Wide-Coverage Parsing for Turkish

Ruket Çakıcı

Doctor of Philosophy

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh

2008

Abstract

Wide-coverage parsing is an area that attracts much attention in natural language processing research. This is due to the fact that it is the first step to many other applications in natural language understanding, such as question answering.

Supervised learning using human-labelled data is currently the best performing method. Therefore, there is great demand for annotated data. However, human annotation is very expensive and always, the amount of annotated data is much less than is needed to train well-performing parsers. This is the motivation behind making the best use of data available. Turkish presents a challenge both because syntactically annotated Turkish data is relatively small and Turkish is highly agglutinative, hence unusually sparse at the whole word level.

METU-Sabancı Treebank is a dependency treebank of 5620 sentences with surface dependency relations and morphological analyses for words. We show that including even the crudest forms of morphological information extracted from the data boosts the performance of both generative and discriminative parsers, contrary to received opinion concerning English.

We induce word-based and morpheme-based CCG grammars from Turkish dependency treebank. We use these grammars to train a state-of-the-art CCG parser that predicts long-distance dependencies in addition to the ones that other parsers are capable of predicting. We also use the correct CCG categories as simple features in a graph-based dependency parser and show that this improves the parsing results.

We show that a morpheme-based CCG lexicon for Turkish is able to solve many problems such as conflicts of semantic scope, recovering long-range dependencies, and obtaining smoother statistics from the models. CCG handles linguistic phenomena i.e. local and long-range dependencies more naturally and effectively than other linguistic theories while potentially supporting semantic interpretation in parallel. Using morphological information and a morpheme-cluster based lexicon improve the performance both quantitatively and qualitatively for Turkish.

We also provide an improved version of the treebank which will be released by kind permission of METU and Sabancı.

Acknowledgements

This thesis is produced in an environment where everyone I will mention here did something small or big but infinitely invaluable to make this happen. I wouldn't be able finish this thesis if any of them were missing in my life.

My first and foremost thanks goes to Mark Steedman. His experience, knowledge and enthusiasm on professional matters and life is truly inspiring. His advice has been life-saving at times. Without him, this thesis would not exist. I thank my examiners, John Carroll and Frank Keller, who gave me invaluable feedback which made the thesis more readable and more complete.

I met Sumru Özsoy, at a very late stage in my PhD. However, her support, guidance and feedback helped me through the difficul times when finishing my PhD. She has given extensive feedback on many parts of the thesis and kept me from making countless mistakes. I also thank her for being warm, caring and positive at all times.

Cem Bozşahin, Deniz Zeyrek and Akira Otani have read parts or all of my thesis and provided invaluable feedback. I thank Cem in particular because I would not have started working in this area if it weren't for him.

I also thank Miles Osborne, my second supervisor, who has always challenged the way I think. I will miss our little chats in the kitchen in BP.

James Curran and Stephen Clark have helped me adapt the C&C parser to Turkish and saved me from getting lost in the depths of the code. I am deeply thankful to them for their guidance and advice. Julia Hockenmaier was always helpful with all sorts of questions regarding CCG and parsing, and gave extensive feedback and advice on the rest of the thesis in various face-to-face and e-mail discussions for which I am thankful. I also thank Jason Baldridge for the parsing scripts used in this thesis for dependency parsing evaluation and also for convincing me to do a PhD in Edinburgh.

I thank Bilge Say in METU and Kemal Oflazer in Sabancı University for providing the METU/Sabancı Turkish Treebank, and also guiding the correction process with extensive feedback.

Sebastian Riedel has been a great friend and office mate. He also provided the dependency structure viewing module which was of great help. I thank him for being one of my best friends over the years. I thank Umut Özge for endless discussions on CCG and Turkish, and also the XML parser for the treebank, Abhishek Arun and James Clark being great officemates. I will not forget our early morning coffee and bagel extravaganza. I am especially grateful to Abhishek for helping me whenever I

needed help, being a true friend and helping me to submit my thesis from overseas. I thank my friend Utku for scripts and his friendship.

I thank Ben Hachey for proving to be friend of hard-times by organising the submission of my thesis together with Avril Heron, who I am also truly and utterly grateful.

Emre Ugur has given me support with his endless enthusiasm. I thank him for being a driving force and a best friend. He has been there for me in the worst of times.

I thank my parents, Şenol and Yalçın, my brother Mehmet and my sisters Gülen and Gizem for always supporting and loving me. I cannot find words to describe my gratitude the time and effort my parents spent on making everything perfect for their children. I wouldn't be able to finish this thesis without them.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ruket Çakıcı*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Parsing is an integral part of natural language understanding systems, since semantic interpretation requires the syntactic information provided by parsing. What makes parsing natural languages hard is the size and high ambiguity of the grammars. This results in a much bigger search space for natural language parsing and requires grammar induction and statistical modeling to guide search.

Creating wide-coverage natural language grammars by hand is impractical. There are two approaches that can be taken. One is *unsupervised* grammar extraction which has been overshadowed by the success of the alternative *supervised* methods. The downside of the supervised methods is that they require labelled data. Although syntactically labelled English corpus is bigger than data available for many languages, even it is considered to be small to make reliable estimates for this language. The aim of this thesis is to identfy the problems that are casued by the limitations on the size of the labelled data and suggest solutions in order to make the most use of available data however limited.

Supervised statistical parsers for English and other languages were created soon after labelled data became available. However, the popularity of statistical parsing increased when lexicalised parsers based on head-word dependency models represented an advance on simple probabilistic context-free grammars (PCFG, Booth and Thompson, 1973). In the late 1990s, a number of lexicalised head-dependency model based parsers were proposed (Magerman, 1995; Collins, 1997; Charniak, 1997; Collins, 1999; Charniak, 2000). This led to a new emphasis on the the dependency based evaluation, an alternative to PARSEVAL which measures the similarity to the human-labeled gold standard context-free trees (Carroll, Briscoe, and Sanfilippo, 1998; Lin, 1998; Collins, 1999; Clark, Hockenmaier, and Steedman, 2002).

Dependency theories developed in parallel to the phrase structure based theories such as Head Driven Phrase Structure Grammars (HPSG) (Pollard and Sag, 1994), Lexical Functional Grammars (LFG) (Bresnan, 1982). Dependency theories are lexicalist theories in the sense that all the syntactic relations are represented as simple directed links between lexical entities. Head-dependent relations are defined between the representational units in the lexicon. Dependency representations allow to directly represent non-projective dependencies, thus are preferred for freer order languages (Skut et al., 1997).

Natural languages are known to require more than context free power (Shieber, 1985). However, the linguistic evidence that they are non-context free does not imply full context sensitive power. Several theories of grammar are conjectured to exhibit limited extra power that is required for certain language phenomena. These theories are considered to be in a subclass of context sensitive grammars which is called the mildly-context sensitive grammars. Certain constrained versions of dependency languages are proven to be equivalent to mildly context sensitive languages (Kuhlmann and Möhl, 2007). This result generalises previous work on the relation between dependency grammars and Lexicalised Tree Adjoining Grammars by Rambow and Joshi (1997). Linear-Indexed Grammars (Joshi, Levy, and Takahashi, 1975) have been argued to be a promising candidate for a class of grammars that have all and only the extra expressive power beyond CFG that is needed to explain the lingusitic phenomena of natural languages (see Gazdar (1988) for discussion). Grammar theories such as Combinatory Categorial Grammar (Steedman, 2000), and Tree Adjoining Grammar (Joshi, Levy, and Takahashi, 1975) are weakly equivalent to Linear-Indexed Grammars and, thus, may well be expressive enough, without overgenerating Thus, the representational structure chosen for annotating a dependency (or any kind of) treebank is crucial. Like parsers, the formalism used in annotation should be expressive enough to represent all the linguistic phenomena specific to that language.

Simpler structures of representation are desirable for the sake of efficiency in annotation process. However, the balance between simplicity of representation and expressiveness of the formalism is of vital importance. The main aim of a parser (computational or human) is to deliver an interpretable meaning representation of the sentence to be parsed. Most of the data can be explained with context-free rules, and thus projective acyclic dependency trees. Long-distance dependencies and other “deep” dependencies are, in fact, rare because of the skewed distributions that apply to all linguistic phenomena. However, they are crucial in recovering predicate-argument structure and

semantic interpretation. Post-processing methods that recover some deep dependencies from context-free phrase-structure trees that are outputs of shallower parsers were suggested in the past (Dienes and Dubey, 2003; Levy and Manning, 2004; Cahill et al., 2008; Johnson, 2002).

Treebank annotations that only represent surface context-free relations limit the possibility of attaining and evaluating wide-coverage recovery of full semantic interpretations. However, treebanks like these are not uncommon, and in fact the METU-Sabancı treebank we take as our starting point is of this kind. The impact aimed in this thesis is to show that a parsing system using an expressive theory of grammar (CCG) can be created from a less expressive method of representation with the use of morphological information and including a limited amount of deep linguistic information in the form of deep dependencies. This system is expected to yield more linguistically meaningful parsing results when compared to the results by parsers that only use the surface syntactic information.

CCG is known for handling difficult phenomena such as coordination and cross-serial dependencies elegantly. CCG parsers are fast, polynomial time parsers that can compose semantic interpretation in the form of logical expressions in parallel to parsing. Multiple-head dependencies and the necessary non-projective dependencies are automatically modeled in CCG. CCG parsers can produce meaningful output to create semantic interpretation very efficiently. Clark, Steedman, and Curran (2004) show that they can semi-automatically be adapted to new domains with only lexical category annotation and training a supertagger or category labeler with this data.

There are high accuracy parsers that parse dependency structures directly. Multilingual dependency parsing is advancing with the introduction of dependency treebanks in many languages. However, the recognition problem of unrestricted non-projective dependency grammars is NP-complete (Neuhaus and Bröker, 1997). In additon, the effect of multiple heads that is required by constructions such as relativisation is not clear.

We argue for deep dependency parsing, and also for morphemic lexicons for agglutinative languages. Turkish is a language with unusual morphosyntactic interactions, free word order and other phenomena such as pro-drop, requires an expressive enough parser. Experimental parsers have been built for Turkish with CCG in the past(Baldridge, 2002; Bozşahin, 2002). We explore the full potential of using an expressive formalism with a challenging language that has characteristics that is very dissimilar to languages most commonly focussed on in computational linguistics re-

search such as English.

1.1 The Thesis

The main thesis of the present work is that a grammar that takes account of the rich productive morphology of the Turkish language on the one hand, and the constrained universal combinatory projection mechanism of CCG on the other, yields a satisfactory basis for wide coverage parsing for that language.

1.2 Contribution

This dissertation makes the following contributions:

- A demonstration of the inadequacy of extreme surface syntactic approaches to dependency annotation for a language with a high degree of freedom in word order and pro-drop. Additional information in the form of secondary dependencies was added to the treebank in an attempt to include missing predicate-argument relations that is lost during extraction, or that is simply absent in case of coordination. We made a number of systematic corrections to the treebank in the form of dependency label corrections, head corrections, morphological annotation corrections and even tokenisation corrections in some cases. A much cleaner version of the treeank which includes crucial long-distance dependencies that were missing results although the annotation is still open to improvement.

- A wide coverage CCG lexicon for a lesser studied language in parsing. In the past, CCG grammars for English (Hockenmaier, 2003a) and German (Hockenmaier, 2006) have been extracted. Especially, the grammar for English, proved to be competitive in overall performance and what is more important, in predicting the long-range dependencies which other systems usually fail to do. In this thesis, we study a language in the lesser studied languages class. It is highly agglutinative unlike English, German, or other more widely researched languages.

  A CCG lexicon is induced from the data which is an improved version of the lexicon presented in Çakıcı (2005). This lexicon is based on the principle that different parts of a word in an agglutinative language can split the set of dependents of that word. One solution to handle this is the separation of these

morphemes (or morpheme sets) and including them in the lexicon as separate entities. Another solution is keeping all the inflected forms of a word in the lexicon as separate entities and possibly using the morphological information to guess the dependents. We believe that the former solution is superior to the latter when there is limited data. We explore these possible solutions and their impacts on recovery of surface and deep linguistic information from dependency data. We provide results for:

1 A Word-based (Lexemic) lexicon (Chapter 5) and parsers (Chapter 8) that predict surface dependencies only.

2 A Morphemic lexicon (Chapter 6) and parsers (Chapter 8) that predict surface dependencies only with this lexicon.

3 Both of these lexicons with C&C parser that provides deep linguistic information in the form of long-range dependencies and so on (Chapter 9).

We claim that these individual investigations show that morpheme-based representation of the lexicon provides both qualitative and quantitative advantage in all parser configurations. In contrast to English, languages with rich morphology seem to benefit from inclusion of morphological information during processing (Eryiğit, Nivre, and Oflazer, 2008; Çakıcı and Baldridge, 2006; Dyer, 2007; Çetinoğlu and Oflazer, 2006). These languages also show similar behaviour in parsing compared to languages without strong morphology (Nivre et al., 2007), (McDonald, 2006).

- A comparison of direct dependency parsing that outputs dependencies without an intermediate level and a parser that is able to output deep linguistic knowledge is done. The output of three parsers: Collins' parser, McDonald's MST-Parser (non-projective and projective configurations), and C&C parser are examined. The outputs of these systems are to some extent incompatible. However, a partial evaluation and comparison is possible. This dissertation gives the first wide-coverage results for dependency parsing of Turkish with a syntactically and semantically expressive grammar formalism.

1.3 Outline

Chapter 2 provides information about the Turkish dependency treebank that is used in this thesis. The characteristics and the limitations of the surface syntactic dependency annotation for Turkish is is discussed with examples and an extensive amount of corrections made to the Turkish dependency treebank are explained. Possible solutions to some of the most important problems are proposed and implemented.

Chapter 3 provides a basic introduction to morphology. The interaction of morphology, syntax and semantics in the view of some interesting linguistic phenomena such as bracketing paradoxes and phrasal scope of morphemes is covered. Computational analysis of morphological parsing and the history of morphological processing is given together with an overview of morphological parsers and disambiguators that were created for Turkish in the past. Morphology in relation to Categorial Grammars are also discussed in this chapter.

Chapter 4 gives an introduction to CCG grammar formalism and discusses the theoretical issues regarding how certain natural language phenomena are treated within CCG and also discusses the generative power of CCG grammars. A review of statistical parsing with CCG is given in this chapter. The systems that are intended for wide-coverage parsing, and CCG parsers for other languages are also briefly introduced.

Chapter 5 outlines the algorithm for mapping dependency structures to a CCG lexicon automatically. An evaluation of this grammar is given together with some examples of CCG derivations. A review of the earlier research on automatically or semi-automatically creating CCG lexicons from other languages is also given.

Chapter 6 demonstrates why a morphemic lexicon is more appropriate for languages like Turkish and creates a linguistically more sound version of the lexicon induction process that is based on the idea that morphemic entities rather than words are not only more appropriate for solving the problems mentioned earlier like phrasal scope, but also improves the coverage and provides a degree of generalisation. The effects of having smaller representational units than words in a CCG lexicon of an agglutinative language is evaluated and results are compared with the results from the word-based lexicon.

Chapter 7 gives an overview of the dependency theory and dependency parsing history. Several influential parsers are reviewed in this chapter, including McDonald's MSTParser which is used in some parsing experiments in Turkish that resulted in state-of-the-art parsing results for Turkish dependency parsing.

In **Chapter 8** the process of experimentation on direct and indirect accounts of dependency parsing are explained and the results of these experiments are discussed. Different parsing models of Turkish are proposed in this chapter to explore the effects of morphology, word-order, scrambling, and formal constraints of projectivity on different parsers. The chapter presents the use of morphological features and use of gold-standard CCG categories as simple features to parsers that directly estimate dependency relations and provide upper bounds for realistic parsing with these features.

Chapter 9 provides a review of the existing wide-coverage CCG parsers and CCG parsing results for Turkish. The inner workings of the parser used for parsing Turkish is explained and the advantages and limitations of the current configuration is explored. The dependency output from this parser is evaluated against a compatible set of dependencies from the treebank, and is compared with the output of the direct dependency parsing experiments performed with the MSTParser. The advantages and the potential for improvement for these models are discussed.

Chapter 10 concludes by summarizing the contributions and the way each chapter sheds light on the issues introduced in this dissertation. Directions for future research are also given in this chapter.

Chapter 2

Data

Turkish is highly-inflected and has more word order flexibility than languages like English. It is an agglutinating language, which means words are formed with linear concatanation of affixes in a compositonal and surface structural sense. In an agglutinating language like Turkish a single word can be a sentence with tense, agreement, polarity, and voice as in (2.1) and translate into a relatively longer English sentence. Morphological structure of the words bears clues about part-of-speech tags, modality, tense, person and number agreement, case, voice and so on. As in (2.1b) predicate-argument structure goes through transformational changes through morphology which makes the morphology-syntax interface more complex.

(2.1) a. Gidemeyebilirdim.
go-Abil-Neg-Possib-Aor-Past-P1sg
I might not have been able to go.

b. Konuşturmalısın.
speak-Caus-Oblig-P2sg
(You) must make (someone) speak.

c. Arabandakiyleyim.
car-Poss2sg-Loc-Rel-Inst-CopPers1sg
I am with the one in your car.

Turkish has relatively free word order. Although it is said to have an SOV base constituent order, it allows both local and long-distance scrambling. The former means that arguments of verbs may swap order within a clause, and the latter means that an argument may appear in a higher clause than that of the verb which subcategorises

for it. Alternative word orders are subject to discourse and information structure restrictions (Erguvanlı, 1979; Hoffman, 1995). The syntactic roles of the arguments are indicated by case marking as in (2.3). There are certain exceptions to free word order. Certain scrambling patterns are not allowed in some structures such as relativisation. Bozşahin (1998) proves this by giving evidence from gapping and coordination, and showing that some permutations such as SO & OSV are not allowed. Subordinated words have relatively restricted argument order. As seen in (2.2) and (2.3) these types of relativisation constructions are always head-final.

(2.2) a.

Çaldığı	şekerleri	ceplerine	dolduruyordu.
steal-PastPart	candy-Plu-Acc	pocket-Plu-Poss-Dat	stuff-Prog-Past

He was stuffing the candies he stole in his pockets.

b.

*Şekerleri	çaldığı	ceplerine	dolduruyordu.
candy-Plu-Acc	steal-PastPart	pocket-Plu-Poss-Dat	stuff-Prog-Past

(2.3) a.

Kitapları	kapıya	gelen	adama	verdim.
books-Acc	door-Dat	come-PresPart	man-Dat	give-Past-P1sg.

b.

Kapıya	gelen	adama	verdim	kitapları.
door-Dat	come-PresPart	man-Dat	give-Past-P1sg	books-Acc.

I gave the books to the man who came to the door

c.

*Gelen	kapıya	adama	verdim	kitapları.
come-PresPart	door-Dat	man-Dat	give-Past-P1sg	books-Acc.

