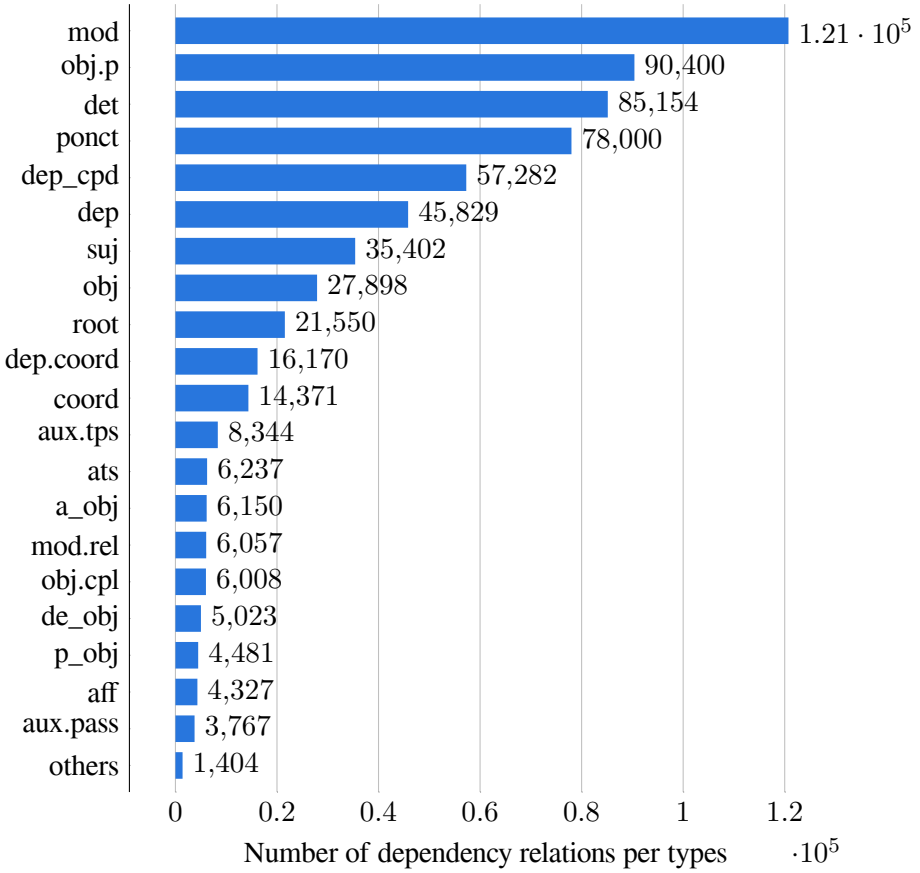


FIGURE 5.5: Distribution of dependency relations in the French Tree Bank corpus

CCG theory assigns two kinds of lexical categories to words: *basic categories* (e.g., S, NP, PP), and *complex categories* obtained by combination of basic categories using application functors $\backslash$ and $/$, e.g., S$\backslash$NP is a complex category that can be assigned to a word that expects a word with lexical category NP on its left, in order to produce an S; S$/$NP means that the NP-word is expected on the right.

In order to assign lexical categories to nodes of the dependency tree, we first process words that have unique lexical categories in the corpus: for example, nouns have lexical category NP, adjectives have lexical category NP$/$NP or NP$\backslash$NP depending whether they are on the left or on the right of the noun, etc. Once we have assigned these unique (or position-dependent, as in adjectives) lexical categories, we move over to verbs.

The main verb of the sentence, which is normally the root of the dependency tree, may have *argument dependencies*, labeled *suj*, *obj*, *a_obj*, *de_obj*, *p_obj*, i.e., correspondences with subject, direct and indirect object, and/or *adjunct dependencies* labeled *mod*, *ats*, etc., representing complementary information such as number, time, place, and so forth. Figures 5.6 and 5.7 illustrate popular dependency relations between main verb with other components in a sentence, i.e., the relation between the main verb and subject, main verb and objects or punctuation. We assign lexical category S$\backslash$NP to a main verb having a subject to its left, and then we add $/$NP (or $\backslash$NP, depending on its position with respect to the verb) *for each direct object or indirect object* (in the order of words in sentence).

Auxiliary verbs followed by other verbs get assigned the lexical category (S$\backslash$NP)$/$(S$\backslash$NP). For example in the sentence “Je voudrais prendre un rendez-vous pour demain” (I would like to make an appointment for tomorrow) (Fig. 5.8), the main verb is “prendre”. It has a subject, so

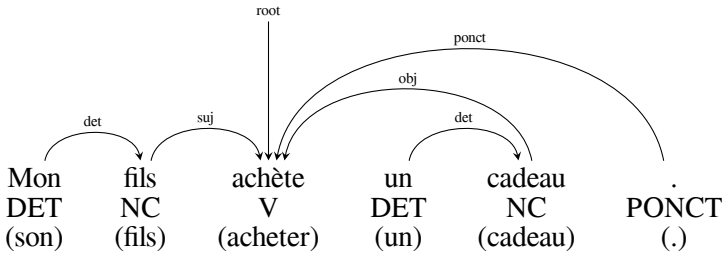


FIGURE 5.6: Dependency tree of the French sentence “My son buys a gift”.

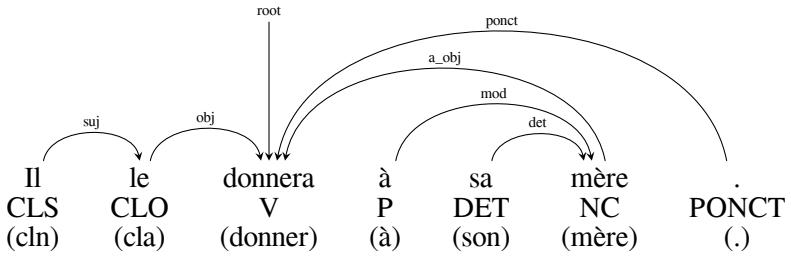


FIGURE 5.7: Dependency tree of the French sentence “He will give it to his mother”.

its lexical category must contain S\NP. Furthermore, it has a direct object dependency pointing to “rendez-vous”. Therefore, we assign lexical category (S\NP)/NP to it. The verb “voudrais,” being auxiliary, gets the lexical category (S\NP)/(S\NP).

POS tag	Example	Lexical categories
DET	un(e) (a,an), le la, l’, les (the)	NP/NP
ADJ	petit (small), pareil (similar), même (same)	NP/NP, NP\NP
NC, NPP	soirée (evening)	NP
ADV	vraiment (really), longuement (long)	(S\NP)/(S\NP), (S\NP)\(S\NP)
CL(R,O,S)	ce (this), se (-self), o n(they)	NP, NP/NP, (S\NP)/(S\NP)
PRO	il (he), nous (we)	NP
CS	que (what), comme (as)	NP, NP/NP, (S\NP)/(S\NP)
P+D,P, P+PRO	au (to), du (of), des (of), à (at), sur (on)	NP/NP, NP\NP, (NP\NP)/NP, (NP/NP)/NP, (S/S)/NP
PROREL	qui (who), que (what), où (where)	(NP\NP)/(S\NP), (NP\NP)/NP, (NP\NP)/S
(DET, ADV, ADJ, PRO)WH	quel (which), comment (how), quand (when)	(S/(S\NP), (S\NP)/(S\NP)
PREF	mi- (half), vice-	NP/NP, (NP/NP)(NP/NP)
I	oui (yes), non (no), bravo	NP

TABLE 5.1: Examples of lexical categories assigned to words

Lexical categories are iteratively assigned to the other words based on syntax functions and dependencies. For example, the noun words always get lexical category NP assigned to them, articles get (NP\NP). Categories (NP\NP) or (NP/NP) are assigned to adjectives depending on their position relatively to nouns. In Table 5.1, we list lexical category examples by word classes based on syntactic analysis and dependency structure. It should be noted that lexical categories assigned to words during this step can be replaced by others at later steps of the process of building the CCG derivation tree.

5.5.2 Binarization of Dependency Trees

Algorithm 5: Chunk extraction

ChunkExtraction $(chunk, sentence)$
Input: Chunk as part of dependency tree (full sentence at begin), dependency tree
Output: List of chunks in global variable $chunkedList$

```
chunkedList ← empty array;
mainVerb ← main verb of chunk;
headConnectedWords ← words connected to mainVerb;
for element ∈ headConnectedWords do
    elementList, newChunk ← empty array;
    add element into elementList and newChunk;
    while elementList is not empty do
        firstElement ← the first item in elementList;
        for word ∈ dependency tree do
            if word has dependency with firstElement then
                add word to elementList and newChunk;
        remove firstElement from elementList;
    add to chunkedList the result of applying recursively ChunkExtraction function
      to newChunk;
return chunkedList
```

Binarization of the dependency tree is performed on the basis of information about the dominant sentence structure for the specific language. In French, most sentences are SVO, as in "*Mon fils (S) achète (V) un cadeau (O)*" (My son buys a gift, cf. Fig. 5.6), or SOV as in "*Il (S) le (O) donnera (V) à sa mère (indirect O)*" (He will give it to his mother, cf. Fig. 5.7). Using this general linguistic property, we can extract and classify the components of the sentence into subjects, direct objects, indirect objects, verbs, and complement phrases.

The proposed algorithm for transforming a dependency tree into a binary tree consists of two steps. First, we extract chunks from the dependency tree by using Algorithm 5 which is based on syntactic information and dependency labels between words. For example, the subject chunk is obtained by finding a word that has a dependency labeled *suj*, the verb chunk corresponds to the root of the dependency structure, direct or indirect object chunks are obtained as words with links directed to the root verb and having labels *obj* or *p_obj*, etc. Next, we build a binary tree for each chunk, as described in Algorithm 6, and then combine the binary trees in the inverse order of the dominant sentence structure. For example if SVO is the dominant structure, we start by building the binary tree of the *object* chunk, then combine it with the binary tree of the

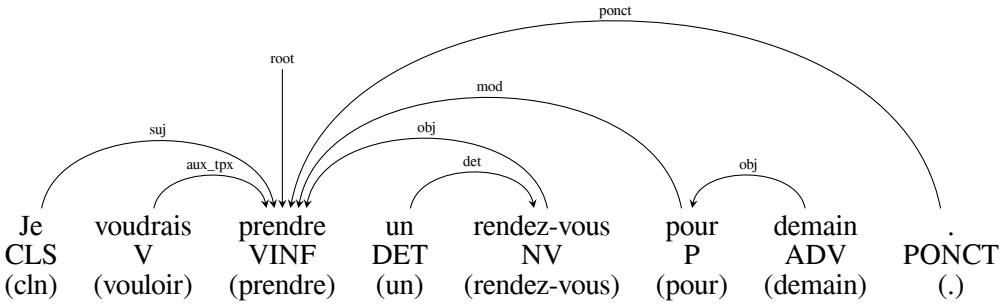


FIGURE 5.8: Dependency tree of the sentence "I would like to make an appointment for tomorrow"

verb chunk, and finally we obtain the binary tree of the *subject* chunk. In Fig. 5.9, the reader can see four chunk groups in the dependency tree, displayed as regions of the binarized tree.

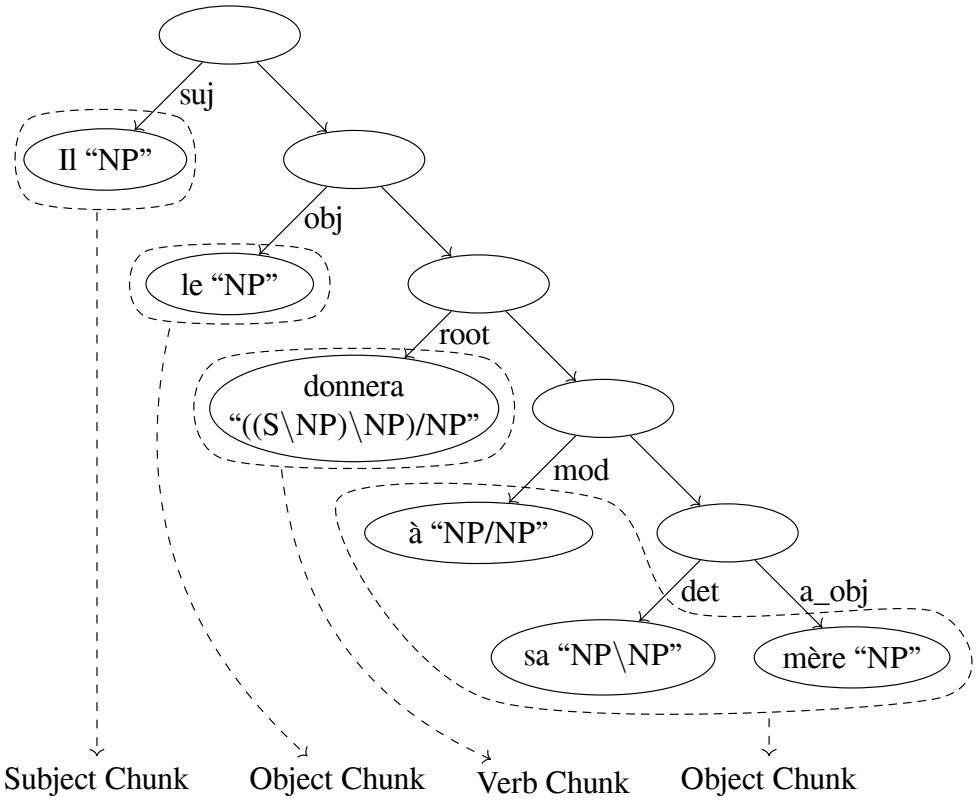


FIGURE 5.9: Chunks (in dotted lines) and binarized tree of sentence "He will give it to his mother"

5.5.3 CCG Derivation Tree Building and Validation

The final step is lexical category assignment verification by the construction of a complete CCG tree. The latter operation is performed iteratively applying combinatory rules. We start from the leaves of the binarized tree (see Fig. 5.9)—for which we have already lexical categories from Step 1—and move upwards by applying combinatory rules.

Combinatory rules usually require two input parameters in order to form a new lexical category except for the type-raising rule that requires a single input parameter. In the binary tree,

Algorithm 6: Tree binarization

BinarizationTree $(chunk)$
Input: Chunk list of sentence
Output: Binary tree

$bTree \leftarrow$ empty; $bChildTree \leftarrow$ empty;
$i \leftarrow$ get length of $chunk$;
while $i \geq 0$ **do**
  if $chunk[i]$ *is a list of words* **then**
    $bChildTree \leftarrow buildTreeWithWords(chunk)$;
  else
    $bChildTree \leftarrow$ perform a recursion on $BinarizationTree$ with $chunk[i]$;
  if $bTree$ *is empty* **then**
    $bTree \leftarrow bChildTree$;
  else
    Create a new temporary tree $bTempTree$;
    $rootNode$ is root node of $bTempTree$;
    Put $bTree$ to the right of $bTempTree$;
    Put $bChildTree$ to the left of $bTempTree$;
    $bTree \leftarrow bTempTree$;
    $bChildTree \leftarrow$ *empty*;
  $i \leftarrow i - 1$;
return *bTree*

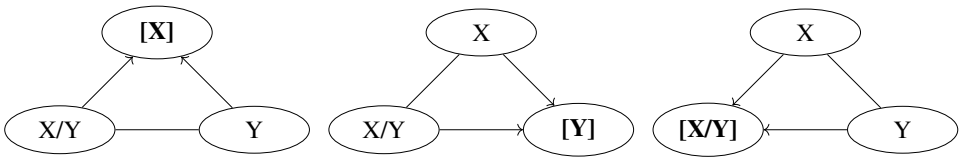


FIGURE 5.10: Inference rules for lexical categories

whenever two nodes have lexical category information assigned to them, we can infer the lexical category of the third node, as in Fig. 5.10. Using this method we move upwards, and if, after having applied all necessary combinatory rules, we obtain S as the root of the tree, then we validate the CCG derivation tree thus obtained. The result of obtaining this method for the tree of Fig. 5.9 can be seen in Fig. 5.11 (next to each node we have displayed the combinatory rule used to obtain its lexical category).

Some type-changing rules

In order to reduce the number of possible lexical category types, in cases such as bare noun phrases, type raising, clausal adjuncts, and sentence modifiers, we can use some unary type-changing rules:

$$\begin{aligned} N &\Rightarrow NP \\ S_{[\text{dcl}]}/NP &\Rightarrow NP\backslash NP \\ S_{[\text{others}]}\backslash NP &\Rightarrow NP\backslash NP \\ S\backslash NP &\Rightarrow S/S \end{aligned} \tag{5.9}$$

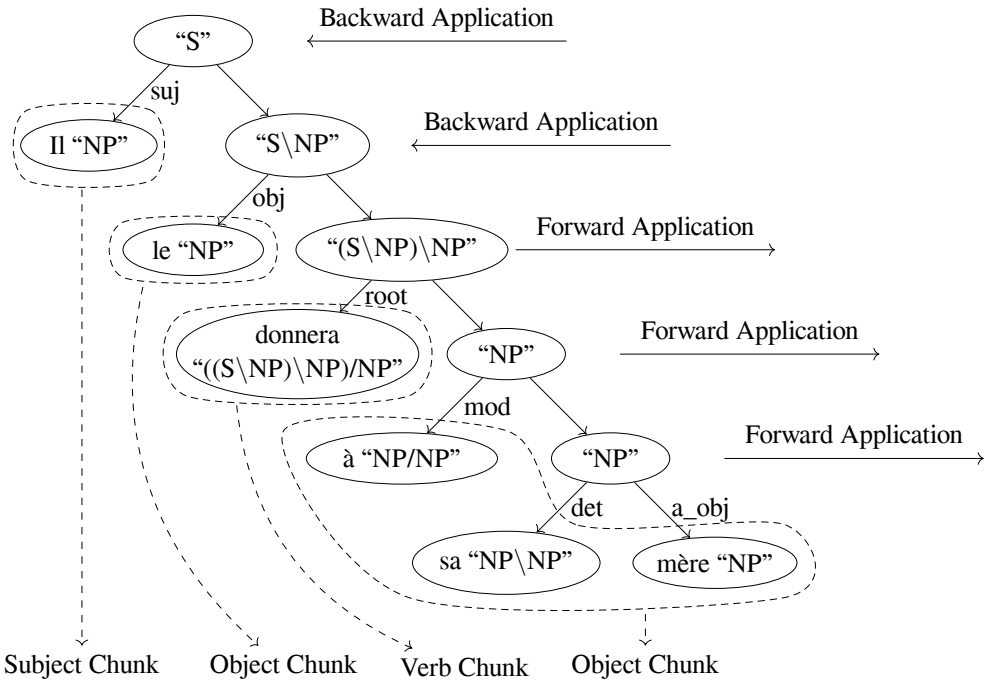


FIGURE 5.11: CCG derivation tree of the sentence "He will give it to his mother."

We can also use binary type-change rules in the case of words and punctuation:

$$\begin{array}{ccccc} NP & , & \Rightarrow & S/S \\ , & \text{NP} & \Rightarrow & S\backslash S \\ , & \text{NP} & \Rightarrow & (S\backslash NP)\backslash(S\backslash NP) \\ \text{X} & . & \Rightarrow & X \end{array} \tag{5.10}$$

In the following we will consider some special cases arising while building CCG derivation trees, such as coordination, subordination, wh-questions, topicalization, and negation.

Coordination construction

In coordination we connect similar components (e.g., nouns, verbs, adjective-nouns, or phrases). Here are some examples where We use label CC in the dependency structure for words *et* (and), *ou* (or), *mais* (but), and so forth:

- Between components, in an elliptical construction: *La France a battu [l'Argentine] (NP) et (CC) [la Belgique] (NP)* (France has beaten [Argentina] and [Belgium]).
- Between components of the same type: *[Le président adore] (S/NP) et (CC) [sa femme déteste] (S/NP) le foot* ([The president loves] and [his wife hates] football).
- Between components with different structures: *Henri [cuisine] (S\NP) et (CC) [écoute de la musique] (S\NP)* (Henri [cooks] and [listens music]).
- Between components without distribution: *[Henri] (NP) et (CC) [sa femme] (NP) gagnent exactement 2000 dollars ensemble* (Henri and his wife earn exactly 2000 dollars together).

Punctuation (e.g., comma or semicolon) can also be used in a way similar to coordination in listing similar components.

Following Hockenmaier and Steedman, 2007, we consider conjuncts as nodes in the binary tree and obtain new lexical categories by inference on adjacent nodes, as in the rules given in Fig. 5.12.

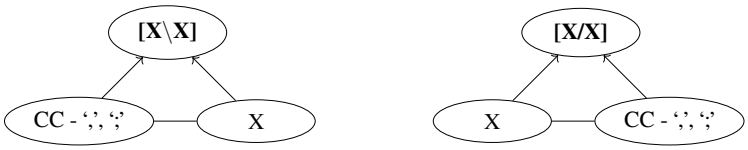


FIGURE 5.12: Coordination rules for lexical categories

Subordination construction

In French, there are three types of clauses: *independent*, *subordinate*, and *main*. Independent clauses stand alone. They are either complete sentences or are attached to other independent clauses through coordinating conjunctions, for example, "*[Henri est prêt] donc [on peut commencer]*" (Henri is ready so we can start). A main clause is the principal component of a sentence with subject and verb.

Let us examine the issue of subordination construction. A subordinate (or *dependent*) clause is used to provide supplementary information about the sentence and is introduced by a subordinating conjunction or a relative pronoun. In addition, it does not express a complete idea and cannot stand alone. For example, "Henri aime le gâteau [*que sa mère a acheté*]" (Henri likes the cake that his mother has bought). We divide subordinating clauses into two groups. The first one uses subordinating conjunctions (*comme* (as, since), *lorsque* (when), *puisque* (since, as), *quand* (when), *que* (that), *si* (if)), annotated with label CS. The second group uses relative pronouns (*qui* (who, what, whom), *que* (whom, what, which, that), *lequel* (what, which, that), *dont* (of which, from what, whose), *où* (when, where)), annotated with label PRO. Lexical categories are assigned according to context, for example (NP\NP)/(S\NP), (NP\NP)/(S/NP) is assigned to relative pronouns as in the case of English (cf. Steedman, 1996).

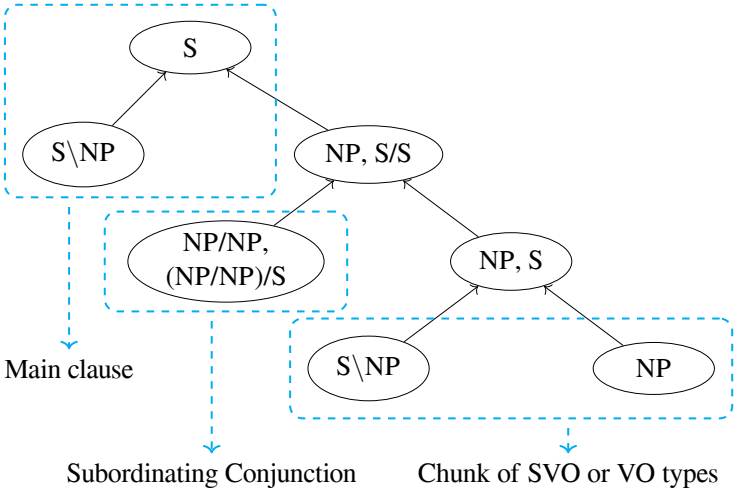


FIGURE 5.13: Subordination rules for lexical categories

Wh-questions construction

French questions starting with interrogative words can take a wide variety of forms. In general, we consider four basic patterns (Boucher, 2010):

- Wh-word in situ and verb in situ: *Vous allez où?* (You go where?)
- Wh-word raised, verb in situ: *Où vous allez?* (Where do you go?)
- Wh-word raised, verb raised: *Où allez-vous?* (Where go you?)
- Wh-word raised + *est-ce que*, verb in situ: *Où est-ce que vous allez?* (Where is it that you go?)

In our approach, interrogative words are separated from the sentence as in Fig. 5.14. Depending on their position in the sentence (head or last), lexical categories assigned to them are $S_{[wh]}/S$, $S_{[wh]}/(S\backslash NP)$, $S\backslash S_{[wh]}$, or $(S\backslash NP)\backslash S_{[wh]}$. We treat the rest of the sentence in the standard way.

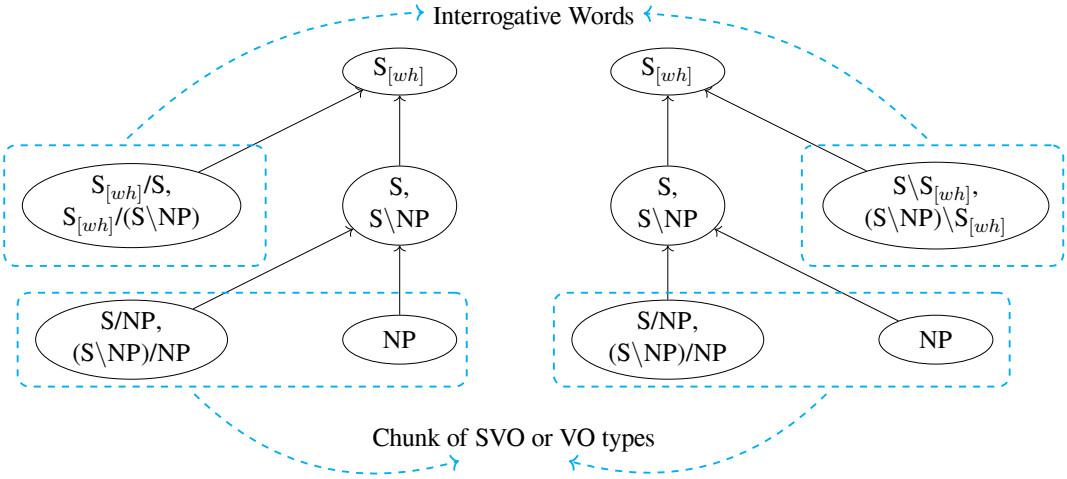


FIGURE 5.14: Wh-question rules for lexical categories

Topicalization construction

Like in English, topicalization concerns expressions where a clause topic appears at the front of a sentence or clause, for example, "Ce livre, je l'aime vraiment" (This book, I really like it). The component before the verb and subject is labeled MOD and can be considered as a sentence or noun phrase. Therefore, we use unary ad hoc type-changing rules such as N $\Rightarrow$ NP, S/NP $\Rightarrow$ NP\NP, S\NP $\Rightarrow$ NP\NP and S\NP $\Rightarrow$ S/S, or binary ad hoc type-changing rules involving punctuation, such as NP ',' $\Rightarrow$ S/S, ',' NP $\Rightarrow$ S\S, ',' NP $\Rightarrow$ (S\NP)\(S\NP), X '.' $\Rightarrow$ X, to transform the CCG derivation tree.

Negation structure construction

Negative sentences are formed by using two negative particles. The first one is *ne* or *n'* (no) and the second one can be *pas* (not) or some other particle such as *plus* (anymore), *rien* (nothing), *jamais* (never) or *personne* (nobody). Usually two negative words surround the conjugated verb, as in *Henri ne mange jamais de viande* (Henri never eats meat). In our process, negative words are filtered and placed in a verb chunk with the lexical categories that have been assigned using the adverb tag set.

