SUPERTAGGING WITH COMBINATORY CATEGORIAL GRAMMAR FOR DEPENDENCY PARSING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURAK KERİM AKKUŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

SUPERTAGGING WITH COMBINATORY CATEGORIAL GRAMMAR FOR DEPENDENCY PARSING

submitted by **BURAK KERİM AKKUŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Dr. Ruket Çakıcı
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Cem Bozşahin
Cognitive Science Department, METU

Dr. Ruket Çakıcı
Computer Engineering Department, METU

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Department, METU

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU

Assoc. Prof. Dr. Tolga Can
Computer Engineering Department, METU

Date:

Name, Last Name: BURAK KERİM AKKUŞ

Signature :

ABSTRACT

SUPERTAGGING WITH COMBINATORY CATEGORIAL GRAMMAR FOR DEPENDENCY PARSING

AKKUŞ, BURAK KERİM
M.S., Department of Computer Engineering
Supervisor : Dr. Ruket Çakıcı

September 2014, 55 pages

Combinatory Categorial Grammar (CCG) categories contain syntactic and semantic information. CCG derivation trees can be used in extracting partial dependency structures by providing the missing information in order to build complete dependency structures. Therefore, CCG categories are sometimes referred to as supertags. The amount of information encoded in supertags makes it possible to create very accurate and fast parsers as supertagging is considered "almost parsing". In this thesis, a maximum entropy based part of speech tagger is presented to improve the performance of CCG supertagging and another maximum entropy classifier is implemented with additional features for supertagging. Morphological features of words of an agglutinative language such as Turkish are used in order to improve the accuracy of POS tagging and supertagging processes. This indicates direct relationships between morphemes and lexical categories. The effects of using the improved supertagger are tested on dependency parsers by means of using supertags as rich parts of speech tags. Additionally, using POS taggers that assign multiple part of speech tags to the ambiguous words is suggested as another potential improvement for supertaggers.

Keywords: Combinatory Categorial Grammar, Dependency Parsing, Part of Speech Tagging, Supertagging, Multitagging, Maximum Entropy Classification

ÖZ

BAĞLILIK AYRIŞTIRMASI İÇİN BİRLEŞENLİ ULAMSAL GRAMER İLE SÜPER ETİKETLEME

AKKUŞ, BURAK KERİM
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Dr. Ruket Çakıcı

Eylül 2014 , 55 sayfa

Birleşenli Ulamsal Gramer'in sınıfları söz dizimi ve anlamıyla ilgili bilgiler içerir. CCG türeme ağaçları eksiksiz bağlılık yapıları oluşturmak için gerekli bilgileri, kısmi bağlılık yapılarınu kullanarak oluşturabilirler. Bu nedenle CCG kategorileri süper etiketler olarak da adlandırılırlar. Süper etiketlerin içerdiği bilgiler oldukça başarılı ve hızlı ayrıştırıcılar oluşturmayı olası kılar. Bu başarı süper etiketlemenin aynı zamanda "neredeyse ayrıştırma" olarak da adlandırılmasına neden olmuştur. Bu tezde, maksimum entropi tabanlı bir sözcük türü etiketleyicisi süper etiketleme başarısı arttırmak için sunulmakta ve yine bir maksimum entropi modeli, süper etiketleme için yeni özelliklerle uygulanmaktadır. Türkçe gibi eklemeli bir dile ait sözcüklerin biçim bilgileriyle ilgili özellikleri sözcük türü ve CCG kategori etiketleme işlemlerinin başarılarını arttırmak için kullanılmıştır. Bu sözcük kategorileriyle biçimsel kategorilerin ilgisini gösterir. Geliştirilen süper etiketleyiciyi kullanmanın etkileri süper etiketleri zengin sözcük türü bilgileri olarak kullanarak test edilmektedir. Ek olarak, sözcüklere birden fazla etiket atayan sözcük türü etiketleyicileri süper etiketleyicileri geliştirmek için olası bir yöntem olarak önerilmektedir.

Anahtar Kelimeler: Birleşenli Ulamsal Gramer, Bağlılık Ayrıştırması, Sözcük Türü Etiketleme, Super Etiketleme, Çoklu Etiketleme, Maksimum Entropi Sınıflandırması

to the stars
and
the winds of Mediterranean

ACKNOWLEDGMENTS

I would like to thank my family for their unconditional love and support: My brother Murat Akkuş, for being there, always and Ender Bulut, for the last year we've been through for our thesis studies.

I would like to express my deepest gratitude to my supervisor Dr. Ruken Çakıcı for her guidance and support though my studies. I am very grateful for the assistance given by İlkcan Keleş in using HPC systems in our department. I would like to offer my special thanks to Assist. Prof. Aykut Erdem, Assist. Prof. Erkut Erdem and Çağdaş Baş for their support and assistance in using computational resources of Hacettepe University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AB	Ajdukiewicz and Bar-Hillel
CG	Categorial Grammar
CCG	Combinatory Categorial Grammar
CFG	Context Free Grammar
CoNLL	Conference on Computational Natural Language Learning
CYK	Cocke–Younger–Kasami
C&C	Clark and Curran
HPSG	Head-driven Phrase Structure Grammar
LFG	Lexical Functional Grammar
LTAG	Lexicalized Tree Adjoining Grammar
MegaM	Maximum Entropy (GA) Model Optimization Package
MSTParser	Maximum Spanning Tree Parser
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PCFG	Probabilistic Context Free Grammar
POS	Part of Speech
PST	Phrase Structure Tree
RASP	Robust Accurate Statistical Parsing
TAG	Tree Adjoining Grammar
WSJ	Wall Street Journal

CHAPTER 1

INTRODUCTION

Dependency graphs are more intuitive in terms of representation for syntactic and semantic relations in a language. Dependency grammars are used popularly for languages with relatively free word order such as Turkish. Combinatory Categorial Grammar [42] is a lexicalized grammar formalism, which handles semantic and syntax transparently. CCG categories -which are referred to as *supertags* throughout this thesis- are complex structures that contain various information such as subcategorization, predicate argument information and semantic interpretation. Using CCG categories as features for dependency parsers has been shown effective in improving coverage and accuracy of the parsers that use this information only superficially [11, 3].

In this thesis, we focus on improving the dependency recovery by improving two pre-parsing steps. These steps are POS tagging and supertagging. Çakıcı [14] uses gold standard tags and shows the potential of using CCG categories as features in the dependency parsing. A real life application begins with raw data and uses taggers to label the data. The errors in each step propagates up to the parsing level. In this thesis, we try to improve the parser performance by reducing these errors.

Initially, we show the importance of POS information for supertagging. We build simple supertaggers that only uses POS tag sequences and show their performance. We compare the results of these taggers with more complex statistical supertaggers that use more informative features.

We show the results of a maximum entopy POS tagger similar to Ratnaparkhi [38] and improve its accuracy by introducing new features to the model. We use Clark and Curran's [18] supertagger and train the supertagger on Turkish morphemic lexicon. We also implement a supertagger that includes different features specific to Turkish language in order to improve the accuracy on Turkish supertagging. We explore the effects of morphologically more informative features on both POS tagging and supertagging performances. We include labels, stems and suffixes extracted with morphological analyses as well as their approximations with prefixes and suffixes of varying lengths.

The proposed taggers with additional features are used to tag data with firstly POS tags then supertags. Various combinations of taggers are used with MSTParser [33] for dependency parsing and the results are compared with the ones achieved with gold standard tags.

We also explore the concept of multitagging for POS tags and supertags. We show that assigning a list of tags can be used to improve the performances of both taggers with a slight increase in complexity. However, the integration of multitagging into dependency parsing is left as a future research problem.

The rest of the thesis is organized as follows. Chapter 2 provides a brief introduction to Combinatory Categorial Grammar (CCG). Only the most basic characteristics of the CCG categories will be given here in order to make the reader familiar with the categories we will be referring to as *supertags* or *CCG categories* throughout the thesis. This chapter also includes a few examples of CCG derivations, which can be mapped to parse trees. Chapter 3 presents dependency parsing and applications of dependency parsing with Combinatory Categorial Grammar in relation to this thesis. Chapter 4 provides detailed information about datasets used in this thesis. We conducted experiments on both English and Turkish and this chapter gives the highlights of and the statistics on these two datasets. Chapter 5 contains supertagging and parsing experiments and their detailed analyses. And finally Chapter 6 concludes the work presented in this thesis and discusses further research possibilities on this topic.

CHAPTER 2

COMBINATORY CATEGORIAL GRAMMAR

CCG [42] is mildly context sensitive and it is an extension of Categorial Grammar of Ajdukiewicz [1] and Bar-Hillel [7].CG is equivalent to Context Free Grammar. CCG provides a transparent interface between surface syntax and underlying semantics, where each syntactic derivation corresponds directly to an interpretable structure. CCG's notion of interpretation is represented in predicate argument structures, bracketings in morphology and syntax is achieved via proper lexical type assignments.

Figure 2.1 shows the category of a simple verb *likes* in CCG. (2.1) presents the complete syntactic type and its semantic interpretation. The details are presented in (2.2). (2.3) shows the derivation of a simple sentence with its semantic interpretation with three words.

(2.1) $$\textit{likes} := (S\backslash NP_{3s})/NP : \lambda x \lambda y.like'\, xy$$

(2.2) $$\overbrace{\textit{likes}}^{string}\ \underbrace{:=}_{\substack{string\\ type\\ descriptor}}\ \overbrace{\overbrace{(S\backslash NP_{3s})/NP}^{syn.\,type} : \overbrace{\underbrace{\lambda x \lambda y.}_{correspondence}\ \underbrace{like'_{\underbrace{(e,(e,t))}_{sem.\ type}}\, xy}_{logical form}}^{interpretation}}^{category}$$

(2.3) $$\begin{array}{ccc}
\text{John} & \text{likes} & \text{Mary} \\
\hline NP : john' & \hline (S\backslash NP)/NP : \lambda x.\lambda y.likes'xy & \hline NP : mary' \\
 & \multicolumn{2}{c}{\underline{\hphantom{(S\backslash NP)/NP : \lambda x.\lambda y.likes'xy\ NP : mary'}}\,>} \\
 & \multicolumn{2}{c}{S\backslash NP : \lambda y.likes'marry'y} \\
\multicolumn{3}{c}{\underline{\hphantom{NP : john'\ (S\backslash NP)/NP : \lambda x.\lambda y.likes'xy\ NP : mary'}}\,<} \\
\multicolumn{3}{c}{S : likes'marry'john'}
\end{array}$$

Figure 2.1: An example CCG category and derivation of a sentence

CCG is used in many different NLP applications including parsing (Hockenmaier and Steedman [27], Clark and Curran [18]), natural language generation (White and Baldridge [46], White et al.[47]), machine translation (Birch et al.[8]), psycholinguistics (Reitter et al.[39]) and so on. We will give a brief introduction to the rules and categories in CCG which is essential for the rest of the thesis.

2.1 Categories in Categorial Grammar

CCG categories consist of single types or functors and arguments. We use the result first notation of Steedman [41]. The result type is shown as the first item in a category and its arguments are presented afterwards with slash type denoting their relative positions according to the functor word.