2.1 METU-Sabancı Treebank

The METU-Sabancı Treebank is a sub-corpus of the METU Turkish Corpus which is a 2 million word corpus of post-1990 written Turkish. The METU Turkish Corpus includes material taken from three daily newspapers, 87 journal issues and 201 books (Atalay, Oflazer, and Say, 2003; Oflazer et al., 2003). The sentences in the treebank are taken from this corpus retaining the proportions of the contributing sources. The dependency treebank has 5620 sentences and 53,796 tokens (with punctuation). The average sentence length is 9.6 tokens and 8 words. Figure 2.1 shows the distribution of the average sentence length. The sentence length changes dramatically among different genres, going up to as high as 53 for scientific articles and news. Sentences

from fiction and literature are relatively shorter consisting of many one-word sentences from dialogues. Note that this graph shows the cumulative average over number of sentences. There is a sharp rise in the average sentence length after about 500^{th} and 4500^{th} sentences. This is reflected in the category type coverage and parsing coverage results later in Chapters 5, 6 and 9.

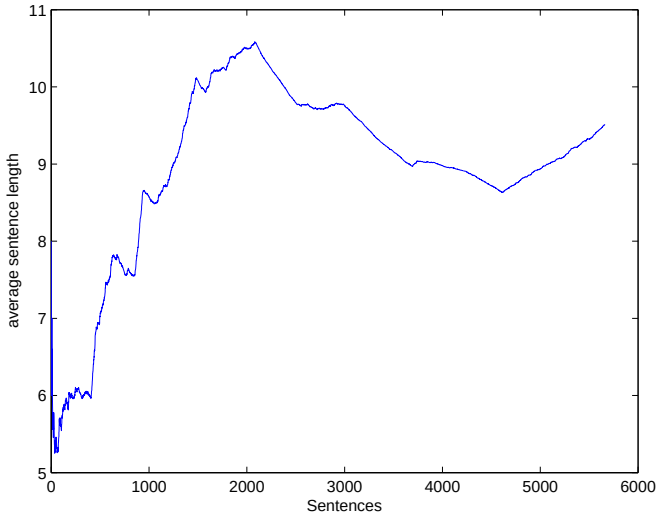


Figure 2.1: Average sentence length throughout the corpus

The words in the treebank occur together with their disambiguated morphological analyses, and surface dependency links which are represented through indexes.

(2.4)

Kapının	kenarındaki	duvara	yaslanıp	bize	baktı	bir	an	.
door-Gen	side-Loc-Rel	wall-Dat	lean-ADS*	we-Dat	looked	one	moment	.

* *ADS = AfterDoingSo*

(He) looked at us leaning on the wall next to the door, for a moment.

Figure 2.2 shows the encoding of the sentence in (2.4) in the treebank. We filled the "LEM" and "MORPH" fields appropriately for illustration purposes. These fields were initially designed to be in the data but then were left blank in the final release. "MORPH" contains the sequence of the morphemes, and "LEM" contains the infinitive form for the verbs and the root for the others. (Inflectional groups) IG field represents the morphological information in terms of derivational boundaries, type info and morpheme names for each word. They are explained in detail in Section 2.2. It is not trivial to map the morph information to the given IGs. This is mainly because IGs are made of information tags rather than morpheme names. Tags such as *Pos (positive)*,

or *Pnon (no possessive marker)* do not correspond to any morphemes, in fact, they represent the lack of corresponding morphemes in these slots, such as the agreement morpheme for the former and the negation morpheme for the latter. There are also *Zero* morphemes in some cases of derivational morphology. Figure 2.3 shows a graphical representation of the sentence in (2.4). Dependencies are from dependent to head in this figure.However, throughout the thesis, dependency links are represented from dependent to head in graphical representations unless otherwise stated. Word to word dependencies are shown in the figure although dependencies are represented between inflectional groups as explained in Section 2.2.

<S No="3">

<W IX="1" LEM="kapı" MORPH="kapı+nHn" IG='[(1,"kapı+Noun+A3sg+P2sg+Gen")]' REL="[2,1,(POSSESSOR)]"> Kapının </W>

<W IX="2" LEM="kenar" MORPH="kenar+nHn+DA+ki" IG='[(1,"kenar+Noun+A3sg+P3sg+Loc")(2,"Adj+Rel")]' REL="[3,1,(MODIFIER)]"> kenarındaki </W>

<W IX="3" LEM="duvar"MORPH="duvar+yA" IG='[(1,"duvar+Noun+A3sg+Pnon+Dat")]' REL="[4,1,(OBJECT)]">duvara </W>

<W IX="4" LEM="dayanmak" MORPH="dayan+Hp" IG='[(1,"dayan+Verb+Pos")(2,"Adv+AfterDoingSo")]' REL="[6,1,(MODIFIER)]"> dayanıp </W>

<W IX="5" LEM="bize" MORPH="biz+yA" IG='[(1,"biz+Pron+PersP+A1pl+Pnon+Dat")]' REL="[6,1,(OBJECT)]"> bize </W>

<W IX="6" LEM="bakmak" MORPH="bak+DH" IG='[(1,"bak+Verb+Pos+Past+A3sg")]' REL="[9,1,(SENTENCE)]">baktı </W>

<W IX="7" LEM="bir" MORPH="bir" IG='[(1,"bir+Det")]' REL="[8,1,(DETERMINER)]"> bir </W>

<W IX="8" LEM="an" MORPH="an" IG='[(1,"an+Noun+A3sg+Pnon+Nom")]' REL="[6,1,(MODIFIER)]"> an </W>

<W IX="9" LEM="." MORPH="." IG='[(1,".+Punc")]' REL="[,()]"> . </W>

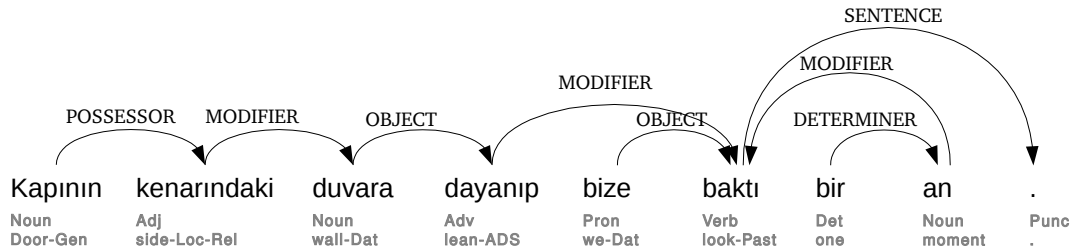
</S>

Figure 2.2: The encoding of the sentence in (2.4) in the dependency treebank

The dependencies in METU-Sabancı Turkish treebank are surface ones, so phenomena such as traces and pro-drop are not modelled. Apart from the constraint of "surface dependencies", there are a few constraints that almost all dependency treebanks with pure-dependency approach adopt such as *single-headedness*.

The syntactic relations used to model the dependency relations are given in Table 2.1. The first two columns show the name and the frequency of the label, the third is the most frequent, basic[1] part-of-speech tag it occurs with and the fourth is the frequency of them occurring together. The most frequent label in the data is MODIFIER. It can

[1]The detailed description of basic and extended POS-tags is given in Chapter 8.



(He) looked at us leaning on the wall next to the door, for a moment.

Figure 2.3: The graphical representation of word-word dependencies in the treebank.

co-occur with a very wide range of part-of-speech tags but the most frequent one is *Adj*. Adverbs, adjectives, relativised verbs, postpositions, and some nominal modifiers have this label. It is the most overloaded relation type in the treebank.

Apart from familiar label names, ETOL is used for constructions very similar to phrasal verbs in English, and COLLOCATION, is used for idiomatic usages and word sequences with certain patterns. However, some frequent collocations are integrated in the dependency structures as unified lexical entries. *Collocation* and *Etol* are mostly used to represent frequent pairs that cannot be represented in the same word-slot. This happens when they are separated by a clitic or a morpheme.

Punctuation marks are excluded from dependency structures unless they participate in a relation, such as the use of comma in coordination. The label SENTENCE links the head of the sentence to the final punctuation mark.[2] Section 2.3 gives an overview of the cases where punctuation is involved in the dependency structure.

2.2 Morphology

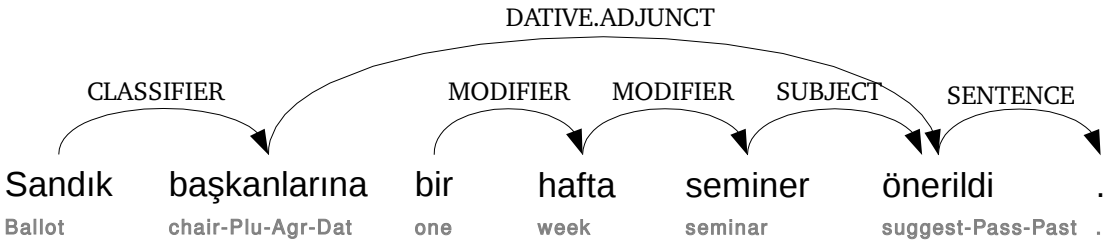
Morphology in the Turkish treebank is represented in *Inflectional Groups* (IGs). Words with more than one IG either have derivational morphology or valency altering suffixes such as causative and passive morphemes for verbs. Some modalities are also annotated in different IGs (*Ability*) whereas some of them are not (*Conditional*). (2.5) shows the OBJECT linked to the second IG of *değiştirmez* because *değiş* is intransitive; it takes an object in this example because of the valency altering property of the causative morpheme *-DHr* that is affixed to the verb.[3]

[2] This is essentially like identifying the final punctuation mark as the *root* symbol, which is how it is treated when dependency parsers are evaluated (Section 8.6).

[3] *-DHr* can be realised in many different ways depending on certain morphophonemic rules that are explained briefly in Section 3.2.5.

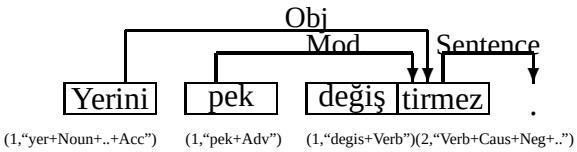
Label	Frequency	POS-tag	pair frequency
ABLATIVE.ADJUNCT	523	Noun	483
APPOSITION	101	Noun	82
CLASSIFIER	2100	Noun	1967
COLLOCATION	36	Noun	24
COORDINATION	2633	Punc	1817
DATIVE.ADJUNCT	1362	Noun	1136
DETERMINER	1915	Det	1840
EQU.ADJUNCT	34	Pron	19
ETOL	13	Adj	8
FOCUS.PARTICLE	18	Conj	14
INSTRUMENTAL.ADJUNCT	271	Noun	252
INTENSIFIER	947	Conj	858
LOCATIVE.ADJUNCT	1134	Noun	1107
MODIFIER	11618	Adj	4535
NEGATIVE.PARTICLE	164	Negp	113
NUM	3	Num	2
OBJECT	8259	Noun	6634
POSSESSOR	1507	Noun	1334
QUESTION.PARTICLE	274	Ques	219
RELATIVIZER	84	Conj	83
SENTENCE	7370	Verb	6165
S.MODIFIER	540	Conj	274
SPEAKER	51	Punc	50
SUBJECT	4536	Noun	3779
VOCATIVE	231	Noun	126

Table 2.1: Names and frequencies of the dependency labels in METU-Sabancı treebank. Last two columns show the POS tag the label occurs with most frequently, and frequency of this pair. Red coloured labels indicate the ones that exist only in the corrected version of the treebank.



One week of training was suggested for ballot chairs.

Figure 2.4: Passive example



(He) doesn't change his place that often.

Figure 2.4 shows how a passive sentence is annotated in a surface dependency framework. The dependents in passive sentences are usually made dependent on the IG with the passive morpheme but this is not consistent throughout the treebank in the official release of the treebank. These types of annotation mistakes have been corrected.

IGs thus play a role in dependency structure. Different IGs can be heads of different dependents. Dependencies always emanate from the final IG of a word (Figure 2.5).

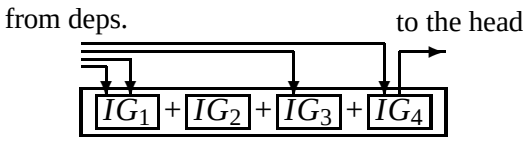


Figure 2.5: The structure of a word

2.2.1 Derivational Morphology

Adverbs

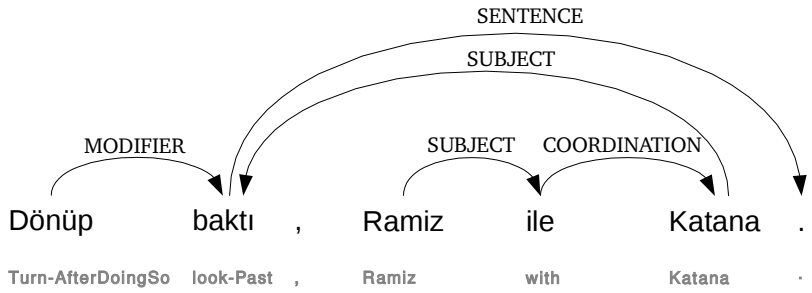
Verb-to-adverb derivation is very productive. As shown in Figure 2.6 verbs that the adverbs are derived from may have complements.These derivational suffixes resemble gerunds in English and behave in the same way. They take sentences as complements and modify the main predicate. Some examples of these morphemes and their corresponding tag names in the treebank are shown in Table 2.2. These morphemes make verb-modifying adverbs out of verbs. The capital letters represent the general form of

morpheme	tag name
-mAdAn	WithoutHavingDoneSo
-(y)Hnca	When
-Hp	AfterDoingSo
-(y)ArAk	ByDoingSo
-ken	While

Table 2.2: Verb-to-Adverb derivational morphemes

the morphemes. These capital letters change with respect to vowel harmony and other phonological processes. **(y)** means there is an optional **y** that drops if the root ends with a consonant. **A** can be **a** or **e** depending on vowel harmony restrictions. In a similar manner **H** can be instantiated as **ı** or **ı**.

Ramiz and *Katana* are shared between *dönüp* and *baktı* since the coordination of these two is the subject of both verbs (stems). But since only surface dependencies are represented, and words can not have two heads, it is not possible to represent these dependendies with the current design of representation.



E: Ramiz and Katana turned and looked.

Figure 2.6: Adverbs

Subordination

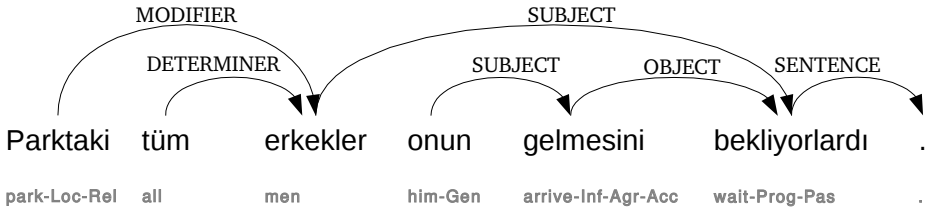
In the treebank, subordination is treated as if the subordinate verb is a noun derived from a verb. Morphemes that are involved in subordination behave as if they are derivational suffixes that create nouns out of verbs. Agreement suffixes or case markers are attached to the nominalised verb exactly in the same way nouns receive case. Figure 2.7 shows how a sentence with subordination is annotated in the Turkish treebank. Subordination morphemes are a diverse class of morphemes. Some of these

are shown in Table 2.3.

morpheme sequence	tense
-dHk+AGR+CASE	present - past
-AcAk+AGR+CASE	future
-mA+AGR+CASE	infinitive

Table 2.3: Subordination morphemes

Subordinated verbs agree with the noun that is in the subject position as AGR tag indicates. Subjects in subordination clauses are marked with genitive case, unlike their corresponding matrix clause counterparts which are always nominative. The final morpheme on the subordinated verb is the appropriate case marker. Figure 2.7 shows an example sentence with the last type of subordination *-mA+AGR+CASE* in the treebank.



All the men in the park were waiting for him to arrive

Figure 2.7: Subordination

These examples show the morpho-syntactic nature of some relations in the data. Subordination is controlled by morphological operations, however the syntactic scope subordination morpheme covers is bigger than the verb it is attached to at the constituent level. *-mesi* demand a sentential complement, and, at the same time, it controls the morphological properties (case marking) of the subject of the verb it is attached to.

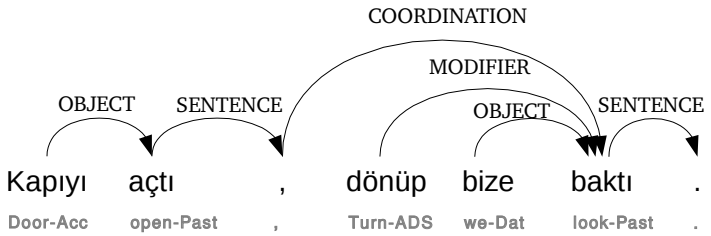
2.2.2 Nominal sentences, copula sentences

The dependency label that identifies the sentence head is SENTENCE in the Turkish treebank. This label marks the head of the sentence together with the final punctuation and it does not necessarily mean the sentence is a full sentence. Fragments are not differentiated. Some sentences consist only of an NP or other types of non-verb

phrases in the data. Since one type of copular verb is regarded as zero morpheme, these sentences cannot be distinguished from copular sentences. Some sentences with non-verbal heads incorporate as morphological information the zero copula IG. However, the remaining sentences are ambiguous between fragments or copular sentences. We do not explicitly annotate non-verbal heads, however, these go through preprocessing for lexicon induction as explained in Chapter 6.

2.3 Punctuation

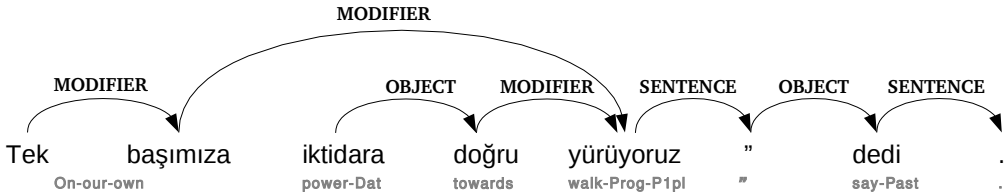
Punctuation marks can sometimes have dependents in METU-Sabancı Treebank. For instance, in coordination, the first conjunct has a link to the comma that separates (or conjoins) the two conjuncts, exactly as *ve (and)* would do (Figure 2.8).



(He) opened the door, turned and looked at us.

Figure 2.8: Comma included in the dependency structure as the conjunct in a sentence coordination.

Punctuation marks can also have different roles such as marking the sentential complements as in Figure 2.9. The head of the sentential complement depends on the intervening punctuation which is a double-quote in this instance.



We walk towards power on-out-own, he said

Figure 2.9: A double quote included in the dependency structure as the head of a sentential complement.

The sentence final punctuation acts as a root node of the whole dependency structure. The head of the sentence gets linked to the final punctuation with relation SENTENCE. Sentential complements have the dependency label SENTENCE as well, but they do not depend on the final punctuation. Therefore, only the combination of SENTENCE label and the final punctuation marks the top level of the dependency tree.