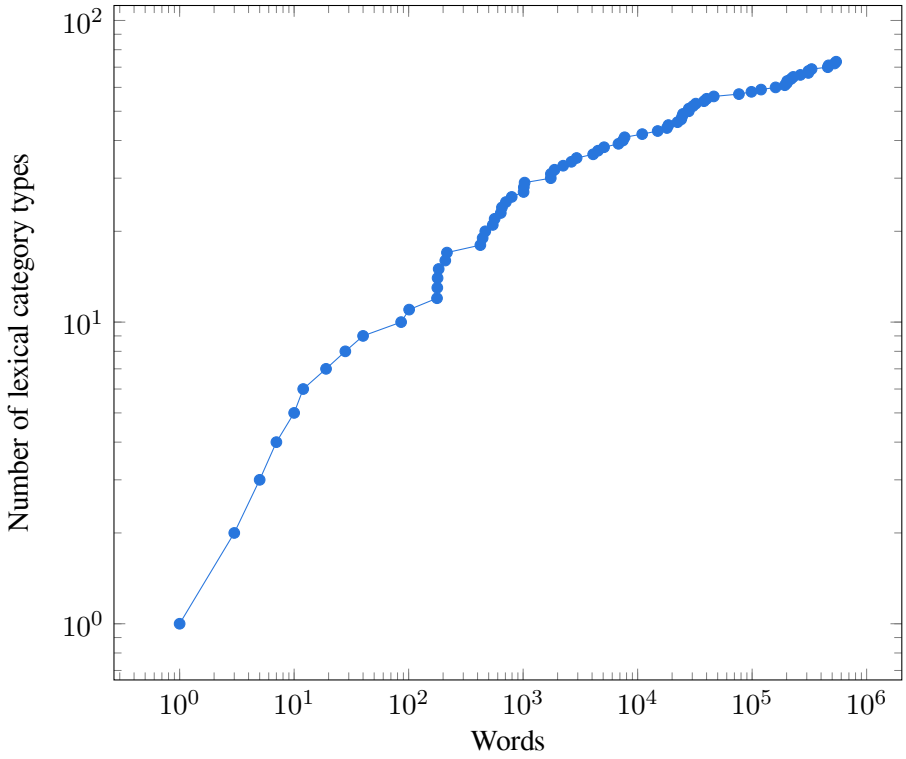


FIGURE 5.15: The growth of lexical category types

5.6 Experiments and Evaluation

In our experiment we used the French TreeBank corpus (FTB) () that contains a million words taken from annotated *Le Monde* news articles from the period 1991–1993. We used the dependency-based version, as in (Candito, Crabbé, and Denis, 2010; Candito et al., 2009). By applying our method to the complete set of 21,550 dependency trees of the corpus, we obtain CCG derivation trees for 94,02 % of the sentences.

All in all, we obtain a set of 73 distinct lexical categories for the complete French corpus. More specifically, we see that the number of lexical categories increases rapidly during the 10.000 first sentences (Fig. 5.15) and that, as expected, there are less newly generated lexical categories towards the last sentences in the corpus. In addition, lexical categories NP/NP and NP, which correspond to articles, adjectives and nouns, are assigned to more than half of the words (Fig. 5.16). In Table 5.2, the reader can see the list of French words with the highest number of distinct lexical categories—notice that the four verbs of the table (*être*, *avoir*, *faire*, *voir* are used both as main and as auxiliary verbs).

Another experiment has been done on a small data set of sentences and the dependency structure obtained through the MaltParser tool. Out of 360 sentences, we obtained only 83% of completed sentences with lexical categories. This result shows that the accuracy of our approach strongly depends on dependency parsing.

5.7 Error analysis

The failure rate of our approach is high when compared to results in other languages, such as 99.44% of successfully parsed sentences in English or 96% in Hindi. There are three main causes of error: incorrect POS tags, incorrect dependencies or dependency labels, and finally errors resulting from complex linguistic issues like sentential gapping, parasitic gaps, etc.

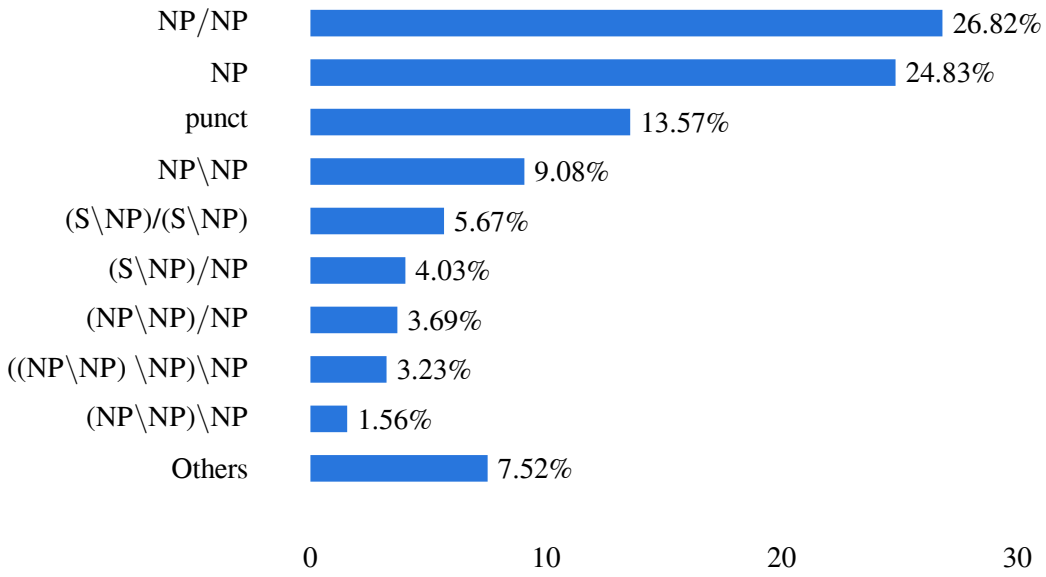


FIGURE 5.16: Distribution of CCG lexical categories

Incorrect POS tags are the main cause leading to erroneous results. These range from low-level errors of dependency parsers to phrases that are inherently ambiguous and can have more than one syntax trees, such as *la belle porte le voile* where *belle*, *porte*, and*voile* can be both noun/verb/noun or adjective/noun/verb.

Dependency relation errors affect the chunking and binarization process, and this results in erroneous lexical categories.

Finally, complex linguistic issues arise when processing of utterances where omission of a word or of a group of words—which otherwise are necessary for the grammatical completeness of a sentence—is tolerated. These issues often result in incorrect identification of verb arguments. For example, in *Henri veut aller au parc et sa mère à la bibliothèque* (Henri wants to go to the park and his mother to the library), the absence of a verb between words *mère* and *à la bibliothèque* results in obtaining incorrect lexical categories for the remaining words.

#	Word	#Lex.Categories	Frequence
1	à (at)	23	15,566
2	être (be)	23	9,715
3	de (of)	22	49,720
4	que (what)	21	5,288
5	avoir (have)	21	7,742
6	comme (as)	20	803
7	pour (for)	20	3925
8	faire (do)	19	1,173
9	voir (see)	18	333
10	en (in)	18	8,081

TABLE 5.2: Words with the highest number of lexical categories and their frequencies

5.8 Conclusion of the Chapter

We have presented a novel approach for building CCG derivation trees by the use of dependency structure information. Our approach involves chunking and binarization of dependency trees. Lexical category assignment is validated by combining lexical categories moving upwards in the binarized tree, and by verifying that the lexical category S is obtained at the root of the tree. We have applied our method on the French Treebank corpus and obtained a 94,02% success rate of validated derivation trees. We expect the obtained French CCG Treebank to be a useful resource for machine learning or deep learning algorithms predicting syntax in the CCG framework.

Chapter 6

CCG Supertagging Using Morphological and Dependency Syntax Information

In the previous chapter we described the creation of the French CCG Tree Bank corpus (20,261 sentences) out of the French Tree Bank corpus by . In order to improve the assignment of lexical categories to words, we have developed a new CCG supertagging algorithm based on morphological and dependency syntax information. We then used the French CCG Tree Bank Corpus, as well as the Groningen Tree Bank corpus for the English language, to train a new BiLSTM+CRF neural architecture that uses (a) morphosyntactic input features and (b) feature correlations as input features. We show experimentally that for an inflected language like French, dependency syntax information allows significant improvement of the accuracy of the CCG supertagging task when using deep learning techniques (Le and Haralambous, 2019).

6.1 Introduction

CCG supertagging plays an important role in parsing systems, as a preliminary step to the build of complete CCG derivation trees. In general, this task can be considered as a sequence labeling problem with input sentence $s_{\text{input}} = (w_1, w_2, \ldots, w_n)$ and the CCG supertags $s_{\text{output}} = (t_1, t_2, \ldots, t_n)$ as output. Input features can be words or features extracted from words, such as suffix, capitalization property or a selection of characters (Lewis, Lee, and Zettlemoyer, 2016; Wu, Zhang, and Zong, 2017; Xu, 2016; Ambati, Deoskar, and Steedman, 2016; Kadari et al., 2018b). We will use morphosyntactic annotations such as lemma, suffix, POS tags and dependency relations (Honnibal, Kummerfeld, and Curran, 2010) to build feature sets. These annotations are extremely useful in order to add additional information about words as well as long-range dependencies in the sentence (Figure 6.1). These novel features allow us to improve accuracy of a supertagging neural network. We also consider adding correlations between features as additional input features of the network and examine the results.

In the past few years, Recurrent Neural Networks (RNN) (Goller and Kuchler, 1996) along with their variants such as Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers, Schmidhuber, and Cummins, 1999) and GRU (Cho et al., 2014) have been proven to be effective for many NLP tasks, and especially for sequence labeling such as POS tagging, Named Entity Recognition, etc. In the CCG supertagging task, different RNN-based models have been proposed and have obtained high accuracy results. Following this trend, we base our model on the Bi-Directional LSTM (BiLSTM) architecture associated with Conditional Random Fields (CRF) as output layer. Thus, we take advantage of the ability to remember the information of previous and next words in the sentence with the BiLSTM network and increase the ability to learn from the relationship of output labels with CRF.

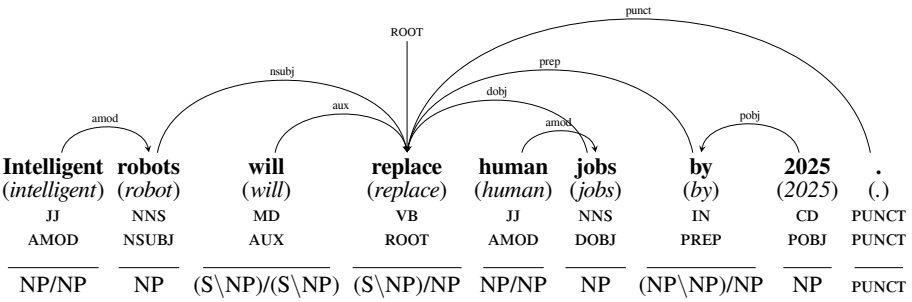


FIGURE 6.1: A sentence with POS tags, dependencies and CCG lexical categories.

6.2 State-of-the-art of the CCG Supertagging Task

One of the first applications of machine learning to CCG supertagging is the development of a statistical parser by Clark and Curran, 2007. They proceed in two main steps: supertagging and combination of lexical categories. Their supertagging approach is based on the log-linear model by using the lexical category set in a local five-word context to obtain a distribution. The model's features are POS tags of words included in the five-word window, plus the two previously assigned lexical categories (to the left). They applied their method on the CCG Bank corpus (Hockenmaier and Steedman, 2007) with 92.6% of accuracy for words and (only) 36.8% of accuracy for complete sentences (that is the percentage of sentences of which *all* words are tagged correctly).

Like many others supervised methods, CCG supertagging requires a sufficiently large amount of labeled training data to achieve good results. Lewis and Steedman, 2014b have introduced a semi-supervised approach to improve a CCG parser with unlabeled data. They have constructed a model for the prediction of CCG lexical categories, based on vector-space embeddings. Features are words and some other information (e.g., POS tagging, chunking, named-entity recognition, etc.) in the context window. Their experiments use the neural network model of (Collobert, 2011) in association with conditional random fields (CRF) (Turian, Ratinov, and Bengio, 2010).

Using RNN for CCG supertaging has been proven to provide better results with a similar set of features and window size in the work of Xu, Auli, and Clark, 2015. However, the conventional RNN is often difficult to train and there still exist problems such as gradient vanishing and exploding, in the layers over long sequences (Bengio, Frasconi, and Simard, 1993; Pascanu, Mikolov, and Bengio, 2013). Therefore, LSTM networks—a special variant of RNNs capable of learning long-term dependencies—were proposed to overcome these RNN limitations. In particular, Bi-directional LSTM network models have been created with the ability to store two-way information, and the majority of literature in the area (Lewis, Lee, and Zettlemoyer, 2016; Wu, Zhang, and Zong, 2017; Xu, 2016; Ambati, Deoskar, and Steedman, 2016; Kadari et al., 2018b) use this model with different training procedures and achieve high accuracy.

The performance of BiLSTM network models has been improved by combining them with a CRF model for the sequence labeling task (Huang, Xu, and Yu, 2015; Steedman and Baldridge, 2011; Ma and Hovy, 2016). Using a BiLSTM-CRF model similar to the one in (Huang, Xu, and Yu, 2015), Kadari et al., 2018a have shown the efficiency of CRF by achieving a higher accuracy in CCG supertagging and multi-tagging tasks.

In most of the above works, similarly to many sequence labeling tasks, the model inputs are words and their features are extracted directly from words. However, we claim that lexical categories assignment to words can use morphological and dependency syntax to enrich the

feature set. In the following section, we present a neural network model based on BiLSTM-CRF architecture with moprhosyntactic features.

6.3 Neural Network Model

6.3.1 Input Features

We will use the following input features for words in sentences:

- the *word* per se (word);
- the *word lemma* (lemma);
- the *POS tag* of the word (postag);
- the *dependency relation* (deprel) of the word with its parent in the dependency tree (and the tag "root" for the head of the dependency tree, which has no parent).

Each one of these features provides predictive information about the CCG supertag label. Therefore, our input sentence will be $s = \{x_1, x_2, \ldots, x_n\}$ where each x_i is a vector of the features $x_i = [\text{word}_i, \text{lemma}_i, \text{postag}_i, \text{deprel}_i]$.

Before describing our model, let us briefly review pre-existing models.

6.3.2 Basic Bi-Directional LSTM and CRF Models

Unidirectional LSTM Model

As mentioned earlier, the shortcomings of standard RNNs in practice involve gradient vanishing and an explosion problem when dealing with long term dependencies. LSTMs are designed to cope with these gradient problems. Basically, a conventional RNN is defined as follows: the input $x = (x_1, x_2, \ldots, x_T)$ feeds the network, and the network computes the hidden vector sequence $h = (h_1, h_2, \ldots, h_T)$, and the output sequence, $y = (y_1, y_2, \ldots, y_T)$, from $t = 1, \ldots, T$ where T is the number of time steps as in the following formulas:

$$h_t = f(Ux_t + Wh_{t-1} + b_h) \tag{6.1}$$

$$y_t = g(Vh_t + b_y), \tag{6.2}$$

where U, W, V denote weight matrices that are computed in training time, b denotes bias vectors and $f(z)$, $g(z)$ are activation functions as follows.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{6.3}$$

$$g(z) = \frac{e^z}{\sum_k e^{z_k}}. \tag{6.4}$$

Based on basic RNN architecture, an LSTM layer is formed from a set of memory blocks (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005). Each block contains one or more recurrently connected memory cells and three gate units: input, output and "forget" gate. More specifically, activation computation in a memory cell at time step t is defined by the

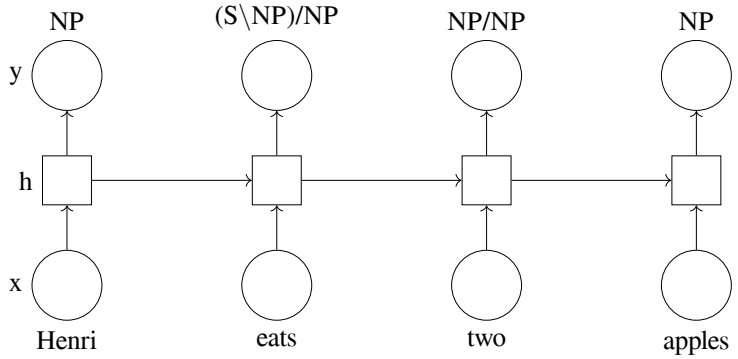


FIGURE 6.2: A simple RNN model

following formulas:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{6.5}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{6.6}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \tag{6.7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{6.8}$$
$$h_t = o_t \odot \tanh(c_t), \tag{6.9}$$

where i_t, f_t, o_t, c_t correspond to input gate, forget gate, output gate and cell vectors, σ is the logistic sigmoid function, tanh is the hyperbolic tangent function, W terms denote weight matrices, and b terms denote bias vectors.

Figure 6.2 illustrates the simple RNN model that includes an input layer x, a hidden layer h and an output layer y. For the CCG supertagging task, the input feature x consists of word embedding vectors, the output layers provides CCG lexical categories.

Bidirectional LSTM Model

In order to assign a supertag to a word, we need to use the word's information and its relations to the previous and next word in the sentence. Two-way information access from past to future and vice-versa gives global information in a sequence. However, the LSTM cell only retrieves information from the past using input and output of the previous LSTM cell. In other words, an LSTM cell does not receive any information from the LSTM cell *following* it. Therefore, a Bi-Directional LSTM (BiLSTM) model has been proposed in (Schuster and Paliwal, 1997; Baldi et al., 2000) to overcome this problem, as follows:

$$\text{Bi-LSTM}_{\text{sequence}}(x_{1:n}) := \text{LSTM}_{\text{backward}}(x_{n:1}) \circ \text{LSTM}_{\text{forward}}(x_{1:n}). \tag{6.10}$$

In general architectures, one may have one forward LSTM layer and one backward LSTM layer for the complete sequence and run them in reverse time. The features of the two layers are concatenated at the level of the output layers. Thus, information from both the past and the future is transmitted to each memory LSTM cell. The hidden state is computed as follows:

$$h_t := f(W_{\overleftarrow{h}}\overleftarrow{h_t} + W_{\overrightarrow{h}}\overrightarrow{h_t}), \tag{6.11}$$

where $\overleftarrow{h_t}$ is the backward hidden sequence and $\overrightarrow{h_t}$ is the forward hidden sequence. As in the illustration in Figure 6.3, we can access both past and future input features for a given word in the BiLSTM model.

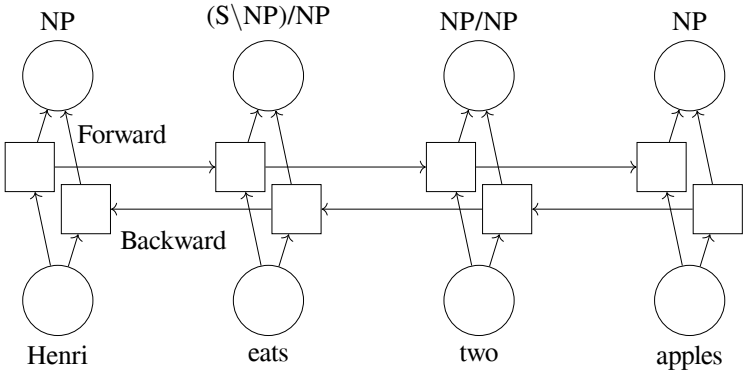


FIGURE 6.3: A bidirectional LSTM model

CRF Model

BiLSTM networks are used to build efficient predictive models of the output sequence based on the features of the input sequence. However, they cannot consider the correlation between output labels and their neighborhoods. In our case, CCG supertags, by nature, *always* have correlations with the previous or next labels, for example, an output CCG supertag of a word is NP/NP (usually an article), which allows us to predict the fact that the next CCG supertag is NP. Figure 6.4 shows a simple CRF model—note that the feature input and output are directly connected, as opposed to LSTM or BiLSTM model network where memory cells are employed.

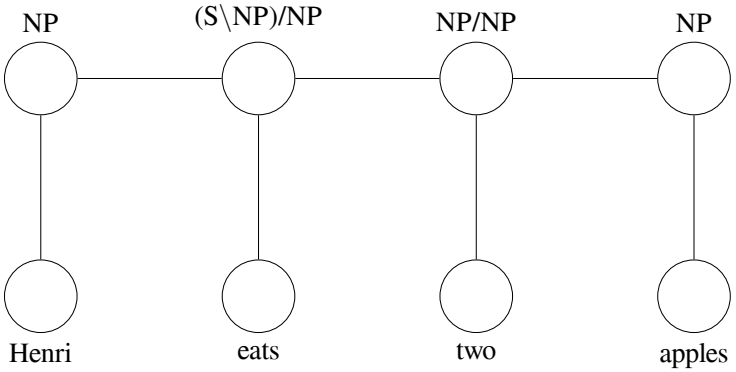


FIGURE 6.4: A CRF model

In order to enhance the ability of predicting the next label from the current label in an output sequence, two approaches can be used:

1. building a tag distribution for each training step and using an heuristic search algorithm to find optimal tag sequences (Vaswani et al., 2016), or

2. focusing on the context using sentence-level information instead of only word-level information. The leading work of this approach is the CRF model of (Lafferty, McCallum, and Pereira, 2001).

We use the second approach in the output layer of our model. The combination of BiLSTM network and CRF network can improve the efficiency of the model by strengthening the relationship between the output labels through the CRF layer, based on the input features through the BiLSTM layer.

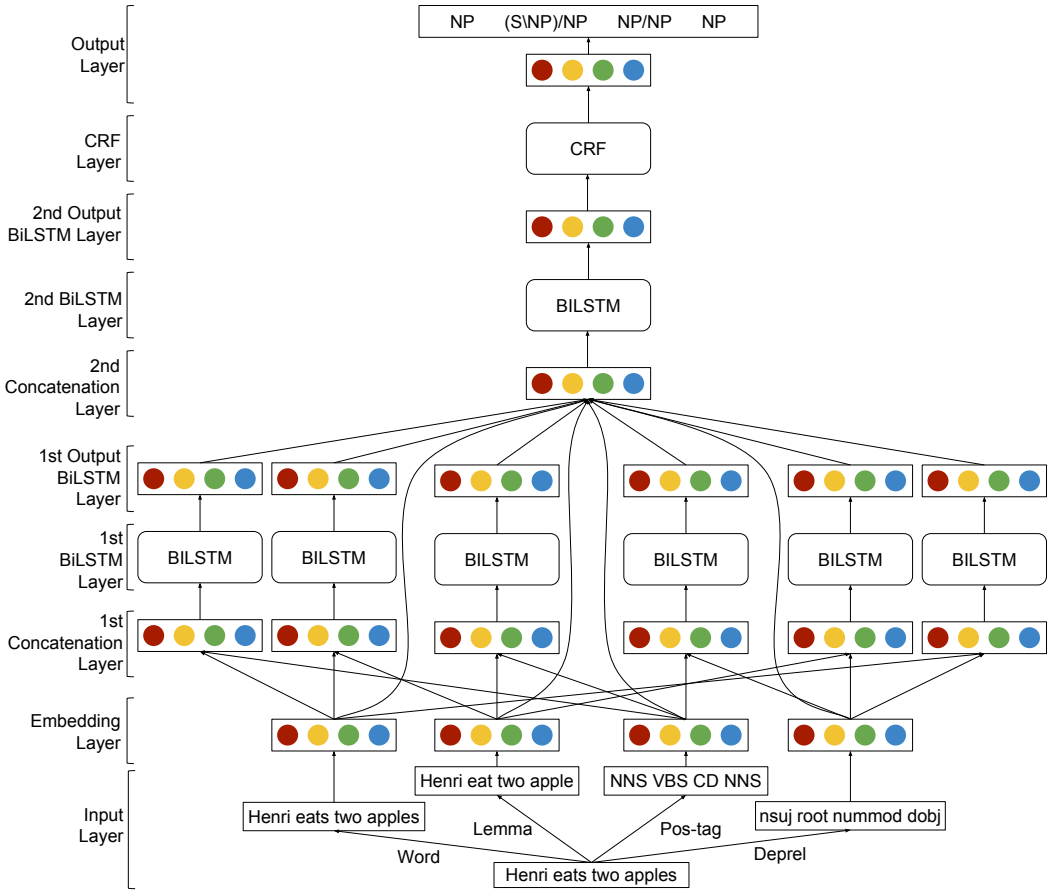


FIGURE 6.5: Architecture of the BiLSTM network with a CRF layer on the output

6.3.3 Our Model for Feature Set Enrichment and CCG Supertagging

In our model (see Fig. 6.5), each input is a set of features: word, lemma, POS tag and dependency relation. These features are vectorized with a fixed size by using the embedding matrix in the embedding layer. In the next layer, the correlation between pairs of features is calculated by combining them. Then, we use a BiLSTM network to memorize and learn the relations with other words in the context of the sentence for these pairs of features. After that, all features are concatenated to become the input of the second BiLSTM network layer. Finally, CCG supertag labels are obtained in the output of the CRF network layer, the input of which is the output of the 2nd output BiLSTM layer.

6.4 Evaluation

6.4.1 Data Set and Preprocessing

We use two different corpora to experiment our model, one in English and one in French. The first corpus is the Groningen Meaning Bank (GMB) corpus Bos et al., 2017 which has been built for deeper semantic analysis on a discourse scope. The second one is our CCG Corpus for French which we extracted from the French Tree Bank (FTP) corpus () by using the dependency analysis of the sentence (see Chapter 5).

In order to obtain a standard data set for the training process, we extract all sentences with annotations for each word such as lemma, POS tag, dependency relation and CCG label, for

TABLE 6.1: Statistics on the corpus

Statistic	GMB	FTB
Sentence	23 451	18 724
Word	1 037 739	570 054
Word token	32 073	28 748
Lemma token	26 987	18 762
POS tagset	43	29
Dependency Label	56	27
CCG Label	636	73

each corpus. As the GMB corpus does not contain dependency relations, we have used the Stanford Parser (De Marneffe, MacCartney, Manning, et al., 2006) to add them a posteriori.

We compare the structures of the two corpora in Table 6.1. In particular, there is a difference in the distribution of sentences according to their length (see also Fig. 6.6). In the GMB corpus, the number of short sentences and long sentences is relatively similar. This is quite different in the FTB corpus where there are more short sentences, and where long sentences spread over a wider range. This difference of the distribution rate in the data sets can affect the training process outcomes of the two corpora.