2.1.1 Atomic category

Atomic categories are a small set of basic categories which includes nouns, prepositions or sentences. There is no limit on the number of items in this set but most general examples are as follows: S, N, NP, PP [41]. This list might change with the requirements of the language.

2.1.2 Complex category

Complex categories are formed by the combination of atomic and other complex categories with the help of two slash operators: forward slash $(/)$ and backward slash $(\backslash)$. Operators are left associative. In certain extensions of Categorial grammar, slashes may have modalities [42].

X/Y and $X\backslash Y$ are categories if X and Y are categories. The number of potential CCG categories is infinite, however they are limited in the context of grammars of natural languages. In our data sets, we only have categories that match observed words in sentences of natural languages.

2.2 Rules

CCG's difference from AB categorial grammar lies in the combinators which provides additional powers other than simple function application. In this section, we present function application and two most important rules of CCG that makes it possible to handle most linguistic phenomena that CFGs cannot resolve.

2.2.1 Function Application

Function applicaiton is the most basic rule, it is the only rule in AB Catgorial Grammar and makes it equivalent to Context Free Grammars. Function application rules are presented in 2.4 and 2.5. Depending on the slash type, the word with category Y is either to the left or to the right of the word with category X/Y or $X\backslash Y$ and the function application produces category X for the phrase derived of those two words. Examples of function application are given in Section 2.3.

(2.4) Forward Application:

$$X/Y{:}\ f \quad Y{:}\ a \quad \rightarrow \quad X{:}\ fa \qquad (>)$$

(2.5) Backward Application:

$$Y{:}\ a \quad X\backslash Y{:}\ f \quad \rightarrow \quad X{:}\ fa \qquad (<)$$

2.2.2 Composition

Composition allows consecutive strings that are not traditionally accepted constituents to form phrases that can take part in different combinations that are not possible for their initial categories or any other category that can be derived with simple function application. Functions may compose into other functions with any of the rules 2.6, 2.7, 2.8 or 2.9 in the appropriate context. Crossing composition rules allow CCG to handle crossing dependencies in languages such as Turkish. Figure 2.2 gives an example for the use of composition. Composition is used to produce the composite verb *might prove* before the verb takes its object.

(2.6) Forward Composition:

$$X/Y : f \quad Y/Z : g \quad \rightarrow \quad X/Z : \lambda x.f(gx) \qquad (>\mathbf{B})$$

(2.7) Backward Composition:

$$Y\backslash Z : g \quad X\backslash Y : f \quad \rightarrow \quad X\backslash Z : \lambda x.f(gx) \qquad (<\mathbf{B})$$

(2.8) Forward Crossing Composition:

$$X/Y : f \quad Y\backslash Z : g \quad \rightarrow \quad X\backslash Z : \lambda x.f(gx) \qquad (>\mathbf{B}_\times)$$

(2.9) Backward Crossing Composition:

$$Y/Z : g \quad X\backslash Y : f \quad \rightarrow \quad X/Z : \lambda x.f(gx) \qquad (<\mathbf{B}_\times)$$

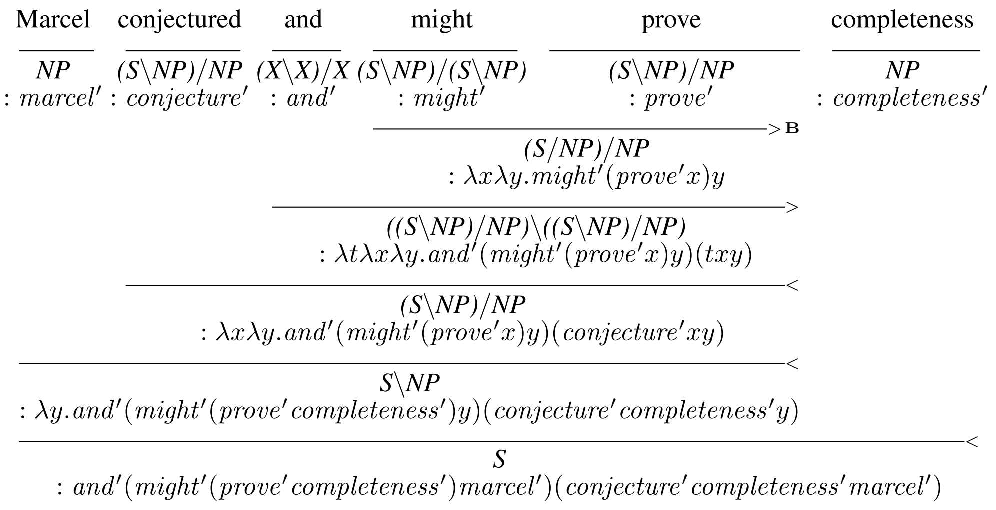

Marcel
conjectured
and
might
prove
completeness
NP : marcel′
(S\NP)/NP : conjecture′
(X\X)/X : and′
(S\NP)/(S\NP) : might′
(S\NP)/NP : prove′
NP : completeness′
>B
(S/NP)/NP : λxλy.might′(prove′x)y
>
((S\NP)/NP)\((S\NP)/NP) : λtλxλy.and′(might′(prove′x)y)(txy)
<
(S\NP)/NP : λxλy.and′(might′(prove′x)y)(conjecture′xy)
<
S\NP : λy.and′(might′(prove′ completeness′)y)(conjecture′ completeness′y)
<
S : and′(might′(prove′ completeness′)marcel′)(conjecture′ completeness′ marcel′)

Figure 2.2: Composite verb example : *might prove*

2.2.3 Type-Raising

Type-raising turns arguments into functions over functions over those arguments. Type-raising rules allow composition for these arguments and lets them take part in coordinations. Type-raising rules are presented in (2.10) and (2.11). In Figure 2.3, two nouns *Marcel* and *I* are type-raised into sentence-making words that take sentence-making words to their right which take a noun phrase as an argument to their left. These actions allow these noun phrases to combine with the transitive verbs following them before the verbs combine with their objects. After this step, resulting categories can combine with a coordinator and share the same object. Without type raising, these kind of derivations will not be possible.

(2.10) Forward Type Raising:

$$A : a \quad \rightarrow \quad T/(T\backslash A) : \lambda f.fa \qquad (>\mathbf{T})$$

(2.11) Backward Type Raising:

$$A : a \quad \rightarrow \quad T\backslash(T/A) : \lambda f.fa \qquad (<\mathbf{T})$$

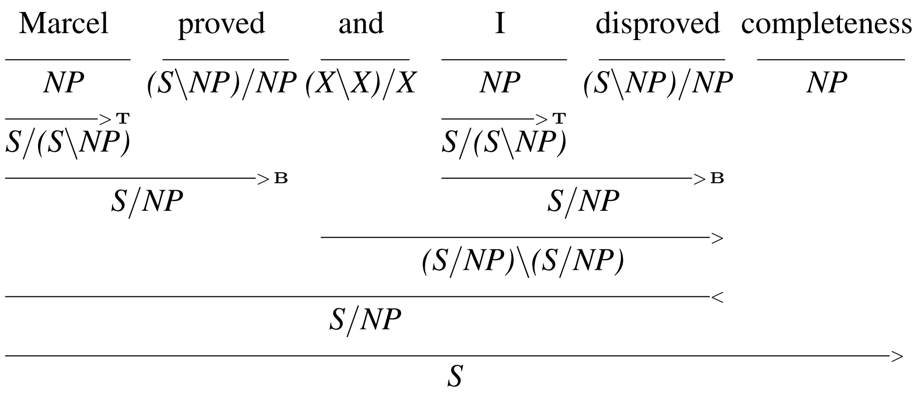


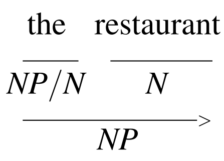
Figure 2.3: Type raising of two noun phrases, their composition with verbs and coordination.

There are other combinators such as substitution which is based on the combinator S [42, 44, 43] but these are not in the scope of this thesis and are omitted in this short introduction to CCG.

2.3 Category Examples

This section provides examples for basic word categories and their combinations with other categories in small phrases or sentences. Only syntactic types are shown and logical interpretations are omitted in order to focus on syntactic derivations. [1]

(2.12) **Determiner**



A determiner looks for a noun, to produce a simple noun phrase.

[1] Examples 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.19 and 2.20 are taken from Johanna Moore's Natural Language Generation course at University of Edinburgh. 2.18 and 2.21 are modified versions of those examples. `http://www.inf.ed.ac.uk/teaching/courses/nlg/`

(2.13) **Adjective**

$$\begin{array}{ccl}
\text{great} & \text{food} & \\
\overline{N/N} & \overline{N} & \\
\cline{1-2}
\multicolumn{2}{c}{N} & >
\end{array}$$

An adjective looks for a noun, to produce a (modified) noun.

(2.14) **Preposition**

$$\begin{array}{ccl}
\text{in} & \text{the restaurant} & \\
\overline{PP/NP} & \overline{NP} & \\
\cline{1-2}
\multicolumn{2}{c}{PP} & >
\end{array}$$

An preposition looks for a noun phrase on its right, to produce a prepositional phrase.

(2.15) **Postposition**

$$\begin{array}{ccl}
\text{one floor} & \text{above} & \\
\overline{NP} & \overline{PP\backslash NP} & \\
\cline{1-2}
\multicolumn{2}{c}{PP} & <
\end{array}$$

An postposition looks for a noun phrase on its left, to produce a postpositional phrase.

(2.16) **Intransitive Verb**

$$\begin{array}{ccl}
\text{Giovanni's} & \text{rocks} & \\
\overline{NP} & \overline{S\backslash NP} & \\
\cline{1-2}
\multicolumn{2}{c}{S} & <
\end{array}$$

An intransitive verb looks to its left for a subject, to produce a sentence.

(2.17) **Transitive Verb**

$$\begin{array}{cccl}
\text{Giovanni's} & \text{serves} & \text{pasta} & \\
\overline{NP} & \overline{(S\backslash NP)/NP} & \overline{NP} & \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} & > \\
\cline{1-3}
\multicolumn{3}{c}{S} & <
\end{array}$$

A transitive verb looks first to its right for an object, then to its left for a subject, to produce a sentence.

(2.18) **Ditransitive Verb**

$$\begin{array}{cccc}
\text{The restaurant} & \text{serves} & \text{Giovanni} & \text{pasta}\\
\overline{NP} & \overline{((S\backslash NP)/NP)/NP} & \overline{NP} & \overline{NP}\\
 & \multicolumn{2}{c}{\overline{\qquad (S\backslash NP)/NP \qquad}{>}} & \\
 & \multicolumn{3}{c}{\overline{\qquad\qquad S\backslash NP \qquad\qquad}{>}}\\
\multicolumn{4}{c}{\overline{\qquad\qquad\qquad\qquad S \qquad\qquad\qquad\qquad}{<}}
\end{array}$$

A ditransitive verb looks first to its right for objects, then to its left for a subject, to produce a sentence.