Apart from these three major uses of punctuation in the dependency structure, punctuation marks do not get involved in the dependency structure.

2.4 Morpheme names

Table 2.4 lists all the morpheme names that exist in the treebank. The version that this information is collected is the version that is published in the CoNLL 2006 shared task for dependency parsing. These are compared with the morpheme names that are in the corrected version of the treebank. The morpheme names that are present in the original treebank version, but missing in the final version are the ones that appear as different tags because of typos or annotation mistakes in the original version. An example to this is the *Since* versus *since* distinction.

2.5 Improvements on the Treebank

We have made a significant number of changes to the treebank in order to improve consistency and correctness. The correction cycle consisted of constant correction of the treebank, explained in Chapter 5, which is in turn followed by using the lexicons and the treebank itself for parsing. Most of the annotation mistakes were found during lexicon induction phase. Most of the time, when a problem was identified, the whole treebank was searched for similar problems and inconsistencies. Among many changes the most important types are listed below:

a. fixing incorrect morphological analyses of frequent words,

b. connecting tokens that previously were not connected to the dependency graph,

c. changing dependency links or labels of some relatively non-frequent types (e.g. intensifiers, appositions) to ensure consistency of annotation,

d. fixing a considerable number of incorrect dependency links or labels,

old	new	old	new
A1pl	A1pl	NotState	NotState
A1sg	A1sg	Noun	Noun
A2pl	A2pl	Num	Num
A2sg	A2sg	Opt	Opt
A3e	*NONE*	ord	Ord
A3pl	A3pl	Ord	Ord
A3sg	A3sg	P1pl	P1pl
Abl	Abl	P1sg	P1sg
Able	Able	P2pl	P2pl
Acc	Acc	P2sg	P2sg
Acquire	Acquire	P3pl	P3pl
Adj	Adj	P3sg	P3sg
Adv	Adv	Pass	Pass
AfterDoingSo	AfterDoingSo	Past	Past
Agt	Agt	PastPart	PastPart
Aor	Aor	PCAbl	PCAbl
As	As	PCAcc	PCAcc
AsIf	AsIf	PCDat	PCDat
Become	Become	PCGen	PCGen
ByDoingSo	ByDoingSo	PCIns	PCIns
Card	Card	PCNom	PCNom
Caus	Caus	PersP	PersP
Cond	Cond	Pnon	Pnon
Conj	Conj	Pos	Pos
Cop	Cop	Postp	Postp
Dat	Dat	Pres	Pres
Demons	DemonsP	PresPart	PresPart
DemonsP	DemonsP	Prog1	Prog1
Desr	Desr	Prog2	Prog2
Det	Det	Pron	Pron
Distrib	Distrib	Prop	Prop
Dup	Dup	Punc	Punc
Equ	Equ	Ques	Ques
FitFor	FitFor	QuesP	QuesP
Fut	Fut	Range	Range
FutPart	FutPart	Real	Real
Gen	Gen	Recip	Recip
Hastily	Hastily	Reflex	Reflex
Imp	Imp	ReflexP	ReflexP
InBetween	InBetween	Rel	Rel
Inf	Inf	Related	Related
Inf2	Inf	since	Since
Ins	Ins	Since	Since
Interj	Interj	SinceDoingSo	SinceDoingSo
JustLike	JustLike	Stay	Stay
Loc	Loc	Time	Time
Ly	Ly	Verb	Verb
Narr	Narr	When	When
Neces	Neces	While	While
Neg	Neg	With	With
NONE	Negp	Without	Without
Ness	Ness	WithoutHavingDoneSo	WithoutHavingDoneSo
Nom	Nom	Zero	Zero

Table 2.4: Morphological tags in METU-Sabancı Turkish treebank data.

e. correcting tokenisation errors that could be fixed. We will explain what we mean by this in the following.

f. disambiguating coordinations that involve the heads of the sentences, which cannot be distinguished with the current design principles in the treebank. The motivation behind this, and the method are explained in Section 2.5.4.

.g adding extra dependencies that are crucial in identifying the type of extraction in relativisation sentences. These links also contain missing predicate-argument relations that may be helpful in future semantic processing studies. The process is explained with examples in Section 2.5.3.

Some additions to the treebank that are not in the original design were necessary in some exceptional cases. For instance, in news articles, when a person is quoted, the periods between the sentences divided the speaker and the verb *dedi (said)* at the very end of the paragraph as in (2.7) during tokenisation of (2.6). Thus, the name of the person quoted would be disconnected from the verb it depends on which is in sentence3, and it will be irrelevant to the dependency structure of sentence1. For these cases, a new dependency label was introduced. The speaker which is referred to as Person-X as shown in (2.6) is linked to the next punctuation mark, which is usually a semi-colon, with the label SUBJECT and the punctuation mark is linked to the top level with the label SPEAKER. This solution does not connect the speaker subject with the verb it depends on, but it, at least, preserves the connectedness and consistency of sentence1 in 2.7. sentence3 is treated as if it is a sentence with a missing subject (pro-drop). SPEAKER also says that the verb of this subject is in another sentence.

(2.6) **Person-X:** sentence1. sentence2. sentence3, said.

(2.7) **Person-X:** sentence1.
sentence2
sentence3, **said**.

More substantial changes had been made to the treebank some of which were mentioned in Çakıcı (2005). These included adding secondary links for long-distance dependencies and arguments shared among conjuncts in sentence or verb coordination

structures. A brief summary of these changes is given in Section 2.5.3 and Section 2.5.4.

The main aim of the corrections is to make the treebank annotation more consistent. In the majority of cases, we opted for aligning the annotation with the most frequently occurring annotation especially when correcting morphological errors. This is not an effort to create a different treebank, but to make the existing one more self-consistent and as correct as possible. The new label types such as SPEAKER were only introduced because the problem could not be solved within the existing design principles of the treebank.

2.5.1 Morphological Changes

Morphological annotation in the Turkish treebank is performed with manual disambiguation among possible morphological parses of a word provided by a morphological analyser (Atalay, Oflazer, and Say, 2003). Morphological analyses that are wrong given the context were sometimes chosen by the annotators. A few examples of these and changes made to correct them are given here.

ikimiz (we both) was annotated as if *iki* is a number and a noun was derived from it. In fact, the reading suggested by this annotation is ”our two”, as in number two. Íkimiz (2.8) –*we (both)*– should be a pronoun like *hepimiz – we (all)*– which is annotated as a pronoun in the treebank. An alternative solution is to change the second IG to a pronoun rather than a noun.

Doğrusu (*in fact*) is an adverb. It can also be a noun in possessive case in a different context. The root – Doğru – has a noun reading, an adjective reading, and a postposition reading. The adverb *Doğrusu* is annotated as shown in (2.9) most of the time, in the Turkish treebank. However, postpositions do not get inflectional morphology and hardly receive derivational suffixes in Turkish. They may take copula morpheme as an affix as all other non-verbal stems do. Thus, this analysis for *doğrusu* as a noun is grammatically and pragmatically incorrect. Instances of this type are replaced by the annotation shown in (2.9) as ‘correct’.

(2.10) is a more general type of annotation error. This word can be analysed as two different types. The first is as a relativised verb. These are annotated as verb-to-adjective morphological derivation in the Turkish treebank. The second is as a verb future tense inflection. Relativised verbs of the first type were mostly annotated as tense-inflected verbs. These types of disambiguation mistakes are corrected by replac-

ing the first annotation in (2.10) with the second line.

(2.8) ikimiz – (1,“iki+Num+Card”)(2,“Noun+Zero+A3sg+P1pl+Nom”)
correct – (1,“iki+Pron+A1pl+P1pl+Nom”)
alternative – (1,“iki+Num+Card”)(2,”Pron+A1pl+P1pl+Nom”)

(2.9) doğrusu – (1,“doğru+Postp+PCDat”)(2,“Noun+Zero+A3sg+P3sg+Nom”)
correct – (1,“doğrusu+Adv”)

(2.10) kurtulamayacak – (1,“kurtul+Verb”)(2,“Verb+Able+Neg+Fut+A3sg”)
correct– (1,“kurtul+Verb”)(2,“Verb+Able+Neg”)(3,“Adj+FutPart+Pnon”)

These are only a few examples of a large set of corrections to morphological annotation in the treebank. These changes are not expected to make significant improvement on the parsing models to be trained on the treebank, but they all contribute to the overall consistency and correctness of the data, and are expected to be useful in tasks such as part-of-speech tagger training.

2.5.2 Wrong IGs

A considerable number of corrections were made on the dependencies among IGs. These are mostly arguments of relativised or subordinated verbs, that were annotated to depend on the last IG instead of the stem verb by mistake. These dependencies were corrected by assigning the dependency head to be the stem, not the secondary IGs. The motivation could be explained by an example case.

There are 396 genitive marked nouns with SUBJECT label. Their heads are the first IGs of the relativised or nominalised verbs. On the other hand, there are 203 nouns of the same sort that (inconsistently) depend on the second(166), the third(25), the fourth(11), and the fifth (1) IGs of the head in the original version of the treebank. This means that the general principle of annotating these must be linking them to the first IG (or the verb stem). Linguistically, if the label is called SUBJECT, we think that the dependency should be to the stem which is the verb. On the other hand, if these are accepted as genuine derivation, then for instance, subordinated verb phrases should be treated like other noun phrases which means subject label should be changed into POSSESSOR. The ideal situation would be to have both of these dependencies. This way, subject dependency would link the noun and the verb, and possessor dependency would link the noun to the aggreement suffix. However, single-headedness restriction

does not allow this. So, we corrected these dependencies as consistent with the similar cases as possible. The same was done for objects and other modifiers of these verbs.

The following example justifies our approach. In this example there are two subject labelled dependencies to the same word. This is not possible under normal circumstances. But, linguistically these subjects are governed by two different IGs. In the original version of the treebank this sentence was wrongly annotated as shown in (2.11). But the corrected dependency structure is as shown in (2.12).

(2.11)

<S No="3">

<W IX="1" ... IG='[(1,"önem+Noun+A3sg+Pnon+Nom")(2,"Adj+With")(3,"Noun+Zero+A3sg+Pnon+Nom")]' REL="[2,1,(OBJECT)]"> Önemli </W>

<W IX="2" ... IG='[(1,"ol+Verb+Pos")(2,"Adj+PresPart")]' REL="[6,3,(SUBJECT)]"> olan </W>

<W IX="3" ... IG='[(1,"öncül+Noun+A3sg+Pnon+Nom")]' REL="[4,1,(SUBJECT)]"> öncül </W>

<W IX="4" ... IG='[(1,"ile+Conj")]' REL="[5,1,(COORDINATION)]"> ile </W>

<W IX="5" ... IG='[(1,"kanıt+Noun+A3sg+Pnon+Gen")]' REL="[6,3,(SUBJECT)]"> kanıtın </W>

<W IX="6" ... IG='[(1,"çeliş+Verb+Neg")(2,"Noun+Inf+A3sg+P3sg+Nom")(3,"Verb+Zero+Pres+Cop+A3sg")]' REL="[7,1,(SENTENCE)]"> çelişmemesidir </W>

<W IX="7" ... IG='[(1,".+Punc")]' REL="[,( )]"> . </W>

</S>

(2.12)

<S No="3">

<W IX="1" ... IG='[(1,"önem+Noun+A3sg+Pnon+Nom")(2,"Adj+With")(3,"Noun+Zero+A3sg+Pnon+Nom")]' REL="[2,1,(OBJECT)]"> Önemli </W>

<W IX="2" ... IG='[(1,"ol+Verb+Pos")(2,"Adj+PresPart")]' REL="[6,3,(SUBJECT)]"> olan </W>

<W IX="3" ... IG='[(1,"öncül+Noun+A3sg+Pnon+Nom")]' REL="[4,1,(SUBJECT)]"> öncül </W>

<W IX="4" ... IG='[(1,"ile+Conj")]' REL="[5,1,(COORDINATION)]"> ile </W>

<W IX="5" ... IG='[(1,"kanıt+Noun+A3sg+Pnon+Gen")]' REL="[6,1,(SUBJECT)]"> kanıtın </W>

<W IX="6" ... IG='[(1,"çeliş+Verb+Neg")(2,"Noun+Inf+A3sg+P3sg+Nom")(3,"Verb+Zero+Pres+Cop+A3sg")]' REL="[7,1,(SENTENCE)]"> çelişmemesidir </W>

<W IX="7" ... IG='[(1,".+Punc")]' REL="[,( )]"> . </W>

</S>

E: *What is important is that premise and evidence do not contradict.*

2.5.3 Relativisation

So far, we have explained how Turkish dependency treebank is made more self-consistent. The corrections made were mostly corrections of the existing structure, without adding

new information. In this section and the next, will describe the major contribution to the METU-Sabancı Treebank: Addition of long distance dependencies.

Long-distance dependencies have an important role in building semantic interpretation. Unfortunately, the issue of deep structure always finds itself in the "Future work" sections of the work on parsing and other studies. So this means everyone knows capturing "deep" linguistic information is important and everyone also knows it is the hard part. We will show here, some problems caused by not including this information in the Turkish dependency treebank, and suggest a simple solution.

Underhill (1972) identifies two types of relative construction in Turkish. Subject extraction and the rest. The second group covers object extraction, extractions from adjunct phrases (or PP), and from possessive construction at a non-subject location.

All of these types have instances in the Turkish treebank. However, there is no explicit encoding of extraction in the treebank; for instance, the heads of the relative clauses are represented as modifiers. Some of this information could be recovered using heuristics that rely on morphology, like the presence of the PRESPART morpheme in (2.13), and part-of-speech of the word. However, this does not help in identifying the type of extraction as shown in (2.14). (2.14a) is an example of extraction from a locative adjunct and (2.14b) shows extraction from object site.

(2.13) Kitabı okuyan adam uyudu.
book+ACC read+PresPart man slept.
The man who read the book slept

(2.14) a. Uyuduğum araba yandı.
sleep+PastPart car burn+Past.
The car I slept in burned.

b. Okuduğum kitap yandı.
read+PastPart book burn+Past.
The book I read burned.

Case information is lost in extractions, so surface dependencies alone cannot differentiate between these cases. The dependency structures for these extractions are the same (Figure 2.10), causing loss of information about the valency of verbs that would lead to incorrect logical forms for these sentences. We fixed this by adding a T.LOCATIVE.ADJUNCT dependency from *araba (car)* to *uyuduğum (sleep+PastPart)*, indicating that the extraction is from the adjunct. Similarly, a T.OBJECT link was

added from *kitap (book)* to *okuduğum (read+PastPart)* to mark the noun that is extracted from the object position, indirectly providing information that *oku (read)* is transitive (Figure 2.10). Similar labels (T.SUBJECT for subject extraction, T.POSSESSIVE for genitive extraction and other types of ADJUNCT extractions) were added to the treebank manually for approximately 1250 instances in about 800 sentences. These serve as additional information about long-distance dependencies in the form of secondary links from extracted arguments to their logical heads (Çakıcı, 2005).

The tense information is lost in the relativisation process as well. In object extraction one can differentiate between past tense and future tense but not past tense and present. However, this information does not affect the overall grammatical structure as much as case information does. Therefore, we will not discuss this issue any further.

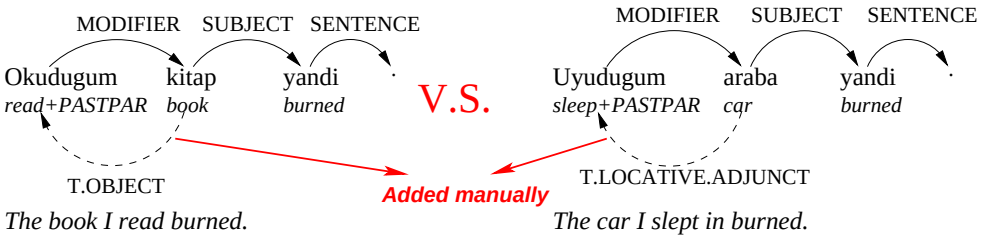
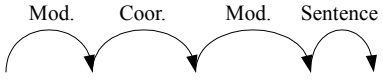


Figure 2.10: The dependencies in (2.14)

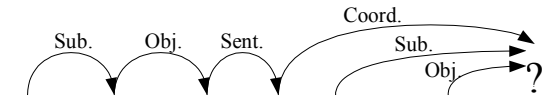
2.5.4 Coordination

There are 1657 sentences with some type of coordination in METU-Sabancı Treebank. The treebank annotation for a typical coordination example is shown in (2.15). Here, the first conjunct is linked to the conjunctive word *ve* with a MODIFIER dependency link and then a COORDINATION link goes from *ve* to the second conjunct and finally, the last conjunct links to the head of the sentence with a MODIFIER link. Despite its simplicity this scheme bears some problems in annotating some structures that are quite common. The example of an argument cluster coordination in (2.16) cannot possibly be represented with the design principle adopted. This causes inconsistencies throughout the treebank among similar sentences as to how they should be annotated.



(2.15) Koşarak ve zıplayarak geldi .
Running and jumping come+Past

He came running and jumping.

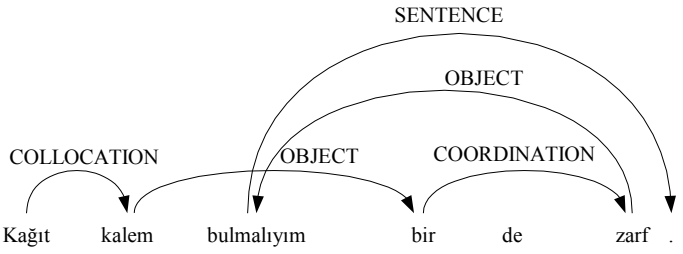


(2.16)

Biri	kuru	diyor	,	biri	kavrulmuş	.
One	*dry*	*says*	,	*one*	*roasted*	

One says dry, another roasted.

It is possible to annotate these types of coordination with gapping, if only one argument coordinates. For instance, in (2.17) objects can be linked to each other before linking to the verb. The approach taken in the treebank was linking both objects to the verb as if the verb is ditransitive. Apart from introcuding non-projective links to the dependency structure, this approach solves the representation problem for gapping in this example. However, if there are two dependents of the missing head such as SOVSO coordination, it is not possible to coordinate two words with one conjunctive, thus, this method does not work for those sentences.



(2.17)

paper	pencil	find-Neces-P1sg	one	too	envelope

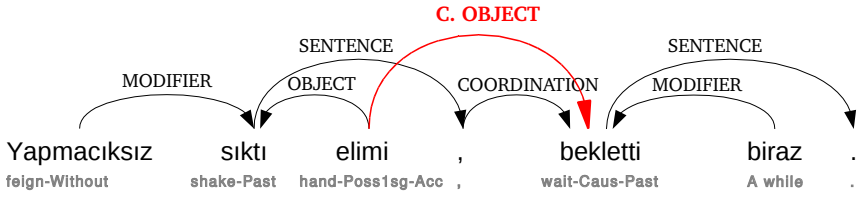
I need to find paper and pencil, and an envelope.