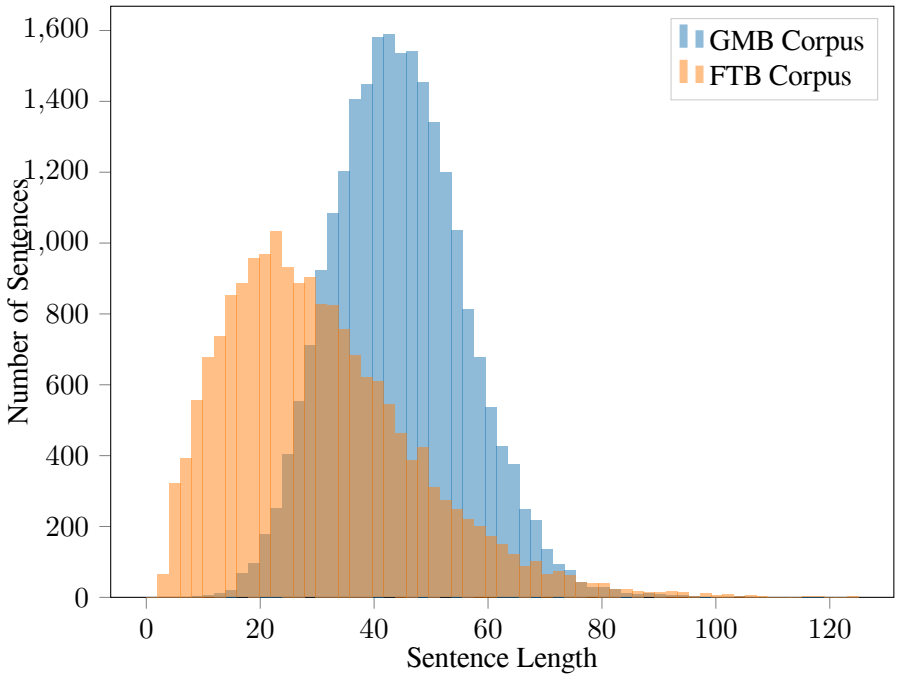


FIGURE 6.6: Histogram of sentence lengths in the corpora

6.4.2 Training procedure

We implement our neural network by using the Keras deep learning library (Chollet, 2018). The data sets are divided into three sets: training set, validation set and test set with the proportion

$$0.8 * (0.8 * (\text{training\_set}) + 0.2 * (\text{validation\_set})) + 0.2 * (\text{test\_set}).$$

The validation set is used to measure performance at each epoch. Final evaluation on the test set is based on the best accuracy results on the validation set.

Pre-trained Word Embeddings

In order to work with numeric data in the neural network, we use pre-trained word embeddings to transform words or lemmas of the corpora into numeric vectors. More specifically, we use Glove (Pennington, Socher, and Manning, 2014) (a 200-dimensional embedding trained on 6 billion words collected from Wikipedia) for the GMB corpus, and Word2vec (Mikolov et al., 2013) (the French version by Fauconnier (Fauconnier, 2015), also with 200 dimensions, and trained on 1.6 billion words collected from the web) for the FTB corpus. For out-of-vocabulary words, we assign embeddings by random samples. Based on the distribution by length of sentences (Fig. 6.6), we assign a fixed length of 120 words to all sentences, so that the input dimension is $[120, 200]$ for each sentence input. Finally, the other features are transformed to numeric vectors by using a one-hot encoding matrix with size depending on their number in the dictionary.

Parameters and Hyperparameters

We fix the number of training examples (batch size) as 32 for each forward or backward propagation. Each training process runs 20 times to evaluate and compare outcomes (epoch). In addition, we have experimented with the number of different hidden states, such as 64, 128, 256, 512 to find a configuration that is optimally consistent with the model. We decided to carry on the experiment with 128 hidden states because this choice optimally balances accuracy and performance.

Optimization algorithm

Choosing an optimizer is a crucial part of the model building process. Our model uses the Root Mean Square Prop (RMSprop) optimizer (Tieleman and Hinton, 2012) which proceeds by keeping an exponentially weighted average of the squares from past gradients. To increase convergence, the learning rate is divided by this average:

$$v_{dw} := \beta v_{dw} + (1 - \beta) \cdot dw^2, \tag{6.12}$$

$$v_{db} := \beta v_{db} + (1 - \beta) \cdot db^2, \tag{6.13}$$

$$W := W - \alpha \frac{dw}{\sqrt{v_{dw}} + \epsilon}, \tag{6.14}$$

$$b := b - \alpha \frac{db}{\sqrt{v_{dw}} + \epsilon}, \tag{6.15}$$

where v_{dw} v_{db} are the exponentially weighted averages from past squares of gradients, dw^2 and db^2 are cost gradient related to the current layer weight, W and b denote weight and bias, α is the learning rate from 0.9 to 0.0001 ($\alpha = 0.01$ is the default setting), β is an hyperparameter to be tuned and ϵ is very small to avoid dividing by zero.

6.4.3 Experimental Results

In order to evaluate the proposed input features and the model, we conduct two sorts of comparisons:

1. we compare the outcomes of different feature sets such as [word], [word, suffix], [word, suffix, cap(italization)], [word, lemma, suffix, cap], [word, lemma, postag, suffix, cap], [word, lemma, postag, deprel, suffix, cap], [lemma, postag, deprel], [lemma, postag], [lemma];

2. we compare the outcomes of different neural network architectures such as BiLSTM (Lewis, Lee, and Zettlemoyer, 2016), standard BiLSTM CRF (Huang, Xu, and Yu, 2015), Double-BiLSTM CRF (Kadari et al., 2018a), and ours.

Evaluation on our test set is shown on Table 6.2 for the French FTB corpus and on Table 6.3 for the English GMB corpus.

Because of the architecture chosen and since we use correlations of features as additional features, our model requires at least two features in the input data. Therefore, we can not produce results on input data with a single feature like [word] or [lemma]. Nevertheless we compare the outcome of other models on our input features, including single input features.

TABLE 6.2: 1-best tagging accuracy comparison results on the test set in the French FTB Corpus

Feature set	BiLSTM	BiLSTM CRF	Double BiLSTM CRF	Our model
word	*78.60*	*78.76*	*77.14*	-
word, suffix	*78.97*	*78.80*	*76.58*	78.90
word, suffix, cap	*78.56*	*78.97*	***75.96***	84.43
word, lemma, suffix, cap	79.16	79.67	78.78	78.49
word, lemma, postag, suffix, cap	81.28	81.84	81.24	81.50
word, lemma, postag, deprel, suffix, cap	83.23	83.95	83.56	84.06
word, lemma, postag, deprel	83.43	83.98	83.70	**85.05**
lemma,postag,deprel	83.00	83.05	83.15	82.40
lemma,postag	80.20	80.37	81.40	80.05
lemma	77.61	77.83	76.66	-

Let us first start with the French corpus. In Table 6.2 we have displayed methods from the literature in italics: word, suffix and cap(italization) as input features, BiLSTM, BiLSTM+CRF and Double BiLSTM+CRF as architectures. As the reader can see, by applying pre-existing methods we obtain a maximum accuracy of 75.96%. By using our input features with pre-existing architectures we obtain a maximum accuracy of 83.98%. By using our architecture with input features used by others we get an accuracy of 84.43%. Both of these results are significantly better than those in previous works. Finally, by combining our input features with our model we manage to gain another 1% and achieve a topmost accuracy of 85.05%.

It is interesting to notice that, even though the lemma feature carries less information than the word feature (as expected), the combination of lemma and POS tag features provides better results than the word feature, and that these results are systematically increased by 2% when dependency relations are added as well.

TABLE 6.3: 1-best tagging accuracy comparison results on the test set in English GMB Corpus

Feature set	BiLSTM	BiLSTM CRF	Double BiLSTM CRF	Our Model
word	*92.83*	*92.49*	*91.16*	-
word, suffix	*93.08*	*92.93*	*91.57*	92.92
word, suffix, cap	***93.30***	*93.20*	*91.48*	**94.31**
word, lemma, suffix, cap	93.33	93.26	91.78	93.38
word, lemma, postag, suffix, cap	93.29	93.02	93.25	92.44
word, lemma, postag, deprel, suffix, cap	93.45	93.18	93.15	92.46
word, lemma, postag, deprel	93.35	93.18	93.24	92.90
lemma, postag, deprel	93.25	93.13	92.98	93.26
lemma, postag	93.21	93.12	92.98	92.95
lemma	90.56	90.13	89.86	-

The accuracy results for the English GMB corpus are displayed on Table 6.3. Here differences are less significant, and the results all lie in the 92–94% range, with a single exception: the case of the lemma feature, where we lose about 2% of accuracy. Nevertheless, when we add the POS tag feature to the lemma feature, we get a slightly better result than the word feature (the difference is about 1%). The best result is an accuracy of 94.31%, obtained by our model, but not with our morphosyntactic features but rather with the legacy word, suffix and cap(italization) features.

A general conclusion could be that morphosyntactic information brings a real advantage for neuronal supertagging of French (a language the verbs of which are highly inflected). It would be interesting to test the model with even more inflected languages such as German, Russian or Greek.

6.5 Related CCG Parsing Works

In the previous chapter, we described our method of obtaining a CCG derivation tree from a sentence with its dependency structure. However, we obtain only a single CCG derivation tree from a given input. In this section, we focus on approaches to construct CCG parser systems that employ a CCG parsing model. In addition, we will see how to parse by using results inherited from CCG supertagging.

We will investigate the task of recovering labeled predicate-argument dependencies from pairs of lexical categories and words obtained through the CCG supertagging task. Numerous CCG parsing systems have been developed for English and some other languages, based on CKY-style parsing algorithms, on parsing via planning shift-reduce algorithms with beam search, or on the A* algorithm. These approaches rely on the bottom-up approach, in which the results of the supertagging task are used as a starting point of the parsing process. Combinatory rules are used to combine valid constituents in order to generate new constituents. The process continues until achievement of valid CCG derivation trees.

6.5.1 Chart parsing algorithms

The CKY algorithm is a popular choice for constituency parsing. Similarly, various CCG parsing system have employed this algorithm to obtain CCG derivation parse tree with an $\mathcal{O}(n^3)$ worst case time complexity. The CKY algorithm is essentially a bottom-up algorithm and begins by combining adjacent words to obtain a span of size two. It then carries on in combining adjacent spans of size three in a higher row. The algorithm terminates when spans covering the whole sentence are reached.

More specifically, given a sentence with n words corresponding with pos $\in \{0, 1, \ldots, n-1\}$. A span is a unit of length of words, e.g., in the sentence "Henri emprunte des livres aux Capucins" (Henri borrows books from the Capucins), the span of the phrase "aux Capucins" is 2, while its position pos is 4 (see Figure 6.7). A set of derivations is defined as the result achieved by parsing each valid (pos, span) pair. The derivations for (pos,span) is then combined with the derivations in (pos, i) and $(\text{pos}+i, \text{span}-i)$ for all $i \in \{1, \ldots, \text{span}-1\}$. By this way, we can parse a sentence by finding the derivations that span the whole sentence $(0, n)$ recursively by finding derivations in $(0, i)$ and $(i, n-i)$ for all $i \in \{1, \ldots, n-1\}$. In order to stay in the limits of polynomial time for the parsing process, we use a dynamic programming approach with a chart data structure to store the derivations for each (pos, span) which is evaluated and can be reused. The chart data structure includes two dimensional array indexed by pos and span, the valid pairs corresponding to pos + span $\leq n$.

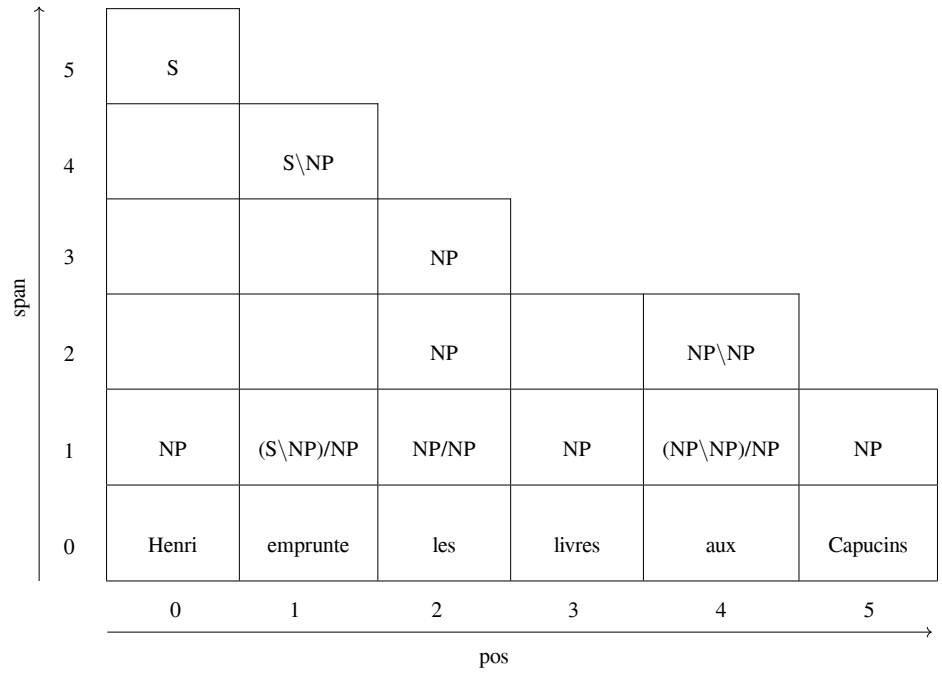


FIGURE 6.7: CKY table parsing for the sentence "Henri borrows books from the Capuchins"

The C&C parser (Clark and Curran, 2003; Clark and Curran, 2004; Djordjevic and Curran, 2006; Djordjevic, Curran, and Clark, 2007) is one of the complete implementations based on CKY-style parsing algorithms to obtain CCG derivation trees from given sentences. This system employs a chart-parsing algorithm along with repair on the chart that reduces the search space based on reusing the partial CKY chart from the previous parsing step. Besides, the use of constraints also improves parsing efficiency by avoiding the construction of useless sub-derivations. For example, the punctuation constraints that apply on semicolons, colons or hyphens reduce the sentence into a set of smaller units that significantly improves the time using for parsing, as well as the accuracy of the parsing result.

The NLTK CCG Parser is another complete CCG parser for English, first developed by (Gange, 2008). The current version of the NLTK parser, implemented by (Nakorn, 2009), has expansions to support feature unification, semantic derivation or probabilistic parsing. The parsing algorithm is build on the packed-chart approach (Clark and Curran, 2004), which is an improvement of the chart parsing. The basic ideas behind the packed chart approach is based on grouping edges. In particular, edges having the same category in the same cell are grouped together. Thus, the parsing algorithm processes the packed edge instead of all edges that have similar structure and category. This helps to improve parsing efficiency up to 10 times (Haoliang et al., 2007).

The OpenCCG Parser[1] (White and Baldridge, 2003) is developed as a component of the OpenNLP toolkit, which is an open source NLP library implemented in Java language. This parser used Multi-Modal CCG, an extension of the CCG version by Mark Steedman using modalities and hierarchies of lexical categories (Baldridge and Kruijff, 2003). Essentially, the parsing algorithm is constructed on a bottom-up chart and an agenda that ranks partial parse solutions based on an n-gram measure (Varges and Mellish, 2001). The first partial parse item in the agenda is fed to the chart in the parsing process.

The StatCCG Parser (Hockenmaier, 2003) uses a standard chart parsing algorithm for probabilistic grammars, in which conditional probability is used to measure derivation parse trees. In particular, the probability of partial derivation parse trees is computed as the product of the relevant probabilities such as expansion, head, non-head and lexical probability. The final probability of a complete derivation tree is the product of the probabilities of each partial derivation parse tree in this tree. Moreover, beam search and dynamic programming are used to identify the constituents within a cell of the chart that have the highest probability parse. These strategies guarantee an efficient parsing process with wide-coverage grammar.

6.5.2 CCG parsing via planning

Answer Set Programming (ASP))—a declarative programming paradigm—may be used to obtain all CCG derivation parse trees for a given sentence, based on a prominent knowledge-representation formalism. The basic idea of ASP is to represent a computational problem by a program whose answer set contains solutions, and then employs an answer-set solver to generate answer-sets for the program (Lierler and Schüller, 2012). The search of all CCG derivation parse trees for a given sentence is regarded as a planning problem. Thus, we can consider this task as answer-set programming.

In order to realize a CCG parsing-via-planning problem, we need to declare states and actions of the program. States are defined as Abstract Sentence Representations (ASR) of sentences, whereas actions are annotated combinators. More specifically, by going through the supertagging task, we obtain a full lexicon, i.e., a mapping function of lexical categories to words. We define an ASR as a sequence of lexical categories annotated by a unique identifier. The words in the sentence are replaced by corresponding lexical categories in the lexicon. Example 11 shows an ASP of the sentence "Henri watches TV" and the sample lexicon {Henri $\vdash$ NP, regarde $\vdash$ (S\NP)/NP, la $\vdash$ NP/NP, télévision $\vdash$ NP}.

(11) a. $\text{Henri}_1\ \text{regarde}_2\ \text{la}_3\ \text{télévision}_4$

b. $[\text{NP}^1, (\text{S}\backslash\text{NP})/\text{NP}^2, \text{NP/NP}^3, \text{NP}^4]$

An equivalent representation via combinatory rules can be given using an instance of a CCG combinator ζ that has a general form like:

$$\frac{X_1, \ldots, X_n}{Y}\zeta,$$

[1] Official site: `http://openccg.sourceforge.net/`.

where $X_1, \ldots, X_n$ are functions and arguments of the left clause of a combinatory rule that is called a precondition sequence of ζ, whereas Y is the result of a combinatory rules or the effect of applying the combinator ζ.

The annotation of combinators is realized by assigning distinct identifiers. Thus, an annotated combinator includes the assignment of a distinct identifier for each element of its precondition sequence, and the identifier of the left most annotated lexical category in the precondition sequence to its effect. For example, the annotated combinator ζ_1 for the phrase "la télévision" (the television) which has a ASR [NP/NP^3, NP^4] is illustrated in Example 12.a.

(12) a. $\frac{NP/NP^3 \;\; NP^4}{NP^3} \; >$

b. $\frac{(S\backslash NP)NP^2 \;\; NP^3}{S\backslash NP^2} \; >$

c. $\frac{NP^1 \;\; S\backslash NP^2}{S^1} \; <$

Similar to that, we can find annotated combinators for other parts of the sentence in Example 11.a. These annotated combinators correspond to Examples 12.a, 12.b, 12.c that are denoted ζ_1, ζ_2, ζ_3, respectively.

Given an ASR in the example 11.b, a sequence of actions ζ_1, ζ_2, ζ_3 forms a plan:

Time 0: [NP^1, (S\NP)/NP^2, NP/NP^3, NP^4]
action: ζ_1
Time 1: [NP^1, (S\NP)/NP^2, NP^3]
action: ζ_2
Time 2: [NP^1, S\NP^2]
action: ζ_3
Time 3: [S^1].

This plan provides a derivation tree for the sentence (11.a). With this approach, the declaration of states and actions that correspond to ASP encoding and the application of combinatory rules become most important because performance time and results achieved depend on them.

The ASPCCGTK toolkit[2] (Lierler and Schüller, 2011; Lierler and Schüller, 2012) is a complete implementation of this approach. Using the output of the C&C supertagger that provides lexical categories for words of a given sentence, this tool creates a set of ASP facts that is used to find CCG derivation parse trees.

6.5.3 Shift-reduce parsing algorithms

Shift-reduce or transition-based parsers have become increasingly popular for dependency parsing. In general, the advantages of these approaches are based on their scoring model, which is defined over transition actions. Efficient parsing is obtained by using a beam search that allows to find the highest scoring action. Using a shift-reduce approach for the CCG parsing task improves operations such as easy treatment of sentences for which finding a spanning analysis is difficult, production of fragmentary analyses for sentences (Zhang, Kordoni, and Fitzgerald, 2007), or creation of local structures.

Similar to the transition-based parsing approach, shift-reduce CCG parsing consists of a stack α of partial derivations, a buffer β for incoming words and a transition system that includes

[2]The official site: `http://www.kr.tuwien.ac.at/staff/former_staff/ps/aspccgtk/`.

a series of actions $\mathcal{T}$ and a set of configurations. In particular, the set of action types is defined as follows:

- SHIFT-X: remove the first word w_i in the buffer α, assign the lexical category X to it ($X(w_i)$), and then push it onto the stack β.
- COMBINE-X: pop the top two nodes $X_k(w_i)$, $X_l(w_j)$ out of the stack α, combine $X_k(w_i)$, $X_l(w_j)$ into a new node $X_h(w_i)$, and then push $X_h(w_i)$ back on the stack α. We can see that this action corresponds to using a CCG combinatory rule.
- UNARY-X: pop the top node $X_k(w_i)$ out of the stack α, transform $X_k(w_i)$ into a new node with category X_l, and then push the new node $X_l(w_i)$ onto the stack α. This actions corresponds to using a type-raising rule in CCG.
- FINISH: the buffer β is empty, that is, all words in β have been shifted onto the stack α. The parsing process is finished.

For decoding in CCG shift-reduce parsing, we can use a greedy local search (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004) or a beam-search (Johansson and Nugues, 2007; Huang, Jiang, and Liu, 2009; Zhang and Clark, 2009; Zhang and Clark, 2011; Vaswani et al., 2016). The authors of (Zhang and Clark, 2011) formulate their decoding algorithm as follows:

- The initial item is defined as the unfinished item, in which the stack α is empty, the buffer β contains all ordered words from a given sentence.
- A candidate item is a tuple (α, β, γ), where α and β represent stack and buffer as above, γ is a boolean value representing the status of the candidate item. A candidate item is thereby marked as finished if and only if the FINISH action is used. In that case no more actions can be applied to it.
- A derivation is the result of a process started with the initial item and consisting of a sequence of actions until the candidate item acquires finished status.

The beam-search algorithm includes an agenda variable that contains the initial item at the beginning step and is used to hold the N-best partial candidate items for each parsing step. A candidate output variable is used to store the current best finished item, which is set empty at the beginning step. At each step of the algorithm, each candidate item from the agenda is exploited and extended by using actions. After this step a number of new candidate items can be created. If a new candidate item has a finished status, a comparison of the score between the current candidate output and the new candidate item is realized. If the current candidate output is none or the score of the new candidate item is higher, the candidate output is replaced by the new candidate item; otherwise, we continue with the candidate output. Otherwise, if the new item has unfinished status, the new candidate is appended to a list that contains new partial candidate items. After every candidate item in agenda has been processed, we will clear all items in the agenda and add the N-best items from the list into the agenda. The process continues with new N-best candidate item in the agenda. The algorithm terminates when the agenda is empty, i.e., when no new candidate times are generated. We obtain a derivation from the final candidate output. The pseudo code of the algorithm is illustrated in Algorithm 7.

The Z&C Parser[3] (Zhang and Clark, 2011) is the first CCG parser based on the transition-based approach. The authors have used a global linear model to score candidate items, features are extracted from each action. The training model is based on the perceptron algorithm (Collins, 2002). The CCG parser of (Xu, Clark, and Zhang, 2014; Ambati et al., 2015) focuses on improving the result of the Z&C Parser with dependency models (Clark and Curran,

[3] Available source code: `https://sourceforge.net/projects/zpar/`.

Algorithm 7: Agenda-based Searching Algorithm

Data: $x_1, \ldots, x_n$ are input words, $grammar$, N is size of agenda
Result: An agenda $\mathcal{A}$
begin
  $A \longleftarrow \emptyset$; // Agenda $\mathcal{A}$ is emptied
  for $initial\_item \in$ *TAG(*$x_1 \ldots x_n$*)* **do**
    PUSH($\mathcal{A}$,initial_item) ;
  $candidate\_ouput \longleftarrow \emptyset$;
  while $\mathcal{A} \neq \emptyset$ **do**
    $list \longleftarrow \emptyset$;
    for $item \in \mathcal{A}$ **do**
      for $action \in grammar.get\_actions(item)$ **do**
        $candidate\_item \longleftarrow item.apply(action)$;
        if $candidate\_item.finished$ *is* $True$ **then**
          if $candidate\_output$ *is* $\emptyset$ *or* $candidate\_item.score \geq candidate\_output.score$ **then**
            $candidate\_output \longleftarrow candidate\_item$;
        else
          $list.append(candidate\_item)$;
    $\mathcal{A} \longleftarrow \emptyset$;
    $\mathcal{A} \longleftarrow$ list.best(N);

2007). Using dependencies provides an elegant solution to handle the spurious ambiguity issue. Moreover, a training data for dependencies can be achieved easier than others, such as syntactic derivation data. The authors have used the dependency oracle (Goldberg and Nivre, 2012) to create dependencies and score the CCG derivations trees through a unification mechanism.

Several shift-reduce CCG parsers such as LSTM-CCG (Xu, 2016) and TranCCG[4] (Ambati, Deoskar, and Steedman, 2016) have used deep learning models and have achieved significant improvements when compared with previous works. In particular, (Xu, 2016) factors a model into five components: U, V, X, Y denote four LSTMs and W denotes a BiLSTM. In the initial step, W obtains sentences converted into embedding vectors as input. We redefine a candidate item as a tuple $(j, \alpha, \beta, \delta)$, where j is the index of the first item of the buffer β and δ denotes the set of CCG dependencies (Clark and Curran, 2007). The stack α for a candidate item at step t ($t \geq 1$) is:

$$\alpha_t = [h_t^U; h_t^V; h_t^X; h_t^Y]. \tag{6.16}$$

The pair $[\alpha_t, w_j]$ represents a candidate item. $[\alpha_0, w_0]$ represents the initial item, where $\alpha_0 = [h_\perp^U; h_\perp^V; h_\perp^X; h_\perp^Y]$, $h_\perp$ is the initial state of the hidden layer of the component. Before obtaining the probability of the i-th action, we need to calculate two affine transformation layers as follows:

$$\begin{aligned} b_t &= f(A[\alpha_t; w_j] + r), \\ a_t &= f(Bb_t + s), \end{aligned} \tag{6.17}$$

[4]Available source code: https://github.com/bharatambati/tranccg

where B, A are weight matrices, r, s are bias vectors, and f is an activation function. The prediction of the next action is based on the probability of the previous action:

$$p(\tau_t^i \mid b_t) = \frac{e^{a_t^i}}{\sum_{\tau_t^k \in \mathcal{T}(\alpha_t, \beta_t)} e^{a_t^k}}, \tag{6.18}$$

where $\mathcal{T}(\alpha_t, \beta_t)$ denotes the set of possible actions for the current candidate item, and $\tau_t^i \in \mathcal{T}(\alpha_t, \beta_t)$.

We continue the training of the neural architecture by a shift-reduce CCG parsing based on a greedy model. Let $\theta = \{U, V, X, Y, W, B, A\}$ represent the weights of the greedy model, the value of the tuple θ will be optimized in different training epochs. At each training epoch, we first get a sentence s_n from the training set, then use beam search to decode s_n to obtain a n-best list of candidate output with the current θ and denote the result as $\mathcal{P}(s_n)$. In the second step, for each derivation y_i in $\mathcal{P}(s_n)$, we compute the F1 score at the sentence level by using the CCG dependency set δ. Then, we compute the log-linear score of the action sequence as follows:

$$\rho(y_i) := \sum_{j=1}^{|y_i|} \log\ f(y_i^j), \tag{6.19}$$

where $|y_i|$ is the number of actions derived in y_i, and $f(y_i^j)$ denotes the softmax action score of the j-th action. In the final step, we minimize the negative F1 score objective for S_n. We repeat these steps for other sentences and other epochs. The objective function is defined as:

$$J(\theta) := -\sum_{y_i \in \mathcal{P}(s_n)} p(y_i \mid \theta)\ F1(\delta_{y_i}, \delta_{s_n}), \tag{6.20}$$

where $F1(\delta_{y_i}, \delta_{s_n})$ is the F1 score at the sentence level for derivation y_i compared with the standard dependency structure δ_{s_n} of s_n, and $p(y_i \mid \theta)$ denotes the probability score of the action sequence y_i:

$$p(y_i|\theta) := \frac{e^{\rho(y_i)}}{\sum_{y \in \mathcal{P}(s_n)} e^{\rho(y)}}. \tag{6.21}$$

6.5.4 A* parsing algorithms

(Klein and Manning, 2003a) introduce an extension of the classic A* search algorithm to tabular PCFG parsing, which is called A* parsing algorithm. A* search essentially is an agenda-based best-first graph search approach that finds the lowest cost parse with a heuristic estimation function without traversing the entire search space. Using the A* parsing approach to build a CCG parser has been realized in works such as (Auli and Lopez, 2011; Lewis and Steedman, 2014a; Lewis, Lee, and Zettlemoyer, 2016). More generally, the A* parser is based on a chart/table and an agenda. The agenda can be a buffer or queue of items with a priority order. Each item in the agenda marks its priority by a cost or score that consists of the items' inside probability and an estimated heuristic upper bound on the outside probability that is computed with respect to the context. These aim to give a the probability of the complete parse. Then, the chart is synthesized in best-first order, until a complete derivation parse for the sentence is achieved.