(2.19) **Adverb**

$$\begin{array}{ccc}
\text{Giovanni's} & \text{totally} & \text{rocks}\\
\overline{NP} & \overline{(S\backslash NP)/(S\backslash NP)} & \overline{S\backslash NP}\\
 & \multicolumn{2}{c}{\overline{\qquad S\backslash NP \qquad}{>}}\\
\multicolumn{3}{c}{\overline{\qquad\qquad\qquad S \qquad\qquad\qquad}{<}}
\end{array}$$

An adverb looks to its right for a verb phrase, to produce another verb phrase which will combine with a subject to produce a sentence. In this example, the adverb modifies an intransitive verb and the resulting phrase acts as an intransitive verb.

(2.20) **Relative Clause**

$$\begin{array}{ccc}
\text{restaurant} & \text{that} & \text{rocks}\\
\overline{N} & \overline{N\backslash N/(S\backslash NP)} & \overline{S\backslash NP}\\
 & \multicolumn{2}{c}{\overline{\qquad N\backslash N \qquad}{>}}\\
\multicolumn{3}{c}{\overline{\qquad\qquad\qquad N \qquad\qquad\qquad}{<}}
\end{array}$$

A relative clause with “that” modifies a noun

(2.21) **Conjunction**

$$\begin{array}{ccc}
\text{cooks} & \text{and} & \text{serves}\\
\overline{S/NP} & \overline{(X\backslash X)/X} & \overline{S/NP}\\
 & \multicolumn{2}{c}{\overline{(S/NP)\backslash(S/NP)}{>}}\\
\multicolumn{3}{c}{\overline{\qquad\qquad S/NP \qquad\qquad}{<}}
\end{array}$$

Conjunction of two verbs with “and”
Here X’s represent S/NP and the actual conjunction category is $((S/NP)\backslash(S/NP))/(S/NP)$. $(X\backslash X)/X$ is a general form for these kind of conjunctions.

One can study the sentences in CCGBank for more detailed examples of CCG categories and derivations. CCGBank is a semi-automatically derived corpus from Penn Treebank and will be further discussed in Section 4 which explains data sets used in this thesis.

CHAPTER 3

PARSING

Parsing is an important step in making sense of sentences. It helps in naming and analysing the words in a sentence and the connections between them. In this section, we will briefly present dependency parsing and mention main approaches in that area. Then we will introduce parsing with combinatory categorial grammar with our main focus on supertagging. We will present the idea of supertagging and multitagging in the domain of CCG with main publications on these topics.

3.1 Dependency Parsing

Using dependencies to represent linguistic structure is much older than using phrase structures. It has roots in the works of ancient Greek, Indian and Arabian linguists [37]. However, modern dependency linguistic theories begin with the work of Tesnière [45].

Both phrase structure trees and dependency structures have their own respective advantages. Dependency grammars are especially well suited for languages with free word order such as Czech and Turkish [9, 36]. Dependency relations are more intuitive for understanding the structure. They are also somewhat closer to semantic relations in the way they link words to each other. This makes interpreting a sentence with dependency links easier. Phrase structure trees contain intermediate nodes, however dependency relations only exist between words.

Building a probabilistic model for dependencies is trivial. Calculating probabilities for dependencies is easier since it is possible to split the whole structure into head-dependent relationships which reduces sparsity of the data. However, dependency relations are not independent of each other and simple independence assumptions harms a probabilistic model based on dependency grammars.

Earlier dependency parsing studies use existing parsers trained on phrase structure trees. [21] translated dependency relations to PSTs and used the parsers build for Penn Treebank. Nivre [34] and Kudo and Matsumoto [29] built deterministic dependency parsers which are time efficient. CoNLL's shared tasks of 2006 and 2007 were focused on dependency parsing. MaltParser of Nivre [35] obtained the second best overall score in CoNLL 2006 and best in CoNLL 2007. McDonald et al. [33] presented a graph based dependency parser which is also used in this study.

3.2 Parsing with Combinatory Categorial Grammar

Parsing with CCG is similar to parsing with CFG. One technical difference is that instead of POS tags words are assigned CCG categories. However, CCG categories contain a lot more information than simple POS tags and internal nodes of CFG can simply be included in a category. Therefore, these categories are called supertags. First step in CCG parsing is assigning each word with a set of lexical categories [27, 26, 18]. This may be achieved by assigning all known categories as candidates or by selecting a set of possible categories with a statistical tagger. After assigning lexical categories, they are processed using CCG's combinatory rules to match the desired sentence. Since our main focus is on improving supertagger accuracy to perform dependency parsing on unseen data we will mainly present supertagging.

One can simply assign each word all the categories available for it in the lexicon without any additional process and move on to combining these categories for parsing as in Hockenmaier and Steedman [27] and Hockenmaier [26] and try to eliminate some of the subtrees on the parsing level with a procedure such as beam search. However, trying a lot of alternatives cost a lot of computational resources. Too many derivations, most of which are undesired are generated. Using a statistical tagger is an alternative to overcome this problem. As an alternative, we can try to find the actual, correct category of the word before passing to parsing stage. By selecting a relatively small number of candidates for each word, corresponding number of derivations can be reduced greatly. This process of statistically assigning lexical categories is called supertagging.

CCG categories contain significant amount of syntactic and semantic information. Supertagging is also called almost parsing because supertags can be considered as fragments of larger trees as is the case for LTAG, where the idea was first proposed [6]. The amount of information makes it possible to create fast parsers after a successful supertagging process.

Supertagging was applied to LTAG [6, 17, 16] before CCG. However, the performance was not satisfying enough to integrate it in a parser. Bangalore and Joshi [6] used a Markov Model tagger to assign LTAG trees to words. LTAG trees are very similar in nature to CCG categories regarding the information they contain.

Clark and Curran [18, 19] used maximum entropy (ME) to train a supertagger for CCG and showed its advantages for parsing in terms of both accuracy and speed. They also proposed a multitagging approach in which they assign more than one category to the ambiguous words. They defined ambiguity in terms of the ratio between the most probable supertag and the other candidates. ME model makes it easy to add new feautures and define a multi-tagger. Multi-tagger assigned a set of categories each with a probability above a given threshold rather than a single category. The reported that per word accuracy for a single category is not high enough to generate derivations that cover all the sentences. Even if a single word is assigned an erroneous category, we may not parse that sentence. When the average length of a sentence is taken into account, there is too much difference between 90% and 95%. Above that level, even one percent counts a lot and actually below 95% is practically useless. For example, C&C supertagger with 92.7% accuracy is only capable of correctly labelling 1 in 3 sentence. When the per-word accuracy increases to 97%, 64.9% of the sentences have all correct categories for their words an 98.7% accuracy on words reflects on the sentences as 81.3%.

Table 3.1 shows the features Clark and Curran [18] use for their supertagger. Here, *w* represents the word and *t* stands for the POS tag, *c* represents the supertag. The

Table 3.1: Features used in Maximum Entropy Model of Clark&Curran for supertagging

WORD	POS	SUPER
w_i	t_i	c_{i-1}
w_{i-1}	t_{i-1}	c_{i-2},c_{i-1}
w_{i-2}	t_{i-2}	
w_{i+1}	t_{i+1}	
w_{i+2}	t_{i+2}	
	t_{i-2},t_{i-1}	
	t_{i-1},t_i	
	t_{i-1},t_{i+1}	
	t_i,t_{i+1}	
	t_{i+1},t_{i+2}	

subscripts denote the place of the word/tag/supertag in the context window where w_i means current word and t_{i-1} means the POS tag of the previous word.

3.2.1 Multi-tagging

As mentioned in the previous sections, supertagging is quite important in parsing with CCG. If the tagger assigns wrong categories the parser cannot recognize the sentence since combinatory rules only combine suitable categories. We can trade a bit of speed for a better tagger if we give alternative categories to the parser when the probability of the category of a word is lower than required.

Clark and Curran [18, 19] implement the CCG supertagger as a Maximum Entropy Model and use it as a multi-tagger with various ambiguity levels. They assign the categories within a certain percentage of the probability of the most likely category as the set of supertags when a single tag cannot provide a parse for the sentence.

Multi-tagging provides a way to attach importance to speed or accuracy of the parser. As the set size increases the chance of finding the correct derivations also increase, however, with a more strict set of categories, parser works faster. Therefore, one can start with a small set of categories per word and enlarge the list of categories received from the supertagger if the parser cannot find a derivation. Alternatively, one can start with a larger set of categories per word and reduce the set size if the process takes more than a desired time.

Table 3.2: Multi-tagging results on WSJ00 with different β values for supertagger

EXPERIMENT	SUPER	S_SUPER	AVG_SUPER
$\beta_{MSUPER} = 1.00$	92.7	36.0	1.000
$\beta_{MSUPER} = 0.99$	92.7	36.3	1.001
$\beta_{MSUPER} = 0.75$	93.5	40.7	1.020
$\beta_{MSUPER} = 0.50$	94.5	45.6	1.052
$\beta_{MSUPER} = 0.10$	97.0	64.9	1.225
$\beta_{MSUPER} = 0.01$	98.4	77.7	1.716
$\beta_{MSUPER} = 0.001$	98.7	81.3	2.988

Table 3.2 shows the multi-tagging results on section 00 of CCBank (section 00 of Penn Treebank). C&C supertagger is trained on the sections 02-21. The table contains results for different β values. Here, column *SUPER* shows the accuracy of the supertagger on the word level, *S_SUPER* represents the accuracy for the whole sentence (i,e. all correctly tagged words). Multi-tagging accuracy is defined as correct if the gold standard tag is among the given multi-tag set. *AVG_SUPER* show the average number of supertags per word. The results show the effects of multi-tagging performance through β values. As β decreases, more supertags are assigned to the words and the accuracy of the supertagger for both words and whole sentences increase as expected.

In this thesis, we present multi-taggers for both part of speech tags and CCG categories. We introduce new features into taggers and use them on unseen data for dependency parsing.

CHAPTER 4

DATA

We use data from two languages for our experiments. For English, CCGBank, a CCG derivation tree corpus created from the Penn Treebank [28], and for Turkish, morphemic and lexemic CCG lexicons of Çakıcı **??** and Çakıcı & Steedman [11].

4.1 English

CCGBank is an annotated English corpus with CCG derivation trees and it is semi-automatically derived from Penn Treebank by Hockenmaier [25] and Hockenmaier and Steedman [28]. Penn Treebank is annotated with phrase structures and it is the largest manually annotated available data for English containing 1 million words [32]. It is the most popular data set for statistical parsing of English and it is the most commonly used corpus for parsing studies. Therefore, early dependency parsing studies translate dependency relations into phrase structures, parse them and then traslate them back [21, 15].

Hockenmaier [25] presents an algorithm for translating phrase structure trees of Penn Treebank into normal-form CCG derivations. Penn Treebank uses null elements to encode various linguistic phenomena such as extraction. This makes it possible to extract CCG derivations for long-distance dependencies through coordination and extraction which are crucial to semantic interpretation. Even though, various specific constructors are used while transforming the Treebank, the basic algorithm is as follows: After preprocessing, firstly determine constituent type, then apply binarization and finally assign categories.