V versus VP coordination

There are about 800 coordinations that involve a SENTENCE label. This is more than half of the total number of sentences with coordination (1517) present in the data. A surface dependency approach does not differentiate between coordination of verbs, verb phrases, and sentences. Both sentences in (2.18) are represented equivalently in the Turkish treebank although (a) is verb coordination, (b) is verb phrase or sentence coordination (with pro-drop). The first sentence is from the treebank and the second sentence is derived from the first by removing the causative morpheme of the verb in (2.18a) *–bekletti (held)–* thus making it intransitive – *bekledi (waited)–* to simplify the example.[4]

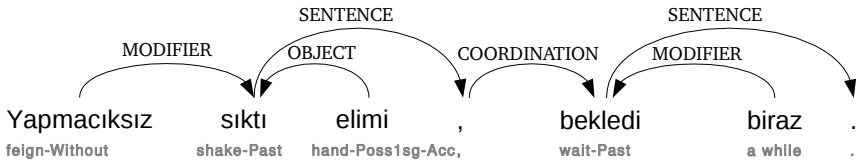
[4]Sentence coordination and verb phrase coordination might be hard to differentiate in Turkish since Turkish is a pro-drop language and usually sentences consist only of verb phrases.

(2.18) a.



((s)he) Shook and held my hand

b.



((s)he) Shook my hand and waited

Our solution for this is similar to the one suggested in the previous section for relativisation. We add secondary dependency links C.OBJECT emanating from *elimi (my hand-ACC)* to *bekletti (held)* in (2.18a) in order to restore the information that *elimi* is the object of *bekletti* as well as *sıktı*. By doing this, we make sure that the missing dependencies are restored and the predicate-argument structure of these verbs will thus be predicted correctly. (2.18a) is thus treated as verb coordination whereas (2.18b) as VP or sentence coordination. Similar links such as C.SUBJECT, C.DATIVE.ADJUNCT etc. are added for other types of dependents that could be shared among conjuncts. We manually added these links to every occurrence of verb coordination with shared arguments in over 800 sentences with SENTENCE coordination.

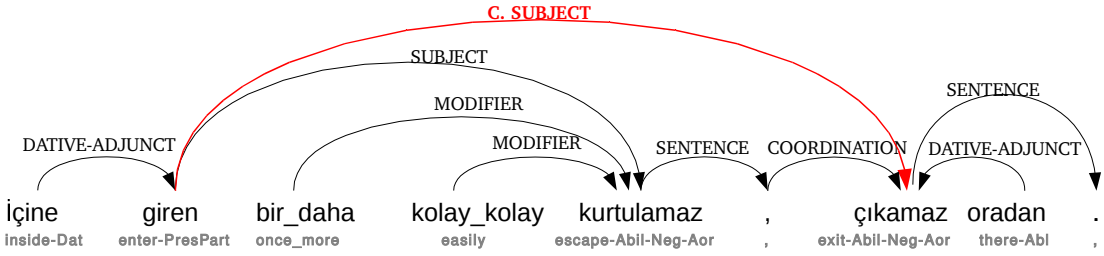


Figure 2.11: Coordination example with secondary edges added

This procedure needs to be done for other types of information for shared arguments for an improved treebank, which we put aside as future work.

Adding these secondary dependencies results in some words being multiple headed. Multiple heads are not supported by the pure-dependency approach adopted and are usually ignored by the dependency parsers most commonly used such as McDonald et al. (2005) and Nivre et al. (2007). This information was not used in any of the

initial parsing experiments to be discussed in Chapter 8. They are only used in guiding the lexicon induction process to obtain correct CCG categories for these structures in order to be able to compare our results with previous work. However, this information is invaluable for training parsers aiming to predict deep dependencies.

2.5.5 Various other changes

Some other changes of small scope when compared to the ones we discussed earlier were made throughout the data as well. These are mostly changes to the dependency labels, indexes or structural changes on the sentence level. Some changes to the morphological annotation of some words were also made and are discussed below.

- Some morphemes especially particles had inconsistent names such as *Demons* versus *DemonsP* in a few sentences. These were corrected together with incorrect annotations caused by typos such as *Postp* versus *PostP*.

- S.MODIFIER was linked to the head verb in some sentences, whereas in the majority of its occurrences it is seen to be linked to the top level (final punctuation). This is regularised by making all S.MODIFIERs linked to the final punctuation when possible.

- Annotation of verbs or nouns that take sentential complements were made consistent. These verbs are "*dedi (said)*", "*diyerek (while saying)*", "Sentence-x *karsılığını (the answer that* sentence-x)". The nouns that take sentential complements such as *karsılığını* were linked with CLASSIFIER label as this was the label that existed in most of the sentences of this sort in the original release of the treebank.

- COORDINATION categories were not compatible in a number of cases. This is possible under some circumstances but in the Turkish treebank, most of the time it was due to wrong annotation. We corrected these problems whenever the incompatibility was at the word level. For IGs we automated the correction process. (Section 6.3.2).

- In copular sentences, copula morpheme is attached to the head noun (or adjective) and this morpheme behaves as the sentence head. When negated, these sentences are headed by the negative particle. Negative particles in Turkish treebank are annotated to depend on the previous verb as NEGATIVE.PARTICLE.

This is not correct when *değil* is the head of the sentence. There are other uses of *değil* where it is used as 1) in “not ... but ...” type coordinations, 2) together with question particle to be used like “isn’t it”. These are not consistently annotated in the original release of the treebank. We isolated and annotated all *değil* cases. The morphology of this word was also erroneous as it was annotated as a verb. However, it behaves exactly like a noun attached with copula verb morphologically. We changed all instances of morphological annotation of this word by the tag *Negp*. This is a new morpheme tag, and it is only used for this purpose.

The sentence in (2.19) is shown annotated in the original treebank in (2.20) and in the corrected version in (2.21). The two versions are quite different, but we would like to draw attention to *degil* labelled as SENTENCE which is the head of the sentence.

(2.19)

Mazim	kalbimde	bir	yara	gibi	değildi	.
past-Poss1sg	heart-Poss1sg-Loc	a	wound	like	not-Past	.

My past was not like a wound in my heart.

(2.20)

<S No=”1”>
<W IX=”1” ... IG=’[(1,”mazi+Adj”)(2,”Noun+Zero+A3sg+P1sg+Nom”)]’ REL=”[2,1,(CLASSIFIER)]”>
Mazim </W>
<W IX=”2” ... IG=’[(1,”kalp+Noun+A3sg+P1sg+Loc”)]’ REL=”[6,1,(LOCATIVE.ADJUNCT)]”> kalbimde
</W>
<W IX=”3” ... IG=’[(1,”bir+Det”)]’ REL=”[4,1,(DETERMINER)]”> bir </W>
<W IX=”4” ... IG=’[(1,”yara+Noun+A3sg+Pnon+Nom”)]’ REL=”[6,1,(OBJECT)]”> yara </W>
<W IX=”5” ... IG=’[(1,”gibi+Postp+PCNom”)]’ REL=”[7,1,(SENTENCE)]”> gibi </W>
<W IX=”6” ... IG=’[(1,”değil+Verb+Past+A3sg”)]’ REL=”[5,1,(NEGATIVE.PARTICLE)]”> değildi
</W>
<W IX=”7” ... IG=’[(1,”...+Punc”)]’ REL=”[,( )]”> ... </W>
</S>

(2.21)

<S No=”1”>
<W IX=”1” ... IG=’[(1,”mazi+Noun+A3sg+P1sg+Nom”)]’ REL=”[6,1,(SUBJECT)]”> Mazim </W>
<W IX=”2” ... IG=’[(1,”kalp+Noun+A3sg+P1sg+Loc”)]’ REL=”[4,1,(MODIFIER)]”> kalbimde </W>
<W IX=”3” ... IG=’[(1,”bir+Det”)]’ REL=”[4,1,(DETERMINER)]”> bir </W>
<W IX=”4” ... IG=’[(1,”yara+Noun+A3sg+Pnon+Nom”)]’ REL=”[5,1,(OBJECT)]”> yara </W>
<W IX=”5” ... IG=’[(1,”gibi+Postp+PCNom”)]’ REL=”[6,1,(MODIFIER)]”> gibi </W>

<W IX="6" ... IG='[(1,"değil+Negp+Past+A3sg")]' REL="[7,1,(SENTENCE)]"> değildi </W>

<W IX="7" IG='[(1,"...+Punc")]' REL="[,( )]"> ... </W>

</S>

- Dependency structures for passive and causative sentences were regularised by checking and correcting the dependents of passive and causative verbs in these sentences. In most sentences of this type object and subject labelled arguments were linked to the IG with passive or the causative morpheme. We ensured this was applied throghout the treebank.

- Annotation of relatively infrequent relation labels such as VOCATIVES, FOCUS.PARTICLE were checked and made more consistent throughout the treebank.

- There were many tokenization errors in the treebank. Question and exclamation marks were ignored as sentence boundaries and several actual sentences separated by these were annotated as a single sentence. We split these sentences whenever possible. This resulted in 57 additions to the treebank. In other cases, sentences were wrongly split in between periods in proper names with only initials and parts of abbreviations. The former is corrected by combining parts of these proper names that were in different sentences, and the latter is corrected by combining the period with the abbreviation and also combining it the rest of the sentence. As a result, the number of sentences changed to 5669 from 5620. Note that these changes do not effect the number of tokens except in abbreviation correction cases where one period is combined with the preceding abbreviation.

- One of the most overloaded words is "Ne–(what)" in the treebank. It can appear as OBJECT, SUBJECT because of the way questions are annotated. It can be a MODIFIER when it is an adverb modifying an adjective, QUESTION.PARTICLE in some questions, and CLASSIFIER some other times. It can also be involved in *either* .. *or* type coordination structures. It can take inflection and be an ADJUNCT as well. This means it takes almost all the labels. Overloading means confusion in annotation. Mistakes indirectly caused by overloading were also corrected throughout the data.

correction type	sentences	changes
secondary links (extraction)	1200	1065
secondary links (coordination)	800	181
morphology	-	1975
wrong dependency	-	4732
tokenisation	49	67

Table 2.5: Summary of major types of corrections

2.6 Summary and Conclusion

Table 2.5 gives a summary of the major corrections made to the treebank. The correction process is still continuing as the lexicons induced from the data are still being tested on the parsers.

This chapter gave an overview of the current state of the Turkish treebank which is the data used for this thesis. Turkish treebank is a relatively new treebank and it is not uncommon in treebank annotation projects that there are a lot of mistakes and design flaws that could not be accounted for in the early stages of treebank design. A lot of the treebanks have second releases (Czech2.0, Penn2.0) that contain corrected versions of the existing or a bigger set of trees. But sometimes, considerable linguistic information needs to be added to account for certain phenomena in natural languages such as argument sharing between conjuncts, long-distance dependencies etc. An example of the second type of improvements is the Tiger treebank (Brants et al., 2004) for German. NEGRA, which was the earlier version of the German treebank lacked long-distance dependencies and information such as argument sharing between coordinate structures. This kind of information is crucial for determining the coordination type (if there are no node labels as in some dependency treebanks) and predicate argument structure. It is not hard to see that a treebank failing to provide information for predicate argument structure is deficient. However, most of the time because of practical issues or for being computationally simple, this issue is ignored.

In this chapter, we explained how a treebank that was initially designed to have only surface dependencies can be modified to include at least some of the crucial information to recover some long-distance dependencies and be able to differentiate certain coordination types which would otherwise be identical. We also demonstrated other changes that did not need any substantial diversion from the principle of the original

release of the treebank. By this, we mean correction of the morphological and syntactic annotation mistakes.

Chapter 3

Morphology

In agglutinative languages the union of words may be compared to mechanical compounds, in inflective languages to chemical compounds. R. Morris.

3.1 Introduction

Before taking its deserved place in linguistic research starting with 90s, morphology was often neglected within the generative framework in linguistics. Followers of generative grammar either considered morphology as part of lexicon which was transparent to syntax, or explained morphology by means of syntactic methodology, i.e. via transformations or X-bar rules that combines morphemes rather than words. (Lieber, 1980; Lieber, 1992; Selkirk, 1982). Chomsky (1970) stated that morphology should be part of lexicon. This has given rise to a tradition of handling morphology with lexical rules. These were simply phonological changes made to stems and affixes without taking syntactic relations into consideration.

Hockett (1954) draws attention to two different approaches to morphology in linguistics. The first is Item and Arrangement which relies on representations of morphemes that are combined (arranged) to form words. The other approach is Item-and-Process in which the rules change the form of the word, by adding, deleting or applying other processes.

A third approach: "Constraint-based morphology" was later introduced by Bird (1990) and was followed by Russell (1993). This adopts the idea that morphology is neither a collection of representational morphemes combining to form words, nor processes that change an original form to obtain the final word. This approach holds that the phonological properties of the words are specified by "constraints" that link

them with semantics and syntax.

Hoeksema and Janda (1988) introduce 4 processes that define morphological operations, some of which can be explained with categorial grammar rules. Some others are handled by introducing new mechanisms that increase the generative power of categorial grammars which are originally context-free.

According to Hoeksema and Janda (1988), Bar-Hillel (1964)'s handling of morphology corresponds to Item-and-Arrangement approach. Dowty (1979), Schmerling (1983), Bach (1983), Bach (1984) adopt an Item-And-Process approach. Hoeksema and Janda state that Item-and-Arrangement is a variant of X-bar theory and corresponds to categorial grammars in terms of expressive power, thus is not powerful enough to explain all morphological phenomena. They emphasise the need to extend categorial grammars to have more than application and concatenation in order to explain some morphological phenomena such as infixation and reduplication. They use Bach's wrapping operations to explain infixation.

The additional operations Hoeksema and Janda (1988) introduce to explain some morphological phenomena boosts the generative power of categorial grammars. Addition is handled with simple application of the CG rules. However, their handling of infixation, which is accomplished by adding wrapping to categorial grammars affects the generative capacity of the categorial grammar. Infixation brings in some context-sensitiveness to the grammar which is acceptable considering the fact that natural languages are not context-free (Shieber, 1985) but in a class that is a subset of context-sensitive languages called mildly context-sensitive languages (Joshi, 1985). Wrapping operation in itself has potentially more power than context-free. Hoeksema and Janda (1988) say that languages with wrapping operations may be in a proper subset of context-sensitive languages. However, it is not clear whether they are equivalent to the proper subset that Combinatory Categorial Grammar, Linear Indexed Grammars belong to. Type changing rules for morphology (lexical rules in some contexts), when used without any restriction on recursivity or on the categories that can be type-changed, boost the generative power of categorial grammars to turing-complete unnecessarily Carpenter (1991; 1992).

3.1.1 Morphology and NLP

Morphology in well known NLP systems is not generally considered a vital component to be focused on. The main reason for this is that English, which is the most studied

language, does not have a complex inflectional component when compared to some other languages. Thus one can easily build an NLP system for English that ignores morphology and still obtain satisfying results. Without proper consideration for the interaction between morphology and other components, mismatches of bracketing occur, or ambiguity that needs to be passed on to syntax and semantics may be ignored which results in deficient systems.

Computational systems of CCG developed so far do not have robust handling of morphology. Hoffman (1995) treats inflected forms in Turkish as separate words and, Hockenmaier (2003a) does the same for English. Bozşahin (2002) proposes a morphemic lexicon with attachment constraints that assumes a morphologically decomposed input.

3.1.1.1 Morphological Analysis

There are different approaches to morphological processing in the literature. The relatively more popular approach for agglutinative languages is finite-state transducer based morphological processing. Finite state transducers take the input word and process it from left to right searching for a valid analysis and they give a string of morphemes that might have created the surface form in question.

Another alternative to morphological parsing is affix-stripping. These strip affixes off the word guided by their lexicon to find a root. When they find a root that is compatible with the morphotactics of the rest of the morphemes they return them as a candidate analysis.

When parsing a word in an agglutinating language like Turkish, these two approaches would work in different ways. An FST parser would start from left to right finding a root from the lexicon, and guided by the transitions and rules of morphotactics scan all possible morphemes that can attach to the root and do the same iteratively for the rest of the input. An affix-stripping approach, on the other hand, would start from the end and strip affixes to reach a compatible root in the end. This approach is not feasible for a language like Turkish due to high ambiguity.

Morphological analysers produce all possible morpheme strings for a surface word. Thus, a morphological processor to be used in practical applications needs a disambiguator module or a disambiguation process afterwards for languages with highly ambiguous morphology like Turkish.

Early work on morphological analysis was a part of English text-to-speech system MITalk. (Sproat, 1992). DECOMP is a suffix-stripping morphological model with

a lexicon that is interactively created from the words in Brown Corpus. AMPLE by Summer Institute of Linguistics uses morphotactics inspired by categorial morphology (Hoeksema, 1984) and it is context-free (Sproat, 1992)[pp.203].

Two-level morphology is introduced by Koskenniemi (1983). Two-levels of "two-level morphology" are the surface and the lexical levels on which the rules are applied in parallel. Two-level morphology is based on three ideas: 1) Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules. 2) The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time. 3) Lexical look-up and morphological analysis are performed in tandem. (Karttunen and Beesley, 2001).

KIMMO is a two-level morphological analyser. It is implemented in C and is very efficient. It produces a combined finite-state transducer that produces the combined output of all the morphological rules implemented. A two-level finite-state transducer works on two tapes one for lexical form and for surface form and accepts if the surface form is compatible with the rules and the lexical form in recognition mode. The current version of KIMMO (PCKIMMO2) uses grammar rules and unification as an interface when building a morphological system , whereas the earlier version uses "alternation classes" (corresponding to states in a finite-state machine).

Generative systems have also been suggested for morphological parsing. One example is Kay's chart parsing model that has two stages (Kay and Kaplan, 1983). The first stage applies the phonological rules and the second phase of dictionary look-up finds the underlying forms for recognition. Rules are used to literally generate the surface forms from an underlying form with the help of a chart parser (Sproat, 1992).

We believe a robust system for agglutinative languages like Turkish cannot be built without proper consideration of morphology for the following reasons:

1 The sparse data problem is unlikely to be overcome regardless of the amount of data provided, because a word may have millions of different inflected forms. This is discussed in Section 3.4.

2 Morphology is often highly interlaced with syntax, semantics and even discourse in languages with strong morphology. Thus, without proper handling of interlacing aspects of syntax, semantics and morphology, analysis of language structures is not possible. We will give some discussion and examples of this issue in Section 3.2.

3 There is a good potential of generalisation with the use of morphology and this

should be facilitated when building NLP systems for a competent but compact outcome. A detailed discussion about this issue is given in Chapter 6.

In this chapter, we discuss some morphological phenomena together with information on how morphology is treated in history. We focus mostly on the interaction of morphology with syntax and semantics. Most of the examples in this chapter are on Turkish. For a detailed description of Turkish morphology and how it is treated see (Lewis, 1967; Underhill, 1972; Hankamer, 1989; Kornfilt, 1997; Oflazer, Göçmen, and Bozşahin, 1994; Pembeci, 1998; Öztaner, 1996). We will also discuss the history of computational morphology to some extent, and give a review of morphological analysers for Turkish. Finally, more discussion on why morphological processing is necessary for NLP applications will be given.