Compared with chart parsing or shift-reduce parsing, A* parsing does not require pruning of the search space of the lexical categories of each word (supertagging task). For example, the C&C parser takes an average of 3.57 lexical categories par word into account when calculating CCG derivation parse trees, whereas A* parsing can consider the complete set of 425 lexical categories per word (Lewis and Steedman, 2014a). That is because the A* parsing algorithm focuses on best-first search for sentence derivation, rather than constructing a complete chart

containing all lexical categories of all words. Moreover, the performance of A* parsing is faster than CKY by up to 5 times, while keeping accuracy (Lewis, He, and Zettlemoyer, 2015).

Each item on the agenda is ranged by cost and computed as the product[5] of the inside probability and an upper bound on the outside probability. More specifically, for a span $w_i, \ldots, w_j$ corresponding to lexical categories $c_i, \ldots, c_j$ of a sentence $S = w_0, \ldots, w_n$, the cost of the partial parse item $y_{i,j}$ corresponding to span $w_i, \ldots, w_j$ is computed as $f(y_{i,j}) := g(y_{i,j}) \times h(y_{i,j})$, where $g(y_{i,j})$ represents the inside probability, computed as:

$$g(y_{i,j}) := \prod_{k=i}^{j} p(c_k|S), \tag{6.22}$$

Whereas $g(y_{i,j})$ denotes the upper bound of the outside (or "context") probability, computed as:

$$h(y_{i,j}) := \prod_{k=1}^{k<i} \max_{c_k} p(c_k \mid S) \quad \times \prod_{k=j+1}^{k\leq n} \max_{c_k} p(c_k \mid S). \tag{6.23}$$

If there are two items in the agenda with the same cost, then the number of dependency relations is used to select among them.

In order to formulate the A* parsing model, let the CCG derivation parse y of sentence S be a list of lexical categories $c_1, \ldots, c_n$. The optimal derivation parse $\hat{y}$ is computed as follows:

$$\hat{y} := \arg\max_{y} \prod_{i=1}^{n} p(c_i \mid S). \tag{6.24}$$

A sentence can have many CCG derivation parse trees for the same initial sequence of lexical categories. Thus, the model needs to be extended by a deterministic heuristic for ranking derivation parses having the same initial lexical categories. In particular, in case two derivation parses have the same probability, one uses the sum of lengths of edges to select among them.

The A* search-based parser begins by finding a derivation tree of the given sentence based on 1-best categories for each word. The objective is to build a complete chart as soon as possible. If this search fails, the process continues by adding one more lexical categories (having the highest probability in the agenda) into the chart and attempting again. This process allows to exclude a number of lexical categories for each word until achievement of a derivation parse tree for the input sentence.

The EasyCCG parser[6] (Lewis and Steedman, 2014a; Lewis, He, and Zettlemoyer, 2015; Lewis, Lee, and Zettlemoyer, 2016) is a complete CCG parsing application based on a supertag-factored A* parsing approach, in which the search of the highest scoring supertag sequence has been combined to build up a complete derivation parse tree. This parser achieves a higher efficiency than the C&C parser, which uses the CKY parsing approach, both in accuracy and time.

The NeuralCCG parser (Lee, Lewis, and Zettlemoyer, 2016) improves A* by combining optimal decoding and global representation via Tree-LSTM (Tai, Socher, and Manning, 2015). This tool currently achieves a state-of-the-art accuracy for CCG parsing task when using a CCG bank as training corpus. In this case, parsing is considered as graph search problem. The parsing model is reformulated as a search for the highest scoring path for the graph. Let a node y in the graph represent a labeled span $w_i, \ldots, w_j$, and let an edge e in the graph represent a combinatory rule production in partial parse. The head node HEAD(e) of the edge e is defined as the parent of the combinatory rule production, its children nodes are the children of the combinatory rule production. A start node $\emptyset$ denotes an empty parse with outgoing edges. A path is a set of edges

[5] We can in fact replace product by sum (Lewis, Lee, and Zettlemoyer, 2016).

[6] Online demo at the address: `http://4.easy-ccg.appspot.com`.

E that begins at $\emptyset$ and ends at a destination node. A derivation parse is defined as a path in the graph. Each edge e has a score (weight) $s(e)$. The score of the derivation parse from path E is computed as:

$$f(E) := \sum_{e \in E} s(e). \tag{6.25}$$

As a result, the search of the complete parse is equivalent to the search of the highest scoring path.

The score of each edge e is a sum of the local score $s_{\text{local}}(e)$ which corresponds to the inside probability in §6.22 and of the global score $s_{\text{global}}(e)$ which corresponds to the upper bound of the context probability in equation 6.23. The global score is computed by using a hidden representation representing the derivation parse of $y = \text{HEAD}(e)$. In the Tree-LSTM neural network (Tai, Socher, and Manning, 2015), we can have a hidden representation of both constituents and a complete sentence from a given sentence $w_1, w_2, \ldots, w_n$ for the derivation parse y in forward and backward directions, as follows:

$$\begin{aligned}
i_y &= \sigma(W_i^R[c_l, h_l, c_r, h_r, x_y] + b_i^R), \\
f_y &= \sigma(W_f^R[c_l, h_l, c_r, h_r, x_y] + b_f^R), \\
o_y &= \sigma(W_o^R[\hat{c}_y, h_l, h_r, x_y] + b_o^R), \\
c_{rl} &= f_y \circ c_l + (1 - f_y) \circ c_r, \\
\hat{c}_y &= \tanh(W_c^R[h_l, h_r, x_y] + b_c^R), \\
c_y &= i_y \circ \hat{c} \circ c_l r, \\
h_y &= o_y \circ \tanh(c_y),
\end{aligned} \tag{6.26}$$

where h_l, c_l, h_r, c_r denote hidden and cell states of left and right childs respectively, x_y represents a learned embedding vector for the lexical category at the root of y, σ is a cost function, W, b denote weights and biases parametrized by the combinatory rule R. The global score is formulated as follows:

$$s_{\text{global}}(e) := \log(\sigma(W.h_y)). \tag{6.27}$$

6.6 Conclusion of the Chapter

We have presented a new CCG supertagging task based on morphological and dependency syntax information, which has allowed us to create a CCG version of the French Tree Bank corpus FTB. We used this corpus to train a new BiLSTM+CRF neural architecture that uses new, morphosyntactic, input features and feature correlations as separate input features. We have experimentally shown that, at least for an inflected language as French, dependency syntax information is useful for improving the accuracy of the CCG supertagging task when using deep learning techniques.

Chapter 7

Towards a DRS Parsing Framework for French

Combinatory Categorial Grammars provide a transparent interface between surface syntax and underlying semantic representation. Moreover, CCGs gain the same expressive power as the lambda calculus because their foundation is combinatory logic. In previous chapters we have achieved a CCG derivation parse tree for a given sentence, by using morphological and dependency syntax information. These results help us to constitute the first bridgehead on the way towards semantic interpretation for natural language inputs. In this chapter, we will focus on Discourse Representation Theory that allows the handling of meaning across sentence boundaries. Based on the foundations of these two theories, along with the work of Johan Bos on the Boxer Framework for English language, we propose an approach to the task of semantic parsing with Discourse Representation Structure for the French language. In addition, we will synthesize experiments, evaluations, and error analysis about our approach (Le, Haralambous, and Lenca, 2019).

7.1 Introduction

Researchers in the domain of computational linguistics have studied for a long time various forms of logic aiming to capture semantic information from natural language data. Among these, Discourse Representation Theory (DRT) is one of the first frameworks for exploring meaning across sentences, with a formal semantics approach Kamp and Reyle, 1993. DRT has been used for various applications such as the implementation of a semantic parsing system or natural language understanding systems. This chapter addresses the build of a semantic parsing application based on CCG and Discourse Representation Structure (DRS) which is an important element of DRT.

CCGs essentially illustrate an explicit relation between syntax and semantic representation Steedman, 1996. They allow access to a deep semantic structure of the phrase and facilitate recovering of non-local dependencies involved in the construction, such as coordination, extraction, control, and raising. In addition, CCGs are compatible with first order logic (FOL) and lambda-expressions. Their use allows analysis of syntax and semantic relationship between words or phrases in the scope of the sentence. Nevertheless, the analysis of CCG is limited in scope of a single sentence. We thus need a higher language formalism analysis level which can effect an analysis on utterances with several sentences.

DRT is one of the theoretical frameworks used for discourse interpretation in which the primary concern is to account for the context dependence of meaning. Indeed, a discourse may be interpretable only when the framework takes account of the contexts of the discourse. Furthermore, DRT can be used to deal with a variety of linguistic phenomena such as anaphoric pronouns or temporal expressions, within or across different sentences. DRS expressions have

two advantages: (1) they provide meaning representation for several sentences in discourse instead of a single sentence only, and (2) they can be translated into First-Order Logic.

In the next sections, after an introduction to semantic parsing, we provide an overview of DRT and its core DRS. In the next section, we present the architecture of Boxer semantic parsing, which is a fundamental in order to construct DRS parsing for French language. Then we describe our approach, experimentation, and evaluation through an example. Finally, we finish the chapter with a summary about our work.

7.2 Discourse Representation Theory

In the early 1980s, a theoretical framework for dynamic semantics has been introduced by Hans Kamp under the name *Discourse Representation Theory* (DRT). The goal was to deal with linguistic phenomena such as anaphoric pronouns, time, tense, presupposition, and propositional attitudes (Kamp, 1981; Kamp and Reyle, 1993). The emergence of DRT made a dynamic approach on natural language semantics possible. In this approach, the meaning of a given sentence is identified in a relationship with its context. In particular, the interaction between a sentence and its context is reciprocal. The analysis of an utterance will depend on its context. Otherwise the context can be transformed into a new context if information from the utterance is used and added to the context.

Unlike other theoretical frameworks using First-Order Logic syntax for semantic representations, DRT uses a semantic representation called "Discourse Representation Structure" (DRS) in order to describe objects mentioned in an utterance and their properties. On the one hand, DRS plays the role of semantic content in which it provides the precise expressive meaning for a natural language inputs. On the other hand, DRS keeps the role of discourse context where it aggregates the interpretation of subsequent expressions occurring in the utterance.

The dynamicity of DRT is represented through three major components. The first and crucial component includes recursive definitions of the set of all well-formed DRSs used in the meaning representation of natural language. The second component is a model-theoretical semantic interpretation used to link the members in the set of DRSs. The third component constitutes an algorithm that integrates natural language input into DRSs.

7.2.1 Type Lambda Expression

The type in the formal language L_λ which we introduce more detail in Appendix A.1 can be used to express both constants and variables in syntactic categories, and it also allows quantification over variables of any category. Syntactic categories can be matched with semantic types specifying object type in categories. More generally, there are two basic semantic types, which we denote by $\langle e \rangle$ and $\langle t \rangle$. Type $\langle e \rangle$ denotes for entities or individuals (i.e., any discrete object), whereas type $\langle t \rangle$ denotes truth values (i.e., propositions, sentence).

Definition 7.2.1 (Recursive definition of types)**.**

1. $\langle e \rangle$ and $\langle t \rangle$ are types.
2. If $\langle a \rangle$ and $\langle b \rangle$ are types, then $\langle a, b \rangle$ is a type.
3. Nothing else is a type.

All pairs $\langle a, b \rangle$ made out of basic or complex types $\langle a \rangle$, $\langle b \rangle$ are types. $\langle a, b \rangle$ can be regarded as the type of functions that map arguments of type $\langle a \rangle$ to values of type $\langle b \rangle$. Some popular examples of complex types are: $\langle e, t \rangle$ (unary predicate), $\langle e, \langle e, t \rangle \rangle$ (binary predicate), $\langle t, t \rangle$ (unary function), $\langle \langle e, t \rangle, \langle e, t \rangle \rangle$ (unary predicate transformer), $\langle \langle e, t \rangle, t \rangle$ (second-order predicate) and so on. The figure 7.1 illustrates an usage of the type-base theory for the analysis of a sentence.

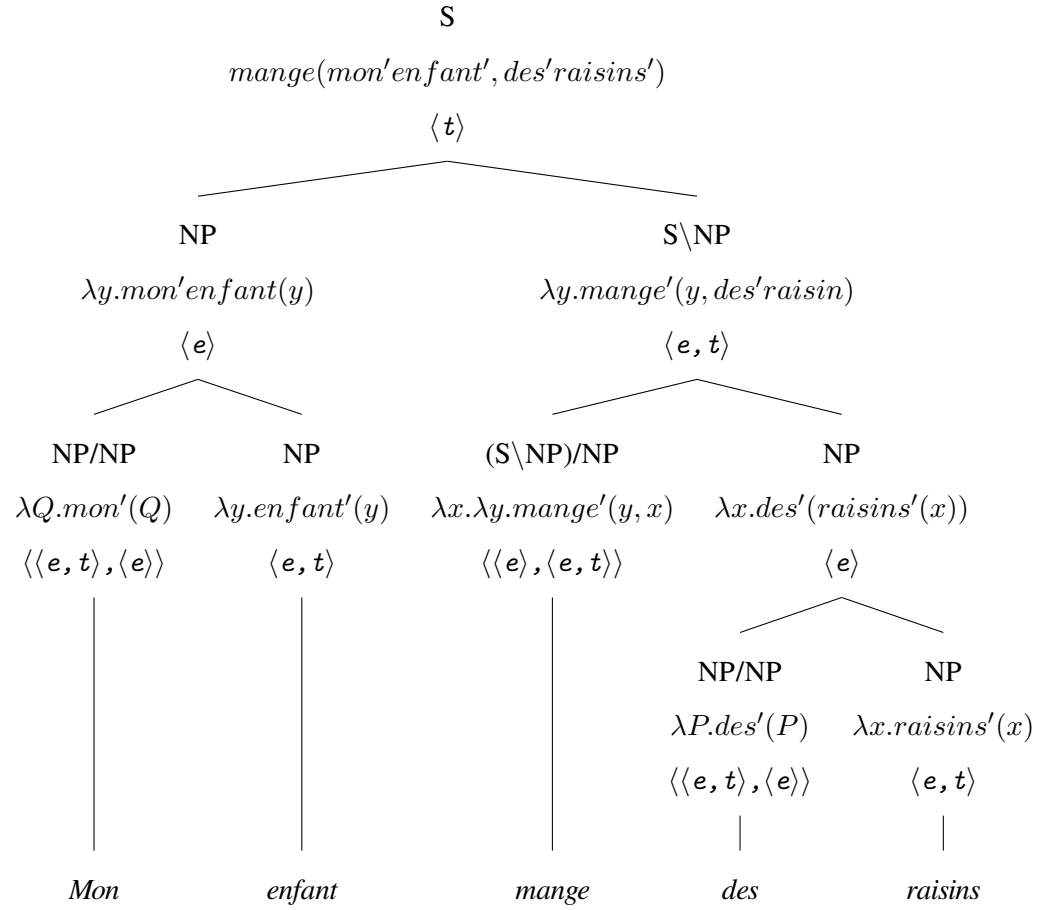


FIGURE 7.1: A type-based expression for the sentence "My child eats grapes"

7.2.2 Discourse Representation Structure

The core of DRT is DRS, which is main component for the construction of semantic representations for natural language texts. The objective of DRS is not only to achieve interpretations of single sentences, but also to represent larger linguistic units, paragraphs, discourses or texts. In general, the meaning representation for large unit sentences through DRS proceeds unit by unit. Each sentence that gets processed contributes its information to the DRS that contains information from preceding sentences.

The representation of DRSs used from this section relies on basic type-theoretic expressions that are compatible with formal semantic representations (Bos et al., 2017). Therefore, the type $\langle e \rangle$ is used for expressions of discourse referents or variables, whereas type $\langle t \rangle$ is used for basic DRS expressions.

$$\begin{aligned} \langle exp_e \rangle &::= \langle ref \rangle | \langle var_e \rangle \\ \langle exp_t \rangle &::= \langle drs \rangle \end{aligned} \tag{7.1}$$

Definition 7.2.2. A basic DRS expression $\langle drs \rangle$ is a set-theoretic object built from two principal components: a set of discourse referents $\langle ref \rangle$ that represents the objects under discussion,

and a set of conditions $\langle condition \rangle$ that are properties of discourse referents, and express relations between them. We use the following notation:

$$\langle drs \rangle ::= \boxed{\begin{array}{|c|}\hline \langle ref \rangle * \\ \hline \langle condition \rangle * \\ \hline \end{array}} \tag{7.2}$$

In general, DRS conditions are of three types: $\langle basic \rangle$, $\langle link \rangle$, and $\langle complex \rangle$:

$$\langle condition \rangle ::= \langle basic \rangle \mid \langle link \rangle \mid \langle complex \rangle \tag{7.3}$$

The basic conditions are properties of discourse referents or relations between them:

$$\begin{aligned} \langle basic \rangle ::= \quad & \langle sym_1 \rangle (\langle exp_e \rangle) \mid \langle sym_2 \rangle (\langle exp_e \rangle, \langle exp_e \rangle) \\ & \mid timex(\langle exp_e \rangle, \langle sym_0 \rangle) \\ & \mid named(\langle exp_e \rangle, \langle sym_0 \rangle, class), \end{aligned} \tag{7.4}$$

where $\langle exp_e \rangle$ denotes expressions of type, $\langle sym_n \rangle$ denotes n-ary predicates, $\langle num \rangle$ denotes cardinal numbers, $timex$ expresses temporal information and $class$ denotes named entity classes.

The link conditions are discourse referent markers or constants which are used for references or separations between these discourse makers or constants:

$$\begin{aligned} \langle link \rangle ::= \quad & \langle exp_e \rangle = \langle exp_e \rangle \mid \langle exp_e \rangle = \langle num \rangle \\ & \mid \langle exp_e \rangle \neq \langle exp_e \rangle \mid \langle exp_e \rangle \neq \langle num \rangle. \end{aligned} \tag{7.5}$$

The complex conditions represent embedded DRSs: implication ($\rightarrow$), negation ($\neg$), disjunction ($\vee$), modal operators expressing necessity ($\Box$) and possibility($\Diamond$). The types of complex conditions are unary and binary:

$$\begin{aligned} \langle complex \rangle &::= \langle unary \rangle \mid \langle binary \rangle \\ \langle unary \rangle &::= \neg \langle exp_t \rangle \mid \Box \langle exp_t \rangle \mid \Diamond \langle exp_t \rangle \mid \langle ref \rangle : \langle exp_t \rangle \\ \langle binary \rangle &::= \langle exp_t \rangle \rightarrow \langle exp_t \rangle \mid \langle exp_t \rangle \vee \langle exp_t \rangle \mid \langle exp_t \rangle ? \langle exp_t \rangle \end{aligned} \tag{7.6}$$

where the condition $\langle ref \rangle : \langle exp_t \rangle$ denotes verbs with propositional content.

Definition 7.2.3. The construction of DRSs from basic DRSs is defined according to the following clauses:

1. If $\langle exp_x \rangle$ is a discourse referent marker, [$\langle exp_x \rangle$: $\emptyset$] is a DRS.
2. If [$\emptyset$: $\langle condition_k \rangle$] is a DRS.
3. If $\langle sym_k \rangle$ is a n-ary predicate and $\tau_1, \ldots, \tau_n$ are terms, then [$\emptyset$:$\langle sym_k \rangle (\tau_1, \ldots, \tau_n)$] is a DRS.
4. If $v = \langle exp_x \rangle$ is a discourse referent marker and τ is a term, then [$\emptyset$:$v = \tau$] is a DRS.
5. If $v = \langle exp_x \rangle$ is a discourse referent marker and τ is a term, then [$\emptyset$:$v \neq \tau$] is a DRS.
6. If $\langle drs \rangle ::= [\langle ref \rangle : \langle condition \rangle]$ is a DRS, then [$\emptyset$: $\neg \langle drs \rangle$] is a DRS.

7. If $\langle drs \rangle ::= [\langle ref \rangle : \langle condition \rangle]$ and $\langle drs\,' \rangle ::= [\langle ref\,' \rangle : \langle condition\,' \rangle]$ are DRSs, then $\langle drs \rangle \oplus \langle drs\,' \rangle ::= [\langle ref \rangle \cup \langle ref\,' \rangle : \langle condition \rangle \cup \langle condition\,' \rangle]$ is a DRS.

8. Nothing else is a DRS.

Let us illustrate DRS through the example of sentence 13.a. This sentence can be analyzed and rewritten under DRS form as follows:

$$[x, y : \text{femme}(x), \text{poisson}(y), \text{acheter}(x, y)].$$

More specifically, the DRS expression contains two discourse referents $\langle ref \rangle = \{x, y\}$, whereas the set of conditions includes $\langle condition \rangle = \{\text{femme}(x), \text{poisson}(y), \text{acheter}(x, y)\}$. Suppose now that the sentence in the example 13.b is followed by sentence 13.a. The DRS expression for the second sentence includes discourse referents $\langle ref \rangle = \{u, v, w\}$, whereas the set of conditions is $\langle condition \rangle = \{\text{donner}(u, v, w), \text{mari}(w), \text{person}_1(v), \text{thing}_1(w)\}$. Thus, The DRS of the second sentence will be rewritten as follows: $[u, v, w : \text{donner}(u, v, w), \text{mari}(w), \text{person}_1(v), \text{thing}_1(w)]$. Finally, we obtain the final DRS expression after integrating the DRS of the second sentence into the DRS of the first sentence, as well as resolving anaphoric problem as follows:

$$\begin{aligned}[x, y, u, v, w \ : &\text{femme}(x),\ \text{poisson}(y),\ \text{acheter}(x, y),\\ &\text{donner}(u, v, w),\ \text{mari}(u),\ v = x,\ w = y].\end{aligned}$$

(13) a. La femme achète des poissons.
(The woman buys fishes.)

b. Elle les donne à son mari.
(She gives them to her husband.)

In order to illustrate the different DRS expressions, we have three formalisms ofr representing the above sentences:

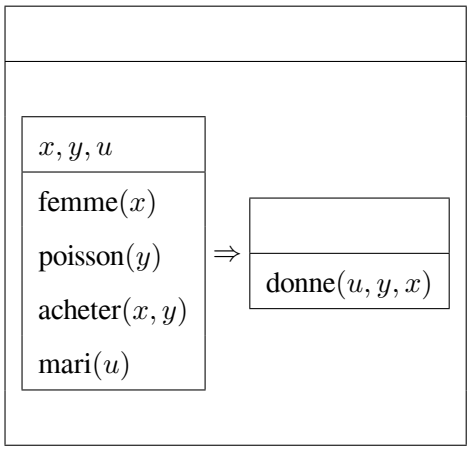
1. The “official” DRS notation:

$$<\{\emptyset\}, <\{x, y, u\}>, \{\text{femme}(x), \text{poisson}(y), \text{acheter}(x, y), \text{mari}(u)\}> \Rightarrow <\{\emptyset\}, \{\text{donne}(u, y, x)\}>.$$

2. The linear notation:

$$[\emptyset : [x, y, u : \text{femme}(x), \text{poisson}(y), \text{acheter}(x, y), \text{mari}(u)] \Rightarrow [\emptyset : \mathit{donne}(u, y, x)]].$$

3. The boxed notation:



We can generalize DRSs through a model-theoretic interpretation. Conventionally, an appropriate model $\mathcal{M}$ for the DRS $\langle drs \rangle$ is an ordinary first-order model $\mathcal{M} = \langle \mathcal{D}, \mathcal{F} \rangle$, where $\mathcal{D}$ denotes the set of individuals occurring in the conditions of $\langle drs \rangle$, and $\mathcal{F}$ is an interpretation function that maps the n-ary predicate in the basic condition of $\langle drs \rangle$ to n-ary relations of $\mathcal{D}$. An embedding function is defined as a partial mapping from discourse referents $\langle ref \rangle$ to elements of $\mathcal{D}$. Thus, the function f verifies in $\mathcal{M}$ if f maps an discourse referents onto an individual in $\mathcal{D}$. For instance, we have the embedding functions in the example 13.a such femme(x), poisson(y) and so on.

Let f, g, and $\langle drs \rangle ::= [\langle ref \rangle : \langle condition \rangle]$ be two embedding functions and a DRS, $f\langle drs \rangle g$ denotes the extension g on f with respect of $\langle drs \rangle$, if domain(g) = domain$(f) \cup \langle condition \rangle$ and for all v in domain(f): $f(v) = g(v)$. The verification of the function f used to verify the DRS $\langle drs \rangle$ in model $\mathcal{M}$ is as follows:

- The function f verifies the DRS $\langle drs \rangle$ if f verifies all conditions in $\langle condition \rangle$.
- The function f verifies an n-ary predicate $\mathcal{P}(v_1, \ldots, v_n)$ if $\langle f(v_1), \ldots, f(v_n) \rangle \in \mathcal{F}(\mathcal{P})$.
- The function f verifies $v = u$ if $f(v) = f(u)$.
- The function f $\neg\langle drs \rangle$ if there is no g such that $f\langle drs \rangle g$ and g verifies $\langle drs \rangle$.
- The function f verifies $\langle drs \rangle \vee \langle drs\,' \rangle$ if f there is a g such that $f\langle drs \rangle g$ and g verifies $\langle drs \rangle$ or $\langle drs\,' \rangle$.
- The function f verifies $\langle drs \rangle \Rightarrow \langle drs\,' \rangle$ if for all $f\langle drs \rangle g$ such that g verifies $\langle drs \rangle$, there is an h such that $g\langle drs \rangle h$ and h verifies $\langle drs\,' \rangle$.
- The function f verifies $\langle drs \rangle(\forall v)\langle drs\,' \rangle$ if for all individuals $d \in \mathcal{D}$ for which there is a g such that $g(v) = d$ and g verifies $\langle drs \rangle$, every h such that $g\langle drs\,' \rangle h$ verifies $\langle drs\,' \rangle$.

The truth-conditional semantics of the DRS $\langle drs \rangle$ is true in the model $\mathcal{M}$ if there exists an embedding function f such that domain(f) = $\langle condition \rangle$ and the function f is satisfied in $\mathcal{M}$.