Preprocessing step includes correcting simple annotation errors and tagging, bracketing for coordinate structures and analysis of noun phrases. Head finding rules of Magerman [31] and Collins [20] is applied for finding heads, complements and adjuncts. CCG derivations require two categories at a time. All the complements on the left are made right branching by inserting dummy non-terminals up until the head. After assigning categories final derivations contain CCG categories for words as leaves and subtree levels contain categories with arguments. For supertagging, this category information is used together with words.

4.2 Turkish

Highly agglutinative structure and relatively free word-order of Turkish are important factors in processing Turkish language. Turkish dependencies may be multi-headed and non-projective [14]. Morphological structure of words makes them interesting to process as different inflectional groups in a single word may have different dependency relations with many other words as heads or dependants. This requires considering smaller-than-word elements as the entities/nodes in dependency relations.

METU-Sabancı Turkish Treebank [40, 36, 4] is a dependency annotated treebank containing 5620 sentences taken from METU Turkish Corpus [40]. The corpus contains 2 million words taken from several different sources such as 87 journal issues and 201 books written after 1990. The Treebank is a subset of the corpus with the same distribution of sources. The variation of sources can be directly observed in average sentence lengths throughout the treebank and also the corpus. The average is 8 words per sentence but this value may rise to 53 for some scientific articles, whereas dialogues contain many one-word sentences [14]. Therefore, a fair distribution of the available data which is quite small compared to English for training and test purposes is important. We use the corrected version of the Turkish treebank as explained in [14].

4.2.1 Morphemic CCG Lexicon

Turkish handles case, tense, number, polarity, modality, relativization and voice through morphology with its agglutinative nature. There are 231K morphemes in Metu Sabancı Turkish Treebank for 54K tokens with an average of 4.31 morphemes per word. A Simple Turkish word may have a translation with several English words. Treating each morheme as a separate lexical entry helps overcoming bracketing problems occuring due to morphemes being bound to the word they are attached to [14, 10].

Çakıcı [13, 14] use METU Sabancı Turkish Treebank to extract a morphemic CCG lexicon for Turkish. Morphological structure of words are annotated with inflectional groups in the treebank. Words are separated into multiple lexical entities with these derivational boundaries based on morphemes. Average number of inflectional groups per word is 1.26. There are many studies that show using morphological features for Turkish improves performance of linguistic tools such as parsers [15, 24].

4.3 Experimental Setup

In the experiments, we used the morphemic and lexemic CCG lexicons for Turkish and CCGBank for English. CCGBank is used in multi-tagging experiments presented in Section 5.2. C&C POS tagger and supertagger is trained on sections 02-21 and tested on section 23. The rest of the experiments are conducted with Turkish morphemic CCG lexicon. It contains 5660 sentences [1] and only around 70% of them can be parsed with CCG with the automatically induced CCG categories.

In the experiments for taggers, we use morphemic lexicon. The sentences are shuffled and randomly split into 85% training and 15% test sets. Maximum entropy taggers are evaluated 10 times with random splits and average results are presented. All the experiments in Section 5.5 are performed with a single split, the same set of training and test files are used in each configuration. Experiments in Section 5.6 are performed similarly, but on the lexemic lexicon instead of the morphemic one.

[1] The number of sentences is different than reported before, because we use the corrected Treebank version in [14] that has more sentences due to corrected tokenization errors

Some statistics about the morphemic lexicon and the training and test splits are as follows. The training data contains 295 distinct supertags, the test data contains 183. With a cut-off value of 5 (used for C&C experiments with Turkish) observed distinct training categories reduce to 140. However, the total number of observed supertags in all the training file decreases to only 58304 from an initial value 58582 for the one without cut-off. Out of 183 categories in the test data 167 of them is observed in the training data which can cover 10700 tokens out of total 10717 test tokens. This is an indication of some rare categories that may be extracted erroneously with the automatic extraction procedure that is used in [14]. When cut-off is applied to the training, these values decreases to 129 and 10645 respectively. The morphemic lexicon contains 32 POS tags.

CHAPTER 5

EXPERIMENTS

In this section, we present the experiments conducted on the data sets and we discuss their results. Experiments begin with an analysis on part of speech tags. Then multi-tagging results on CCGBank are presented for POS tagging and CCG supertagging. After these initial experiments, we present maximum entropy taggers for both part of speech tagging and CCG category assignments. Finally, dependency parsing results on Turkish morphemic and lexemic CCG lexicons are presented with newly developed taggers.

5.1 Part of Speech Information for Supertagging

Word categories as parts of speech play an important role in CCG supertagging as CCG categories are syntactic types closely related to these categories. In this section, several baseline supertaggers that only use part of speech information are presented. The supertaggers presented here only use n-grams in a window of several tags without any knowledge of previous supertags or words in the sentences. CCG only allows adjacent categories to combine, therefore, looking at consecutive POS tags to assign a supertag can be seen as the simplest algorithm for supertagging.

Table 5.1: Number of n-grams in each setting and average number of supertags for each n-gram.

N-GRAM	NUM	AVG	CUT-OFF	NUM$_{CUT}$	AVG$_{CUT}$
1-GRAM { C }	32	33.16	-	32	33.16
2-GRAM { P C }	600	5.95	20	254	10.73
3-GRAM { P C N }	4368	2.43	5	1377	4.63
5-GRAM {PP P C N NN}	29154	1.23	5	1223	3.04
3-GRAM {PP P C }	4366	2.65	5	1382	5.27

These supertaggers are trained on the Turkish morphemic lexicon and they assign the most common category associated with the POS tag sequence. The n-gram tagger utilises the principle of adjacency of CCG. For the n-grams with more than one POS tag a back-off mechanism is used. If a 5-gram in test data is not observed in the training data, 3-gram version of the POS sequence is used for prediction, if 3-gram is not seen then 2-gram and 1-gram tags are used. Although its accuracy is lower than the 3-gram with previous and next POS tags, an alternative 3-gram version is presented using the tag before the previous tag of the current word instead of the next one, concerning the head-final nature of Turkish.

Table 5.1 presents the number of n-grams observed in training data in the *NUM* column and the average number of supertags associated with those n-grams in each configuration in the *AVG* column. Two sets of experiments are performed on each n-gram configuration. The first one uses all observed tag sequences, the second one discards a sequence if it is observed less than a given cut-off value shown by *CUT-OFF* to eliminate noise. The results of the experiments with cut-off values are presented in columns with subscript *CUT*. N-grams are shown with the POS tags in curly braces. *C* denotes current tag, *P* means previous tag and *N* is the next POS tag. *PP* is the one before previous and *NN* is the one after the next tag.

Table 5.2: N-gram supertaggers for Turkish with POS features.

EXPERIMENT	ACCURACY	COVERAGE	CUT-OFF	ACCURACY$_{CUT}$	COVERAGE$_{CUT}$
1-GRAM { C }	41.10	100.0	-	41.10	100.0
2-GRAM { P C }	50.76	99.86	20	50.85	96.22
3-GRAM { P C N }	59.31	97.30	5	59.06	88.50
5-GRAM {PP P C N NN}	58.47	63.56	5	59.52	31.71
3-GRAM {PP P C }	51.70	97.31	5	52.01	88.56

Supertagging accuracy of the n-gram taggers are presentend in Table 5.2. The scores are averages of 10 experiments with random splits on Turkish morphemic lexicon (85% for training, 15% for test). We obtain close to 60% accuracy with only using POS Tags in a 3-gram model with previous and next POS tags. C&C[18] supertagger which uses the features presented in Table 3.1 has accuracy at around 70% for Turkish on the same data. This demonstrates that POS tagging accuracy is very important for supertagging. In this table *COVERAGE* shows how many POS n-grams in the test data are directly observed in the training data and therefore how much back-off is used for supertagging. Since cut-off eliminates singular tag pairs and treats them as noise, the average number of tags associated with an n-gram is higher in those configurations and the coverage is lower. 5-gram experiments are the ones clearly affected by using thresholds for n-gram dictionaries due to sparseness as shown as the number of 5-grams in Table 5.1. 4-gram experiments are skipped for the sake of symmetry and simplicity.

Table 5.3: N-gram multi-taggers ($\beta = 0.5$ and $\beta = 0.1$) for Turkish with POS features with the same cut-off values.

	β=0.5				β=0.1			
EXPERIMENT	**ACC**	**NUM**	**ACC$_{CUT}$**	**NUM$_{CUT}$**	**ACC**	**NUM**	**ACC$_{CUT}$**	**NUM$_{CUT}$**
C	55.03	1.43	54.68	1.43	84.33	3.76	84.44	3.82
P C	64.24	1.59	64.07	1.56	87.62	2.91	87.44	3.72
P C N	71.55	1.43	71.20	1.56	86.57	1.99	88.08	3.23
PP P C N NN	68.92	1.56	71.28	1.54	78.41	1.22	87.56	2.75
PP P C	65.70	1.48	66.00	1.70	85.31	2.14	87.32	3.66

Table 5.3 presents a multi-tagging approach similar to the one used in C&C supertagger. Some POS tags sequences are very distinct and can only be associated with a single supertag, but some may have more than one supertag that can relate to them. Therefore, we may want to assign more than one category in some cases. Here, β value defines a threshold based on the most common CCG category associated with the POS sequence. For example, $\beta = 0.5$ means all supertags with probabilities of at least $0.5 \times P$ where P is the probability of the most common category, are included in the multi-tag set. Smaller β values increase the accuracy since more tags can be associated with a word. The multi-tag set is assumed correct if it contains the gold-standard category of the word. NUM columns show the average number of supertags assigned to a word in multi-tagging configuration. Experiments are presented with two β values and with/without applying cut-off. Several n-gram examples are presented in Tables 5.4 and 5.5 with distinct CCG categories as well as ambiguous ones. Table 5.4 shows how the n-gram length helps to solve ambiguity and Table 5.5 shows how POS tags change the outcome in the same n-gram category. In both tables, *TAG SEQUENCE* shows the n-grams, *SEQ. COUNT* shows the number of times that n-gram is observed in the training data and *SUPERTAG COUNTS* shows the most common categories associated with that n-gram.

Table 5.4: Longer n-gram sequences help in some cases. Here, 5-gram sequence shows single verb sentences whereas bigram and trigram sequences are ambiguous.

TAG SEQUENCE	SEQ. COUNT	SUPERTAG COUNTS
Verb	15695	(S 4309) (S\NP 3911) ((S\NP[nom])\NP 1930)
BGN Verb	458	(S 336) (S\NP 57) (S/S 25)
BGN Verb Punc	112	(S 89) (S\NP 10) (S\NP[nom] 5) (S/S 3) ((S\NP)/(S\NP) 3) (NP 2)
BGN BGN Verb Punc END	59	(S 59)

Table 5.5: Punctuations and associated POS tag sequences.