3.2 Morphology and Syntax

Spencer (1991) categorises languages as: isolating, agglutinating, inflectional and polysynthetic. Isolating languages are those that do not have morphology whatsoever such as Chinese and Vietnamese. Inflectional languages, have combined morphemes of different features which are difficult to map to surface forms. Polysynthetic languages make use of combination of words to form a full sentence. The fourth class is Agglutinative defined below. Turkish belongs to the class of agglutinative languages. The definition and the Swahili example (3.1) for agglutination given below are taken from Trask (1993). The Turkish example in (3.2) shows the agglutinative aspect of Turkish morphology.

> AGGLUTINATION A type of morphological structure in which words can be readily divided into a linear sequence of distinct morphemes, each of which typically has a fairly consistent shape and a single consistent meaning or function.

(3.1)

a-	-li-	-ku-	-on-	-a
he	-past	you	see	indicative

he saw you

(3.2)

öl	-dür	-e	-me	-z	sin
die	-Caus	-Abil	-Neg	-Pres	P2sg

you can/may not kill

(3.3)

yom	-ase	-rare	-na	-i
read	-Caus	-Abil	-Neg	-Pres

X cannot make Y read.

In (3.2), we see a typical one-word-sentence in Turkish where the verb *öl – die* is decorated with a decomposable string of morphemes to mark voice (-Caus), modality (-Abil), polarity (-Neg), tense (-Pres) and person (-P2sg). This is not particularly a longer-than-average word for Turkish, and it constitutes a sentence on its own as the Swahili example in (3.1) does.

There are many agglutinative languages e.g. Hungarian, Japanese and Turkish. The Japanese counterpart of (3.2) is (3.3) although the meaning is slightly different in this example.[1] Agglutination could be seen as an advantage because it makes generalisation easier. It could be argued that segmentable systems are much easier to model than inflectional systems.

Generalising a system to process an agglutinative language with heavy morphology-syntax interface may be more complex than isolating languages with no morphology or languages with simpler morphology. On the other hand, agglutinative languages have advantages over inflecting languages where the morpheme boundaries are not as clear, and morphological information cannot be represented in a string of morphemes. This is because segmentation of morphemes for agglutinative languages is easier as opposed to segmentation in inflectional languages. However, this task has its challenges. Morphemes may have grammatical functions such as case and agreement, changes of predicate-argument structure and so on. This interaction between morphology, syntax and semantics, sometimes, leads to mismatches of bracketing in different levels of representation in some theories, which are usually called bracketing paradoxes in the literature. This issue will be discussed in more detail in Section 3.2.1.

3.2.1 Bracketing Paradoxes

Bracketing paradoxes (or bracketing mismatches) are conflicts between certain levels of representation in language. We find two types of bracketing paradoxes mostly studied on in the literature. In the first type, morphological grouping is inconsistent with the constituent structure suggested by syntax. A typical example is : He'll do it. *he'll* is a morphological unit but *'ll do it* is a constituent (Trask, 1993). The second type is the mismatch of phonology and interpretation.

[1] Thanks to Akira Otani for this example.

Pesetsky (1979) observed that the morphophonemic restrictions while attaching a morpheme may be different from the semantic bracketing as the following example from English shows.

(3.4) a. [*un*-[*happy*-*er*]]

b. [[*un*-*happy*]-*er*]

In English, an adjective may take comparative form only if it is monosyllabic, trochaic or disyllabic. For example *happy* or *easy*. So *happy* can take *-er* but *unhappy* cannot, normally. The bracketing should be as in (3.4a) according to the morphophenemic restrictions. Siegel(1974)'s Level ordering hypothesis says that affixation takes place in two linearly ordered blocks, which are separated by the word stress rules. Thus, un- being a "class II" affix is attached after -er in ordering. Thus the phonological constraint is met. However, the actual meaning of the word is *more unhappy* rather than *not happier*. This means the semantic interpretation has the bracketing in (3.4b).[2] Bracketing paradoxes appear in many languages and have been studied by Fukushima (1999), Bozşahin (2002), Müller (2003) and so on.

The type of bracketing paradoxes in Turkish that we discuss here mostly fall in the first category mentioned above. The attachment and phonological characteristics of a bound morpheme is constrained by the word it is attached to whereas it has semantic scope over the whole phrase. Examples (3.5) and (3.6) show an ambiguous case where bracketing choices make a difference to the semantic interpretation. These examples show that morphemes may or may not have phrasal scope even though they are identified with the word they are attached to. In (3.5) the scope of *-lı – with –* is [long leaves] whereas in (3.6), it is only [leaves]. Thus they yield different interpretations depending on the bracketing. There is no way to identify this ambiguity and pass it on to syntax or semantics level without treating the bound morpheme *-lH* as a separate item that defines its own semantic scope.

(3.5)

[Uzun yaprak]	-lı	ağaç
Long leaf	-With	tree

The tree with long leaves

(3.6)

[Uzun]	[yaprak-lı]	ağaç
Long	leaf-With	tree

The long tree with leaves

[2] Some authors such as Stump (1991) and Sproat (1992) claim that this is not a mismatch.

3.2.1.1 Bracketing Paradoxes and Coordination

Examples in 3.7 show that bracketing paradoxes appear in coordination as well. The noun phrase in 3.7a is a typical coordination example where two noun phrases *tuz -salt* and *yağlı yiyecekler - fatty food* coordinate. However, because suspended affixation is possible in Turkish, the bracketing in (3.7b) which is a noun coordination is also possible. The semantic outcome of this bracketing is $food(x) \wedge (with(salt,x) \wedge with(fat,x))$.

(3.7) a.

tuz	ve	[yağ	-lı]	yiyecekler
salt	and	fat	-With	food

salt and food with fat

b.

[tuz	ve	yağ]	-lı	yiyecekler
salt	and	fat	-With	food

food with salt and fat

(3.8a) and (3.8b) display the bracketing ambiguity between the syntactic and semantic levels of bracketing. We will not discuss this types of ambiguity in this chapter. This example is given only for a better understanding of the issue at hand. This example shows that the English version has a bracketing ambiguity, as well.

(3.8) a.

[genç	kızlar]	ve	erkekler
young	girl-Plu	and	man-Plu

b.

[genç	[kızlar	ve	erkekler]]
young	girl-Plu	and	man-Plu

young girls and men

3.2.2 Morphemes with wider scope

3.2.2.1 Subordination

In example (3.9) it is clear that the semantic scope of *-nH* is the whole constituent of *adamın kitabı bana vermesi*. It could also be argued that agreement morpheme *-sH* has the phrasal scope shown in (3.9). However, the phonological characteristics of both morphemes are determined by the word they attached to, thus *-nH* is instantiated as *-ni*.

(3.9)

Ahmet	[adam-ın	[kitab-ı	bana	ver]-me-si]	-ni	iste-di.
Ahmet	man-Gen	book-Acc	me	give-Inf-Agr	-Acc	want-Past

Ahmet wanted the man to give the book to me.

3.2.2.2 Adverbials and Gerunds

The derivational morpheme *-madan* attaches to verbs to make them adverbs. This is another example of phrasal scope derivational morpheme having scope over the whole phrase *yemeğin hepsi bit*, not just the verb. Taking into account the wider scope restrictions of these morphemes is vital for the correct semantic interpretation.

(3.10)

[[Yemeğin	hepsi	bit]-meden]	tatlı	yiyemezsin.
food-Gen	all-Poss3sg	finish-WHDS	desert	eat-Abil-Neg-Aor-P2sg

-WHDS = -WithoutHavingDoneSo

You may not have desert before finishing your food.

3.2.3 Suspended Affixation

Suspended affixation is common in Turkish. It can be explained as inflectional morphemes having scope over the coordinating clause. Inflectional and some derivational morphemes[3] can have scope over the whole coordinating clause. (3.11) is an example of a group of nominal inflectional morphemes that have scope over the phrase *kapı ve pencere – door and window*. The semantic structure for this example is $(poss(plu(door \wedge window)))$ whereas the attachment is as seen in the example.

In verbal morphology a non-final conjunct can be underspecified for modality and tense (second tense) and person as in (3.12). Morphemes with wider scope are usually attached after tense in verbal morphology (second tense or modality) whereas in nominal morphology there are no restrictions. (3.13) shows an example where the past tense morpheme has wider-than-word scope. This morpheme is attached after the main tense morpheme which is *-mHş –Narr* has a usage that is similar to the past perfect in English. It has scope over the whole VP coordination consisting of two conjuncts. (3.14) shows the same phenomena with verb coordination as opposed to VP coordination in the previous example. However, (3.15) that has the first tense morpheme right after coordination is implausible.

[3] Although the suffixes in question may not be called derivational morphemes by some linguists, we will stick with the treebank terminology at this point

(3.11)
ev-in [kapı ve pencere] -ler -i
house-Gen door and window -Plu -Poss3sg
[doors and windows] of the house

(3.12)
[gider ve uyur] -sam
go-Pres and sleep-Pres -Cond-P1sg
if I [go and sleep]

(3.13)
şimdi [yapraklar dökülmüş ve havalar soğumuş]-tu.
Now [leaves fall-Narr and air cool-Narr]-Past
Now, the leaves had fallen and weather had cooled.

(3.14)
Soba [sönmüş ve soğumuş]-tu.
Stove [die-down-Narr and cool-Narr]-Past .
The stove had died down and cooled.

(3.15)
*babam [kay ve düş]-tü.
father-Poss1sg [slip and fall]-Past.
Intended reading: My father slipped and fell.

(3.16)
Kapıdaki kız [alımlı ve güzel] -di.
door-Loc-Rel girl attractive and pretty -Past
the girl at the door was [attractive and pretty]

Productive derivational morphemes can also take part in such constructions. *-lı – with –* is considered as a derivational morpheme that makes an adjective out of a noun in Turkish. The same holds for *-(y)dH* copula morpheme in (3.16) which is also considered as a derivational morpheme to make verbs out of nouns (or noun phrases). This type of suspended affixation is the most frequent type.

Suspended affixation is an interesting phenomenon which is not exclusive to Turkish. Examples of this are seen in Hindi, Japanese and so on. A detailed analysis of suspended affixation in Turkish is given in Kabak (2007) for interested readers.

3.2.4 -ki relativisation

An interesting type of relativisation is *-ki* relativisation in Turkish. The head can drop in constructions which leads to the specifier taking the inflectional properties of the

dropped noun such as number, case, and so on. *-ki* is regarded as an adjective derived from a noun in Turkish treebank. And the phenomenon above is represented by deriving a noun from the adjective which itself was derived from a noun as in (3.17).

(3.17) IG='[(1,"üst+Noun+A3sg+P1sg+Loc")(2,"Adj+Rel")(3,"Noun+Zero+A3pl+Pnon+Nom")]'
Üstümdekiler – the ones on/above me

The morphological structure of the relativised noun is the same in both sentences in (3.18) except case marking, although the first one is a derived adjective while the second one behaves as an inflected noun. This may either be a case of "headless relativisation" which involves head-deletion according to some theories or it can simply be explained as an adjective that goes through nominal morphology which is common in Turkish. The latter means that the noun is first turned into an adjective by *-ki* relativiser and the resulting adjective goes through nominal inflection. This type of overloading is often explained by zero morphemes that change types in METU Sabancı Treebank. (3.19) is an example of a headless relative clause.

(3.18) a.
masadaki geline baktım.
table-Loc-Rel bride-Dat look-Past-P1sg
I looked at the bride at the table

b.
masadakine baktım.
table-Loc-Rel-Dat look-Past-P1sg
I looked at the one at the table

(3.19)
Bu öngörülerin doğruluğunu sınayacaklar yine deney
This hypothesis-Plu-Gen truth-Agr-Acc test-FutPart-Plu again experiment

ve gözlemlerdir
and observation-Plu-Cop

FutPart = relativiser

What will test the truth of these hypotheses is, again, experiments and observations.

3.2.5 Morphology and Argument Structure

The most significant aspect of morphology/argument structure interaction is observed in voice changing structures. In Turkish, voice changing affects not only verbal morphology but also case suffixes on the arguments. Other types of voice changing morphemes are passives, reflexives and reciprocals.

3.2.5.1 Causatives

Causative morphemes are *-DHr* and *-t*. *-DHr* is used for verbs ending with a consonant and *-t* is used for verbs ending with a vowel or when there is double causativisation or if a passive verb is being made into a causative verb. There are exceptions to these phonological rules such as if a verb ends with a -t sometimes the causative form is achieved by -Hr instead of -DHr. Interested reader should refer to Underhill (1986).

Causative forms of passive verbs and vice versa are quite common in Turkish as well as double passivised or double causativised verbs. More than two occurrences of passive or causative morphemes attached to one verb is not common.

The causative morpheme is *-DHr*, which has a number of phonologically and morphologically conditioned allomorphs. The allomorphs are:

- -DHr is the most common allomorph, which is attached to the verbs ending in consonants.
- -t is used after vowels or words ending with *-(H)r* or *-Hl* e.g. *uyu-t*, *delir-t*, *karar-t*.
- -Ht is used for a small number of exceptions that end with certain sounds such as *-rk*. *korkut*.
- -Hr is another allomorph that is rare. *yat-ır*, *bat-ır*, *bit-ir*
- -Ar is used for a limited number of one syllable verbs such as *çıkar*.

The causative morpheme has the effect of adding an argument to the basic argument structure of the predicate. Thus, causativisation of a intransitive verb yields a transitive structure as in (3.21). The causative morpheme brings the subject of an intransitive verb to object position. Causativisation of a transitive verb yields a ditransitive structure as in (3.20). The subject of *oku – read* becomes dative marked oblique.

Causativisation, despite resembling a morphological operation affected purely by morphophonemic properties of the verb it is attached to, affects the argument structure of the verb it is attached to through syntax by means of case marking.

(3.20) a.
Ahmet kitab-ı oku-du.
Ahmet book-Acc read-Past
Ahmet read the book.

b.
Ben Ahmet-’e kitab-ı oku-t-tu-m.
I Ahmet-Dat book-Acc read-Caus-Past-P1sg

I made Ahmet read the book.

(3.21) a.
Yangın sön-dü.
fire-Nom extinguish-Past-P3sg
The fire extinguished.

b.
Ben yangın-ı sön-dür-dü-m.
I fire-Acc extinguish-Caus-Past-P1sg
I extinguished the fire.

3.2.5.2 Passives

Passive and causative constructions in Turkish are achieved through morphology. Passive morphemes are *-Hl* and *-(H)n*. *-Hl* is for verbs that end with a consonant and *-(H)n* is for verbs that end with a vowel or the sequence *-Hl*. They are also used to make double passives one after the other one. They directly change the valency of the verb they are attached to without the need for an auxiliary. The passive morpheme is assigned a category that takes a *TV* or *IV* category and turns it into a *IV* or *S* category respectively.

Voice is altered through bound morphemes in Turkish. Treatment of this kind of type changes can be theory-specific. But in this research we will assume that passive morpheme is a type changing morpheme that changes a transitive verb into an intransitive one. *-(H)n* and *-Hl* are the morphemes for passive construction. *-(H)n* is used if the word ends with *l* and *-Hl* is used otherwise. The subject of the passive sentence is the direct object of the underlying "deep structure", which is why the accusative case marker in (3.22) disappears in (3.23). Impersonal passives are also common. These passivise intransitive verbs and they are formed by the same morphological operations. In terms of effects on argument structure passive morphemes do exactly the opposite of causativisation morphemes, as expected.

(3.22)
Kahya yüzüğü bul-du.
Butler ring-Acc read-Past
The butler found the ring

(3.23)
Yüzük bul-un-du.
ring find-Pass-Past
The ring was found.

3.3 Morphological Processors for Turkish

As previous discussion indicates, morphology plays a vital role in Turkish natural language processing. There are many morphological analysers designed for Turkish. (Güngör, 1995; Öztaner, 1996; Pembeci, 1998; Çakıcı, 2002; Hankamer, 1989; Oflazer, 1994; Köksal, 1973)

The first finite-state morphological analyser that is designed for Turkish is *keçi* (Hankamer, 1986). *keçi* incorporates ordered phonological rules and finite-state morphotactics. The phonological rules in *keçi* are applied one after the other until a perfectly matching surface form is found, before moving on the to finding the next morpheme. This is different from the two-level approach where both lexical and surface restrictions are checked in parallel and all together given a phonological context. Sproat calls this procedure "generate-and-test" incorporating *ordered* phonological rules following Generative Phonology by Chomsky and Halle (1968) (Sproat, 1992)[pp.190].

keçi has finite-state morphotactics. It does not incorporate any context-free elements. Each affix determines the next state of the automaton and the optionality is obtained by ε-transitions.

Oflazer (1994) presents a wide-coverage morphological analyser with 23,000 lexical entities. He uses PC-KIMMO (Antworth, 1990) which is an earlier version of PC-KIMMO-2 that does not have word-grammar rules. Oflazer implements the morphotactic rules of Turkish that are explained in Oflazer, Göçmen, and Bozşahin (1994) by alternation classes of PC-KIMMO that correspond to states in an FST. The morphophonemic rule component uses 22 phonetic rules to explain the phonetic phenomena in Turkish. This system is the first wide-coverage morphological analyser of Turkish that handles special cases and exceptions. This morphological analyser was also used for morphological analysis in METU-Sabancı Treebank (Atalay, Oflazer, and Say, 2003; Oflazer et al., 2003).

3.3.1 Disambiguators

When the level of morphological ambiguity is considered in Turkish, morphological disambiguators that choose between different analyses (very similar to parse selection) are vital for practical NLP systems with a morphological processing component. Some morphological ambiguity should be passed on to the parsing level but this is not a primary concern. There are a few disambiguators for Turkish. The disambiguators defined in Oflazer and Tür (1996) and Oflazer and Tür (1997) are two of the early

disambiguators that use hybrid models of hand crafted rules and voting constraints modelling the context of the word to be tagged. A purely statistical model is created by Hakkani-Tür, Oflazer, and Tür (2002).

Yüret and Türe (2006) use decision trees and train a separate model for each of the morphological features the morphological analyser creates. These features are the 126 morphological tags that Oflazer (1994)'s morphological analyser creates. The system decides on the resulting morphological analysis by independently voting on each of the parses to find the most likely one. They report a tagging result of 96% when a separate classifier is trained for each tag and 91% when decision lists are used to tag the data without the help of a morphological analyser. This system is trained on whole tags (a string of morphological features). The training data was semi-automatically disambiguated corpus of 1 million words and test data is a manually created set of 958 instances.