Definition 7.2.4. The translation from DRS to FOL is defined as follows:

- For DRSs, if $\langle drs \rangle ::= [\langle ref \rangle \Leftrightarrow \{v_1, \ldots, v_n\}{:}\langle condition \rangle \Leftrightarrow \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}]$ then, $\langle drs \rangle^{\langle fol \rangle} ::= \exists v_1, \ldots, \exists v_n(\mathcal{C}_1^{\langle fol \rangle} \wedge \cdots \wedge \mathcal{C}_m^{\langle fol \rangle})$.
- For basic or link conditions $\mathcal{C}^{\langle fol \rangle} ::= \mathcal{C}$.
- For negations: $(\neg\langle drs \rangle)^{\langle fol \rangle} := \neg\langle drs \rangle^{\langle fol \rangle}$

- For implications: $(\langle drs \rangle \Rightarrow \langle drs\,' \rangle)^{\langle fol \rangle} ::= \forall v_1, \ldots \forall v_n((C_1^{\langle fol \rangle} \wedge \cdots \wedge C_m^{\langle fol \rangle}) \rightarrow \langle drs\,' \rangle^{\langle fol \rangle})$

DRS expressions are compatible with first-order logic (FOL) through specific steps of translation like in Definition 7.2.4 (Van Eijck and Kamp, 1997; Blackburn et al., 1999; Bos, 2004). First, we consider each discourse referent as a first-order quantifier. Then, DRS conditions are interpreted into a conjunctive formula of FOL. Finally, the embedded DRSs such as implication, negation, disjunction are translated to corresponding formulas (Bos, 2008). For instance, the FOL which is equivalent with the DRS for utterances in the example 13 is transformed as follows:

(14) $\forall x \forall y \forall u((\text{femme}(x) \wedge \text{poisson}(y) \wedge \text{acheter}(x, y) \wedge \text{mari}(u)) \rightarrow (\text{donner}(u, y, x)))$.

7.2.3 The CCG and DRS Interface

Categorial type transparency is one of the important principles applied to CCG (see chapter 5). It states that there is an equivalence of representations between lexical categories and logical forms. Each syntactic category will be mapped to an unique semantic type in order to ensure a transparency between a CCG derivation and an expression under using semantic types.

In general, we can transform a CCG derivation parse tree to DRS expressions by defining an equivalent mapping between lexical categories and semantic types. For instance, if the primitive lexical categories employed in French CCG corpus are NP(Noun Phrase), S (Sentence), the S category is associated with DRS expression of type $\langle t \rangle$, whereas the NP categories correspond to DRS expression of type $\langle e, t \rangle$. With complex categories where the direction of slash will point out whether the argument come from its left if backward slash is used , or its right if forward slash is presented, the category S\NP/NP which correspond to a transitive verb phrase, requires a NP as argument on its left and has a DRS expression of type $\langle \langle e \rangle, \langle e, t \rangle \rangle$. Or the lexical category NP/NP which indicates an article or an adjective, requires a category NP as argument on its right, and owns the semantic type $\langle \langle e, t \rangle, \langle e \rangle \rangle$. Figure 7.1 shows some examples of such mappings.

Typed λ-expressions can also become a bridge for connecting between lexical categories and DRS expressions. If $\lambda x.\varphi$ is a λ-expression, then we have that x is a variable of type $\langle e \rangle$ and φ is a formula of type $\langle t \rangle$. Table 7.1 shows some examples of equivalences between types, lexical categories and λ-expressions.

7.3 DRS Parsing Based on Boxer

In Chapter 4 we proposed an architecture for building a French semantic parsing framework. We have gone through the steps required to obtain syntactic or grammar analysis for a given utterance. Before going into semantic parsing based on Boxer and DRS, let us recall the previous tasks in this framework.

7.3.1 Syntactic and grammar analysis

We employ dependency grammars to analyze the structure of given sentences. In particular, a dependency structure of sentences is a result of the analysis and description of the dependency of the linguistic units, e.g., words, which are connected to each other by dependency paths. Most of the time, the root of the dependency tree corresponds to the main verb and all other words are either directly or indirectly connected to this verb by directed edges. Each edge has a label for describing the relation between the two words. These labels belong to a set of syntactic functions, e.g., subject, object, oblique, determiner, attribute, etc. Syntactic functions are grammatical relations playing an important role in recognizing components of the sentence.

TABLE 7.1: Some translations between types, λ-expressions and lexical categories

Types	λ-expressions	Lexical Categories	Examples
$\langle e \rangle$	$constants$	N	Henri $\rightarrow henri'$, Paris $\rightarrow paris'$
$\langle e, t \rangle$	$\lambda x.\varphi(x)$	NP	enfant (child) $\rightarrow \lambda x.enfant'(x)$
		S\NP	sourire $\rightarrow \lambda x.sourire'(x)$
$\langle e, \langle e, t \rangle \rangle$	$\lambda x.\lambda y.\varphi(y, x)$	(S\NP)/NP	prendre (take) $\rightarrow \lambda x.\lambda y.prendre'(y, x)$
$\langle \langle e, t \rangle, e \rangle$	$\lambda P.\varphi(P)$	NP/NP	les $\rightarrow \lambda P.les'(P)$
$\langle \langle e, t \rangle, \langle e, t \rangle \rangle$	$\lambda P.P$	(S\NP)/(S\NP)	être (be) $\rightarrow \lambda P.P$
$\langle e, e \rangle$	$\lambda x.x$	(NP\NP)/NP	de (of) $\rightarrow \lambda x.x$
$\langle t, \langle t, t \rangle \rangle$	$\lambda P.\lambda Q[P \wedge Q]$	X/X, X\X	et (and) $\rightarrow \lambda P.\lambda Q[P \wedge Q]$
	$\lambda P.\lambda Q[P \vee Q]$	X/X, X\X	ou (or) $\rightarrow \lambda P.\lambda Q[P \vee Q]$
$\langle t, t \rangle$	$\lambda P.\neg P$	(S\NP)/(S\NP)	pas (not) $\rightarrow \lambda P.\neg P$
$\langle \langle e, t \rangle, t \rangle$	$\lambda P.\exists x.P(x)$	NP	[quelque_chose] (something) $\rightarrow \lambda P.\exists x.P(x)$
$\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$	$\lambda P.\lambda Q.\exists x.$ $[P(x) \wedge Q(x)]$	NP	[quelque] (some) $\rightarrow$ $\lambda P.\lambda Q.\exists x.[P(x) \wedge Q(x)]$
	$\lambda P.\lambda Q.$ $\partial[\exists x[P(x)]] \wedge \forall x$ $[P(x) \rightarrow Q(x)]$	NP	tout (every) $\rightarrow \lambda P.\lambda Q.\partial[\exists x[P(x)]] \wedge$ $\forall x[P(x) \rightarrow Q(x)]$

For the input discourse, syntactic information on words and their interrelations can be obtained via a dependency parser. There exist nowadays various dependency parsers for French such as MaltParser (Candito, Crabbé, and Denis, 2010), Stanford Parser (Green et al., 2011), MSTParser (McDonald, Lerman, and Pereira, 2006) , SpaCy (Honnibal, Goldberg, and Johnson, 2013; Honnibal and Johnson, 2015), and Grew Parser (Guillaume and Perrier, 2015) (see p. 60 for a comparison). We have chosen to use MaltParser in order to obtain morphosyntactic information on words in sentences. We keep the following information for every word: lemma, part-of-speech tag and dependency relation.

We used MElt (see chapter 4) to effect tokenization and morphological analysis like stems, root words, prefixes or suffixes for each word of the input sentence. Sequentially, the output of this step is used as the input for Maltparser in order to analyze dependency structures of the given sentence. For instance, Table 7.2 illustrates an analysis result for a sequence "You can test the compatibility of your computer with our website". First, the sequence is analyzed by MElt tool and the achieved result is presented in the first 6 columns which consists of numerical

TABLE 7.2: The original representation in CONLL format of dependency analysis for the sentence "You can test the compatibility of your computer with our website" by using MElt and Maltparser

ID	Word	Lemma	U_{postag}	X_{postag}	Features	Head	DepRel
1	Vous	cln	CL	CLS	s=suj	2	suj
2	pouvez	pouvoir	V	V	m=ind|n=p|p=2|t=pst	0	root
3	tester	tester	V	VINF	m=inf	2	obj
4	la	le	D	DET	g=f|n=s|s=def	5	det
5	compatibilité	compatibilité	N	NC	g=f|n=s|s=c	3	obj
6	de	de	P	P	_	5	dep
7	votre	son	D	DET	n=s|s=poss	8	det
8	ordinateur	ordinateur	N	NC	g=m|n=s|s=c	6	obj
9	avec	avec	P	P	_	3	mod
10	notre	son	D	DET	n=s|s=poss	11	det
11	site	site	N	NC	g=m|n=s|s=c	9	obj
12	internet	internet	A	ADJ	s=qual	11	mod
MElt						Maltparser	

order starting at 1, word form, lemma or stem of word form, U_{postag} correponding to universal POS tag, X_{postag} denoting French POS tag. Secondly, Maltparser employs these results as input features in order to find dependency relations between words of the given sentence. The result of Maltparser is shown in the two last columns: the first one contains the head of the current token and has a value of ID or zero, and the second one contains the name of dependency relations. The label for root of the dependency tree is marked by "root" and the head value is always zero.

7.3.2 Extraction of CCG derivation tree

In order to obtain a CCG derivation tree for each input French sentence, we have used the empirical approach introduced in Chapter 5 and briefly explained below. Using the syntax and dependency information obtained in the previous step, we process words which have unique lexical categories, e.g., nouns have lexical category NP, adjectives have lexical category NP/NP or NP\NP depending on whether they are on the left of on the right of the noun, etc. Once we have assigned these unique (but position-dependent, since, for example, adjectives in French can be located on both sides of the noun) lexical categories, we move over to verbs. The lexical category S\NP is assigned to a main verb having a subject to its left, and then we add a /NP (or a \NP, depending on its position with respect to the verb) for each direct object or indirect object (in the order of words in the sentence).

The next step is to binarize the dependency tree on the basis of information about dominant sentence structure: In French, most sentences are SVO or SOV. Using this general linguistic property, an algorithm has been proposed in Chapter 5 to extract and classify the components of the sentence into subject, direct object, indirect object, verbs, and complement phrases. This algorithm aims to transform a dependency tree into a binary tree. It is subdivided into two steps:

1. Chunks are extracted from the dependency tree based on syntactic information and dependency labels between words. For example, the subject chunk is obtained by finding a word that has a dependency labeled *suj*; the verb chunk corresponds to the root of the dependency structure; direct or indirect object chunks are obtained as words with links directed to the root verb and having labels *obj* or *p_obj*, etc.

2. A binary tree is built for each chunk, then binary trees are combined in inverse order of the dominant sentence structure. For example if SVO is the dominant structure, we start by building the binary tree of the *object* chunk, then combine it with the binary tree of the *verb* chunk, and finally we obtain the binary tree of the *subject* chunk.

For each input sentence we obtain a single CCG derivation tree, corresponding to its dependency tree input. The output CCG derivation tree is modified to be compatible with Boxer's input format. At the same time, the sentence is analyzed in order to extract named entity components (e.g., *Location*, *Person*, *Date*, *Time*, *Organization*, etc.) and chunk phrases by using the French models of SpaCy application.

7.3.3 Boxer Semantic Parsing

Implemented in the Prolog language with publicly available source code, the Boxer application is designed to provide semantic analysis of discourses for English with CCG derivation trees as input and meaning representation under the form of DRS as output. In order to do the same in French, we had to adapt the source code to the specific characteristics of the French language.

% Primaire: ***'NP V NP'*** ('allow-64')

% Syntax: [np:***'Agent'***,v,np:***'Theme'***]

% CCG: (s:dcl\np)/np

% Roles: [***'Theme'***,***'Agent'***]

% Example: ***'Luc approuve l'attitude de Léa'***

VerbNet:

(approuver, (s:dcl\np)/np, [***'Theme'***,***'Agent'***]).

(autoriser, (s:dcl\np)/np, [***'Theme'***,***'Agent'***]).

(supporter, (s:dcl\np)/np, [***'Theme'***,***'Agent'***]).

(tolérer, (s:dcl\np)/np, [***'Theme'***,***'Agent'***]).

FIGURE 7.2: An excerpt of the Verbnet lexical resource for French

Verbs are the central component of most sentences. Once a verb is given, we are able to know the components that can be attached to it. For example, the verb "to buy" must be followed by a direct object, and the verb "to sleep" cannot since it is intrasitive. Relationships between a verb and its noun phrase arguments are illustrated by thematic roles (e.g., Agent, Experience, Theme, Goal, Source, etc.). In Boxer, verbs and their thematic roles are extracted from the VerbNet lexical resources corpus (Schuler, 2005). For French, we have used the French

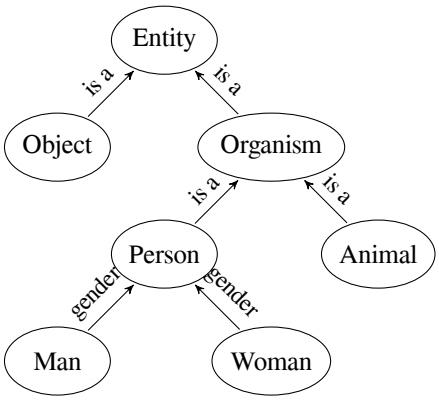


FIGURE 7.3: An ontology resource example

VerbNet corpus (Pradet, Danlos, and De Chalendar, 2014) (see an example in Fig. 7.2), while ontologies have provided hierarchical or equivalent relationships between entities, concepts, etc. (Fig. 7.3).

Issues concerning anaphora and presupposition triggers introduced by noun phrases, personal pronouns, possessive pronouns, reflexive pronouns, demonstrative pronouns, etc., are processed on a case-by-case basis, based on the resolution algorithm proposed in (Bos, 2003). Finally, the meaning representation of the discourse analysis is exported in two different formats: FOL and DRS.

7.4 Experiment and Evaluation

7.4.1 Experiments

We illustrate the capabilities of French discourse analysis via our architecture by the following example: "*Tous les soirs, mon voisin met sa voiture au garage. Il arrose ses rosiers avec son fils.*" (Every evening, my neighbor puts his car in the garage. He waters his rose bushes with his son). We have two sentences in this text, containing possessive pronouns, personal pronouns, and noun phrases.

We first apply a parser to obtain dependency relations. We then obtain a CCG derivation tree as output of the CCG parsing stage. The results are represented (cf. Fig. 7.4) in a format which is compatible with the input format of Boxer. Each word is handled as a term (*t*) together with the following information: *CCG lexical category*, *original word*, *lemma*, *POS tag label*, *chunks* and *named entity information*.

The reader can see the output of the example in two formats: FOL (Figure 7.6) and DRS (Figure 7.5). Boxer for French can analyze correctly linguistic phenomena such as possessive pronouns (*ses, mon, sa, son*), propositional quantifiers (*tout*) and noun phrases (*sa voiture au garage*). However, there is still room for improvement, for example, we do not obtain the chronological order of actions in the example.

On the other hand, we experimented our system with 4,525 sentences from the French TreeBank corpus in order to have an overview on a wide-coverage corpus. The length of the sentences in our experimentation is limited to 20 words because the FTB corpus was extracted from French newspapers, the sentences are thus regularly long and complex compared to the simple and short sentences of a discourse. Finally, we have obtained 61,94% of sentences which can be analyzed successfully by our system. By analyzing errors that occurred in our outcomes, we figure out two main causes. The first one derives from errors in dependency analysis or CCG analysis step. The second one originates from the lack of semantic representation definition on the CCG lexical in Boxer for French.

FIGURE 7.4: CCG derivation tree for the sentence "*Tous les soirs, mon voisin met sa voiture au garage*".

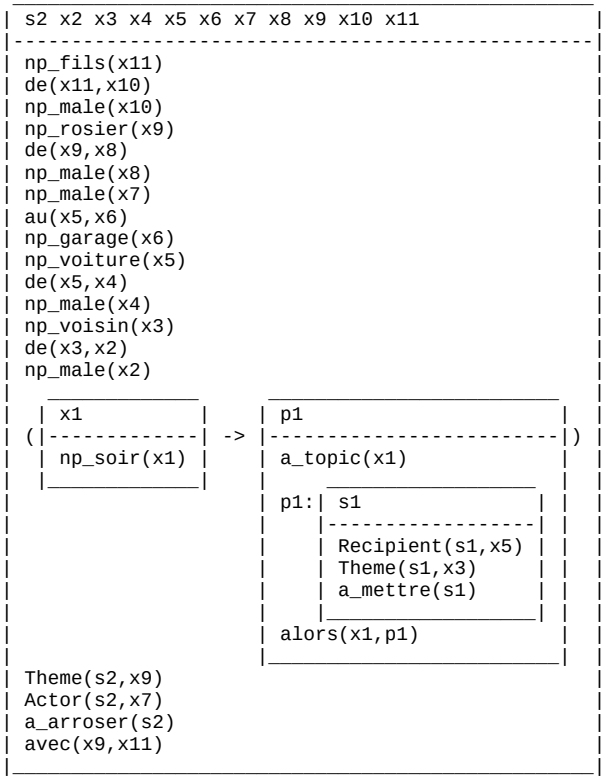


FIGURE 7.5: DRS output of the utterance

$\exists\ z3\ z7\ z8\ z9\ z10\ z11\ z12\ z13\ z14\ z5\ z6.(np\_fils(z6) \wedge de(z6, z5) \wedge np\_male(z5) \wedge np\_rosier(z14) \wedge de(z14, z13) \wedge np\_male(z13) \wedge np\_male(z12) \wedge au(z10, z11) \wedge np\_garage(z11) \wedge np\_voiture(z10) \wedge de(z10, z9) \wedge np\_male(z9) \wedge np\_voisin(z8) \wedge de(z8, z7) \wedge np\_male(z7) \wedge \forall\ z4.(np\_soir(z4) \rightarrow \exists\ z1.(a\_topic(z4) \wedge \exists\ z2.(Recipient(z2, z10) \wedge Theme(z2, z8) \wedge a\_mettre(z2)) \wedge alors(z4, z1))) \wedge Theme(z3, z14) \wedge Actor(z3, z12) \wedge a\_arroser(z3) \wedge avec(z14, z6))$

FIGURE 7.6: FOL output of the utterance

7.4.2 Error Analysis

Semantic parsing is a difficult task in the natural language processing field. We obtain a parse of French discourse step by step, and in order to obtain the semantic representation accurately, we have to ensure accuracy of previous analysis stages. If there is an error in them, this will fatally lead to errors in the results. For example, incorrect POS tags are one of the leading causes of erroneous results. Also phrases can be inherently ambiguous and therefore can have more than one syntax trees, such as *la belle porte le voile* where *belle*, *porte*, *voile* can be both noun/verb/noun or adjective/noun/verb. In addition, complex linguistic issues arise in the processing of utterances, where omission of a word or of a group of words—which otherwise are necessary for the grammatical completeness of a sentence—is tolerated. These issues often result in incorrect identification of verb arguments. For example, in *Henri veut aller au parc et sa mère à la bibliothèque* (Henri wants to go to the park and his mother to the library), the absence of a verb between words *mère* and *à la bibliothèque* may result in obtaining incorrect lexical categories for the remaining words.

7.5 Conclusion of the Chapter

We have proposed an empirical approach towards building a semantic representation application for the French language, based on the CCG framework (for analyzing in the scope of the sentence) and on DRS (for dealing with semantic relations between the sentences of a discourse). Syntactic information and dependency relations between words are analyzed and extracted using a dependency parser. After that, the information is used to build a CCG derivation tree for each sentence. Finally, sentences in the form of CCG derivation trees are treated by Boxer, which we have adapted to French language by direct intervention on the source code, and we obtain a semantic representation of the discourse in FOL or in boxed format. In future research, we plan to build a corpus with discourses and their meaning representation in DRS form, using this application. We also plan to use deep neural network models to improve the robustness of the results obtained.

Chapter 8

Conclusion and Perspectives

In this thesis, we concentrated on study different aspects for building a meaning representation framework from natural language texts which are increasingly generated by conversations and discussions on the social networking platforms. In a general context, the rise of many social networking web-based services along with internet-based applications across multiple mobile platforms allow user to create and publish theirs content freely and without limitations. User-generated contents are created in the form of short text, articles, images, videos or mixed of other formats, depending on what kind of data the social networking platform supports. In our research, we dealt with user-generated texts that can be easily created by the vast majority of users and can be used to explore the users' sentiments or intentions. This helps to reduce the effort of agents in an organization or company that are responsible for gathering or receiving information, and for interacting with their customers on social networking platforms.

Capturing and representing semantics of natural language texts has been realized by many different research teams working with diverse methods, paradigms and ideologies for many different purposes, such as machine translation, question answering, automated reasoning or code generation. In particular, meaning representation of a given sentence requires not only the identification of roles and objects in the sentence but also the use of automated reasoning. Meaning representation also enables exploring and analyzing data by parsing natural language utterance corresponding to databases, or parsing questions, commands from conversational agents or chatbots such as Alexa, Siri, Cotana. More generally, the most basic idea of meaning representation is how to achieve a machine-understandable representation of given natural language utterances.

With the purpose of finding an effective way of analysis and representation of semantics for given natural language utterances, we have examined and discussed the various prominent works ranging from using rule-based methods to current deep neural network approaches. With the limitation of a massive amount of data about pairs of natural language utterances and their meaning representations that have been becoming a mandatory requirement for the deep learning approaches, we have decided to use the empirical approach and proposed a general architecture for a meaning representation framework using French natural language input. In the first module of the architecture, we start by analyzing morphological information for each word. From that, we achieve lemma, POS tag and features of the word in text. These crucial data are used as input for extracting relationships between words and constituents by using dependency grammar. As the result of this step, we obtain the syntactic and dependency information of every word of the input utterance.

In the second module in the architecture, the bridge between syntax and semantic is constructed based on CCGs, and this helps us in obtaining a syntax-semantic transparent interface. The result of the previous module is employed as input of the process of extraction of a CCG derivation tree. More particularly, this process consists of two stages: the first one is the task of assignment of lexical categories to each word depending on its position and its relationship with other words in the sentence; the second one focuses on the binarization of the dependency tree into a binary tree. Parsing of the CCG derivation tree is realized on the binary tree by applying combinatory rules defined in CCG theory.

Analysis of CCGs is encapsulated within the scope of a single sentence. Nevertheless, a natural language utterance can regularly be created with more than two sentences. The analysis of an utterance will need to be considered in its context. Therefore, in the general meaning context of the utterance, each sentence will contribute information to the common context—and sometimes may change its meaning. Otherwise, the context also gives information to explain the meaning of the sentence. In the last module of our proposed architecture, we construct a meaning representation for a given utterance based on DRT theory, DRS format and the Boxer tool by Johan Bos and is team. Accordingly, the analysis result of CCG derivation tree of the previous module is regarded as the input for this module, besides of the additional information about chunks and named entities in the sentence. The transformation of the input CCG derivation tree into a DRS format allows us to process linguistic phenomena such as anaphora, coreference and so on. As a result, we obtain ether a logical form or data in DRS boxing format.

8.1 Limitations and Perspectives

8.1.1 Limitations

We attempted to explore, propose and build the most appropriate solution for a general model aiming to capture meaning representation from natural language input. However, there are still many limitations that need to be tackled:

- The building of CCG derivation trees for each given sentence must employ syntax and dependency information obtained by other tools. Therefore, we are dependent on the accuracy of results provided by these tools. In other words, our approach can collapse completely when syntax and dependency information for the input is not correct as expected.
- In the CCG grammar formalism, a sentence can own more than one CCG derivation trees, depending on the relationships between constituents of the sentence. However our approach provides always only a single CCG derivation tree for each sentence because the CCG derivation tree is a directly obtained from the structure of the dependency tree.
- The diversity and complexity of French syntax structures has caused many difficulties in the analysis process. Thus, the results achieved for the semantic representation framework are still weak and an extra effort is needed to improve them by adding special semantic patterns for French language, as well as handling mutli-word problems.

8.1.2 Perspectives

In the NLP field, meaning representation or semantic interpretation plays nowadays a crucial role in the building of machine-understandable representation systems for natural languages such as conversational agents, robotic navigation, language translation, automated reasoning and so on. From the result of our work, we propose a summary of some promising research directions:

- The construction of French CCG parser becomes an indispensable requirement because its application potential is quite large, and this for various NLP tasks. With results obtained in this thesis such as the French CCG corpus, the CCG supertagging model and a state-of-the-art in the development of a CCG parser, we believe that the complete implementation of such a tool is perfectly feasible.
- In the CCG supertagging model, we have proven that the usage of embedding features from lemma, postag and dependency information helps to improvd the accuracy of our model in comparison with other models. The transformation of characters, words, or

sentences into a numerical vector representation is a mandatory requirement when using deep learning techniques. Based on these transformations, we have operations such as character embedding, word embedding and sentence embedding, respectively. Similarly, we can propose a novel feature embedding, called *meaning embedding* or meaning vector representation. Using our work, we can obtain a DRS or logical form representation for each natural language utterance.

Appendix A

Supporting Information

A.1 The λ-calculus

In order to obtain meaning representation, we need to fulfill different steps and a number of tasks as in the architecture we propose. In this process, First-Order Logic augmented with λ-calculus is employed as an inference-allowing formalism between human language and interpreted machine language. We can capture and represent the semantic representations for lexical items and then use compositionality to obtain semantic representation of constituents, in First-Order Logic and λ-calculus.

Lambda calculus was first introduced by Alonzo Church in the 1930s, in his research on the foundations of mathematics (Church, 1936). λ-calculus is essentially a branch in mathematical logic used for expressing computations based on abstraction functions and applications that is rely on variable binding and substitution. Nowadays, λ-calculus has becomes quite popular with applications in different areas of mathematics, linguistics and computer science. More specifically, λ-calculus is a foundational theory for the development of functional programming languages. λ-calculus also plays a crucial role in the semantic representation of natural language in computational linguistics.

In general, We distinguish between typed λ-calculus and untyped λ-calculus. Typed λ-calculus includes constraints about types of variables and functions, in which the type of a variable, which plays the role of argument of a function, must be compatible with the type of this function. By contrast, there is no restriction involving type in untyped λ-calculus. In formal linguistics, we are mostly interested with typed λ-calculus.

λ-calculus is regarded as a formal language L_λ. The expressions of λ-calculus are called λ-expressions or λ-terms. Essentially, a λ-calculus expression consists of symbols for variables and functions, the symbol λ and parentheses. If $\mathcal{E}$ is the set of all λ-expressions, and $x, y, z, \ldots$ are variables, then we have the following properties:

- if x is a variable, then x belong to $\mathcal{E}$;
- if x is a variable and $P \in \mathcal{E}$, then $\lambda x.P$ (we call it an *abstraction*) belongs to $\mathcal{E}$;
- if $Q \in \mathcal{E}$ and $S \in \mathcal{E}$, then the *application* of Q to S, denoted QS, belongs to $\mathcal{E}$.