TAG SEQUENCE	SEQ. COUNT	SUPERTAG COUNTS
Punc	10351	(. 5449) (, 1851) (conj 1837)
Verb **Punc** END	3873	(. 3715) (... 151)
Noun_Nom **Punc** Noun_Nom	343	(conj 164) (, 120)
Noun_Nom **Punc** Noun_Gen	76	(, 58) (conj 7)
Noun_Nom **Punc** Noun_Loc	54	(, 38) (conj 7)
Noun_Nom **Punc** Noun_Acc	36	(, 32) (conj 1)

The results presented in this section show that the preceding and the following word categories are the most informative ones. This result is expected as CCG only allows adjacent categories to combine. Using 5 POS tags however did not contribute much to the accuracy of the supertagger and in some cases such as the multi-taggers without cut-off it has a negative effect. The trigram results with only previous tags are not much different from the bigram results. One would expect that verb in a sentence like "Ali kitap okudu" would benefit from the knowledge of two previous nouns, however, the outcome does not support that claim. This also shows -similarly with the 5-gram results- that when you move further from the main word denoted with *C* - *(current*

POS tag) the information gain is not significant.

5.2 Multiple Part of Speech Tags for Multi-Supertagging on English

Multi-tagging is the process of assigning more than one tag or category to the words when the uncertainty levels of labels for that word is above a certain threshold. Clark and Curran [18] defines that threshold in terms of some percentage over the most probable label. When probability of a category is not decisive over other categories, they propagate all categories above the limit to the next level and leave the decision to further stages. They experimented with assigning multiple supertags for parser and obtained great improvements in both coverage and accuracy as shown in Table 3.2. Curran er al. [22] also experimented with multi POS tagging for CCG and demonstrate its benefits for supertagging. In this section, we also experiment with assigning multiple part of speech tags to the words to improve the accuracy of the supertagger.

C&C uses part of speech tags in a window of 5 words as features for the supertagger. Therefore, the accuracy of the supertagger is directly related to the accuracy of the POS tagging procedure. Tables 5.6 and 5.7 show the accuracy of the supertagger under various conditions. Columns show tagging accuracies on words (POS and SUPER) and sentences (S_POS and S_SUPER) meaning all tags are correct in a sentence. In multi-tagging experiments, the given set of tags is accepted as correct if the expected tag -i.e. gold standard- is among the the given ones. β values control the thresholds for multi-tags. For example if $\beta = 0.5$, the most probable tags and the tags with probabilities higher than half of the most probable tag are assigned to the word. $\beta = 0.01$ is the most relaxed configuration for supertagger and it is also the the default value. Therefore we used $\beta = 0.01$ for the supertagger as fixed and experimented with different values of β for the POS tagger.

Taggers are trained on CCGBank with the sections 2-21. Tests are conducted on the section 23 of CCGBank -and also Penn Treebank, WSJ23 - Test data contains 2407 sentences with 55371 words with an average of 23 words per sentence. The length of the sentences directly affects the accuracy of the tagging on the whole sentence. Even

1% improvement on the per word accuracy greatly improves the tagging accuracy for the whole sentence.

Table 5.6: Multi-tagging experiments on WSJ23 with C&C Tools where $\beta_{MPOS} = 0.1$, $\beta_{MSUPER} = 0.01$

EXPERIMENT	POS	SUPER	S_POS	S_SUPER
SUPERTAGGING with GENERATED POS TAGS	97.2	92.0	56.6	34.7
SUPERTAGGING with GOLD POS TAGS	*	93.3	*	39.3
MULTI-TAGGING with GENERATED POS TAGS	97.2	97.8	56.6	71.3
MULTI-TAGGING with GOLD POS TAGS	*	98.6	*	78.9
MULTI-TAGGING with GENERATED MULTI-POS TAGS	99.3	98.5	86.3	78.0

The rows with the gold-standard POS tags (2 and 4 in Table 5.6 and 1 in Table 5.7) contain the maximum values achieved for both single and multiple tagging of supertags.

When we assign single POS tags to each word, the supertagging accuracy is 97.2%. In addition, only 56.6% of all the sentences have all of their parts of speech tags correct. Instead of assigning a single label when the POS categories are ambiguous with probabilities close to each other, multiple POS tags can be assigned to inform supertagger of possible other context. Table 5.6 compares the results of experiments of single and multiple tagging with $\beta_{MPOS} = 0.1$, $\beta_{MSUPER} = 0.01$. As word tagging accuracy improves to 99.3%, sentence tagging accuracy increases to 86.3% for POS tags. We improve the accuracy of the supertagger for sentences from 34.7% (single POS tags and single supertags) to 78% (multiple POS tags and multiple supertags). The best we can get in supertagging with C&C is 78.9% by using gold-standard POS tags. This is expected to result in higher parsing accuracy. Appenddix A provides a detailed multi-tagging analysis for an example sentence.

Different values of β for POS tagger controls the variations of tagged sentences that are input to the supertagger. As β decreases, the possibility of assigning multiple labels increases and more variations of sentences with different combinations of POS tags are input to the supertagger. Table 5.7 compares different values of β. First 2 rows contain single POS tags. Last 2 columns contain average number of tags per word for POS tags and supertags. As the number of POS tags assigned to each word increases, the average number of supertags also increases as expected. Multi-tagging accuracy increases from 71.3% to 78.0%.

Table 5.7: Multi POS Tagging experiments on WSJ23 with different β values for Multi POS Tagger

EXPERIMENT	POS	SUPER	S_POS	S_SUPER	A_POS	A_SUPER
GOLD POS TAGS $\beta_{MSUPER} = 0.01$	*	98.6	*	78.9	-	1.72
GENERATED POS TAGS $\beta_{MSUPER} = 0.01$	97.2	97.8	56.6	71.3	-	1.73
$\beta_{MPOS} = 0.5, \beta_{MSUPER} = 0.01$	98.2	98.1	69.6	74.2	1.03	1.81
$\beta_{MPOS} = 0.1, \beta_{MSUPER} = 0.01$	99.3	98.5	86.3	78.0	1.10	2.06
$\beta_{MPOS} = 0.01, \beta_{MSUPER} = 0.01$	99.7	98.7	93.3	79.6	1.23	2.59

When $\beta = 0.50$, each sentence given 2.06 different POS tagged variations on average. THese variations are extracted with possible combinations of multple POS tags for each word. Then these version of the sentences are passed to the supertagger. This slight increase in complexity brings 3% improvement in supertagging accuracy on sentence level. When we decrease β to 0.10, variations increase to 30.14. However, the distribution is not harmonious. There is a sentence with over 13000 combinations of POS tags. When this sentence is removed, average number of variations shrink to 24 and when the top 10 sentences with the highest number of variations are removed, the average is 15 different POS tagged sentences for each multi-pos-tagged sentence. A pruning strategy can be adapted to eliminate this kind of extreme sentences. With this small sacrifice we can achieve an higher accuracy as 78.0% which is very close to the expected maximum with the gold standard POS tags as 78.9%. On the other hand, if we further pursue multi-tagging with $\beta = 0.01$, variations increase to 2800 different POS tagged variations per sentence and this is a great compromise for the improvement acquired even though this value reduces to 670 if we remove top 10 ambiguous sentences. We hardly make any decisions for POS tagging and leave everything to the supertagger with this configuration. Sentence coverage for supertags is higher than supertagging with gold standard POS tags. However, training/testing time increases as we try every variation and it is unreasonable in practice.

5.3 A Maximum Entropy POS Tagger for Turkish Morphemic CCG Lexicon

Table 5.8: Features used by Ratnaparkhi's English POS Tagger

CONDITION	FEATURE
w_i is not rare	w_i
w_i is rare	w_i prefixes of length 1-4
	w_i suffixes of length 1-4
	w_i contains hyphen
	w_i contains number
	w_i contains uppercase
for all w_i	w_{i-2}
	w_{i-1}
	w_{i+1}
	w_{i+2}
	t_{i-1}
	t_{i-2},t_{i-1}

We adapted the Maximum Entropy Tagger proposed by Ratnaparkhi [38] to Turkish morphemic CCG lexicon with various new features and modifications. Maximum Entropy models are easier to adapt with new features and several morphological features are added to the tagger for Turkish. The taggers are implemented with NLTK Maximum Entropy Classifier [30] with MegaM [23] for optimization. The accuracy of the initial tagger on this corpus with the same feature set used by Ratnaparkhi is 87.88%. These features are presented in Table 5.8. *Rare word* in this context means the ones that are observed less than 5 times in the training data.

Table 5.9: Feature set of Turkish POS Tagger

CONDITION	FEATURES	
for all w_i	w_i	
	w_i prefixes of length 1-4	
	w_{i-2}	w_i is suffix
	w_{i-1}	w_i suffix
	w_{i+1}	w_i is morphemic tag
	w_{i+2}	w_i morphemic tag
	t_{i-1}	w_i base tag
	t_{i-2},t_{i-1}	w_i contains hyphen
	w_i prefix of length 5	w_i contains number
	w_i suffixes of length 1-4	w_i contains uppercase
	w_{i-2} prefix of length 5, if word	w_{i+1} prefix of length 5, if word
	w_{i-1} prefix of length 5, if word	w_{i+2} prefix of length 5, if word

Table 5.10: Letter modifications based on Turkish sound harmony

TRANSITION TABLE			
a, e → A	i, ı → I	u, ü → I	o, ö → O
c, ç → C	g-ğ, k → K	b, p → P	d, t → T

Derivational and inflectional morphemes are usually suffixed in Turkish morphology. Because of the agglutinative nature of Turkish, NLP applications for this language can benefit from morphological information. There is a fixed set of suffixes that can be used and these suffixes adapt to the word they are attached to by means of vowel harmony and assimilation rules. Therefore, a transition table is used to change the letters in suffix features. The letters in the suffixes and their approximations with varying lengths are transformed according to the transitions presented in Table 5.10 in feature extraction process. This modification improves the accuracy of the tagger to 89.86%. Another feature we hypothesize that is useful for Turkish is the stems of the words. Several studies including Akkuş and Çakıcı [2] and Can et al. [12] have shown that a fixed-length pseudo-stem that uses 5 character prefixes can be used to approximate stemming with an actual morphological analysis in Turkish. Adding 5-letter prefixes as features increases the accuracy upto 91.30%.

Morphemic lexicon [14] has additional morphological information. It presents some inflectional suffixes with POS tags such as ("-nDA"|Loc) separated from their words (such as pençesinde|Noun_Loc)). These tokens are labelled as suffixes and are added to the feature list. In addition, suffixes with lengths from 1 to 4 are used for all words instead of only rare ones. Additionally, derivational suffixes are also presented as separate entities such as ("Adj+With"|Adj) for (önemli|Noun_Nom) in the morphemic lexicon. These tokens are labelled as morphemic tags and are used as features for the tagger. Rare word elimination is removed since data size is limited compared to English and prefixes of length 1 to 4 letters are extracted for all words. Finally, 5-letter pseudo-stems are extracted for all words in a window of 5, two preceding and two following words. At the end, the average accuracy of 10 random splits of training(85%) and test(15%) sentences is increased to 92.8%. The new features are presented in Table 5.9 and the extracted features of a word is presented in Table 5.11.

A multi-pos-tagger is also proposed with two simple methods. Multi-taggers output a set of POS tags with associated probabilities. Since POS tagger uses the tags assigned to the words preceding the current word as its features, integrating multiple tags is a problem. However, even though these tags are used as features, they are not the most informative ones. As presented in Table 5.12, many of the most informative features of the Maximum Entropy Model are related to the words, prefixes and suffixes. Therefore, two models are proposed, one uses no history and another uses the most probable tags as history. The accuracy on sentence level is presented in Table 5.13 with comparison to the baseline tagger with features presented in Tables 5.8 and 5.9. X_{BEST} denotes the multi-tagger uses the most propable tag as history and X_{HIST} denotes no history. β values work in the same way as the ones presented in the previous sections.