3.4 Importance of morphology for NLP applications

Research in this fields shows that morphological processing for languages like Turkish is inevitable (Hankamer, 1989; Bozşahin, 2002; Jurafsky and Martin, 2000). Şehitoğlu and Bozşahin (1999) shows that generation of all inflected forms for 40 Turkish root forms results in about 2800 entries even with limited inflectional morphology. Inflectional suffixes can create around 40,000 word forms for verbal paradigm without recursive processes like causativisation.[4] (Jurafsky and Martin, 2000). Nominal paradigm with *-ki* relativisation can create millions of forms.[5] According to the FLH, each complex word has its own, separate entry in the mental lexicon. Two variants of the FLH have dominated psycholinguistic accounts in recent years. The FLH-A version holds that complex words have their own lexical entries which include a representation of their morphological structure. To provide an example, the word forgetful has a lexical entry of its own, but the entry contains a morphological analysis of the word: (for-(get)-ful). The B version, in turn, presupposes that every word has its own entry and that all entries for morphologically complex words are linked to a basic entry for the uninflected or root word. Thus, forgetful has its own lexical entry in the mental dictionary, which, along with the entries for unforgettable, forgettable, forgetting, etc., is

[4] Adding these may create infinite number of forms.

[5] This is because relativised form of the noun can be nominalised and get inflectional morphemes again as shown in Section 3.2.4.

linked to the basic entry get. Under this version of the FLH, the basic entry is called the nucleus and the remaining entries clustered around the nucleus are its satellites.

Hankamer (1989) discusses extensively that Full Listing Hypothesis (FLH) (Butterworth, 1983) is untenable with evidence from Turkish morphology. According to FLH each word has its own entry in the mental lexicon. Some variations hold that complex words have their own entries in additon to their morphological analyses. Hankamer gives figures on how much memory is needed to store such information together with the numbers of exhaustive generation of inflectional forms of nouns and verbs in Turkish. He shows that even without recursion (which exists in Turkish morphology) the numbers go up as much as 1.8 million per verb root and 9.2 million per noun root. With recursion these numbers go up to 26.7 million and 216.6 million respectively. This means for a lexicon with 20000 noun roots and 10000 verb roots a lexicon of 200 billion entries would be required. This is infeasible even for human processing and Hankamer uses this to suggest that the humans could not possibly be storing all the inflected (and derived) word forms as lexical entities in the memory. He calculates together with the figures from Sagan (1985) that it would only be possible to store only 125 billion morphologically complex words.

If even human mind does not have the ability to store all inflectional forms in the memory, it would be irrational and unnecessarily costly to design an NLP system with hundreds of billions of lexical entries to fetch from.[6] We observe in real data that the number of instances of the verb *git – go* 177 in the Turkish treebank in Figure 3.1. A language model without morphological analysis will treat each distinct inflected form of the verb as a new instance. This might easily cause sparse data problem.

.....	frequency	cat/word
total occurrences	177	N/A
distinct cat-word pairs	128	1.38
distinct category types	14	12.64

Figure 3.1: The figures for the verb *git (go)*.

NLP systems for Turkish incorporate morphological complexities. This is either by employing a morphological analyser, or using sub-word units in defining the relationships in sentences. (Çakıcı and Baldridge, 2006; Çetinoğlu and Oflazer, 2006;

[6]This is discussed in Section 6.2.2.1 in detail.

Eryiğit and Oflazer, 2006). Eryiğit, Nivre, and Oflazer (2006) and Eryiğit, Nivre, and Oflazer (2008) show that morphological features can be very useful in improving the performance of a dependency parse. The most prominent features are nominal case and relativisation morphemes. It is also established that use of morphology helps in machine translation systems. Oflazer and Durgar El-Kahlout (2007) show that using sub-word level entities in machine translation gives an improvement of 24% BLEU scores compared to a baseline for English to Turkish translation when morphotactical knowledge is provided for generation. Dyer (2007) also shows that the other direction in translation i.e from morphologically complex language to English also benefits from the use of morphological knowledge. There are still unresolved problems such as the ones mentioned in Çetinoğlu and Oflazer (2006) such as phrasal scope of inflectional morphemes, bracketing mismatches of coordinated structures, such as suspended affixation and so on.

Chapter 4

Combinatory Categorial Grammars

Combinatory Categorial Grammar (Ades and Steedman, 1982; Steedman, 2000) is an extension to the classical Categorial Grammar (AB) of Ajdukiewicz (1935) and Bar-Hillel (1953). AB, and extensions to it, are lexicalist theories. Surface structure is constructed with the help of lexical categories of the word in categorial grammars. Categories are either atomic (*NP*) or complex like (*(S\NP)/NP*). The category of a word specifies its predicate-argument relations and directionality of its arguments in categorial grammars. In this respect, a word with syntactic category *(S\NP)/NP* expects two noun phrases: one to the right of the word and one to the left and becomes a sentence when combined with them.[1] A lexical item in a categorial grammar can be represented as the triplet:

$\phi := \sigma : \lambda$ where ϕ is the phonological form, σ is its syntactic type, and λ its semantic type. Some examples are shown in (4.1).[2]

(4.1) a. $book := N : \mathsf{book}$

b. $oku := (S\backslash NP_{\text{nom}})\backslash NP_{\text{acc}} : \lambda x.\lambda y.\mathsf{read}\,xy$

In AB, there are two kinds of application rules defined by the direction of application. These are shown in (4.2).

(4.2) Forward Application ($>$): $X/Y : f \quad Y : a \quad \Rightarrow \quad X : fa$

Backward Application ($<$): $Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : fa$

[1] We assume the notation that the result category is always on the left and argument category is always on the right with respect to the slash.

[2] The example in (4.1b) is a Turkish transitive verb.

An example AB derivation is shown in (4.3). The directionality and lexical type requirements for arguments determine and guide the syntactic structure formation while the semantic interpretation is composed in parallel. Categorial grammars are known for this property of parallel construction of syntactic and semantic structure.[3]

(4.3)
$$\dfrac{\dfrac{\underline{\text{John}}\quad\underline{\text{likes}}\quad\underline{\text{Marry.}}}{NP\colon john' \quad (S\backslash NP)/NP\colon \lambda x.\lambda y.likes' xy \quad NP\colon marry'}\,(>)}{S\backslash NP\colon \lambda y.likes' marry' y}\,(<)$$
$$S\colon likes' marry' john'$$

4.1 Combinatory Rules and Principles

AB is weakly equivalent to context-free (Bar-Hillel, Gaifman, and Shamir, 1964). Since natural languages require slightly more generative power than context-free, additional abilities that increase expressivity are required to explain linguistic phenomena that CFGs cannot.

In addition to functional application rules, CCG has the ability to combine functions with combinatory operators such as functional composition. Composition (**B**), type raising (**T**) that are adopted from Curry and Feys' (1958) combinatory logic. Combinatory operations, while preserving the transparency of syntax and semantics during derivations, increase the expressiveness to that of Linear Context Free Rewriting Systems, a multilevel proper subset of mildly context-sensitive grammars. CCG is in the first known trans-context free level together with TAG and others.

CCG is designed to predict long-distance dependencies, as well as surface ones automatically while combining lexical categories representative of predicate-argument relations and directionality. CCG is a lexicalist theory. This means all language-specific characteristics are specified in the lexicon by type-logical modalities on the slashes i.e. slash types.[4] The rules only specify how these categories can be combined on a higher level. The language specific properties such as word order or unbounded dependencies are implicitly coded in the lexicon.

Composition (4.4) and type-raising (4.5) are used to handle syntactic coordination and extraction in languages by providing a means to construct constituents that are not accepted as constituents in other theories. Hoffman (1995) claims that flexible

[3] The interested reader is referred to Steedman (2000) for a review on semantic interpretation construction during parsing, and to Bos et al. (2004) for an example of wide-coverage semantic parsing.

[4] See Steedman and Baldridge (to appear) for an extensive discussion. We will omit slash types here for ease of reading, and because they are not particularly important in the focus of this dissertation.

constituency is crucial for handling the scrambling of arguments in languages like Turkish.

(4.4) Forward Composition (>**B**): $X/Y: f \quad Y/Z: g \quad \Rightarrow \quad X/Z: \lambda x.f(gx)$

Backward Composition (<**B**): $Y\backslash Z: g \quad X\backslash Y: f \quad \Rightarrow \quad X\backslash Z: \lambda x.f(gx)$

(4.5) Forward Type Raising (>**T**): $X: a \quad \Rightarrow \quad T/(T\backslash X): \lambda f.fa$

Backward Type Raising (<**T**): $X: a \quad \Rightarrow \quad T\backslash(T/X): \lambda f.fa$

Steedman also defines a generalised version of composition shown in (4.6) where n is bounded by the degree of valency in the lexicon in order to retain the mildly-context sensitive power of CCG.

(4.6) Generalised Forward Composition (> $\mathbf{B}^n$):

$X/Y: f \quad (Y/Z)/\$: ...\lambda z.gz... \quad \Rightarrow \quad X/Z/\$: ...\lambda z.f(gz...)$

Where \$ is formally defined as the following in Steedman (2000)[pp.42].

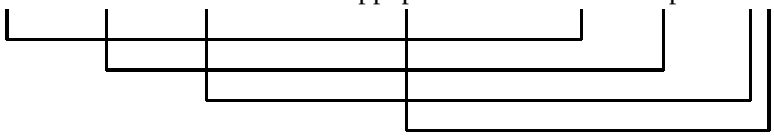
(4.7) *The \$ convention*

For a category $\alpha, \{\alpha\$\}$, (respectively,$\{\alpha/\$\}, \{\alpha, \backslash\$\}$) denotes the set containing α and all functions (respectively, leftward functions, rightward functions) into a category in $\{\alpha\$\}$ (respectively, $\{\alpha/\$\}, \{\alpha\backslash\$\}$

CCG handles phenomena such as "non-constituent" coordination seamlessly (without the use of movement or deletion) because it has a more flexible account of constituency. This property also ensures that the crossing dependencies in the famous hippopotamus example in Dutch (4.8) are correctly predicted with the derivation in (4.9).

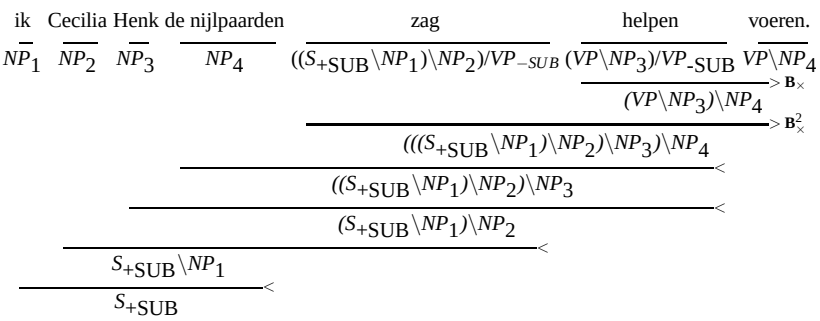
(4.8)

omdat	ik	Cecilia	Henk	de	nijlpaarden	zag	helpen	voeren.
because	I	Cecilia	Henk	the	hippopotamuses	saw	help	feed



'...because I saw Cecilia help Henk feed the hippopotamuses.'

(4.9)



In (4.9) which is taken from Steedman (2000), the existence of the forward crossing composition rule in CCG (4.10), which is excluded for most English categories, makes it possible to cluster the verbs before they combine with their subjects, thus making it possible to handle crossing dependencies of this kind.

(4.10) Forward Crossed Composition ($> \mathbf{B}_\times$):

$$X/Y\colon f \quad Y\backslash Z\colon g \quad \Rightarrow \quad X\backslash Z\colon \lambda x.f(gx)$$

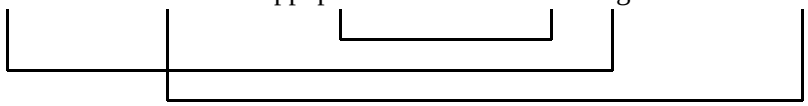
Backward Crossed Composition ($< \mathbf{B}_\times$):

$$Y/Z\colon g \quad X\backslash Y\colon f \quad \Rightarrow \quad X/Z\colon \lambda x.f(gx)$$

Languages that allow scrambling give rise to frequent crossed dependencies. The following Turkish example shows crossing dependencies that are caused by arguments that are scrambled out of their "canonical" order. In SOV order which is often referred to as the canonical word order in Turkish, *Ahmet'in* and *Mehmet* are swapped resulting in a nesting structure.

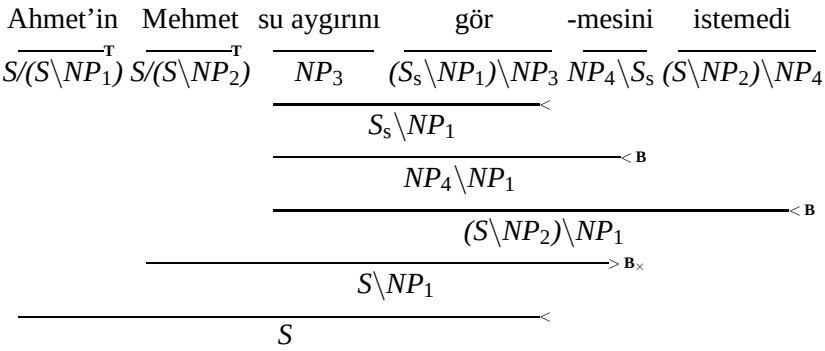
(4.11)

Ahmet'in	Mehmet	su aygırını	görmesini	istemedi.
Ahmet-Gen	Mehmet	hippopotamus-Acc	see-Inf-Agr-Acc	want-Neg-Past



Mehmet didn't want Ahmet to see the hippopotamus.

(4.12)



There are other operations such as substitution (4.13) in CCG that are adopted from combinatory logic. Backward crossed substitution (4.14), for instance, is used to handle parasitic gaps as in (4.15) (Steedman, 2000).

(4.13) Forward Substitution ($>\mathbf{S}$): $(X/Y)/Z\colon f \quad Y/Z\colon g \quad \Rightarrow \quad X/Z\colon \lambda x.fx(gx)$

Backward Substitution ($<\mathbf{S}$): $Y\backslash Z\colon g \quad (X\backslash Y)\backslash Z\colon f \quad \Rightarrow \quad X\backslash Z\colon \lambda x.fx(gx)$

(4.14) Backward Crossed Substitution ($<\mathbf{S}_\times$):

$$Y/Z: g \quad (X\backslash Y)/Z: f \quad \Rightarrow \quad X/Z: \lambda x.fx(gx)$$

(4.15) (articles) which I will file without reading

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(N\backslash N)/(S/NP)}\quad \dfrac{}{S/VP}\quad \dfrac{}{VP/NP}\quad \dfrac{}{(VP\backslash VP)/VP_{ing}}}{(VP\backslash VP)/NP}\,{>}\mathbf{B}}{VP/NP}\,{<}\mathbf{S}_\times}{S/NP}\,{>}\mathbf{B}}{N\backslash N}\,{>}\mathbf{B}\quad \dfrac{}{VP_{ing}/NP}}{}$$

Transparent composition of syntactic structures and semantic interpretations, and flexible constituency make CCG a preferred formalism for long-range dependencies and non-constituent coordination in many languages e.g. English, Turkish, Japanese, Irish, Dutch, Tagalog (Steedman, 2000; Baldridge, 2002)

In addition to parameters limiting the degree of recursivity (in $\mathbf{B}^n$), or the range and domain of the rules as in Type raising, there are principles that limit the unbounded nature of some rules, or the properties of the lexicon. This way the generative power of the grammar is kept under control by, for instance restricting rules that change directionality and allowing uncontrolled permutation which could cause the grammar to be Turing-complete. Some of these principles are (Steedman, 2000):

> **Principle of Head Categorial Uniqueness**: "A single nondisjunctive lexical category 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". This means that the dependencies of both the extracted and non-extracted arguments of the predicate should be specified by the same lexical category. The category of the head cannot change depending on which of these situations it is in. A lexicon that is as compact as possible is preferred.
>
> **Principle of Lexical Head Government**: "Both bounded and unbounded syntactic dependencies are specified by the lexical syntactic type of their head."
>
> **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 category."

Directional Inheritance Principle "If the category that results from the application of a combinatory rule is a function category, then the slash defining directionality for a given argument in that category will be the same as the one(s) defining directionality for the corresponding argument(s) in the input functions."

Principle of Consistency "All syntactic combinatory rules must be consistent with the directionality of the principal function." This means, for instance, backward application cannot, by definition, apply to X/Y. Principle of Consistency prevents copmbinatory rules that override lexical directionality, such as $YX/Y => X$.

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

The principles of Inheritance, Consistency and Adjacency restrict the set of possible rules in order to control the expressive power of the grammar, whereas the principles of Uniqueness, Lexical Head Government and Transparency govern the lexicon. However, these principles do not eliminate the fact that some rules are not needed in some languages simply because of the inherent characterics of the language. For instance, English employs backward crossing composition, but not the forward counterpart. On the other hand, a language with a freer word order could be expected to have both.

Baldridge (2002) and Baldridge and Kruijff (2003) move this choice to the lexicon by the use of modalities defined over slashes. These modalities put limitations on the applicability of rules to categories in a similar way to type logical modalities (Morrill, 1994). For example, many function categories in English bear a slash type that makes them incompatible with crossed composition rules. The result is a system where the rule set is also determined by the combination properties of the categories in the lexicon giving rise to a higher degree of lexicalism. An example of this is the conjunction category shown in (4.16). The $\star$ feature on the slashes limits the combination of this category only by means of application and nothing else. This gives rise to the derivation set in (4.17). An overview and examples for a wide variety of linguistic phenomena are given in Steedman and Baldridge (to appear).

(4.16) and $:= (X\backslash_\star X)/_\star X$

(4.17)

Marcel	conjectured	and	proved	completeness
NP	*(S\NP)/NP*	*(X\⋆X)/⋆X*	*(S\NP)/NP*	NP
		((S\NP)/NP)\⋆((S\NP)/NP) (>)		
		(S\NP)/NP (<)		
		S\NP (>)		
	S (<)			

Another use of modalities is suggested by Bozşahin (2002). Bozsahin proposes a morphemic lexicon, elements of which adhere to combinatory categorial grammar principles. He introduces what he calls "morphosyntactic modalities" to restrict and control the attachment characterics of morphemes. These modalities appear on the atomic categories as opposed to slashes on Baldridge's scheme. $\bowtie$ denotes equality and $\triangleleft$ means "up to and equals". Phrasal scope versus word scope problem is elegantly solved in the lexicon keeping the syntax semantics interface transparent. Bozsahin's modalities help simulate the morphotactic order of the attachment of suffixes while restricting some of them in a way to encourage phrasal attachment. This is acquired by means of delaying the morpheme attachment until after "constituent" formation. The example in (4.18) shows the categories for the English noun *toy* and the plural morpheme *-s*. (4.19) shows how the correct semantics is constructed by delaying affixation by modalities.