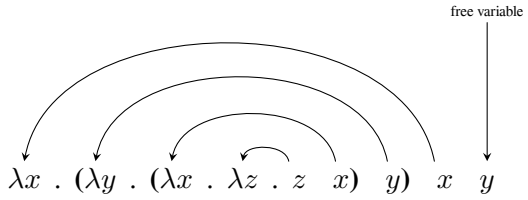
For instance, here are some λ-expressions and the corresponding explanations:

- Px – a function P applied to an argument x;
- $(Px)\,y$ – a function P that applied to two arguments x, y (in fact, a function P applied to x, and the result applied to y), this is equivalent to the mathematical expression $P(x, y)$;
- $P(Qx)$ – a function P applied to the result of applying a function Q to x, this is equivalent to the mathematical expression $P \circ Q(x)$;
- $\lambda x.x$ – the identify function (notice that this is a function, it can be applied to any other function: $(\lambda x.x)(P) = P$;

- $\lambda x.(\lambda y.x\ z)\ v$ – an abstraction and application.

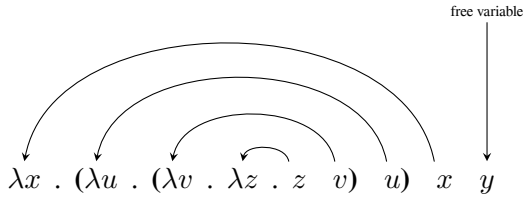
A variable is a symbol used to represent an unspecified term or basic object. We have *bound* variables that are used in the scope of an abstraction, and free variables. For instance, given a λ-expression as in example 16, we have two kinds of variables: v and z are free variable because there is no abstraction operator λ bounding them, whereas x and y, are bound variables. Here is an example of various bound variables with different scopes:

(15)



Bound variables can be renamed to increase readability of the λ-expression by a human:

(16)



λ-expressions that contain no free variable are called *closed* and they are equivalent to terms in combinatory logic.

We have some writing conventions to reduce the number of parentheses:

- function applications is left-associative: $(P_1P_2P_3\dots P_n) := (((P_1P_2)P_3)\dots P_n)$;
- variables are listed after λ: $\lambda xyz.P := \lambda x.\lambda y.\lambda z.P$;
- the body of abstractions extends to the right: $\lambda x.Q\ S := \lambda x.(Q\ S)$.

Let us give a First-Order Logic+λ-expression example:

(17) a. $\lambda x.\lambda y.\text{watch}(y, x)$

b. $(\lambda x.\lambda y.\text{watch}(y, x))(\text{television}, \text{henri})$

c. $\lambda y.\text{watch}(y, \text{henri})(\text{television}) \Rightarrow \text{watch}(\text{television}, \text{henri})$.

We have two variables x, y corresponding to prefixes λx, λy that bind the occurrences of x, y in the expression (17).a. The aim of abstracting over two variables is to mark the slots in which we want the substitution to be made, as in expression (17).b. With argument $(television, henri)$ and functor $\lambda x.\lambda y.watch(y, x)$, the operation of (17).b is also called functional application. Finally in (17).c give we reduce the functional application by replacing λ-bound variables by arguments.

The process of simplification and evaluation clarifies the meaning of a given λ-expression. Simplifying and evaluating λ-expressions is reducing the expression until there is no more reduction. More generally, we have four kinds of transformation and reduction operations defined on λ-expressions, namely α-conversion, substitution, β-reduction and η-reduction:

- α-conversion is the process of changing names of bound variables used in λ-expressions. For instance, from the λ-expression $\lambda x.\lambda y.\text{own}(y, x)$, we can obtain a new λ-expression $\lambda z.\lambda y.\text{own}(y, z)$. New λ-expressions that differ only by α-conversion are called α-equivalent. We regularly apply α-conversion before carrying out substitution because the variables must be distinguished. The aim is to prevent errors in the binding process.

- Substitution is considered as the process of replacing all occurrences of a variable by a λ-expression. Thus, the substitution, written as [expression to be substituted/variable to substitue] is defined as:

  - If x is a variable and P is a λ-expression, then $x[P/x] \equiv P$
  - If x, y are variables and P is a λ-expression, then $y[P/x] \equiv y$, $\forall x \neq y$
  - If x is a variable and $P,\ Q,\ S$ are λ-expressions, then $(P\ Q)[S/x] \equiv (P[Q/x])(Q[S/x])$
  - If x is a variable and P, Q areλ-expressions, then $\lambda x.P[Q/x] \equiv \lambda x.P$
  - If x, y are variables and P, Q are λ-expressions, then $(\lambda y.P)[Q/x] \equiv \lambda y.(P[Q/x])$, $\forall x \neq y$ and y must be bound variable in P
  - If x, y, z are variables and $P,\ Q$ are λ-expressions, then $(\lambda y.P)[Q/x] \equiv \lambda z.(P[z/y][Q/x])$, $\forall x \neq y$ and y must be bound variable in P

- β-reduction is regarded as a basic computational step of the λ-calculus that allows us to simplify λ-expressions. Conventionally, given a variable x and λ-expressions $P,\ Q$, β-reduction is defined by the following "β-rules":

$$\begin{array}{lll} (\lambda x.P)\ Q & \text{becomes} & P[Q/x]\text{by } \beta\text{-reduction} \\ P \longrightarrow_\beta Q & \Longrightarrow & (P\ S) \longrightarrow_\beta (Q\ S) \\ P \longrightarrow_\beta Q & \Longrightarrow & (S\ P) \longrightarrow_\beta (S\ Q) \\ P \longrightarrow_\beta Q & \Longrightarrow & \lambda x.P \longrightarrow_\beta \lambda x.Q \end{array} \tag{A.1}$$

  β-reduction have the same idea with functional applications.

- η-reduction is based on principle of extensionality, in which in the same context, two functions are identical if they return the same result for given arguments. Conventionally, if x is a bound variable in the λ-expression P, η-reduction is defined by the following rule:

$$\lambda x.(Px) \text{ becomes } P \text{ by } \eta\text{-reduction.} \tag{A.2}$$

  With η-reduction, we can switch from $\lambda x.(Px)$ to P and vice-versa, depending on each particular context. For instance, we can rewrite the λ-expression like $\lambda x.\text{eats}(x)$ simply as eats by applying η-reduction.

The purpose of applying transformations and β-reduction is to compute a value which is equivalent to a function in λ-calculus. We say that a λ-expression achieves a normal form, if it can not be reduce any more with the rules in β-reduction or η-reduction.

A.2 Combinatory Logic and The λ-calculus

The initial ideas of combinatory logic were introduced in a talk by Moses Schönfinkel in the 1920s, and a paper of his (Schönfinkel, 1924). Seven years later, Haskell Curry independently

rediscovered combinators and gave important contributions to foundations of logic and mathematics, as well as computational aspects (Curry, 1930). See (Seldin, 2006) for more details about the history of the development of this field.

In general, combinatory logic is an applicative language that becomes a foundational theory to construct functional programming languages such as Haskell, Brook and Curry language. Similar to λ-expressions in the λ-calculus, combinatory logic is constructed around the notion of *combinatory term* (combinatory expression). Conventionally, a combinatory term is made up of variables, atomic constants including basic combinators and applications. The set of combinatory terms $\mathcal{E}^{cl}$ is defined inductively as follows:

- Variable: if x is a variable, then x is a combinatory term.
- Primitive Function: if P is a basic combinator, then P is a combinatory term.
- Application: If M and N are combinatory terms, then $(M\ N)$ is a combinatory term.

Instead of abstraction and application like in the λ-calculus, combinatory calculus has only one operation—application—to manufacture functions. In addition, combinatory logic builds a limited set of primitive functions, based on basic combinators, besides other functions that can be constructed. Primitive functions essentially are combinators or functions that contain no bound variable. Therefore, the combinatory terms do not use the λ operator symbol.

Combinators play an crucial role in combinatory logic, in which they are considered as an abstract operator. Using combinators, we can obtain new "complex" operators from given operators. Based on original works of Haskell Curry and his colleagues (Curry et al., 1958), we have the following basic set of combinators (see also (Haralambous, 2019), an introduction to (Smullyan, 1985), which describes combinators as talking birds in a "certain enchanted forest"):

- **I**: This combinator expresses a variable as function of itself, it is called *elementary identificator*. The combinator **I** is defined as:

$$\begin{aligned} \mathbf{I}x &\equiv x \\ \mathbf{I} &\equiv \lambda x.x \end{aligned} \tag{A.3}$$

- **C**: give a function f of two arguments, we have combinator $\mathbf{C}f$, which is the converse. This combinator is called *elementary permutator*.

$$\begin{aligned} \mathbf{C}fxy &\equiv fyx \\ \mathbf{C} &\equiv \lambda fxy.fyx \end{aligned} \tag{A.4}$$

- **W**: given a function f of two arguments, we have combinator $\mathbf{W}f$, which has one argument getting duplicated into two arguments. This combinator is called *elementary duplicator*.

$$\begin{aligned} \mathbf{W}fx &\equiv fxx \\ \mathbf{W} &\equiv \lambda fxy.fxx \end{aligned} \tag{A.5}$$

- **K**: given a constant c, this constant can be expressed as a function with combinator **K**, which is called *elementary cancellator*.

$$\begin{aligned} \mathbf{K}cx &\equiv c \\ \mathbf{K} &\equiv \lambda fx.f \equiv \lambda xy.x \end{aligned} \tag{A.6}$$

- **B**: given two functions f and g, combinator **B** expresses their composition. This combinator is called *elementary compositor*.

$$\begin{aligned} \mathbf{B}fgx &\equiv f(gx) \\ \mathbf{B} &\equiv \lambda fgx.f(gx) \equiv \lambda fxy.f(xy) \end{aligned} \tag{A.7}$$

- **S**: given two terms f, g, the combinator **S** intertwines them as functions of x:

$$\begin{aligned} \mathbf{S}fgx &\equiv fx(gx) \\ \mathbf{S} &\equiv \lambda fgx.fx(gx) \equiv \lambda xyz.xz(yz) \end{aligned} \tag{A.8}$$

- **Φ**: similarly to **S**, this combinator intertwines the two first arguments of a function f:

$$\begin{aligned} \mathbf{\Phi}fabx &\equiv f(ax)(bx) \\ \mathbf{\Phi} &\equiv \lambda fghx.f(gx)(hx) \equiv \lambda fxyz.f(xz)(yz) \end{aligned} \tag{A.9}$$

- **Ψ**: a variation of **Φ** where a function is distributed into the arguments of another function:

$$\begin{aligned} \mathbf{\Psi}fgxy &\equiv f(gx)(gy) \\ \mathbf{\Psi} &\equiv \lambda fgxy.f(gx)(gy) \equiv \lambda fxyz.f(xy)(xz) \end{aligned} \tag{A.10}$$

The commonly used reducibility relation in combinatory calculus is called weak reducibility $(\rightarrow_w)$. A weak reduction of a relation $M \longrightarrow_w N$ is defined inductively as the following clauses:

1. $M \longrightarrow_w M$.
2. $\mathbf{S}fgx \longrightarrow_w fx(gx)$, $\mathbf{K}cx \longrightarrow_w c$, $\mathbf{I}x \longrightarrow_w x$.
3. If $M \longrightarrow_w N$ and $N \longrightarrow_w O$, then $M \longrightarrow_w O$.
4. If $M \longrightarrow_w N$, then $OM \longrightarrow_w ON$.
5. If $M \longrightarrow_w N$, then $MO \longrightarrow_w NO$.

All combinators can be generated or derived from basic combinators. We can see that any combinator can be rewritten by an equivalent λ-expression. In standard combinatory calculus, the basic combinators has only combinators **I**, **K** and **S** because we can redefine other combinators by using them[1]. For instance, the combinator **B** can be redefined as:

$$\begin{aligned} \mathbf{B}fgx &\equiv \mathbf{S}(\mathbf{KS})\mathbf{K}fgx & \\ &\longrightarrow_w \mathbf{KS}f(\mathbf{K}f)gx & \text{by constracting } \mathbf{S}(\mathbf{KS})\mathbf{K}f \text{ to } \mathbf{KS}f(\mathbf{K}f) \\ &\longrightarrow_w \mathbf{S}(\mathbf{K}f)fg & \text{by constracting } \mathbf{KS}f \text{ to } \mathbf{S} \\ &\longrightarrow_w \mathbf{K}fx(gx) & \text{by constracting } \mathbf{S}(\mathbf{K}f)gx \\ &\longrightarrow_w f(gx) & \text{by constracting } \mathbf{K}fx. \end{aligned} \tag{A.11}$$

[1] In fact, as can be seen in (Haralambous, 2019), $\{\mathbf{S}, \mathbf{K}\}$ is already a basis of the set of combinators, and this basis can be reduced to a single element ι defined as $\iota x := x\mathbf{SK}$.

TABLE A.1: A comparison of rules between λ-calculus and combinatory logic

Combinatory logic		λ-calculus	
Axiom-schemes			
(**I**)	**I**x = X	(α)	$\lambda x.P = \lambda y.[y/x]Q$ if y is not a free variable in Q
(**K**)	**K**cx = c		
(**S**)	**S**fgx = fx(gx)	(β)	$(\lambda x.P)Q = [Q/x]P$
(ρ)	$M = M$	(ρ)	$P = P$
Rules of inferences			
(μ)	$M = N \Longrightarrow OM = ON$	(μ)	$P = Q \Longrightarrow SP = SQ$
(υ)	$M = N \Longrightarrow MO = NO$	(υ)	$P = Q \Longrightarrow PS = QS$
(τ)	$M = N$ and $N = O \Longrightarrow M = O$	(υ)	$P = Q$ and $Q = S \Longrightarrow P = S$
(σ)	$M = N \Longrightarrow N = M$	(υ)	$P = Q \Longrightarrow Q = P$
(ξ)	$M = N \Longrightarrow [x].M = [x].N$	(ξ)	$P = Q \Longrightarrow \lambda x.P = \lambda x.Q$

A combinatory term M is in *weak normal form* if and only if M does not contain any term among $\mathbf{I}x$, $\mathbf{K}fx$, $\mathbf{S}fgx$ (weak redex). Furthermore, we have also a *strong normal form*, which is based on strong reduction. We say that the combinatory term N is a strong reduction of the term M, which expresses as $M >\!\!- N$, if and only if $M >\!\!- N$ is provable by using axioms and rules of combinatory logic.

In general, combinatory logic and λ-calculus are two systems of logic that share lot of common features. For instance, in Table A.1, we can see that there is an equivalence between the two formal systems through inference rules such as reflexiveness, transitivity, symmetry, right and left monotony. We can have an equivalence transformation from combinatory terms to λ-expressions (a "λ-mapping"), as follows:

- $x_\lambda \equiv x$.
- $\mathbf{I}_\lambda \equiv \lambda x.x$, $\mathbf{K}_\lambda \equiv \lambda xy.x$, $\mathbf{S}_\lambda \equiv \lambda xyz.xz(yz)$.
- $(M\ N)_\lambda \equiv M_\lambda\ N_\lambda$.

By contrast, we can also make an equivalent transformation from λ-expression to combinatory terms which is called "H_η-mapping" and is defined inductively as follows:

- $x_{H_\eta} \equiv x$.
- $(P\ Q)_{H_\eta} \equiv P_{H_\eta}\ Q_{H_\eta}$.
- $(\lambda x.P)_{H_\eta} \equiv [x]^\eta.(P_{H_\eta})$

Bibliography

Abelson, Robert and Roger C. Schank (1977). *Scripts, plans, goals and understanding. An inquiry into human knowledge structures*. New Jersey: Psychology Press.

Abend, Omri and Ari Rappoport (2013). "Universal conceptual cognitive annotation (UCCA)". In: *Proceedings of ACL 2013, Sofia, Bulgaria*, pp. 228–238.

Abney, Steven P. (1991). "Parsing by chunks". In: *Principle-based parsing*. Springer, pp. 257–278.

Abzianidze, Lasha et al. (2017). "The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations". In: *Proceedings of EACL 2017, Valencia, Spain*.

Ahn, Kisuh et al. (2005). "Question Answering with QED at TREC 2005". In: *Proceedings of TREC 2005, Gaithersburg, Maryland*.

Aichner, Thomas and Frank Jacob (2015). "Measuring the degree of corporate social media use". In: *International Journal of Market Research* 57.2, pp. 257–276.

Ajdukiewicz, Kazimierz (1935). "Die syntaktische Konnexität". In: *Studia philosophica* 1, pp. 1–27.

Akbik, Alan, Tanja Bergmann, and Roland Vollgraf (2019). "Pooled Contextualized Embeddings for Named Entity Recognition". In: *Proceedings of NAACL 2019, Minneapolis, MN*, pp. 724–728.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). "Contextual string embeddings for sequence labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics, Melbourne, Australia*, pp. 1638–1649.

Akbik, Alan et al. (June 2019). "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". In: *Proceedings of NAACL 2019, Minneapolis, MN*. Minneapolis, Minnesota, pp. 54–59.

Akhundov, Adnan, Dietrich Trautmann, and Georg Groh (2018). "Sequence labeling: A practical approach". arXiv:1808.03926.

Allen, James (1995). *Natural language understanding*. Pearson.

Allen, James et al. (2007). "Deep linguistic processing for spoken dialogue systems". In: *Proceedings of the Workshop on Deep Linguistic Processing, Prague, Czech Republic*, pp. 49–56.

Ambati, Bharat Ram, Tejaswini Deoskar, and Mark Steedman (2016). "Shift-reduce CCG parsing using neural network models". In: *Proceedings of NAACL 2016, San Diego, California*, pp. 447–453.

— (2018). "Hindi CCGbank: A CCG treebank from the Hindi dependency treebank". In: *Language Resources and Evaluation* 52.1, pp. 67–100.

Ambati, Bharat Ram et al. (2015). "An incremental algorithm for transition-based CCG parsing". In: *Proceedings of NAACL 2016, San Diego, CA*, pp. 53–63.

Amri, Samir and Lahbib Zenkouar (2018). "Amazigh POS Tagging Using TreeTagger: A Language Independent Model". In: *International Conference on Advanced Intelligent Systems for Sustainable Development, Tangier, Morocco*. Springer, pp. 622–632.

Andor, Daniel et al. (2016). "Globally normalized transition-based neural networks". arXiv:1603.06042.

Andreas, Jacob, Andreas Vlachos, and Stephen Clark (2013). "Semantic parsing as machine translation". In: *Proceedings of ACL 2013, Sofia, Bulgaria*, pp. 47–52.

Androutsopoulos, Ion, Graeme D. Ritchie, and Peter Thanisch (1995). "Natural language interfaces to databases–an introduction". In: *Natural language engineering* 1.1, pp. 29–81.

Antoine, Jean-Yves, Abdenour Mokrane, and Nathalie Friburger (2008). "Automatic Rich Annotation of Large Corpus of Conversational transcribed speech: the Chunking Task of the EPAC Project". In: *Proceedings of LREC 2008, Marrakech, Morocco*. Citeseer.

Armstrong, Susan, Pierrette Bouillon, and Gilbert Robert (1995). *Tagger overview (Technical Report)*. https://www.issco.unige.ch/en/staff/robert/tatoo/tagger.html. [Online; accessed 1-January-2020].

Artzi, Yoav and Luke Zettlemoyer (2011). "Bootstrapping semantic parsers from conversations". In: *Proceedings of EMNLP 2011, Edinburgh, Scotland*, pp. 421–432.

Auli, Michael and Adam Lopez (2011). "Efficient CCG parsing: A* versus adaptive supertagging". In: *Proceedings of ACL 2011, Portland OR*, pp. 1577–1585.

Avinesh, PVS and G. Karthik (2007). "Part-of-speech tagging and chunking using conditional random fields and transformation based learning". In: *Proceedings of Shallow Parsing for South Asian Languages, Hyderabad, India* 21.

Awasthi, Pranjal, Delip Rao, and Balaraman Ravindran (2006). "Part of speech tagging and chunking with HMM and CRF". In: *Proceedings of NLP Association of India (NLPAI) Machine Learning Contest 2006*.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe (1998). "The Berkeley FrameNet project". In: *Proceedings of the 17th international conference on Computational linguistics, Montréal, Québec*, pp. 86–90.

Baldi, Pierre et al. (2000). "Bidirectional dynamics for protein secondary structure prediction". In: *Sequence Learning*. Vol. 1828. Springer Lecture Notes in Computer Science. Springer, pp. 80–104.

Baldridge, Jason and Geert-Jan M. Kruijff (2003). "Multi-modal combinatory categorial grammar". In: *Proceedings of EACL 2003, Stroudsburg, PA*, pp. 211–218.

Baldridge, Jason, Thomas Morton, and Gann Bierner (2002). *The OpenNLP Maximum Entropy Package*. Tech. rep. URL: https://sourceforge.net/projects/maxent/.

Ballesteros, Miguel et al. (2016). "Training with exploration improves a greedy stack-LSTM parser". arXiv:1603.03793.

Ballesteros, Miguel et al. (2017). "Greedy transition-based dependency parsing with stack LSTMs". In: *Computational Linguistics* 43.2, pp. 311–347.

Banarescu, Laura et al. (2013). "Abstract meaning representation for sembanking". In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, Sofia, Bulgaria*, pp. 178–186.

Bar-Hillel, Yehoshua (1953). "A Quasi-Arithmetical Notation for Syntactic Description". In: *Language* 29, pp. 47–58.

Bar-Hillel, Yehoshua et al. (1960). *On categorial and phrase-structure grammars*. Weizmann Science Press.

Basile, Valerio et al. (2012). "Developing a large semantically annotated corpus". In: *Proceedings of LREC 2012, İstanbul, Turkey*, pp. 3196–3200.

Béchet, Frédéric (2013). *LIA_TAGG: A Statistical POS Tagger and Syntactic Bracketer*. https://gitlia.univ-avignon.fr/jean-francois.rey/otmedia/tree/630a18786e1f92f097d4df348f
tools/lia_ltbox/lia_tagg. [Online; accessed 1-January-2020].

Bellman, Richard (1966). "Dynamic programming". In: *Science* 153.3731, pp. 34–37.

Bengio, Yoshua, Paolo Frasconi, and Patrice Simard (1993). "The problem of learning long-term dependencies in recurrent networks". In: *IEEE International Conference on Neural Networks 1993*. IEEE, pp. 1183–1188.

Berners-Lee, Tim, James Hendler, Ora Lassila, et al. (2001). "The semantic web". In: *Scientific american* 284.5, pp. 28–37.

Birch, Alexandra, Miles Osborne, and Philipp Koehn (2007). "CCG supertags in factored statistical machine translation". In: *Proceedings of the second Workshop on Statistical Machine Translation, Prague, Czech Republic*, pp. 9–16.

Biskri, Ismaïl, Jean Pierre Desclés, and Christophe Jouis (1997). "La Grammaire Catégorielle Combinatoire Applicative appliquée au français". In: *Actes de LTT 1997, Tunis*, pp. 25–27.

Blache, Philippe et al. (2016). "4Couv: A New Treebank for French". In: *Proceedings of LREC 2016, Portorož, Slovenia*, pp. 1546–1551.

Blackburn, Patrick et al. (1999). "Inference and computational semantics". In: *Third International Workshop on Computational Semantics (IWCS-3), Tilburg, The Netherlands*.

Blanc, Olivier et al. (2010). "Partial Parsing of Spontaneous Spoken French". In: *Proceedings of LREC 2010, Valletta, Malta*. Citeseer.

Bohnet, Bernd et al. (2018). "Morphosyntactic tagging with a meta-biLSTM model over context sensitive token encodings". arXiv:1805.08237.

Booth, Taylor L. (1969). "Probabilistic representation of formal languages". In: *10th annual symposium on Switching and Automata Theory (SWAT 1969), Waterloo, Ontario*. IEEE, pp. 74–81.

Bos, Johan (2003). "Implementing the binding and accommodation theory for anaphora resolution and presupposition projection". In: *Computational Linguistics* 29.2, pp. 179–210.

— (2004). "Computational semantics in discourse: Underspecification, resolution, and inference". In: *Journal of Logic, Language and Information* 13.2, pp. 139–157.

— (2008). "Wide-coverage semantic analysis with boxer". In: *Proceedings of the 2008 Conference on Semantics in Text Processing, Stroudsburg, PA*, pp. 277–286.

— (2015). "Open-domain semantic parsing with boxer". In: *Proceedings of the 20th nordic conference of computational linguistics (NODALIDA 2015), Vilnius, Lithuania*, pp. 301–304.

Bos, Johan, Cristina Bosco, and Alessandro Mazzei (2009). "Converting a dependency treebank to a categorial grammar treebank for Italian". In: *Eighth international workshop on treebanks and linguistic theories (TLT8), Milan, Italy*. Educatt, pp. 27–38.

Bos, Johan et al. (2017). "The Groningen Meaning Bank". In: *Handbook of Linguistic Annotation*. Ed. by Nancy Ide and James Pustejovsky. Vol. 2. Springer, pp. 463–496.

Boucher, Paul (2010). "L'interrogation partielle en français: l'interface syntaxe/sémantique". In: *Syntaxe et sémantique* 1, pp. 55–82.

Boyd, Danah M. and Nicole B. Ellison (2007). "Social network sites: Definition, history, and scholarship". In: *Journal of computer-mediated Communication* 13.1, pp. 210–230.

Brants, Thorsten (2000). "TnT: a statistical part-of-speech tagger". In: *Proceedings of the sixth conference on Applied natural language processing, Seattle, WA*, pp. 224–231.

Bresnan, Joan (1982). *The mental representation of grammatical relations*. MIT Press.

Candito, Marie, Benoît Crabbé, and Pascal Denis (2010). "Statistical French dependency parsing: treebank conversion and first results". In: *Proceedings of LREC 2010, Valletta, Malta*, pp. 1840–1847.

Candito, Marie, Benoît Crabbé, and Mathieu Falco (2009). "Dépendances syntaxiques de surface pour le français". In: *Rapport technique, Université Paris* 7.

Candito, Marie and Djamé Seddah (2012). "Le corpus Sequoia: annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical (The Sequoia Corpus: Syntactic Annotation and Use for a Parser Lexical Domain Adaptation Method)[in French]". In: *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN*, pp. 321–334.

Candito, Marie et al. "Deep syntax annotation of the Sequoia French treebank". In: *Proceedings of LREC 2014, Reykjavik, Iceland*.

Candito, Marie et al. (2009). "Analyse syntaxique du français: des constituants aux dépendances". In: *Actes de TALN 2009, Senlis, France*.

Candito, Marie et al. (2010). “Benchmarking of statistical dependency parsers for French”. In: *Proceedings of COLING 2010, Beijing, China*, pp. 108–116.

Çakıcı, Ruken (2005). “Automatic induction of a CCG grammar for Turkish”. In: *Proceedings of the ACL student research workshop, Ann Arbor, MI*, pp. 73–78.

Chang, Chih-Chung and Chih-Jen Lin (2011). “LIBSVM: A library for support vector machines”. In: *ACM transactions on intelligent systems and technology (TIST)* 2.3, p. 27.

Chaplot, Devendra Singh and Ruslan Salakhutdinov (2018). “Knowledge-based word sense disambiguation using topic models”. In: *Proceedings of AAAI-18, New Orleans, LA*.

Charniak, Eugene (2000). “A maximum-entropy-inspired parser”. In: *Proceedings of NAACL 2000, Seattle, WA*, pp. 132–139.

Chen, Aihui et al. (2014). “Classifying, measuring, and predicting users’ overall active behavior on social networking sites”. In: *Journal of Management Information Systems* 31.3, pp. 213–253.