Table 5.11: Feature set for word "incelerken" in (... ait|Postp performansını|Noun_Acc incelerken|Verb "Adv+While"|Adv başlıca|Adj ...)

FEATURE LABEL	FEATURE
t-1	Noun_Acc
t-2 t-1	Postp Noun_Acc
suffix(4)	rKAn
suffix(3)	KAn
suffix(2)	An
suffix(1)	n
prefix(1)	i
prefix(2)	in
prefix(3)	inc
prefix(4)	ince
prefix(5)	incel
w-2	ait
w-1	performansını
w+1	"adv+while"
w+2	başlıca
w-2_p	ait
w-1_p	perfo
w+2_p	başl

5.4 A Maximum Entropy Supertagger for Turkish Morphemic CCG Lexicon

Maximum Entropy Model of Clark and Curran [18] is reimplemented in Python with NLTK Library and MegaM with the same feature set presented in Table 3.1. The accuracy of the initial implementation dropped to 67.79% vs. 69.88% with its C++ implementation. It is possible that we miss certain optimizations or implementation details, however, we expect to see a similar change in C&C implementation if the same features are added to that implementation. Starting from this baseline, new features are added to the model based on word categories and morphological information. The first improvement is achieved with suffixes and morphemic tags. Adding these features increased the accuracy by 0.4%. Adding prefix and suffixes with varying lengths improved the supertagger upto 69.7%.

Table 5.12: Most informative features of the Maximum Entropy POS Tagger: Most of the features are related to the words, suffixes and morphological analyses.

RATIO	FEATURE
15.279	w+1==“noun+inf” and label is Verb
15.049	w+1==“verb+pass” and label is Verb
13.499	contains-uppercase==True and label is Loc
13.469	t-1==<START> and label is Adv
-13.121	label is Loc
13.003	w+1==“noun+pastpart” and label is Verb
11.815	w+1==“verb+caus” and label is Verb
11.193	w+1==“adj+pastpart” and label is Verb
10.534	w+1==“adj+prespart” and label is Verb
10.370	w+1==“adj+with” and label is Noun_Nom
9.991	w+1==“noun+zero” and label is Num
9.824	w+1==“noun+ness” and label is Adj
-9.754	w-1==<START> and label is Adv
9.495	w+1==“-ya” and label is Noun_Dat
9.227	w+1==“verb+able” and label is Verb
9.176	w+1==“noun+zero” and label is Adj
8.904	w+1==“-nda” and label is Noun_Loc
8.679	suffix(1)==A and label is Noun_Dat
8.177	w+1==“adj+rel” and label is Noun_Loc
8.170	w+1==“noun+futpart” and label is Verb
8.095	suffix(2)==lA and label is Noun_Ins
8.082	suffix(2)==TA and label is Noun_Loc
7.964	w+1==“noun+zero” and label is Postp
7.664	w+1==“verb+acquire” and label is Noun_Nom
7.582	w+1==“-dan” and label is Noun_Abl
7.578	suffix(3)==TAn and label is Noun_Abl
7.359	w+1==“-ya” and label is Pron_Dat
7.324	w+1==“-yla” and label is Noun_Ins
7.314	w+1==“adj+futpart” and label is Verb
6.985	w+1==“noun+ness” and label is Noun_Nom

Table 5.13: Comparison of Part of Speech Taggers for Turkish on single-tagging and multi-tagging modes.

	BASE		NEW	
	WORD	**SENT**	**WORD**	**SENT**
POSTAGGER	87.00	28.86	93.24	51.59
MULTIPOS$_{HIST}$, $\beta = 0.5$	90.30	38.28	95.23	61.60
MULTIPOS$_{HIST}$, $\beta = 0.1$	96.24	64.66	98.10	80.45
MULTIPOS$_{BEST}$, $\beta = 0.5$	89.65	36.16	95.09	60.90
MULTIPOS$_{BEST}$, $\beta = 0.1$	96.40	66.08	98.23	81.51

Table 5.14: Proposed features additional to the feature set of [18]

WORD	**POS**	**SUPER**
$w_{i-p(5)}$	t_{i-case}	prefix(1)
$w_{i-1-p(5)}$	$t_{i-1-case}$	prefix(2)
$w_{i-2-p(5)}$	$t_{i-2-case}$	prefix(3)
$w_{i+1-p(5)}$	$t_{i+1-case}$	prefix(4)
$w_{i+2-p(5)}$	$t_{i+2-case}$	
w_i contains number	t_{i-1},t_i ,t_{i+1}	suffix(4)
w_i contains hyphen	w_i is suffix	suffix(3)
w_i contains uppercase	w_i suffix	suffix(2)
w_i is morphemic tag	w_i morphemic tag	suffix(1)
w_i is morphemic base	w_i morphemic rest	

Suffixes and morphemic tags are treated separately and are added as new features. Case information for nouns is added and nominal case is also treated as a separate feature since CCG categories have nominal sub-categorization for noun phrases. These additional features increase the accuracy by 0.2%. Experiments with more detailed morphological information such as stems and morphological analyses instead of fixed length prefix and suffixes are also performed. However, these additions did not im-

Table 5.15: Comparison of Supertaggers on single and multi-tagging mode.

	C&C		**BASE**		**NEW**	
	WORD	**SENT**	**WORD**	**SENT**	**WORD**	**SENT**
SUPERTAGGER	68.97	22.14	67.65	21.79	68.34	21.08
MULTI-TAGGER, $\beta = 0.5$	77.11	29.09	-	-	-	-
MULTI-TAGGER, $\beta = 0.1$	96.05	71.50	-	-	-	-
MULTI-TAGGER$_{HIST}$, $\beta = 0.5$	-	-	73.08	26.74	73.51	24.97
MULTI-TAGGER$_{HIST}$, $\beta = 0.1$	-	-	83.13	35.10	83.45	35.22
MULTI-TAGGER$_{BEST}$, $\beta = 0.5$	-	-	70.22	12.60	72.93	14.84
MULTI-TAGGER$_{BEST}$, $\beta = 0.1$	-	-	86.26	35.45	87.06	36.98

prove the performance (68.51%). This is probably due to the fact that we are already working with a morphemic lexicon which includes most of the features that can be extracted from morphological analyses and prefix and suffixes include the necessary information that can be derived from the full morphological analyses.

A multi-tagger is also built for supertagging with the same procedure described for the POS tagger. Accuracy results for the supertaggers are presented in Table 5.15. Since C&C multi-tagging procedure is different, its results are given in separate rows. These results are achieved with gold-standard POS tags. There is a slight increase in the performance of the tagger with additional features but the change is not as significant as the POS tagger. The most informative features in training the superagger are presented in Table 5.16. These are mainly supertags of the previous words, POS tags and words with morphological tags such as *Verb+Zero*. The supertagger mostly uses word categories rather than the words themselves. Therefore, rather than a direct contribution with morphological information on the supertagger level, a supertagger can benefit from morphological features through POS tagging. An example feature set of the word "incelemeye" is presented in Table 5.17[1].

[1] we use *s-supertag* to denote supertags contrary to *c-category* as used by C&C

Table 5.16: Most of the features are the ones with previous supertags, part of speech tags and morphological tags.

RATIO	FEATURE
60.978	contains-uppercase==True and label is “S\(S/S)”
-51.068	label is “S\(S/S)”
11.238	w+1_p==“"verb” and label is “S”
9.611	t==“verb” and label is “S”
8.721	s-1==“NP[nom]” and label is “NP[nom]\NP[nom]”
8.037	t==“verb” and label is “S\NP[nom]”
8.010	w+1_p==“"verb” and label is “S\NP”
-7.928	t+1==“noun” and label is “(S\NP)\S”
7.874	w+1_p==“"verb” and label is “S\S”
7.835	s-1==“NP” and label is “NP\NP”
7.768	label is “NP/NP”
7.673	t==“verb” and label is “S*bs*NP”
7.540	s-1==“NP[nom]/NP[nom]” and label is “NP[nom]”
7.508	w+1_p==“"adj+” and label is “NP”
7.217	s-1==“S/S” and label is “(S/S)\(S/S)”
-6.978	contains-uppercase==True and label is “(S\NP)\S”
6.872	label is “NP”
6.833	w+1_p==“"noun” and label is “NP”
6.766	t==“verb” and label is “(S\NP[nom])\NP”
6.619	s-1==“(S\NP[nom])\NP” and label is “NP\(S\NP[nom])”
6.593	w+1==“"verb+zero"” and label is “NP”
6.554	s-1==“(S\NP[nom])\NP” and label is “S\(S\NP[nom])”
6.520	s-1==“NULL” and label is “NULL”
6.471	s-1==“(S/S)/(S/S)” and label is “S/S”
6.164	w+1_p==“"noun” and label is “NP\NP”
6.119	is_nom==True and label is “NP[nom]\NP”
6.076	s-1==“(NP/NP)/(NP/NP)” and label is “(NP/NP)\(NP/NP)”
5.973	s-1==“NP” and label is “NP[nom]\NP”
5.934	s-1==“(S\NP[nom])\NP” and label is “(NP/NP)\(S\NP[nom])”
5.906	s-1==“((S\NP[nom])/(S\NP[nom]))\NP” and label is “S\NP[nom]”

Table 5.17: Feature set for word “incelemeye” in (... dikkati|Noun_ Nom|NP ile|Conj|((S\NP)/(S\NP))\NP incelemeye|Verb|S\NP "Noun+Inf"|Noun_ Dat|NP\S başladı|Verb|(S\NP[nom])\NP ...)

FEATURE LABEL	FEATURE	FEATURE LABEL	FEATURE
“w-2”	“dikkati”	“t+1_case”	“dat”
“w-2_p”	“dikka”	“t-2_case”	“nom”
“w-1”	“ile”	“t-2”	“noun”
“w-1_p”	“ile”	“t-1”	“conj”
“w”	**“incelemeye”**	**“t”**	**“verb**
“w+1”	“"noun+inf"’,	“t+1”	“noun”
“w+1_p”	“"noun”		
“w+2”	“başladı”	“t+2”	“verb”
“w+2_p”	“başl”		
“prefix(1)”	“i”	“t-2 t-1”	“noun conj”
“prefix(2)”	“in”	“t-1 t”	“conj verb”
“prefix(3)”	“inc”	“t-1 t t+1”	“conj verb noun”
“prefix(4)”	“ince”	“t-1 t+1”	“conj noun”
“prefix(5)”	“incel”	“t t+1”	“verb noun”
“suffix(4)”	“mAyA”	“t+1 t+2”	“noun verb”
“suffix(3)”	“AyA”		
“suffix(2)”	“yA”	“s-1”	“((S\NP)/(S\NP))\NP”
“suffix(1)”	“A”	“s-2 s-1”	“NP ((S\NP)/(S\NP))\NP”

5.5 Turkish Dependency Parsing on Morphemic Lexicon

In this section, combinations of the proposed models are used for Dependency Parsing of Turkish using the morphemic CCG Lexicon in Çakıcı and Steedman [11]. In order to parse a raw sentence, one needs a POS tagger to assign word categories to be used by supertagger as features. Then, a supertagger is needed to assign CCG categories that dependency parser uses. The sub-perfect accuracy on both of these stages reflects on the accuracy of the dependency parser. Çakıcı [14] has shown the benefits of using supertags as fine grained tags in dependency parsing with MSTParser, however gold standard POS tags are used in Çakıcı [14] and POS taggers that are necessary for parsing raw data are not included to the experiments. Tables 5.18 and 5.19 present accuracies of the taggers. Supertagger results are presented with gold-standard POS tags, baseline tagger with the features of Ratnaparkhi [38] and the improved one with the new features as presented in Section 5.3.