(4.18) toy := $^{b}\triangleleft N/^{b}\triangleleft N$: λxtoy x

-PLU := $^{n}\triangleleft N\backslash^{b}\triangleleft N$: λxplu$x$

(4.19) a.

toy	gun -s
$^{b}\triangleleft N/^{b}\triangleleft N$	$^{n}\triangleleft N$:plu gun (<)
$^{n}\triangleleft N$:*toy(plu gun) (***)	

because n-num (n) $\not\leq$ n-base (b)

b.

toy	gun	-s
$^{b}\triangleleft N/^{b}\triangleleft N$	$^{b}\triangleleft N$	$^{n}\triangleleft N\backslash^{b}\triangleleft N$
$^{b}\triangleleft N$:toy gun (>)		
$^{n}\triangleleft N$: plu(toy gun) (<)		

4.2 Generative Power of CCG

Shieber (1985) proved that natural languages falls into a category more powerful than context-free grammars (CFG) in Chomsky hierarchy using evidence on Swiss-German cross-serial dependencies. Earlier, Bar-Hillel, Gaifman, and Shamir (1964) had proved that AB grammars are context-free. Hence they are not expressive enough for natural grammars. Combinatory rules of CCG extend the capabilities of categorial grammars in a very limited manner. CCG is at level 1 of an infinite hierarchy of Mildly Context Sensitive systems of which CFG is level 0. So CCG is *very* mildly Context Sensitive, in fact, nearly context-free. Tree-Adjoining Grammars and Linear-Indexed Grammars which are weakly equivalent to CCG are also in this level (Vijay-Shanker and Weir, 1994).

Dutch crossing dependencies, shown in the previous section, require more than context-free generative power (Wall, 1972). Handling these require order changing operations like crossed composition of CCG. These combinatory rules introduce the extra generative power to explain the linguistic phenomena that cannot be explained with CFGs.

Another extension of categorial grammars, namely, Categorial Type Logics (CTL) (Morrill, 1994; Moortgat, 1997; Oehrle, to appear) have more power than context-sensitive if there is no restriction on modalities (Carpenter, 1999), but a subset of CTLs were proved to be equivalent to CCGs (Kruijff and Baldridge, 2000). Multimodal-CCG is proved to be weakly equivalent to CCG in generative power (Baldridge, 2002). Hoffman (1995) proves that restricted Multiset-CCG is weakly equivalent to CCG but the unrestricted version is more expressive than needed.

4.3 Word Order

There have been several variations in CCG mainly to handle word order variation and scrambling in an efficient and compact way. Only those that are relevant to Turkish are given here. Hoffman (1995) proposed Multiset-CCG to handle free word order with CCG. The significance of Multiset-CCG is that the set of arguments can be defined rather than one-at-a-time argument selection. This is shown to work well to handle free word order particularly in Turkish (Hoffman, 1995). However, non-restricted version of Multiset-CCG is not weakly equivalent to CCGs, and has more generative power.

Set-CCG is proposed by Baldridge (2000) which has the advantage of handling

scrambling in languages effectively. Set-CCG is also proven to be strongly equivalent to CCG in Baldridge (2000). Underspecification of directionality in Multiset-CCG is not available in Set-CCG. The rationale is that languages are head-consistent in word order, but argument order may vary within a language. For example, the multi-set category $S|\{NP_{nom}, NP_{acc}\}$ for a verb captures all six variations of subject-object-verb (SOV), lexically. The set-CCG category $S\backslash\{NP_{nom}, NP_{acc}\}$ states that all arguments must be to the left of the verb, hence only SOV and OSV are defined lexically. The latter is also consistent with Bozşahin (to appear) who claims only SOV and OSV orders are lexical and other orders are derived by processes like detopicalisation.

Bozşahin supports this argument with the fact that some constructions such as subordination are strictly head-final. Long-distance scrambling out of these phrases is allowed, but post-verbal scrambling in subordination is very uncommon.

4.4 Parsing CCGs

4.4.1 Parsing and Ambiguity

The advantages of a relaxed account of constituency are not limited to the ones shown above. Gapping and argument cluster coordination are other examples. However, allowing words to combine with each other more freely brings in an efficiency issue which is one of the most discussed issues in CCG, namely, the "spurious ambiguity". Spurious ambiguity (Wittenburg, 1986) occurs when several syntactic derivations yield the same semantic structure. According to Steedman (2000) different orders of combination are crucial in determining the information structure. Steedman shows that the ambiguity caused by this flexibility is needed to account for different prosodic bracketings. However, in practical parsing applications, one-to-one matching of semantic classes to derivation trees is preferred by various authors for efficiency, simplicity and performance reasons (Hockenmaier, 2003a; Clark and Curran, 2007b).

Many authors worked on removing spurious ambiguity in equivalent or less expressive versions of CCG such as Lambek calculus (Hepple, 1990; Hendriks, 1993; König, 1989; Hepple and Morrill, 1989). There are several studies on resolving spurious ambiguity in CCG.

We discuss some of these methods to tackle "spurious" ambiguity in parsing in this section. We discuss three approaches each using a different level of representation in order to recognize and remove the unnecessary ambiguity in parsing. Pareschi and

Steedman (1987) use feature structures and redundancy in parsing is limited with the help of both the syntax and semantics. Karttunen (1989) proposes a purely semantics-driven method of subsumption checks to eliminate equivalent semantic classes from the chart. Eisner (1996a) on the other hand aims to provide a purely syntactic driven approach with an argument that semantic information is not always available during parsing.

4.4.1.1 Pareschi and Steedman (1987)

Pareschi and Steedman use an all-path parser and do bottom up incremental parsing. They use a special method to remove the redundant analyses from the chart. They use feature structures that represent the syntactic and semantic information in the categories and derivations. If an analysis has the same edges and has a feature structure equivalent to the feature structure of an analysis covering the same span, it is not added to the chart. They ensure all required edges are added by introducing an operation called *Revealing* which is based on the property of *Parametric neutrality*. This means that the syntactic type of any two categories involved in a combinatory rule determines the type of the third category. This ensures possible correct analyses requiring backward operations in a left-to-right incremental parser are kept in the chart and saves search space because the parser tries new parse candidates involving leftward looking categories instead of reanalysing.

Hepple (1987) shows that Pareschi and Steedman (1987)'s algorithm is incomplete and gives an extensive analysis of the cases where it fails to provide a correct analysis or a genuine ambiguity because of disallowing a certain construction earlier in the process of parsing.

4.4.1.2 Karttunen's Approach

Karttunen (1989) proposes a redundancy elimination method that is semantics driven. Karttunen (1989) uses subsumption to eliminate from the chart equivalent (or subsuming) semantic analyses for a given sentence part. This has been successfully applied to English and Japanese real text parsing by Komagata (2004). Komagata (2004) parses text from two languages with the CCG parser defined in Komagata (1997). Komagata (2004) runs the parser on 22 sentences from a created Japanese corpus, and a fragment of English corpus (197 sentences). Komagata shows considerable improvement on the efficiency of the parsers when disambiguation is applied.

4.4.1.3 Eisner's Normal Form Derivations

Spurious ambiguity can be avoided by keeping only one derivation tree in the chart per semantic class. This requires the knowledge of semantic types, which is not always available during parsing and it is very costly (Eisner, 1996a). Eisner shows that spurious ambiguity can be completely eliminated by forming normal form derivations that do not contain certain chains of composition.[5] Eisner (1996a) shows that using only syntactic restrictions such as blocking rightward chains of forward composition or backward chains of leftward composition are enough to eliminate spurious ambiguity during parsing. According to Eisner's restrictions, a category that is the result of a forward composition cannot be the primary (left) functor in a forward composition and forward application. Similarly, the result of a backward composition cannot be the primary (right) functor of a backward composition or backward application. These also hold for order changing rules i.e. crossed composition.

Eisner (1996a) provides a safe and complete way to normalise pure CCG derivations by restricting the combination properties of the result of composition. The overhead of Eisner's algorithm is constant ($O(1)$ time) as opposed to an extra factor of n for Karttunen subsumption. This is achieved by marking certain constituents by tags during derivation to block them from further combinations. Vijay-Shanker and Weir (1990) does a similar kind of tagging (Hockenmaier and Steedman, 2005).

Eisner shows that these restrictions can easily be applied to CKY parsing, and claims that they can easily be incorporated into other parsing algorithms in particular that of Vijay-Shanker and Weir (1993).

Eisner assumes a CCG without type raising, as mentioned before. Thus, his method does not guarantee to produce normal form structures for practical CCG systems, but it is shown to shrink the search space by eliminating equivalent syntactic parses, thus improving the efficiency of the parser (Clark and Curran, 2007b).

4.4.2 The Issue of Representation

Predicate-argument structure is the main level of representation in CCG. Derivation history is considered to be merely a guide during parsing. CCG derivation trees are at most binary. They are also very expressive because of the information contained in the CCG categories. The labels in a derivation tree such as the ones in CCGBank

[5] Note that Eisner excludes type-raising in his proofs and algorithms. There has been no study on how type-raising may affect Eisner's method of eliminating spurious ambiguity in CCGs to our knowledge. However, it has been argued that once type-raising is lexical his method is complete.

implicitly have information of inherent local and long-range dependencies. The choice of tree data structure also has a motive that they are more common in parsing literature and there are many polynomial-time parsers for phrase-structure grammars. CCG derivation inherently have the head argument information, thus an annotation of this information before parsing as this person Collins (1996) does is not necessary.

Hockenmaier (2003a) models on derivation trees that are in normal form. C&C models both on derivation trees and dependencies directly and shows that there is not much of an accuracy difference between the statistical disambiguation models trained with dependencies and the ones trained with derivation trees. All use CCGBank data (Hockenmaier and Steedman, 2007) for both experiments although possibly slightly different version at different times as CCGbank is going through improvement.

It is shown that using normal-form derivation trees to represent a derivation sequence helps with the efficiency problem caused by spurious ambiguities in wide-coverage systems (Hockenmaier, 2003a; Clark and Curran, 2007b).

Hockenmaier (2003a) models on CCG derivation trees, but there is a restriction on these. “The derivation trees in CCGbank are in normal-form which, in this case, means type-raising and composition are used only when necessary, e.g. for relative clauses, right node raising and argument cluster coordination.” (Hockenmaier and Steedman, 2005)[pp.12] This means that the parser favours function application over combinatory rules such as composition and type raising, and use the combinatory rules only when necessary. This improves the efficiency by reducing the number of entities in the chart and also narrows the search space. A similar method was suggested by Wittenburg (1987).

We discussed several ways to tackle the issue of the so-called “spurious ambiguity” in the previous sections. One possible way to avoid spurious ambiguity is to model on the predicate-argument structures directly, because spurious ambiguity is caused by different choices in the derivation paths but derived dependencies, or predicate-argument relations are shared for any unique semantic interpretation. Clark, Hockenmaier, and Steedman (2002) takes the side of modeling over dependencies which can be derived from the only primary level of representation in CCG: predicate-argument structure. However, Clark and Curran’s (2002) model is an unsound probability model as is Collins (1996) which it originated from. The aim in Clark, Hockenmaier, and Steedman (2002) is “to demonstrate that accurate, efficient wide-coverage parsing is possible with CCG, even with an over-simplified statistical model.”

Clark and Curran (2007a) describe the dependency (in the form of predicate-argument

relations) models and evaluate their parser on DepBank (King et al., 2003), which is an English dependency treebank extracted from the Penn Treebank, making the output compatible to DepBank parser RASP (Briscoe and Carroll, 2006). They use the dependencies derived from the gold-standard normal-form parses in CCGBank. They show that a parsing model trained with dependencies perform as well as a model trained with normal-form derivation structures in the CCGBank (Hockenmaier and Steedman, 2007).

Clark and Curran (2007b) provides state-of-the-art parsing speed and accuracy. This parser parses about 50 sentences per second on avearge on an 18-node cluster. A supertagger that has a high supertagging accuracy assigns CCG categories to words before parsing is performed. This provides a significant boost in the speed of the parser because, most of the time, the supertagger assigns the correct CCG category to the word in the first try. This is consistent with Bangalore and Joshi's (1999) claim that supertagging is almost parsing. We will discuss this parser and some of the others in detail in Chapter 9.

Zettlemoyer and Collins (2005) present a direct approach and map logical structures to the sentences without any kind of syntactic interference. Zettlemoyer and Collins (2007) provide results for the online learning of CCG parsing to create logical forms.

So far, we have seen statistical CCG parsers that train on relatively big corpora. There are also parsers of smaller scale for English and other languages. OpenCCG is the extended project originated from *Grok* (Baldridge, 2002). *Grok* was intended as a library for performing NLP tasks with CCG as described in Baldridge (2002) and Baldridge and Kruijff (2003). OpenCCG is an extended and improved version of Grok with a generation module (White and Baldridge, 2003). Small scale grammars for many languages such as English, Tagalog and Turkish are included in OpenCCG.

Hoffman (1995) describes a CCG parser that is capable of capturing different information structural variations that are caused by word order in Turkish. Hoffman's formalism relies on multi-sets of arguments in order to model free word-order in Turkish. Multi-Set CCG parser was implemented as a question answering system that predicts information structure given the word-order variation in questions.

Cha and Lee (2000) and Cha, Lee, and Lee (2002) describe a CCG parser for Korean language which is similar to Turkish in many respects such as morphosyntactic relations. They built a smaller coverage statistical CCG parser that is inspired by Hoffman's multiset-CCG formalism (KCCG). Their system has similarities with the ideas

presented in this thesis, in the sense that morphemes have representational status in the lexicon and they may have phrasal scope as described for Turkish in Chapter 3. Cha, Lee, and Lee (2002) acquire high precision and recall of up to 87.67/87.03% on morphemic Korean data. The evaluation is done using PARSEVAL.

Bozşahin (2002) presents a morpheme-based CCG grammar for Turkish. The attachment characteristics of morphemes are regulated by a lattice of morpho-syntactic modalities. Bozşahin argues for the necessity of a morphemic as opposed to word-based account of lexicon and explains that some morphemes have phrasal semantic scope, rather than the scope of the word they are attached to. He provides a grammar and a lexicon for a relatively small coverage parser for Turkish that uses Eisner's constraints in an attempt to eliminate "spurious" ambiguity.

Chapter 5

Inducing a CCG Lexicon

Within the last decade, there has been significant increase in the amount of annotated data available for different languages which has given rise to an increase in number of wide-coverage multilingual parsers. Phrase structure representation following Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993), dependency structure annotation as in Czech Prague Dependency Treebank (Hajič, 1998; Böhmová et al., 2003), and hybrid representations e.g. German Tiger Corpus (Brants et al., 2002)) are the most popular representation types.

Dependency Parsers (Kudo and Matsumoto, 2000; Nivre, 2003; McDonald et al., 2005; Nivre, 2006; McDonald and Pereira, 2006) as well as traditional phrase-structure parsers were developed that use dependency treebanks as data. However, phenomena such as non-projective dependencies or long-distance dependencies were sometimes ignored even by the studies that provide the most satisfying parsing results on linguistic data. We believe deep linguistic information such as long-distance dependencies should not be ignored even when parsing accuracy on the rest of the data is high. Ignoring these phenomena would significantly degrade the usefulness of a particular language processing application. Crossing dependencies as well as non-crossing ones, long-range dependencies as well as local ones should be acknowledged in an NLP system, because though rare, they are crucial to semantic interpretation when they do occur.

A CCG parser requires a lexicon that is equipped with syntactic and semantic categories, a list of the rules assumed, and a model to evaluate outcomes or guide search. Creation of such a lexicon requires considerable manpower when it is done manually. There are efforts to create CCG lexicons automatically (Çakıcı, 2005) or semi-automatically (Hockenmaier, 2003a; Hockenmaier, 2006; Hockenmaier and Steedman,

2007) for different languages. These use existing corpora in phrase structure or dependency formats to create CCG derivation trees and CCG lexicons.

Clark and Curran (2006) show that statistical parsing models trained on partial data that is obtained by using only CCG categories perfom almost as well as systems trained on CCG normal-form derivation trees. With the view that "supertagging is *almost* parsing" (Bangalore and Joshi, 1999), given the correct categories parsing is fast and accurate (Clark and Curran, 2004b). However, supertaggers themselves need tagged data to be trained on. In the absence of large amounts of data, generalisation becomes important. In this and the next chapter, we explain how a dependency treebank can be turned into a CCG lexicon automatically and evaluate the resulting lexicon in Section 5.9 and propose a morphemic account of lexicon in an attempt to explain morphosyntactic behaviour and solve the possible sparse data problem up to some degree with the use of morphology.

5.1 Relevant Work

Hockenmaier and Steedman (2007) and Hockenmaier (2006) present wide-coverage CCG lexicons for English and German that are derived from WSJ portion of the Penn Treebank and the Tiger Treebank respectively. Hockenmaier and Steedman (2007) translate the Penn Treebank phrase structure trees into CCG normal-form derivation trees. The end product of this translation process is a set of binary trees which represent the steps of a CCG derivation. Each level of a subtree represents the daughters combining with CCG rules to yield the result parent. The leaves of these trees give CCG categories for words. There are several stages of the translation process that converts phrase structure trees to the representations of CCG derivations. Some of these steps are shown in 5.1.

(5.1) *for each tree:* τ
preprocess(τ)
determineConstituentType(τ)
binarise(τ)
assignCategories(τ)

Preprocessing involves correcting tagging and minor annotation errors, and some other procedures such as correction of the bracketing for coordinate structures, reanalysis of NPs which are represented as flat trees in the Penn treebank, deletion of

null elements etc. Some preprocessing steps involve only simple changes to the tags whereas some require transformations of the whole tree (bracketing changes).

Head finding which is included in the step for determining the constituent type uses the head-finding rules of Magerman (1994). A modified version of the process which is explained in Collins (1999) is used to determine the type of constituents as heads, complements and adjuncts.

Binarisation, as the name suggests binarises the otherwise non-binary parse trees in order to render them consistent with the nature of CCG derivations which requires categories to combine two at a time. After head-finding rules are applied, all the complements that are to the left of the head are made right-branching by inserting dummy non-terminals in the tree up-until the head, and all the complements to the right of the head are made left-branching in the same way. This is demonstrated on a simple tree in Figure 5.1.

Assigning categories is the last major step in the translation process. The categories are assigned to the internal nodes and the leaf nodes according to the following procedure: first the root node, which is usually S, is assigned its category. Then the complement nodes are assigned their categories depending on their label in the original tree. After that, the adjuncts are given their categories. This is a little more complicated than the steps so far. Forward composition and forward crossing composition are accounted for to prevent the inflation of the size of adjunct categories and for generalisation purposes. Hockenmaier and Steedman (2007) ignore the morphological features of the heads of adjuncts when creating the adjunct categories for the same reason. Finally, after adjuncts and complements are assigned categories, the heads are assigned their categories according to the following procedure.

The heads, adjuncts and complements in Figure 5.2(1) are differentiated as shown in Figure 5.2(2). The head of a parent with category X gets the category X if the other child is an adjunct. If not, depending on the directionality of the sister of the head (with category Y) the head gets the category X/Y or X\Y. Our toy tree in Figure 5.2(1) would look like Figure 5.2(3) after this step.

Figure 5.4 shows the outcome of the translation procedure which is applied to the Penn tree in Figure 5.3. Traces like *RNR* are successfully translated into CCG categories that hold this information in the categories implicitly. Some NPs e.g. *the quality* also go through type raising and this is shown in the trees with the use of unary branching. The part of speech tags are omitted from the examples for simplicity.