Chen, Danqi and Christopher Manning (2014). “A fast and accurate dependency parser using neural networks”. In: *Proceedings of EMNLP 2014, Doha, Qatar*, pp. 740–750.

Chiang, David (2005). “A hierarchical phrase-based model for statistical machine translation”. In: *Proceedings of ACL 2005, Ann Arbor, MI*, pp. 263–270.

Cho, Kyunghyun et al. (2014). “On the properties of neural machine translation: Encoder-decoder approaches”. arXiv:1409.1259.

Choi, Jinho D. (2016). “Dynamic feature induction: The last gist to the state-of-the-art”. In: *Proceedings of NAACL 2016, San Diego, CA*, pp. 271–281.

Choi, Jinho D. and Martha Palmer (2012). “Fast and robust part-of-speech tagging using dynamic model selection”. In: *Proceedings of ACL 2012, Jeju Island, Korea*, pp. 363–367.

Chollet, François (2018). *Deep Learning with Python*. Manning Publications.

Chomsky, Noam (1956). “Three models for the description of language”. In: *IRE Transactions on information theory* 2.3, pp. 113–124.

— (1957). “Syntactic Structures (The Hague: Mouton, 1957)”. In: *Review of Verbal Behavior by BF Skinner, Language* 35, pp. 26–58.

— (1959). “On certain formal properties of grammars”. In: *Information and control* 2.2, pp. 137–167.

— (1975). *The logical structure of linguistic theory*. Kluwer Academic Publishers.

Chrupała, Grzegorz, Georgiana Dinu, and Josef Van Genabith (2008). “Learning morphology with morfette”. In: *Proceedings of LREC 2008, Marrakech, Morocco*, pp. 2362–2367.

Chu, Yoeng-Jin (1965). “On the shortest arborescence of a directed graph”. In: *Scientia Sinica* 14, pp. 1396–1400.

Church, Alonzo (1936). “An unsolvable problem of elementary number theory”. In: *American journal of mathematics* 58.2, pp. 345–363.

Church, Kenneth W. and Robert L. Mercer (1993). “Introduction to the special issue on computational linguistics using large corpora”. In: *Computational linguistics* 19.1, pp. 1–24.

Clark, Stephen and James R. Curran (2003). “Log-linear models for wide-coverage CCG parsing”. In: *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pp. 97–104.

— (2004). “Parsing the WSJ using CCG and log-linear models”. In: *Proceedings of ACL 2004, Barcelona, Spain*, p. 103.

— (2007). “Wide-coverage efficient statistical parsing with CCG and log-linear models”. In: *Computational Linguistics* 33.4, pp. 493–552.

Clark, Stephen, Mark Steedman, and James R. Curran (2004). “Object-extraction and question-parsing using CCG”. In: *Proceedings of EMNLP 2004, Barcelona, Spain*.

Clément, Lionel, Bernard Lang, and Benoît Sagot (2004). “Morphology based automatic acquisition of large-coverage lexica”. In: *Proceedings of LREC 2004, Lisbon, Portugal*, pp. 1841–1844.

Cocke, John (1970). *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences.

Collins, Michael (2002). "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms". In: *Proceedings of EMNLP 2002, Philadelphia, PA*, pp. 1–8.

— (2003). "Head-driven statistical models for natural language parsing". In: *Computational linguistics* 29.4, pp. 589–637.

Collobert, Ronan (2011). "Deep learning for efficient discriminative parsing". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Ft. Lauddersdale, FL*, pp. 224–232.

Constant, Mathieu et al. (2011). "Intégrer des connaissances linguistiques dans un CRF: application à l'apprentissage d'un segmenteur-étiqueteur du français". In: *Actes de TALN 2011, Montpellier, France*. Vol. 1, p. 321.

Constant, Matthieu and Anthony Sigogne (2011). "MWU-aware part-of-speech tagging with a CRF model and lexical resources". In: *Proceedings of the workshop on multiword expressions: from parsing and generation to the real world*, pp. 49–56.

Covington, Michael A. (2001). "A fundamental algorithm for dependency parsing". In: *Proceedings of the 39th Annual ACM Southeast Conference, Athens, GA*. Citeseer, pp. 95–102.

Curran, James R., Stephen Clark, and Johan Bos (2007). "Linguistically motivated large-scale NLP with C&C and Boxer". In: *Proceedings of the 45th Annual Meeting of the ACL, Prague, Czech Republic*, pp. 33–36.

Curry, Haskell Brooks (1930). "Grundlagen der kombinatorischen Logik". In: *American journal of mathematics* 52.4, pp. 789–834.

Curry, Haskell Brooks et al. (1958). *Combinatory logic*. Vol. 1. North-Holland Amsterdam.

De Marneffe, Marie-Catherine, Bill MacCartney, Christopher D. Manning, et al. (2006). "Generating typed dependency parses from phrase structure parses". In: *Proceedings of LREC 2006, Genoa, Italy*. Vol. 6. 2006, pp. 449–454.

Debusmann, Ralph (2006). "Extensible Dependency Grammar: a modular grammar formalism based on multigraph description". PhD thesis. Universität Saarland, Saarbrücken.

Dekhtyar, Michael I., Alexandre Ja. Dikovsky, and Boris Karlov (2015). "Categorial dependency grammars". In: *Theor. Comput. Sci.* 579, pp. 33–63.

Deliri, Sepideh and Massimiliano Albanese (2015). "Security and privacy issues in social networks". In: *Data Management in Pervasive Systems*. Springer, pp. 195–209.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22.

Denis, Pascal and Benoît Sagot (2009). "Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort". In: *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*.

— (2010). "Exploitation d'une ressource lexicale pour la construction d'un étiqueteur morphosyntaxique état-de-l'art du français". In: *Actes de TALN 2010, Montréal, Québec*.

— (2012). "Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging". In: *Language resources and evaluation* 46.4, pp. 721–736.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". arXiv:1810.04805.

Dixon, Robert M.W. (2010a). *Basic Linguistic Theory. Grammatical Topics*. Oxford University Press.

— (2010b). *Basic linguistic theory volume 1: Methodology*. Vol. 1. Oxford University Press.

— (2012). *Basic Linguistic Theory: Further Grammatical Topics*. Oxford University Press.

Djordjevic, Bojan and James R. Curran (2006). "Efficient combinatory categorial grammar parsing". In: *Proceedings of the Australasian Language Technology Workshop 2006, Sydney, Australia*, pp. 3–10.

Djordjevic, Bojan, James R. Curran, and Stephen Clark (2007). "Improving the efficiency of a wide-coverage CCG parser". In: *Proceedings of the Tenth International Conference on Parsing Technologies, Prague, Czech Republic*, pp. 39–47.

Dong, Li and Mirella Lapata (2016). "Language to logical form with neural attention". arXiv:1601.01280.

Dozat, Timothy, Peng Qi, and Christopher D. Manning (2017). "Stanford's graph-based neural dependency parser at the conll 2017 shared task". In: *Proceedings of CoNLL 2017, Vancouver, Canada*, pp. 20–30.

Draicchio, Francesco et al. (2013). "Fred: From natural language text to RDF and OWL in one click". In: *Extended Semantic Web Conference, Montpellier, France*. Springer, pp. 263–267.

Duchier, Denys (1999). "Axiomatizing dependency parsing using set constraints". In: *Sixth Meeting on Mathematics of Language, Orlando, FL*. URL: https://www.ps.uni-saarland.de/Publications/documents/duchier-mol6.pdf.

Duchier, Denys and Ralph Debusmann (2001). "Topological dependency trees: A constraint-based account of linear precedence". In: *Proceedings of the ACL 2001, Toulouse, France*, pp. 180–187.

Earley, Jay (1970). "An efficient context-free parsing algorithm". In: *Communications of the ACM* 13.2, pp. 94–102.

Edmonds, Jack (1967). "Optimum branchings". In: *Journal of Research of the national Bureau of Standards B.* 71.4, pp. 233–240.

Edunov, Sergey et al. (2018). "Understanding back-translation at scale". arXiv:1808.09381.

Eisner, Jason (1997). "An empirical comparison of probability models for dependency grammar". arXiv:cmp-lg/9706004.

Eisner, Jason and Giorgio Satta (1999). "Efficient parsing for bilexical context-free grammars and head automaton grammars". In: *Proceedings of ACL 1999, College Park, Maryland*, pp. 457–464.

Eisner, Jason and Noah A. Smith (2005). "Parsing with soft and hard constraints on dependency length". In: *Proceedings of the Ninth International Workshop on Parsing Technology, Vancouver, British Columbia*, pp. 30–41.

Eisner, Jason M. (1996). "Three new probabilistic models for dependency parsing: An exploration". In: *Proceedings of COLING 1996, Copenhagen, Danemark*, pp. 340–345.

Ekbal, Asif, S. Mondal, and Sivaji Bandyopadhyay (2007). "POS Tagging using HMM and Rule-based Chunking". In: *The Proceedings of SPSAL, IJCAI, Hyderabad, India* 8.1, pp. 25–28.

El-Taher, Ahmed I. et al. (2014). "An Arabic CCG approach for determining constituent types from Arabic Treebank". In: *Journal of King Saud University-Computer and Information Sciences* 26.4, pp. 441–449.

Fan, Rong-En et al. (2008). "LIBLINEAR: A library for large linear classification". In: *Journal of machine learning research* 9.Aug, pp. 1871–1874.

Fan, Xing et al. (2017). "Transfer learning for neural semantic parsing". arXiv:1706.04326.

Fauconnier, Jean-Philippe (2015). "French Word Embeddings". http://fauconnier.github.io.

Federgruen, Awi and Michal Tzur (1991). "A simple forward algorithm to solve general dynamic lot sizing models with n periods in 0 (n log n) or 0 (n) time". In: *Management Science* 37.8, pp. 909–925.

Fernández-González, Daniel and Carlos Gómez-Rodríguez (2019). "Left-to-Right Dependency Parsing with Pointer Networks". arXiv:1903.08445.

Fillmore, Charles J. et al. (1976). "Frame semantics and the nature of language". In: *Annals of the New York Academy of Sciences: Conference on the origin and development of language and speech*. Vol. 280. 1, pp. 20–32.

Flanagan, David (2008). *MQL Reference Guide*. Freebase.

Forbes, Katherine et al. (2003). "D-LTAG system: Discourse parsing with a lexicalized tree-adjoining grammar". In: *Journal of Logic, Language and Information* 12.3, pp. 261–279.

Foreman, Curtis (2018). "Types of social media and how each can benefit your business". URL: https://blog.hootsuite.com/types-of-social-media/.

Forney, G. David (1973). "The Viterbi algorithm". In: *Proceedings of the IEEE* 61.3, pp. 268–278.

Gaifman, Haim (1965). "Dependency systems and phrase-structure systems". In: *Information and control* 8.3, pp. 304–337.

Gange, Graeme (2008). "433-460 Project: Chart Parsing for Combinatory Categorial Grammars". https://www.researchgate.net/publication/265141403_433-460_Project_Chart_Parsing_for_Combinatory_Categorial_Grammars.

Gangemi, Aldo et al. (2017). "Semantic web machine reading with FRED". In: *Semantic Web* 8.6, pp. 873–893.

Garg, Rajiv, Michael D. Smith, and Rahul Telang (2011). "Measuring information diffusion in an online community". In: *Journal of Management Information Systems* 28.2, pp. 11–38.

Garnelo, Marta and Murray Shanahan (2019). "Reconciling deep learning with symbolic artificial intelligence: representing objects and relations". In: *Current Opinion in Behavioral Sciences* 29, pp. 17–23.

Garside, RG, Geoffrey Leech, and Eric Atwell (1984). "The Grammatical Tagging of Unrestricted English Text". In: *International Conference on the Methodology and Techniques of Machine Translation, Cranfield, UK*.

Gazdar, Gerald et al. (1985). *Generalized phrase structure grammar*. Harvard University Press.

Ge, Ruifang and Raymond J. Mooney (2005). "A statistical semantic parser that integrates syntax and semantics". In: *Proceedings of the ninth conference on computational natural language learning*, pp. 9–16.

Gendner, Véronique et al. (2004). "Les annotations syntaxiques de référence PEAS". URL: http://atoll.inria.fr/passage/PEAS_reference_annotations_v2.html.

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins (1999). "Learning to forget: Continual prediction with LSTM". In: *Proceedings of ICANN 99, Edinburgh, UK*. IET.

Giménez, Jesús and Lluís Màrquez (2004). "SVMTool: A general POS tagger generator based on Support Vector Machines". In: *Proceedings of LREC 2004, Lisbon, Portugal*.

Goldberg, Yoav and Joakim Nivre (2012). "A dynamic oracle for arc-eager dependency parsing". In: *Proceedings of COLING 2012, Mumbai, India*, pp. 959–976.

— (2013). "Training deterministic parsers with non-deterministic oracles". In: *Transactions of the association for Computational Linguistics* 1, pp. 403–414.

Goller, Christoph and Andreas Kuchler (1996). "Learning task-dependent distributed representations by backpropagation through structure". In: *Proceedings of International Conference on Neural Networks (ICNN'96)*. Vol. 1. IEEE, pp. 347–352.

Gómez-Rodríguez, Carlos and Joakim Nivre (2013). "Divisible transition systems and multiplanar dependency parsing". In: *Computational Linguistics* 39.4, pp. 799–845.

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5-6, pp. 602–610.

Green, Spence et al. (2011). "Multiword expression identification with tree substitution grammars: A parsing tour de force with French". In: *Proceedings of EMNLP 2011, Edinburgh, UK*, pp. 725–735.

Grover, Claire and Richard Tobin (2006). "Rule-Based Chunking and Reusability." In: *Proceedings of LREC 2006, Genoa, Italy*, pp. 873–878.

Guillaume, Bruno and Guy Perrier (2015). "Dependency Parsing with Graph Rewriting". In: *Proceedings of the 14th International Conference on Parsing Technologies*, pp. 30–39.

Haoliang, Qi et al. (2007). "Packed Forest Chart Parser". In: *Journal of Computer Science* 3.1, pp. 9–13.

Haralambous, Yannis (2019). "Ne vous moquez pas de l'oiseau moqueur : un aperçu de la logique combinatoire, avec des applications en linguistique mathématique". In: *Quadrature* 113, pp. 22–34.

Harper, Mary P. and Randall A. Helzerman (1995). "Extensions to constraint dependency parsing for spoken language processing". In: *Computer Speech and Language* 9.3, pp. 187–234.

Harper, Mary P. et al. (1995). "Implementation issues in the development of the PARSEC parser". In: *Software: Practice and Experience* 25.8, pp. 831–862.

Harris, Zellig Sabbettai (1982). *A grammar of English on mathematical principles*. John Wiley & Sons Incorporated.

Hays, David (1960). "Grouping and dependency theories". In: *Proceedings of the National Symposium on Machine Translation*. UCLA, pp. 257–266.

— (1964a). "Dependency theory: A formalism and some observations". In: *Language* 40, pp. 159–525.

— (1967). *Introduction to computational linguistics*. London: Macdonald & Co.

Hays, David G. (1964b). "Dependency theory: A formalism and some observations". In: *Language* 40.4, pp. 511–525.

Hellwig, Peter (1986). "Dependency unification grammar". In: *Proceedings of COLING 1986, Bonn, Germany*.

Hendrix, Gary G. et al. (1978). "Developing a natural language interface to complex data". In: *ACM Transactions on Database Systems (TODS)* 3.2, pp. 105–147.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hockenmaier, Julia (2003). "Data and models for statistical parsing with Combinatory Categorial Grammar". PhD thesis. University of Edinburgh.

— (2006). "Creating a CCGbank and a wide-coverage CCG lexicon for German". In: *Proceedings of COLING/ACL 2006, Sydney, Australia*, pp. 505–512.

Hockenmaier, Julia and Mark Steedman (2007). "CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank". In: *Computational Linguistics* 33.3, pp. 355–396.

Honnibal, Matthew, Yoav Goldberg, and Mark Johnson (2013). "A non-monotonic arc-eager transition system for dependency parsing". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, Sofia, Bulgaria*, pp. 163–172.

Honnibal, Matthew and Mark Johnson (2015). "An improved non-monotonic transition system for dependency parsing". In: *Proceedings of EMNLP 2015, Lisbon, Portugal*, pp. 1373–1378.

Honnibal, Matthew, Jonathan K. Kummerfeld, and James R. Curran (2010). "Morphological analysis can improve a CCG parser for English". In: *Proceedings of COLING 2010, Beijing, China*, pp. 445–453.

Honnibal, Matthew and Ines Montani (2017). "Industrial-strength Natural Language Processing". `https://spacy.io`.

Hovy, Eduard et al. (2006). "OntoNotes: the 90% solution". In: *Proceedings of NAACL 2006, Minneapolis, MN*, pp. 57–60.

Huang, Liang, Wenbin Jiang, and Qun Liu (2009). "Bilingually-constrained (monolingual) shift-reduce parsing". In: *Proceedings of EMNLP 2009, Singapore*, pp. 1222–1231.

Huang, Luyao et al. (2019). "GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge". arXiv:1908.07245.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). "Bidirectional LSTM-CRF models for sequence tagging". arXiv:1508.01991.

Hudson, Richard and Richard A. Hudson (2007). *Language networks: The new word grammar*. Oxford University Press.

Hudson, Richard A. (1991). *English word grammar*. B. Blackwell.

Jackendoff, Ray (1992). *Semantic structures*. MIT press.

Järvinen, Timo and Pasi Tapanainen (1997). *A dependency parser for English*. Tech. rep. University of Helsinki, Department of General Linguistics.

Joachims, Thorsten (1998). "Text categorization with support vector machines: Learning with many relevant features". In: *European conference on machine learning, Chemnitz, Germany*. Springer, pp. 137–142.

Johansson, Richard and Pierre Nugues (2006). "Investigating multilingual dependency parsing". In: *Proceedings of CoNLL 2006, New York, NY*, pp. 206–210.

— (2007). "Incremental dependency parsing using online learning". In: *Proceedings of EMNLP/-CoNLL 2007, Prague, Czech Republic*, pp. 1134–1138.

Johansson, Stig, Geoffrey N. Leech, and Helen Goodluck (1978). *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computer*. Tech. rep. Department of English, University of Oslo.

Johnson, Tim (1984). "Natural language computing: the commercial applications". In: *The Knowledge Engineering Review* 1.3, pp. 11–23.

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi (1975). "Tree adjunct grammars". In: *Journal of computer and system sciences* 10.1, pp. 136–163.

Kadari, Rekia et al. (2018a). "CCG supertagging via Bidirectional LSTM-CRF neural architecture". In: *Neurocomputing* 283, pp. 31–37.

— (2018b). "CCG supertagging with bidirectional long short-term memory networks". In: *Natural Language Engineering* 24.1, pp. 77–90.

Kahane, Sylvain (2012). "Why to choose dependency rather than constituency for syntax: a formal point of view". In: *Meanings, Texts, and other exciting things: A Festschrift to Commemorate the 80th Anniversary of Professor Igor A. Mel′cuk*. Ed. by J. Apresjan et al. Moscow: Languages of Slavic Culture, pp. 257–272.

Kamp, Hans (1981). "A theory of truth and discourse representation". In: *Formal methods in the study of language*. Ed. by J. Groenendijk, T.M.V. Janssen, and M. Stockhof. 135. Mathematical Centre.

Kamp, Hans and Uwe Reyle (1993). *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Springer.

Kaplan, Ronald M. (1973). "A general syntactic processor". In: *Natural language processing* 8, pp. 194–241.

Kaplan, Ronald M. and Joan Bresnan (1982). "Lexical-Functional Grammar: A Formal System for Grammatical Representation". In: *The Mental Representations of Grammatical Relations*. Ed. by Joan Bresnan. The MIT Press, pp. 173–281.

Kaplan, Ronald M. et al. (2004). *Speed and accuracy in shallow and deep stochastic parsing*. Tech. rep. Palo Alto Research Center CA.

Karlberg, Michael (2011). "Discourse theory and peace". In: *The encyclopedia of peace psychology*.

Kasami, Tadao (1966). *An efficient recognition and syntax-analysis algorithm for context-free languages*. Tech. rep. Coordinated Science Laboratory Report no. R-257.

Kate, Rohit J. and Raymond J. Mooney (2006). "Using string-kernels for learning semantic parsers". In: *Proceedings of COLING/ACL 2006, Sydney, Australia*, pp. 913–920.

Kay, M. (1980). *Algorithm Schemata and Data Structures in Syntactic Processing*. Tech. rep. Xerox Palo Alto Research Center CSL-80-12.

Kim, J-D et al. (2003). "GENIA corpus—a semantically annotated corpus for bio-textmining". In: *Bioinformatics* 19.suppl_1, pp. i180–i182.

Kim, Yoon (2014). "Convolutional neural networks for sentence classification". arXiv:1408.5882.

Kiperwasser, Eliyahu and Yoav Goldberg (2016). "Simple and accurate dependency parsing using bidirectional LSTM feature representations". In: *Transactions of the Association for Computational Linguistics* 4, pp. 313–327.

Kitaev, Nikita and Dan Klein (2018). "Constituency parsing with a self-attentive encoder". arXiv:1805.01052.

Klein, Dan and Christopher D. Manning. "Fast Exact Inference with a Factored Model for Natural Language Parsing". In: Advances in Neural Information Processing Systems.

— (2003a). "A parsing: fast exact Viterbi parse selection". In: *Proceedings of NAACL 2003, Edmonton, Canada*, pp. 40–47.

— (2003b). "Accurate unlexicalized parsing". In: *Proceedings of ACL 2003, Sapporo, Japan*, pp. 423–430.

Kübler, Sandra, Ryan McDonald, and Joakim Nivre (2009). "Dependency parsing". In: *Synthesis Lectures on Human Language Technologies* 1.1, pp. 1–127.

Kudo, Taku and Yuji Matsumoto (2001). "Chunking with support vector machines". In: *Proceedings of NAACL 2001, Pittsburgh, PA*, pp. 1–8.

Kwiatkowski, Tom et al. (2010). "Inducing probabilistic CCG grammars from logical form with higher-order unification". In: *Proceedings of EMNLP 2010, Cambridge, MA*, pp. 1223–1233.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *Proceedings of ICML 2001, Williamstown, MA*, pp. 282–289.

Lake, Brenden M. et al. (2017). "Building machines that learn and think like people". In: *Behavioral and brain sciences* 40. DOI: Buildingmachinesthatlearnandthinklikepeople.

Lambek, Joachim (1988). "Categorial and categorical grammars". In: *Categorial grammars and natural language structures*. Springer, pp. 297–317.

Lawrence, Steve et al. (1997). "Face recognition: A convolutional neural-network approach". In: *IEEE transactions on neural networks* 8.1, pp. 98–113.

Le, Ngoc Luyen and Yannis Haralambous (2019). "CCG Supertagging Using Morphological and Dependency Syntax Information". In: *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLING)*.

Le, Ngoc luyen, Yannis Haralambous, and Philippe Lenca (2019). "Towards a DRS Parsing Framework for French". In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pp. 357–363.

Lee, Kenton, Mike Lewis, and Luke Zettlemoyer (2016). "Global neural CCG parsing with optimality guarantees". arXiv:1607.01432.

Lefeuvre, Anaïs, Richard Moot, and Christian Retoré (2012). "Traitement automatique d'un corpus de récits de voyages pyrénéens: analyse syntaxique, sémantique et pragmatique dans le cadre de la théorie des types". In: *SHS Web of Conferences*. Vol. 1. EDP Sciences, pp. 2485–2495.

Leiner, Barry M. et al. (2009). "A brief history of the Internet". In: *ACM SIGCOMM Computer Communication Review* 39.5, pp. 22–31.

Lewis, Martha (2019). "Compositionality for Recursive Neural Networks". arXiv:1901.10723.

Lewis, Mike, Luheng He, and Luke Zettlemoyer (2015). "Joint A* CCG parsing and semantic role labelling". In: *Proceedings of EMNLP 2015, Lisbon, Portugal*, pp. 1444–1454.

Lewis, Mike, Kenton Lee, and Luke Zettlemoyer (2016). "LSTM CCG parsing". In: *Proceedings of NAACL 2016, San Diego, California*, pp. 221–231.

Lewis, Mike and Mark Steedman (2014a). "A* CCG parsing with a supertag-factored model". In: *Proceedings of EMNLP 2014, Doha, Qatar*, pp. 990–1000.

— (2014b). “Improved CCG parsing with semi-supervised supertagging”. In: *Transactions of the ACL* 2, pp. 327–338.

Lhoneux, Miryam de, Miguel Ballesteros, and Joakim Nivre (2019). “Recursive Subtree Composition in LSTM-Based Dependency Parsing”. arXiv:1902.09781.

Lierler, Yuliya and Peter Schüller (2011). “Parsing combinatory categorial grammar with answer set programming: Preliminary report”. arXiv:1108.5567.

— (2012). “Parsing combinatory categorial grammar via planning in answer set programming”. In: *Correct Reasoning*. Springer, pp. 436–453.

Lim, Soojong, Changki Lee, and Dongyul Ra (2013). “Dependency-based semantic role labeling using sequence labeling with a structural SVM”. In: *Pattern Recognition Letters* 34.6, pp. 696–702.

Ling, Wang et al. (2015). “Finding function in form: Compositional character models for open vocabulary word representation”. arXiv:1508.02096.

Litman, Diane et al. (2009). “Using natural language processing to analyze tutorial dialogue corpora across domains and modalities”. In: *Artificial Intelligence in Education*. IOS Press, pp. 149–156.

Liu, Jiangming, Shay B. Cohen, and Mirella Lapata (2018). “Discourse representation structure parsing”. In: *Proceedings of ACL 2018, Melbourne, Australia*, pp. 429–439.

Liu, Jiangming and Yue Zhang (2017a). “In-order transition-based constituent parsing”. In: *Transactions of the Association for Computational Linguistics* 5, pp. 413–424.

— (2017b). “Shift-reduce constituent parsing with neural lookahead features”. In: *Transactions of the Association for Computational Linguistics* 5, pp. 45–58.

Liu, Yinhan et al. (2019). “Roberta: A robustly optimized bert pretraining approach”. arXiv:1907.11692.

Lodhi, Huma et al. (2002). “Text classification using string kernels”. In: *Journal of Machine Learning Research* 2.Feb, pp. 419–444.

Loula, Joao, Marco Baroni, and Brenden M. Lake (2018). “Rearranging the familiar: Testing compositional generalization in recurrent networks”. arXiv:1807.07545.

Ma, Xuezhe and Eduard Hovy (2016). “End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF”. arXiv:1603.01354.

Magerman, David M. (1994). “Natural language parsing as statistical pattern recognition”. arXiv:cmp-lg/9405009.

Marcus, Mitchell et al. (1994). “The Penn Treebank: annotating predicate argument structure”. In: *Proceedings of the Workshop on Human Language Technology, Plainsboro, NJ*, pp. 114–119.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini (1993). “Building a large annotated corpus of English: The Penn Treebank”. In: *Computational linguistics* 19.2, pp. 313–330.

Màrquez, Lluís and Horacio Rodríguez (1998). “Part-of-speech tagging using decision trees”. In: *Proceedings of the European Conference on Machine Learning, Chemnitz, Germany*. Springer, pp. 25–36.