Table 5.18: Accuracy for POS Taggers used for Dependency Parsing

	BASE	NEW
WORD POS ACCURACY	87.00	93.24
SENTENCE POS ACCURACY	28.86	51.59

Table 5.19: Accuracy for Supertaggers used for Dependency Parsing

	C&C		BASE		NEW	
	WORD	SENT	WORD	SENT	WORD	SENT
GOLD POS TAGS	68.97	22.14	67.65	21.79	68.34	21.08
BASE POS TAGGER	64.45	16.61	62.71	16.73	64.27	17.08
NEW POS TAGGER	66.83	19.91	65.43	19.91	66.31	19.91

Table 5.20 presents the results of dependency parsing with MSTParser for labelled and unlabelled accuracy. Top-left cell contains the results with gold-standard POS tags and gold-standard CCG categories. This is the upperbound for the parser that we compare our results with. On the left are POS taggers and on the top are the supertaggers. Each cell on the table shows the results on the combination of those two. Raw sentences are POS tagged with the tagger on the left and supertagged with the tagger on the top, whics are then passed to the dependency parser. The first row contains unlabelled accuracy and unlabelled complete accuracy. The second row contains labelled accuracies.

Table 5.20: MSTParser results for each configuration in terms of labelled and unlabelled dependencies.

GOLD	SUPER	C&C SUPER		BASE SUPER		NEW SUPER	
POS	89.66 53.12 85.80 41.11	WORD	SENT	WORD	SENT	WORD	SENT
GOLD POS	UNLABELLED	81.71	32.27	81.49	31.92	82.03	31.33
	LABELLED	74.93	21.79	74.82	22.61	75.39	21.67
BASE POS	UNLABELLED	79.21	26.38	78.74	26.50	79.99	26.73
	LABELLED	71.04	16.25	70.33	17.43	71.74	17.90
NEW POS	UNLABELLED	80.44	29.56	80.48	30.51	80.82	29.68
	LABELLED	72.93	19.20	72.83	20.73	73.36	19.79

Table 5.20 shows the effects of the tagger on dependency parsing. POS taggers are working very close to the gold standard, and the performances of the dependency parser with gold standard POS tags and the POS tags assigned by the improved model are very close. This shows that the rest of the gap between the dependency parser with the gold standard tags and the dependency parsers with the automatically assigned tags is due to the supertagger performance.

Supertagger performance is not improved with additional features as much as the POS tagger. There may be many reasons behind this. First of all, the data is very sparse for Turkish compared to English (60K vs. 1M). Additionally, there are only 32 POS tags bu 10 times more supertags in the morhemic lexicon and the supertagger uses a lot more features than the POS tagger. The supertags are also automatically induced and they might have inherited the dependency errors in the treebank. This is also evidential by the fact that only around 70% of the sentences can be parsed and assigned a derivational tree by the CCG parser.

Secondly, we may be evaluating the tagger erroneously due to the categories induced in the lexicon. There are many words including long, type-raised categories that are made of the same categories recursively due to the automatic induction. For example, modifiers such as adverbs and adjectives are assigned categories in the form of X/X, but the arguments (X's) may be atomic or complex categories. Sometimes these complex categories are very long and complex that they are very rare. Simple, shorter categories are more common and this feature is used by Baldridge [5] in his supertagger. The supertagger may assign a simple and correct category but the gold standard data may contain a more complex but also correct category. In that case the assigned tag is evaluated as wrong even though both categories (gold and assigned) may give valid parses. This also may be the reason behind the slight difference in dependency parsing results between C&C supertagger and our proposed supertagger even though our supertagger accuracy is slightly behind C&C.

Thirdly, we add morphological information to improve the performance of the supertagger on the morphemic lexicon. The morphological features we extract may be implicitly presented in the morphemic lexicon and therefore they may not be adding significantly to the supertagger. POS tagger provides morphological information in the form of tags to the supertagger. This is evident from the fact that most of the informative features of the clasifier are related to word categories (POS tags).

5.6 Turkish Dependency Parsing on Word-based (Lexemic) Lexicon

All experiments performed on the morphemic lexicon are also conducted on the lexemic version. This section presents the results of the dependency parsing on non-morphemic lexicon and together with the accuracies of the taggers.

Table 5.21: Accuracy for POS Taggers used for Dependency Parsing

	BASE	NEW
WORD POS ACCURACY	78.75	86.98
SENTENCE POS ACCURACY	22.85	39.46

Table 5.22: Accuracy for Supertaggers used for Dependency Parsing

	C&C		BASE		NEW	
	WORD	SENT	WORD	SENT	WORD	SENT
GOLD POS TAGS	61.83	24.03	60.21	24.85	60.83	25.80
BASE POS TAGGER	54.33	18.49	52.89	18.14	53.67	19.79
NEW POS TAGGER	57.75	21.44	56.53	22.61	56.81	22.38

Table 5.21 presents the POS tagger results for baseline tagger and the improved POS tagger with features presented in previous sections. Additional features are more beneficial for the lexemic lexicon which proves our claim that morphological information is carried into the supertagger through morphemic tags in the morphemic lexicon. The lack of fine grained tag names like the ones in the morphemic lexicon in these experiments are compensated with the morphological features. Table 5.22 presents supertagger results for Clark and Curran's supertagger, its reimplementation, and the new supertagger with additional features. Supertagger accuracies are presented with the POS taggers described and the gold-standard POS tags similar to Table 5.19.

Table 5.23 presents dependency parsing results with MSTParser on lexemic Turkish CCG lexicon. Top left cell contains the results with gold-standard part-of-speech tags and CCG categories. Each row contains POS tags from the given alternatives on the leftmost column and each column contains supertags assigned by the supertaggers presented in the topmost row. In each cell unlabelled accuracies are on the first line and labelled accuracies are on the second one. The item on the left is word accuracy and the one on the right is sentence accuracy.

The results on word-based lexicon show similar behaviour to the morphemic lexicon. The improvements in the taggers are reflected to the results as shown in differences between the cell containing result of Base POS / Base Super combination and the cell containing New POS / New Super combination. Compared to the morphemic lexicon, the effects of the improvements on POS tagging are higher in lexemic version. On the other hand, labelled dependency accuracies of experiments on morphemic lexicon

Table 5.23: MSTParser results for each configuration in terms of labelled and unlabelled dependencies.

GOLD	**SUPER**	**C&C SUPER**		**BASE SUPER**		**NEW SUPER**	
POS	**88.94 56.30 81.01 38.04**	**WORD**	**SENT**	**WORD**	**SENT**	**WORD**	**SENT**
GOLD POS	**UNLABELLED**	79.96	38.87	79.07	38.63	79.47	38.75
	LABELLED	65.25	22.14	64.37	22.14	64.76	21.91
BASE POS	**UNLABELLED**	75.53	32.16	74.32	31.68	75.13	31.45
	LABELLED	58.27	16.37	56.60	15.43	57.82	16.96
NEW POS	**UNLABELLED**	77.35	35.22	76.90	35.57	77.14	35.10
	LABELLED	61.28	18.26	60.86	18.96	60.95	18.96

are closer to the accuracies presented with gold standard tags. Unlabelled dependency recoveries are around 10% lower than the ones with gold standard tags in both lexemic and morphemic lexicons.

CHAPTER 6

CONCLUSION

In this thesis, we performed dependency parsing on Turkish sentences. Previous studies have focused on gold-standard annotated data with POS tags and CCG categories and shown the improvements that can be achieved by using CCG supertags as fine grained tags for dependency parsing [14]. We started with untagged data and performed two tagging steps before passing data to the dependency parser. The performances of the taggers directly affect the performance of dependency parsing and we focus on improving these two taggers.

The importance of POS tagging for supertagging procedure is shown with simple supertaggers that only use POS information in a limited window. The results suggest that the POS tags can used as main features of the supertaggers. Multi-tagging experiments are performed for multiple POS tag assignments and improvements are demonstrated when POS tag ambiguity is preserved to a certain extent before supertagging.

A maximum entropy POS tagger is proposed with various new features for improvements. Morphological features and their approximations with prefixes and suffixes are added to the maximum entropy model. These additions proved to be beneficial for POS tagging and improved the accuracy of the tagging process. This improvement is also reflected on to the following supertagging and dependency parsing processes. The parser results with the gold-standard POS tags and the ones tagged with the proposed POS tagger are very close. Additionally, this model is adapted to multiple POS tagging with two simple approaches results of which are reported.

Maximum entropy model of Clark and Curran [18] for supertagging is reimplemented and modified with additional features similar to the POS tagging model. This model is used with the gold-standard POS tags as well as the output of the POS tagger presented in this thesis. We show performance of the parser with the gold-standard tags are still within reach when automatic taggers utilised on the same data.

The morphemic CCG lexicon used in this study as the gold standard data is automatically extracted and contains annotation errors due to the nature of the process. Only around 70% of the sentences have a valid CCG derivation. Many of the mentioned errors can be fixed by manual or semi-automatic lexicon generation. Further improvements and corrections are necessary to improve the performance of NLP applications using this data set.

Multi-tagging presents promising results in improving the coverage and the accuracy of the taggers. Preserving ambiguity in the lower levels of NLP pipeline and leaving the decisions to the upper levels are shown to be beneficial. Clark and Curran [18] uses multi-tagging to successfully improve CCG parsing performance. The MST-Parser only accepts single categories as fine-grained tags and we could not adapt it to the multi-taggers. Future studies will focus on integrating multi POS tagger and multi supertagger with dependency parsing.

REFERENCES

[1] Kazimierz Adjukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935. English translation “Syntactic Connexion" by H. Weber in McCall, S. (Ed.) *Polish Logic*, pp. 207–231, Oxford University Press, Oxford, 1967.

[2] Burak Kerim Akkuş and Ruket Çakıcı. Categorization of turkish news documents with morphological analysis. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 1–8, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[3] Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. Improving dependency parsers using combinatory categorial grammar. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 159–163, 2014.

[4] Nart B Atalay, Kemal Oflazer, Bilge Say, et al. The annotation process in the Turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, 2003.

[5] Jason Baldridge. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 57–64. Association for Computational Linguistics, 2008.

[6] Srinivas Bangalore and Aravind K Joshi. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265, 1999.