Marsden, P. (2011). “Simple Definition of Social Commerce (with Word Cloud & Definitive Definition List)”. In: *Social Commerce Today*. URL: https://socialcommercetoday.com/social-commerce-definition-word-cloud-definitive-definition-list/.

Maruyama, Hiroshi (1990). “Structural disambiguation with constraint propagation”. In: *Proceedings of ACL 1990, Pittsburgh, PA*, pp. 31–38.

McDonald, Ryan (2006). “Discriminative training and spanning tree algorithms for dependency parsing”. PhD thesis. University of Pennsylvania.

McDonald, Ryan, Kevin Lerman, and Fernando Pereira (2006). “Multilingual dependency analysis with a two-stage discriminative parser”. In: *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pp. 216–220.

McDonald, Ryan and Joakim Nivre (2011). "Analyzing and integrating dependency parsers". In: *Computational Linguistics* 37.1, pp. 197–230.

McDonald, Ryan et al. (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*, pp. 523–530.

McDonald, Ryan et al. (2013). "Universal dependency annotation for multilingual parsing". In: *Proceedings ACL 2013, Sofia, Bulgaria*, pp. 92–97.

Mel′cuk, Igor Aleksandrovic et al. (1988). *Dependency syntax: theory and practice*. SUNY press.

Melis, Gábor, Tomáš Kočiskỳ, and Phil Blunsom (2019). "Mogrifier LSTM". arXiv:1909.01792.

Merlo, Paola and Gabriele Musillo (2008). "Semantic parsing for high-precision semantic role labelling". In: *Proceedings of CoNLL 2008, Manchester, UK*, pp. 1–8.

Mi, Haitao and Liang Huang (2015). "Shift-Reduce Constituency Parsing with Dynamic Programming and POS Tag Lattice". In: *Proceedings of NAACL 2015, Denver, CO*, pp. 1030–1035.

Mikolov, Tomas, Quoc V. Le, and Ilya Sutskever (2013). "Exploiting similarities among languages for machine translation". arXiv:1309.4168.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Proceedings of NeurIPS 2013, Lake Tahoe, NV*, pp. 3111–3119.

Mohammed, Mona Ali and Nazlia Omar (2011). "Rule based shallow parser for Arabic language". In: *Journal of Computer Science* 7.10, pp. 1505–1514.

Montague, Richard (1970a). "English as a formal language". In: URL: https://philpapers.org/rec/MONEAA-2.

— (1970b). "Pragmatics and intensional logic". In: *Synthese* 22.1-2, pp. 68–94.

— (1970c). "Universal grammar". In: *Theoria* 36.3, pp. 373–398.

Montanari, Ugo (1974). "Networks of constraints: Fundamental properties and applications to picture processing". In: *Information sciences* 7, pp. 95–132.

Moortgat, Michael (1997). "Categorial type logics". In: *Handbook of logic and language*. Elsevier, pp. 93–177.

Moot, Richard (1998). "Grail: An automated proof assistant for categorial grammar logics". In:

— (1999). "Grail: an interactive parser for categorial grammars". In: *Proceedings of VEXTAL*. Vol. 1999, pp. 255–261.

— (2010). "Wide-coverage French syntax and semantics using Grail". In: *Actes de TALN 2010, Montréal, Québec*.

— (2015). "A type-logical Treebank for French". In: *Journal of Language Modelling* 3.1, pp. 229–264.

Morrill, Glyn V. (1994). *Categorial Logic of Signs*.

— (2012). *Type logical grammar: Categorial logic of signs*. Springer Science & Business Media.

Moser, Christine, Peter Groenewegen, and Marleen Huysman (2013). "Extending social network analysis with discourse analysis: Combining relational with interpretive data". In: *The influence of technology on social network analysis and mining*. Springer, pp. 547–561.

Moshovos, Andreas et al. (2018). "Exploiting Typical Values to Accelerate Deep Learning". In: *Computer* 51.5, pp. 18–30.

Mrini, Khalil et al. (2019). "Rethinking Self-Attention: An Interpretable Self-Attentive Encoder-Decoder Parser". In: arXiv:1911.03875.

Nadejde, Maria et al. (2017). "Predicting target language CCG supertags improves neural machine translation". arXiv:1702.01147.

Nakorn, Tanin Na (2009). *Combinatory Categorial Grammar Parser in Natural Language Toolkit*.

Nasr, Alexis, Frédéric Béchet, and Alexandra Volanschi (2004). "Tagging with hidden Markov models using ambiguous tags". In: *Proceedings of COLING 2004, Geneva, Switzerland*, p. 569.

Ney, Hermann (1991). "Dynamic programming parsing for context-free grammars in continuous speech recognition". In: *IEEE Transactions on Signal Processing* 39.2, pp. 336–340.

Nguyen, Dat Quoc, Mark Dras, and Mark Johnson (2017). "A novel neural network model for joint POS tagging and graph-based dependency parsing". arXiv:1705.05952.

Nguyen, Dat Quoc et al. (2014). "RDRPOSTagger: A Ripple Down Rules-based Part-Of-Speech Tagger". In: *Proceedings of EACL 2014, Gothenburg, Sweden*, pp. 17–20.

— (2016). "A robust transformation-based learning approach using ripple down rules for part-of-speech tagging". In: *AI Communications* 29.3, pp. 409–422.

Nguyen, Le-Minh, Akira Shimazu, and Xuan-Hieu Phan (2006). "Semantic parsing with structured SVM ensemble classification models". In: *Proceedings of COLING/ACL 2006, Sydney, Australia*, pp. 619–626.

Nivre, Joakim (2003). "An efficient algorithm for projective dependency parsing". In: *Proceedings of the Eighth International Conference on Parsing Technologies, Nancy, France*, pp. 149–160.

— (2004). "Incrementality in deterministic dependency parsing". In: *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together, Barcelona, Spain*, pp. 50–57.

— (2008). "Algorithms for deterministic incremental dependency parsing". In: *Computational Linguistics* 34.4, pp. 513–553.

Nivre, Joakim, Johan Hall, and Jens Nilsson (2004). "Memory-based dependency parsing". In: *Proceedings of HLT-NAACL 2004, Boston, MA*, pp. 49–56.

— (2006). "Maltparser: A data-driven parser-generator for dependency parsing." In: *Proceedings of LREC 2006, Genoa, Italy*. Vol. 6, pp. 2216–2219.

Nivre, Joakim and Mario Scholz (2004). "Deterministic dependency parsing of English text". In: *Proceedings of COLING 2004, Geneva, Switzerland*, p. 64.

Nivre, Joakim et al. (2006). "Labeled pseudo-projective dependency parsing with support vector machines". In: *Proceedings of CoNLL 2006, New York, NY*, pp. 221–225.

Nivre, Joakim et al. (2016). "Universal dependencies v1: A multilingual treebank collection". In: *Proceedings of LREC 2016, Portorož, Slovenia*, pp. 1659–1666.

Noord, Rik van (2019). "Neural Boxer at the IWCS shared task on DRS parsing". In: *Proceedings of the IWCS Shared Task on Semantic Parsing, Gothenburg, Sweden*.

Noord, Rik van et al. (2018). "Exploring neural methods for parsing discourse representation structures". In: *Transactions of the Association for Computational Linguistics* 6, pp. 619–633.

Nothman, Joel et al. (2013). "Learning multilingual named entity recognition from Wikipedia". In: *Artificial Intelligence* 194, pp. 151–175.

Novak, Petra Kralj et al. (2015). "Sentiment of emojis". In: *PloS one* 10.12, e0144296.

Obar, Jonathan A. and Steven S. Wildman (2015). "Social media definition and the governance challenge-an introduction to the special issue". In: *Obar, JA and Wildman, S.(2015). Social media definition and the governance challenge: An introduction to the special issue. Telecommunications policy* 39.9, pp. 745–750.

Okumura, Akitoshi and Kazunori Muraki (1994). "Symmetric pattern matching analysis for English coordinate structures". In: *Proceedings of the fourth conference on Applied natural language processing*, pp. 41–46.

Osborne, Timothy (2019). *A Dependency Grammar of English*. John Benjamins.

Paliwal, Gunjan (2015). "Social media marketing: opportunities and challenges". PhD thesis. Massachusetts Institute of Technology. URL: http://hdl.handle.net/1721.1/98997.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury (2005). "The proposition bank: An annotated corpus of semantic roles". In: *Computational linguistics* 31.1, pp. 71–106.

Pammi, Sathish Chandra and Kishore Prahallad (2007). "POS tagging and chunking using decision forests". In: *IJCAI Workshop on Shallow Parsing for South Asian Languages, Hyderabad, India*, pp. 33–36.

Park, Do-Hyung, Jumin Lee, and Ingoo Han (2007). "The effect of on-line consumer reviews on consumer purchasing intention: The moderating role of involvement". In: *International journal of electronic commerce* 11.4, pp. 125–148.

Partee, Barbara (1975). "Montague grammar and transformational grammar". In: *Linguistic inquiry* 6.2, pp. 203–300.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning, Atlanta, GA*, pp. 1310–1318.

Pei, Wenzhe, Tao Ge, and Baobao Chang (2015). "An effective neural network model for graph-based dependency parsing". In: *Proceedings of ACL 2015, Denver, CO*, pp. 313–322.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of EMNLP 2014, Doha, Qatar*, pp. 1532–1543.

Petrov, Slav, Dipanjan Das, and Ryan McDonald (2011). "A universal part-of-speech tagset". arXiv:1104.2086.

Petrov, Slav and Dan Klein (2007). "Improved inference for unlexicalized parsing". In: *Proceedings of NAACL 2007, Rochester, NY*, pp. 404–411.

Petrov, Slav et al. (2006). "Learning accurate, compact, and interpretable tree annotation". In: *Proceedings of COLING/ACL 2006, Sydney, Australia*, pp. 433–440.

Pollard, Carl and Ivan A. Sag (1994). *Head-driven phrase structure grammar*. University of Chicago Press.

Poon, Hoifung (2013). "Grounded unsupervised semantic parsing". In: *Proceedings of ACL 2013, Sofia, Bulgaria*, pp. 933–943.

Poon, Hoifung and Pedro Domingos (2009). "Unsupervised semantic parsing". In: *Proceedings of ACL 2009, Singapore*, pp. 1–10.

Pradet, Quentin, Laurence Danlos, and Gaël De Chalendar (2014). "Adapting VerbNet to French using existing resources". In: *Proceedings of LREC 2014, Reykjavik, Iceland*.

Prasad, Rashmi et al. (2005). "The Penn Discourse TreeBank as a resource for natural language generation". In: *Proceedings of the Corpus Linguistics Workshop on Using Corpora for NLG*.

Prasad, Rashmi et al. (2008). "The Penn Discourse TreeBank 2.0." In: *Proceedings of LREC 2008, Marrakech, Morocco*.

Qi, Peng et al. (2019). "Universal dependency parsing from scratch". arXiv:1901.10457.

Qi, Peng et al. (2020). "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages". arXiv:2003.07082.

Quan, Huangmao, Jie Wu, and J. Shi (2011). "Online social networks & social network services: A technical survey". In: *Pervasive Communication Handbook*. Ed. by Syed Ijlal Ali Shah, Mohammad Ilyas, and Hussein T. Mouftah. CRC Press. DOI: 10.1201/b11271-22.

Ramshaw, Lance A. and Mitchell P. Marcus (1999). "Text chunking using transformation-based learning". In: *Natural language processing using very large corpora*. Springer, pp. 157–176.

Rana, Kulwant Singh, Anil Kumar, et al. (2016). "Social media marketing: Opportunities and challenges". In: *Journal of Commerce and Trade* 11.1, pp. 45–49.

Ratnaparkhi, Adwait (1996). "A maximum entropy model for part-of-speech tagging". In: *Proceedings of EMNLP 1996, Philadelphia, PA*, pp. 133–142.

Raymond, Ruifang Ge and J. Mooney (2006). "Discriminative reranking for semantic parsing". In: *Proceedings of COLING/ACL 2006, Sydney, Australia*, pp. 263–270.

Reddy, Siva et al. (2016). "Transforming dependency structures to logical forms for semantic parsing". In: *Transactions of the Association for Computational Linguistics* 4, pp. 127–140.

Robert, Gilbert (1998). *TATOO: ISSO TAgger TOOl*. https://www.issco.unige.ch/en/staff/robert/tatoo/tatoo.html. [Online; accessed 1-January-2020].

Roberts, Sam GB and Robin IM Dunbar (2011). "Communication in social networks: Effects of kinship, network size, and emotional closeness". In: *Personal Relationships* 18.3, pp. 439–452.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Sagot, Benoît (2016). "External Lexical Information for Multilingual Part-of-Speech Tagging". arXiv:1606.03676.

Sagot, Benoît et al. (2006). "The Lefff 2 syntactic lexicon for French: architecture, acquisition, use". In: *Proceedings of LREC 2006, Genoa, Italy*, pp. 1–4.

Sang, Erik F. and Sabine Buchholz (2000). "Introduction to the CoNLL-2000 shared task: Chunking". arXiv:cs/0009008.

Schluter, Natalie and Josef Van Genabith (2008). "Treebank-based acquisition of LFG parsing resources for French". In: pp. 2909–2916.

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

— (1999). "Improvements in part-of-speech tagging with an application to German". In: *Natural language processing using very large corpora*. Springer, pp. 13–25.

Schönfinkel, Moses (1924). "Über die Bausteine der mathematischen Logik". In: *Mathematische annalen* 92.3-4, pp. 305–316.

Schuler, Karin Kipper (2005). "VerbNet: A broad-coverage, comprehensive verb lexicon". PhD thesis. University of Pennsylvania.

Schuster, Mike and Kuldip K. Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

Seddah, Djamé et al. (2010). "Lemmatization and lexicalized statistical parsing of morphologically rich languages: the case of French". In: *Proceedings of NAACL 2010, Minneapolis, MN, First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pp. 85–93.

Seldin, Jonathan P. (2006). "The logic of Curry and Church". In: *Handbook of the History of Logic* 5, pp. 819–873.

Sgall, Petr et al. (1986). *The meaning of the sentence in its semantic and pragmatic aspects*. Springer Science.

Sha, Fei and Fernando Pereira (2003). "Shallow parsing with conditional random fields". In: *Proceedings of NAACL 2003, Edmonton, Canada*, pp. 134–141.

Singh-Miller, Natasha and Michael Collins (2007). "Trigger-based language modeling using a loss-sensitive perceptron algorithm". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing, Honolulu, HI*. Vol. 4. IEEE, pp. IV–25.

Singla, Ankush, Elisa Bertino, and Dinesh Verma (2019). "Overcoming the Lack of Labeled Data: Training Intrusion Detection Models Using Transfer Learning". In: *2019 IEEE International Conference on Smart Computing, Bologna, Italy*. IEEE, pp. 69–74.

Sleator, Daniel DK and Davy Temperley (1995). "Parsing English with a link grammar". arXiv:cmp-lg/9508004.

Smullyan, Raymond (1985). *To Mock a Mockingbird, and Other Logic Puzzles including an Amazing Adventure in Combinatory Logic*. Alfred A. Knopf.

Socher, Richard et al. (2011). "Parsing natural scenes and natural language with recursive neural networks". In: *Proceedings of the 28th international conference on machine learning, Bellevue, WA*, pp. 129–136.

Steedman, Mark (1996). *Surface structure and interpretation*. MIT Press.

— (1999). "Quantifier scope alternation in CCG". In: *Proceedings of ACL 1999, College Park, MD*, pp. 301–308.

Steedman, Mark (2000a). “Information structure and the syntax-phonology interface”. In: *Linguistic inquiry* 31.4, pp. 649–689.

— (2000b). *The syntactic process*. Vol. 24. MIT press Cambridge, MA.

Steedman, Mark and Jason Baldridge (2011). “Combinatory categorial grammar”. In: *Non-Transformational Syntax: Formal and explicit models of grammar*, pp. 181–224.

Sun, Guang-Lu et al. (2005). “A maximum entropy markov model for chunking”. In: *2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China*. Vol. 6. IEEE, pp. 3761–3765.

Šuster, Simon and Walter Daelemans (2018). “Clicr: A dataset of clinical case reports for machine reading comprehension”. arXiv:1803.09720.

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning (2015). “Improved semantic representations from tree-structured long short-term memory networks”. arXiv:1503.00075.

Tallerman, Maggie (2013). *Understanding syntax*. Routledge.

Taylor, Ann, Mitchell Marcus, and Beatrice Santorini (2003). “The Penn treebank: an overview”. In: *Treebanks*. Springer, pp. 5–22.

Tellier, Isabelle et al. (2012). “Apprentissage automatique d’un chunker pour le français”. In: *Actes de JEP-TALN-RECITAL 2012, Grenoble, France*, pp. 431–438.

Templeton, Marjorie and John Burger (1983). “Problems in natural-language interface to DBMS with examples from EUFID”. In: *Proceedings of the first conference on Applied natural language processing, Santa Monica, CA*, pp. 3–16.

Tesnière, Lucien (1934). “Comment construire une syntaxe”. In: *Bulletin de la Faculté des Lettres de Strasbourg* 7, pp. 219–229.

— (1959). *Eléments de syntaxe structurale*. Klincksieck.

Thompson, Frederick B. et al. (1969). “REL: A rapidly extensible language system”. In: *Proceedings of COLING 1969, Stockholm, Sweden*. ACM, pp. 399–417.

Tieleman, Tijmen and Geoffrey Hinton (2012). “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2, pp. 26–31.

Tong, Amo, Ding-Zhu Du, and Weili Wu (2018). “On misinformation containment in online social networks”. In: *Advances in Neural Information Processing Systems*, pp. 341–351.

Torrey, Lisa and Jude Shavlik (2010). “Transfer learning”. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, pp. 242–264.

Toutanova, Kristina and Christopher D. Manning (2000). “Enriching the knowledge sources used in a maximum entropy part-of-speech tagger”. In: *Proceedings of EMNLP 2000, Hong Kong*, pp. 63–70.

Toutanova, Kristina et al. (2003). “Feature-rich part-of-speech tagging with a cyclic dependency network”. In: *Proceedings of NAACL 2003, Edmonton, Canada*, pp. 173–180.

Tse, Daniel and James R. Curran (2010). “Chinese CCGbank: extracting CCG derivations from the Penn Chinese treebank”. In: *Proceedings of COLING 2010, Beijing, China*, pp. 1083–1091.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). “Word representations: a simple and general method for semi-supervised learning”. In: *Proceedings of ACL 2010, Uppsala, Sweden*, pp. 384–394.

Urieli, Assaf and Ludovic Tanguy (2013). “L’apport du faisceau dans l’analyse syntaxique en dépendances par transitions: études de cas avec l’analyseur Talismane”. In: *Actes de TALN 2013, Les Sables d’Olonne, France*, publication–en.

Van Eijck, Jan and Hans Kamp (1997). “Representing discourse in context”. In: *Handbook of logic and language*. Elsevier, pp. 179–237.

Varges, Sebastian and Chris Mellish (2001). “Instance-based natural language generation”. In: *Proceedings of NAACL 2001, Pittsburgh, PA*, pp. 1–8.

Vaswani, Ashish et al. (2016). "Supertagging with LSTMs". In: *Proceedings of NAACL 2016, San Diego, California*, pp. 232–237.

Vrandečić, Denny and Markus Krötzsch (2014). "Wikidata: a free collaborative knowledge base". In:

Waltz, David (1975). *Understanding line drawings of scenes with shadows*. McGraw-Hill.

Wang, Fang and Milena Head (2007). "How can the web help build customer relationships?: an empirical study on e-tailing". In: *Information & Management* 44.2, pp. 115–129.

Wang, Yushi, Jonathan Berant, and Percy Liang (2015). "Building a semantic parser overnight". In: *Proceedings of ACL/IJCNLP 2015, Beijing, China*, pp. 1332–1342.

Wattenhofer, Mirjam, Roger Wattenhofer, and Zack Zhu (2012). "The YouTube social network". In: *Sixth International AAAI Conference on Weblogs and Social Media, Dublin, Ireland*.

Weiss, David et al. (2015). "Structured training for neural network transition-based parsing". arXiv:1506.06158.

White, Aaron Steven et al. (2016). "Universal decompositional semantics on universal dependencies". In: *Proceedings of EMNLP 2016, Austin, TX*, pp. 1713–1723.

White, Aaron Steven et al. (2019). "The Universal Decompositional Semantics Dataset and Decomp Toolkit". arXiv:1909.13851.

White, Mary (2016). "What types of social networks exist". In: *Love to know*. URL: https://socialnetworking.lovetoknow.com/What_Types_of_Social_Networks_Exist.

White, Michael and Jason Baldridge (2003). "Adapting Chart Realization to CCG". In: *Proceedings of EACL 2003, Stroudsburg, PA*.

Wink, Diane M. (2010). "Social networking sites". In: *Nurse educator* 35.2, pp. 49–51.

Wong, Yuk Wah and Raymond J. Mooney (2006). "Learning for semantic parsing with statistical machine translation". In: *Proceedings of NAACL 2006, New York, NY*, pp. 439–446.

Woods, William A. (1973). "Progress in natural language understanding: an application to lunar geology". In: *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition, New York, NY*. ACM, pp. 441–450.

Wu, Huijia, Jiajun Zhang, and Chengqing Zong (2017). "A Dynamic Window Neural Network for CCG Supertagging". In: *Proceedings of AAAI-17, San Francisco, CA*, pp. 3337–3343.

Xiao, Chunyang, Marc Dymetman, and Claire Gardent (2016). "Sequence-based structured prediction for semantic parsing". In: *Proceedings of the ACL 2016, Berlin, Germany*. Vol. 1, pp. 1341–1350.

Xu, Wenduan (2016). "LSTM shift-reduce CCG parsing". In: *Proceedings of EMNLP 2016*, pp. 1754–1764.

Xu, Wenduan, Michael Auli, and Stephen Clark (2015). "CCG supertagging with a recurrent neural network". In: *Proceedings of ACL/IJCNLP 2015, Beijing, China*. Vol. 2, pp. 250–255.

Xu, Wenduan, Stephen Clark, and Yue Zhang (2014). "Shift-reduce CCG parsing with a dependency model". In: *Proceedings of ACL 2014, Baltimore, MA*, pp. 218–227.

Xue, Naiwen et al. (2005). "The Penn Chinese TreeBank: Phrase structure annotation of a large corpus". In: *Natural language engineering* 11.2, pp. 207–238.

Yamada, Hiroyasu and Yuji Matsumoto (2003). "Statistical dependency analysis with support vector machines". In: *Proceedings of the Eighth International Conference on Parsing Technologies, Nancy, France*, pp. 195–206.

Yang, Zhilin et al. (2019). "XLNet: Generalized Autoregressive Pretraining for Language Understanding". arXiv:1906.08237.

Yasunaga, Michihiro, Jungo Kasai, and Dragomir Radev (2017). "Robust multilingual part-of-speech tagging via adversarial training". arXiv:1711.04903.

Ying, Rex et al. (2018). “Graph convolutional neural networks for web-scale recommender systems”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK*. ACM, pp. 974–983.

You, Yang et al. (2019). “Large batch optimization for deep learning: Training bert in 76 minutes”. arXiv:1904.00962.

Young, Steve et al. (2010). “The hidden information state model: A practical framework for POMDP-based spoken dialogue management”. In: *Computer Speech & Language* 24.2, pp. 150–174.

Young, Tom et al. (2018). “Recent trends in deep learning based natural language processing”. In: *ieee Computational intelligence magazine* 13.3, pp. 55–75.

Younger, Daniel H. (1967). “Recognition and parsing of context-free languages in time n3”. In: *Information and control* 10.2, pp. 189–208.

Yuret, Deniz and Ferhan Türe (2006). “Learning morphological disambiguation rules for Turkish”. In: *Proceedings of HLT/NAACL 2006, New York, NY*, pp. 328–334.

Zelle, John M. and Raymond J. Mooney (1996). “Learning to parse database queries using inductive logic programming”. In: *Proceedings of AAAI-96, Portland, Oregon*, pp. 1050–1055.

Zettlemoyer, Luke S. and Michael Collins (2012). “Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars”. arXiv:1207.1420.

Zhai, Feifei et al. (2017). “Neural models for sequence chunking”. In: *Proceedings of AAAI-17, San Francisco, CA*.

Zhai, Michael, Johnny Tan, and Jinho D. Choi (2016). “Intrinsic and extrinsic evaluations of word embeddings”. In: *Proceedings of AAAI-16, Phoenix, AZ*.

Zhang, Xun et al. (2016). “Transition-based parsing for deep dependency structures”. In: *Computational Linguistics* 42.3, pp. 353–389.

Zhang, Yi, Valia Kordoni, and Erin Fitzgerald (2007). “Partial parse selection for robust deep processing”. In: *ACL 2007 Workshop on Deep Linguistic Processing*, pp. 128–135.

Zhang, Yue and Stephen Clark (2009). “Transition-based parsing of the Chinese treebank using a global discriminative model”. In: *Proceedings of the 11th International Conference on Parsing Technologies*, pp. 162–171.

— (2011). “Shift-reduce CCG parsing”. In: *Proceedings of ACL/HLT 2011, Portland, OR*, pp. 683–692.

Zhang, Yue and Joakim Nivre (2011). “Transition-based dependency parsing with rich non-local features”. In: *Proceedings of ACL/HLT 2011, Portland, OR*, pp. 188–193.

— (2012). “Analyzing the effect of global learning and beam-search on transition-based dependency parsing”. In: *Proceedings of COLING 2012, Mumbai, India*, pp. 1391–1400.

Zhou, Junru and Hai Zhao (2019). “Head-driven phrase structure grammar parsing on Penn Treebank”. arXiv:1907.02684.

Zhu, Muhua et al. (2013). “Fast and accurate shift-reduce constituent parsing”. In: *Proceedings of ACL 2013, Sofia, Bulgaria*, pp. 434–443.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Analyse de la structure de représentation du discours pour le français

Mot clés : Grammaire Catégorielle Combinatoire, Structure de Représentation du Discours, Structure de Dépendance, Contenu Généré par l'Utilisateur

Resumé : Nous nous intéressons aux contenus textuels tels que des discours, énoncés, ou conversations issus de la communication interactive sur les plateformes de réseaux sociaux en nous basant sur le formalisme des grammaires catégorielles combinatoires (GCC) et sur la théorie de représentation du discours. Nous proposons une méthode pour l'extraction d'un arbre de GCC à partir de l'arbre de dépendances de la phrase, ainsi qu'une architecture générale pour construire une interface entre la syntaxe et la sémantique de phrases françaises.

Title : French Language DRS Parsing

Keywords : Combinatory Categorial Grammar, Discourse Representation Structure, Dependency Structure, User-Generated Content

Abstract : In the rise of the internet, user-generated content from social networking services is becoming a giant source of information that can be useful to businesses on the aspect where users are viewed as customers or potential customers for companies. We are interested in textual content such as speeches, statements, or conversations resulting from interactive communication on social network platforms based on the formalism of Combinatory Categorical Grammars (GCC) and on the theory of representation of discourse. We propose a method for extracting a GCC tree from the sentence dependency tree, as well as a general architecture to build an interface between the syntax and the semantics of French sentences.