[7] Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, pages 47–58, 1953.

[8] Alexandra Birch, Miles Osborne, and Philipp Koehn. Ccg supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16. Association for Computational Linguistics, 2007.

[9] Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. The Prague dependency treebank. In *Treebanks*, pages 103–127. Springer, 2003.

[10] Cem Bozşahin. The combinatory morphemic lexicon. *Computational Linguistics*, 28(2):145–186, 2002.

[11] Ruket Çakıcı and Mark Steedman. A wide-coverage morphemic CCG lexicon for Turkish. In *Parsing with Categorial Grammars Workshop ESSLLI 2009 Bordeaux, France Book of Abstracts*, pages 11–15, 2009.

[12] Fazlı Can, Seyit Koçberber, Erman Balçık, Cihan Kaynak, H Çağdaş Öcalan, and Onur M Vursavaş. Information retrieval on Turkish texts. *Journal of the American Society for Information Science and Technology*, 59(3):407–421, 2008.

[13] Ruken Çakıcı. Automatic induction of a CCG grammar for Turkish. In *Proceedings of the ACL student research workshop*, pages 73–78. Association for Computational Linguistics, 2005.

[14] Ruket Çakıcı. *Wide-coverage parsing for Turkish*. PhD thesis, The University of Edinburgh, 2009.

[15] Ruket Çakıcı and Jason Baldridge. Projective and non-projective Turkish parsing. In *Proceedings of the 5th International Treebanks and Linguistic Theories Conference*, pages 43–54, 2006.

[16] John Chen, Srinivas Bangalore, Michael Collins, and Owen Rambow. Reranking an n-gram supertagger. In *Proceedings of the TAG+ Workshop*, pages 259–268, 2002.

[17] Stanley F Chen and Ronald Rosenfeld. A gaussian prior for smoothing maximum entropy models. Technical report, DTIC Document, 1999.

[18] Stephen Clark and James R Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, 2007.

[19] Stephen Clark and James R Curran. Supertagging for efficient wide-coverage CCG parsing. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*, pages 221–233, 2010.

[20] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[21] Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. A statistical parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 505–512. Association for Computational Linguistics, 1999.

[22] James R Curran, Stephen Clark, and David Vadas. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 697–704. Association for Computational Linguistics, 2006.

[23] Hal Daumé III. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August 2004.

[24] Gülşen Eryiğit and Kemal Oflazer. Statistical dependency parsing of Turkish. In *Proceedings of the 11th EACL*, pages 89–96, Trento, 2006.

[25] Julia Hockenmaier. *Data and models for statistical parsing with Combinatory Categorial Grammar*. PhD thesis, The University of Edinburgh, 2003.

[26] Julia Hockenmaier. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 359–366. Association for Computational Linguistics, 2003.

[27] Julia Hockenmaier and Mark Steedman. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 335–342. Association for Computational Linguistics, 2002.

[28] Julia Hockenmaier and Mark Steedman. CCGBank: a corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396, 2007.

[29] Taku Kudo and Yuji Matsumoto. Japanese dependency structure analysis based on support vector machines. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 18–25. Association for Computational Linguistics, 2000.

[30] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics, 2002.

[31] David Mitchell Magerman. *Natural Language Parsing As Statistical Pattern Recognition*. PhD thesis, Stanford University, Stanford, CA, USA, 1994. UMI Order No. GAX94-22102.

[32] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[33] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics, 2005.

[34] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer, 2003.

[35] Joakim Nivre. *Inductive dependency parsing*. Springer, 2006.

[36] Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. Building a Turkish treebank. In *Treebanks*, pages 261–277. Springer, 2003.

[37] W Keith Percival. Reflections on the history of dependency notions in linguistics. *Historiographia linguistica*, 17(1-2):29–47, 1990.

[38] Adwait Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA, 1996.

[39] David Reitter, Julia Hockenmaier, and Frank Keller. Priming effects in Combinatory Categorial Grammar. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 308–316. Association for Computational Linguistics, 2006.

[40] Bilge Say, Deniz Zeyrek, Kemal Oflazer, and Umut Özge. Development of a corpus and a treebank for present-day written Turkish. In *Proceedings of the eleventh international conference of Turkish linguistics*, pages 183–192, 2002.

[41] Mark Steedman. *The syntactic process*, volume 35. MIT Press, 2000.

[42] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax Oxford: Blackwell*, pages 181–224, 2011.

[43] Anna Szabolcsi. On combinatory categorial grammar. In *Proceedings of the Symposium on Logic and Language*, volume 87, pages 151–163, Debrecen, Budapest: Akademiai Kiado, 1987.

[44] Anna Szabolcsi. Bound Variables in Syntax: Are There Any? In Renate Bartsch, Johan van Benthem, and Peter van Emde Boas, editors, *Semantics and Contextual Expression*, pages 295–318. Foris, Dordrecht, 1989.

[45] Lucien Tesnière. *Eléments de syntaxe structurale*. Librairie C. Klincksieck, 1959.

[46] Michael White and Jason Baldridge. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126, 2003.

[47] Michael White, Rajakrishnan Rajkumar, and Scott Martin. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)*, pages 267–276, 2007.

APPENDIX A

MULTITAGGING ENGLISH

A.1 Input sentence

Table A.1: Tokenized input sentence

John	both	likes	and	recommends	the	movie	.

A.2 Supertagged Sentence

Table A.2: Supertagged sentence with default POS tagger and supertagger

WORD	POS	SUPER
John	NNP	N
both	DT	(NP\NP)/N
likes	NNS	N
and	CC	conj
recommends	VBZ	(S[dcl]\NP)/NP
the	DT	NP[nb]/N
movie	NN	N
.	.	.

A.3 Multi Supertagged Sentence

Table A.3: Multitagged sentence with the default POS tagger and category probabilities.

WORD	POS	#	SUPER	PROB$_{SUPER}$
John	NNP	3	N	0.800833
			N/N	0.157488
			NP/NP	0.0416787
both	DT	8	(NP\NP)/N	0.489516
			N/N	0.146728
			NP[nb]/N	0.146423
			(S\NP)/(S\NP)	0.0676818
			NP/NP	0.0640752
			((S\NP)\(S\NP))/N	0.0544851
			NP\NP	0.0140347
			(S[adj]\NP)/(S[adj]\NP)	0.0057501
likes	NNS	1	N	0.991768
and	CC	1	conj	1
recommends	VBZ	4	(S[dcl]\NP)/NP	0.794285
			((S[dcl]\NP)/PP)/NP	0.147138
			(S[dcl]\NP)/S[dcl]	0.0205725
			((S[dcl]\NP)/(S[adj]\NP))/NP	0.00858627
the	DT	1	NP[nb]/N	0.998092
movie	NN	1	N	1
.	.	1	.	1

A.4 Multi POS Tagged Sentence

Table A.4: Multi POS tagged sentence and POS probabilities.

WORD	#	POS	PROB$_{POS}$
John	1	NNP	1
both	1	DT	0.932881
likes	3	NNS	0.483528
		NN	0.315563
		VBZ	0.200909
and	1	CC	1
recommends	1	VBZ	1
the	1	DT	1
movie	1	NN	1
.	1	.	1

A.5 Parsed Sentences

Table A.5: Parsed sentences with CCG categories for each POS tagged variation and their grammatical relations. Notice the difference in tags and dependencies when *likes* tagged as a *verb*.

WORD	POS	SUPER	GRAMMATICAL RELATIONS
John	NNP	N	dobj both_1 likes_2
both	DT	(NP\NP)/N	det movie_6 the_5
likes	NNS	N	dobj recommends_4 movie_6
and	CC	conj	conj and_3 recommends_4
recommends	VBZ	(S[dcl]\NP)/NP	conj and_3 both_1
the	DT	NP[nb]/N	ncmod _ John_0 both_1
movie	NN	N	xmod _ John_0 recommends_4
.	.	.	ncsubj recommends_4 John_0 _
John	NNP	N	dobj both_1 likes_2
both	DT	(NP\NP)/N	det movie_6 the_5
likes	NN	N	dobj recommends_4 movie_6
and	CC	conj	conj and_3 recommends_4
recommends	VBZ	(S[dcl]\NP)/NP	conj and_3 both_1
the	DT	NP[nb]/N	ncmod _ John_0 both_1
movie	NN	N	xmod _ John_0 recommends_4
.	.	.	ncsubj recommends_4 John_0 _
John	NNP	N	conj and_3 recommends_4
both	DT	(S\NP)/(S\NP)	conj and_3 likes_2
likes	VBZ	(S[dcl]\NP)/NP	det movie_6 the_5
and	CC	conj	dobj recommends_4 movie_6
recommends	VBZ	(S[dcl]\NP)/NP	dobj likes_2 movie_6
the	DT	NP[nb]/N	ncmod _ recommends_4 both_1
movie	NN	N	ncmod _ likes_2 both_1
.	.	.	ncsubj recommends_4 John_0 _
			ncsubj likes_2 John_0 _

A.6 Multitagged Sentence - all tags combined

Table A.6: Category probabilities for each POS tag for *likes*. Empty cells mean that category is not among the supertags given in that configuration. Even though only one POS tag is different in these three versions, since supertagger uses a window of 5 words, probabilities of the same categories are different in each configuration.

WORD	POS	SUPER	*likes*=NNS	*likes*=NN	*likes*=VBZ
John	NNP	N	0.800833	0.844543	0.96891
		N/N	0.157488	0.105609	0.0275499
		NP/NP	0.0416787	0.0498477	
both	DT	(NP\NP)/N	0.489516	0.58231	0.0173873
		N/N	0.146728	0.0936997	
		NP[nb]/N	0.146423	0.118799	
		NP			0.0174419
		NP/NP	0.0640752	0.0650045	0.0155345
		NP\NP	0.0140347	0.00860581	
		(S\NP)/(S\NP)	0.0676818	0.0256594	0.931944
		((S\NP)\(S\NP))/N	0.0544851	0.0914259	
		(S[adj]\NP)/(S[adj]\NP)	0.0057501		
likes	NNS	N	0.991768	0.993035	0.179977
	NN	(S[dcl]\NP)/NP			0.656936
	VBZ	S[dcl]\NP			0.113411
		(S[dcl]\NP)/PP			0.0127988
		((S[dcl]\NP)/PP)/NP			0.00954901
		(S[dcl]\NP)/(S[adj]\NP)			0.00937118
		(S[dcl]\NP)/S[dcl]			0.00685088
and	CC	conj	1	1	0.999227
recommends	VBZ	N		0.0150418	0.0109052
		(S[dcl]\NP)/NP	0.794285	0.742061	0.932469
		((S[dcl]\NP)/PP)/NP	0.147138	0.124226	0.031921
		S[dcl]\NP)/NP)/NP		0.0108737	
		(S[dcl]\NP)/S[dcl]	0.0205725	0.0647049	
		((S[dcl]\NP)/(S[adj]\NP))/NP	0.00858627	0.0149339	
		((S[dcl]\NP)/(S[to]\NP))/NP		0.010912	
the	DT	NP[nb]/N	0.998092	0.997408	0.997881
movie	NN	N	1	1	1
.	.	.	1	1	1