After initial category assignment by the procedure defined above there are a few

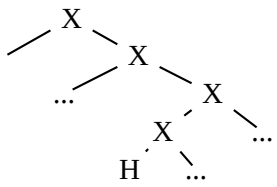


Figure 5.1: Binarisation principle

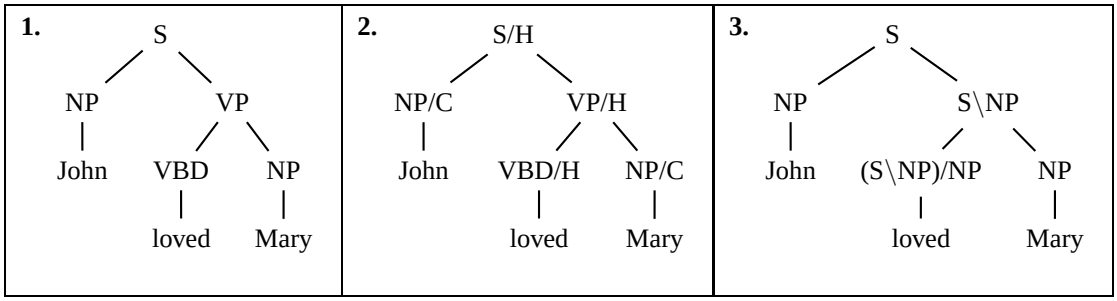


Figure 5.2: The translation example

post processing steps. One of them is the removal of the traces which were kept until the end of the translation process to get the long distance dependencies. Another is treating argument clusters and assigning them the correct categories. There are other steps which we will not mention here. Interested readers should refer to Hockenmaier (2003a) and Hockenmaier and Steedman (2007) for a detailed overview. However, we will refer to some aspects of the translation process in later sections for comparison and discussion purposes.

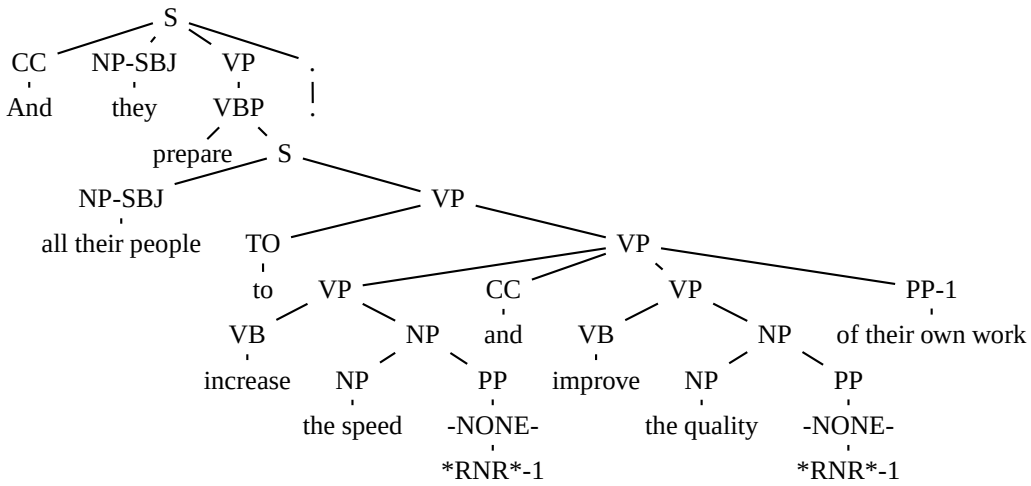


Figure 5.3: Penn Tree

Hockenmaier (2006) describes how this translation algorithm may be adapted to

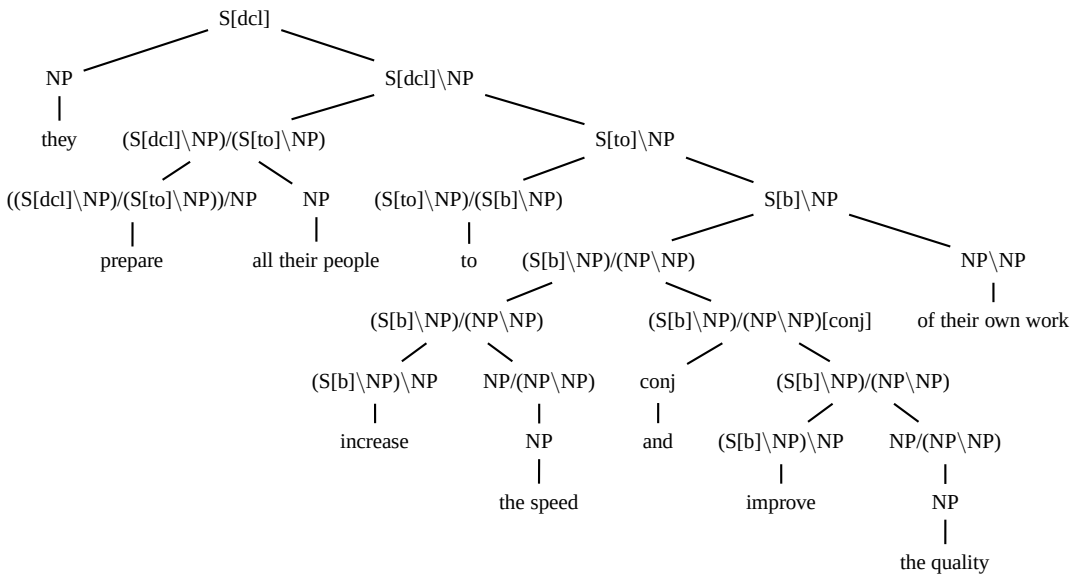


Figure 5.4: The outcome of the translation of the Penn tree in Figure 5.3. Taken from Hockenmaier and Steedman (2005)

account for German, which has more relaxed word-order and richer morphology than English. Another important difference between English and German CCG bank inductions is that the German treebank adopts a hybrid framework for representation which is dependency-based but inherits some phrase structure elements such as internal nodes. This means the translation procedure should map dependency tree to derivation trees. This brings in a few interesting challenges. For instance, there are crossed dependencies, in other words, crossed branches of the trees which we do not see in phrase structure grammars.

(5.2) *for each Tiger graph:* τ

$tigertree = createTree(\tau)$

$preprocess(tigertree)$

$translatetoCCG(tigertree)$

$verifyDerivation(tigertree)$

This does not bring any complexities to the translation of the dependencies to CCG categories as CCG categories are designed to represent such dependencies as well as non-crossing ones inherently. However, since in the Penn Treebank, these were represented differently with traces and null elements, a different path should be taken for translating a dependency treebank which is more related to the focus of research in this

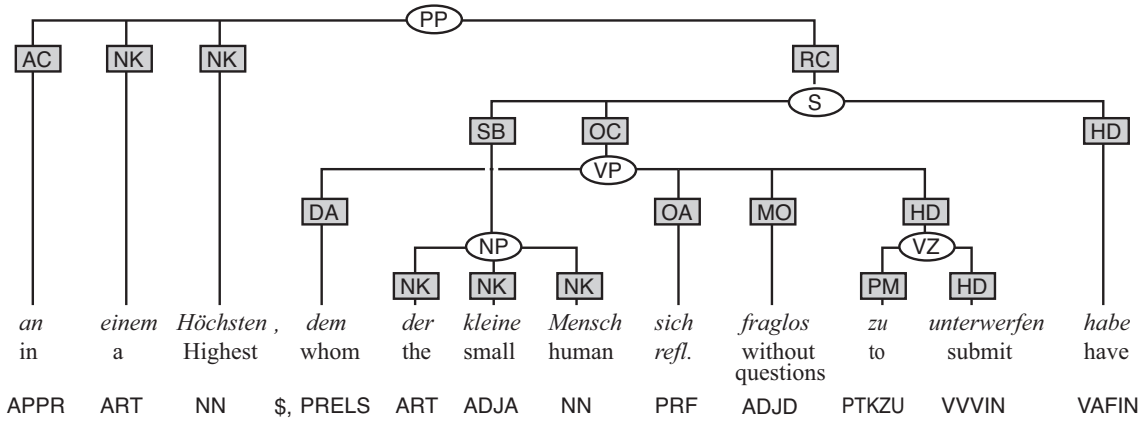


Figure 5.5: Tiger Graph

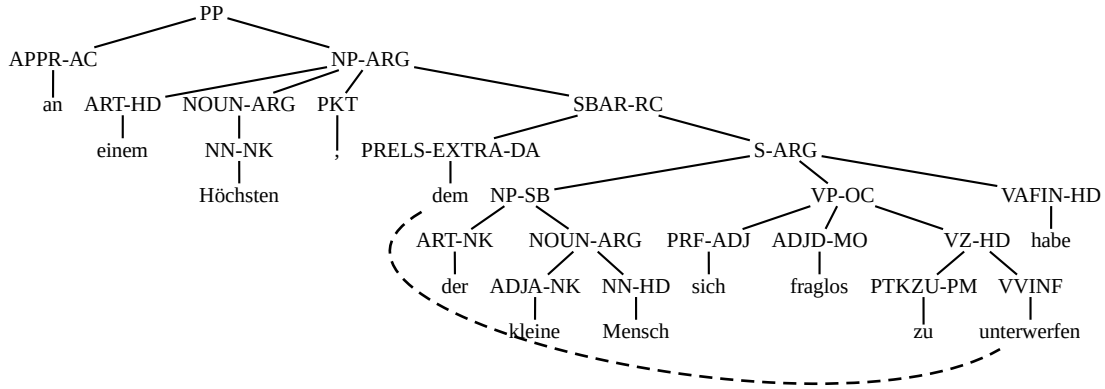


Figure 5.6: Planar Tiger tree

thesis. The algorithm that translates Tiger trees to CCG derivation trees is shown in (5.2).

The algorithm takes Tiger graphs as input and as such they are not planar trees. These graphs need to be turned into flat trees with non-crossing edges since CCG derivations are represented as binary trees of such kind. These trees then go through a preprocessing stage where some changes are made to the Tiger treebank trees to make them compatible with the translation algorithm that is designed for Penn Treebank style trees. A typical example of a Tiger treebank sentence is given in Figure 5.5.

Hockenmaier (2006) first creates a planar tree by raising the tree that causes the crossing to the higher level. The result of this is seen in Figure 5.6. This tree is then transformed into the tree in Figure 5.7 where the pre-terminal nodes are CCG categories for words and internal structure represents the derivation tree of the parse.

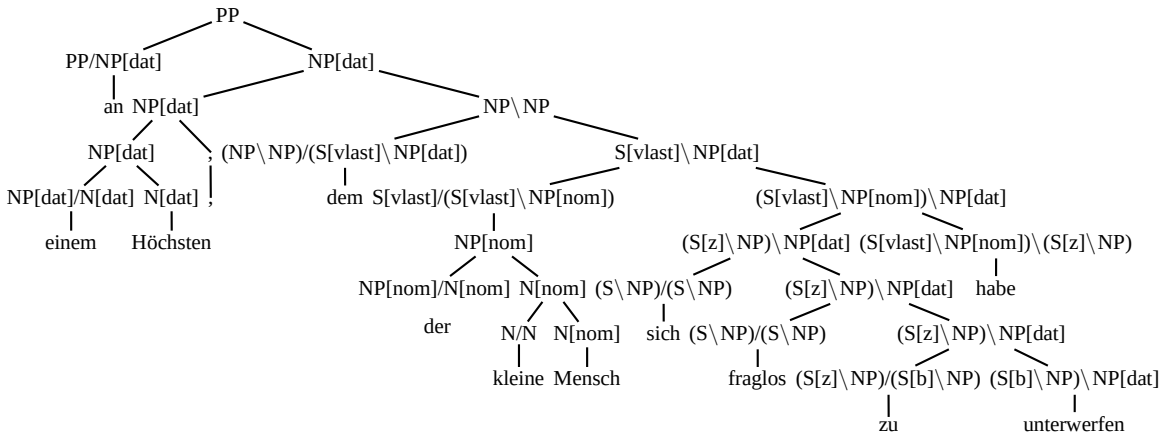


Figure 5.7: Tiger CCG tree (All Tiger images are taken from Hockenmaier (2006))

Other grammars were extracted from the Penn Treebank and other treebanks as well. Lexicalised TAGs (LTAGs) have been extracted from the Penn Treebank by Xia (1999) and Chen and Vijay-shanker (2000) by a similar strategy similar to Hockenmaier's. These studies process the whole treebank as there are others who used only portions (Neumann, 1998). Other wide-coverage deep grammars extracted from the Penn Treebank are: HPSG (Miyao, Nonomiya, and Tsujii, 2003), LFG (Cahill et al., 2002; Cahill et al., 2004). Extraction algorithms for LFG create F-structures automatically by making use of the configurational, categorial and trace information in Penn-II phrase structure trees (Cahill et al., 2008). Moortgat and Moot (2001; 2002) extract a CTL grammar from Spoken Dutch Corpus CGN (Hoekstra et al., 2001).

5.2 Algorithm

The lexicon induction procedure is recursive on the arguments of the head of the main clause. It is called for every sentence and gives a list of the words with categories. This procedure is then applied to all of the sentential conjuncts in case of coordination (Figure 5.8).

After the head of the first conjunct that is connected to the top level is found, the CCG catgory is assigned depending on how many arguments it has. Extracted as well as in-situ arguments are taken into account when this is done. Then all of the dependents of this head are visited assigning them CCG categories.

The words with labels object and subject are assigned *NP* and *NP[nom]* categories respectively, with the exception of sentential complements. If the word is an adjunct it is given a CCG category depending on how many arguments the head word has in

```
recursiveFunction(index i, Sentence s)
headcat = findheadscat(i)
//base case
if myrel is “MODIFIER”
        handleMod(headcat)
elseif “COORDINATION”
        handleCoor(headcat)
elseif “OBJECT”
        cat = NP
elseif “SUBJECT”
        cat = NP[nom]
elseif “SENTENCE”
        cat = S
.
.
if hasObject(i)
        combCat(cat,“NP”)
if hasSubject(i)
        combCat(cat,“NP[nom]”)
//recursive case
        forall arguments in arglist
                recursiveFunction(argument,s);
```

Figure 5.8: The lexicon induction algorithm

between the adjunct and itself.

Sentential modifiers are always assigned category *S/S* or *S\S* depending on their position in the sentence. All the other dependency types that are present in Turkish dependency treebank are assigned *X\X* or *X/X X* being the head category type.

A typical examples from METU-Sabancı Treebank is given in Figure 5.9. The CCG categories that are assigned to the words in the dependency graph are shown in Figure 5.10.

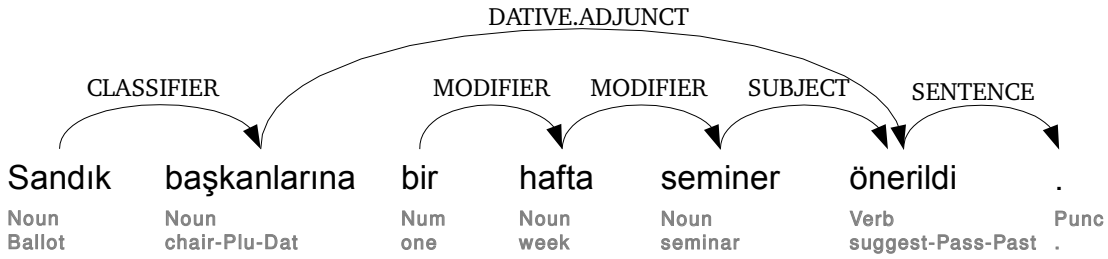


Figure 5.9: Turkish treebank tree

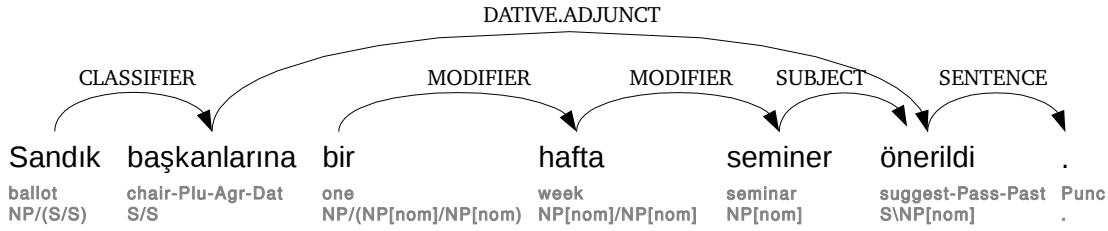


Figure 5.10: After CCG categories are assigned

5.3 Pro-drop

The pronoun subject of a sentence and the genitive pronoun in possessive constructions usually drop. In fact, pronouns are mostly used in Turkish either if there is an ambiguous reference in the discourse, or for contrastive and emphatic purposes (Göçmen, Şehitoğlu, and Bozşahin, 1995). Pro-drop information is not included in the Turkish dependency treebank, which is consistent with the surface dependency approach taken (Oflazer et al., 2003).

A *[nom]* (for nominative case) feature is added by us to the NPs to remove the ambiguity of predicate argument structure of verbs. All sentences have a subject.[1] They are either marked by morphology, or inferred from discourse if they are dropped. Thus, a verb with a category $S\backslash NP$ is assumed to be transitive with a dropped subject. This information is expected to be useful together with lexical rules in generalising the lexicon for use with a CCG parser. Figure 5.11 shows how explicit use of *[Nom]* feature helps in identifying the verb categories with a dropped subject.

There are many types of pro-drop in Turkish such as object pro-drop which cannot be recovered from the current version of the data. This means some verbs will have

[1] This includes the passive sentences in the treebank

	original	pro-drop
transitive	(S\NP[nom])\NP	S\NP
intransitive	S\NP[nom]	S
ditransitive	((S\NP[nom])\NP)\NP	(S\NP)\NP

Figure 5.11: Categories with and without dropped subjects

a different distribution over their predicate-argument structure leading to data sparsity and loss in accuracy. The counterpart to the English example in (5.3) about pronoun resolution, is in (5.4). All 4 combinations of object dropping are possible leading to different meanings as well as information structures as inclusion of pronouns in some cases marks topic. This has a similar effect achieved through intonation in the English example (5.3). Genitive subjects of relativisation are also marked with [nom] feature for the time being. We will leave the solution to this problem for future work.

(5.3) Bill called John, a republican, and then he insulted him.

(5.4) a. Bill John'a cumhutiyetci dedi , sonra hakaret etti.
bill john-Dat republican said , then insulted
intended reading: Bill called John a republican, and then Bill insulted John.

b. Bill John'a cumhutiyetci dedi , sonra o hakaret etti.
bill john-Dat republican said , then he insulted
intended reading: Bill called John a republican, and then John insulted Bill.

c. Bill John'a cumhutiyetci dedi , sonra ona hakaret etti.
bill john-Dat republican said , then he-Dat insulted
intended reading: Bill called John a republican, and then Bill insulted John.

d. Bill John'a cumhutiyetci dedi, sonra o ona hakaret etti.
bill john-Dat republican said , then he he-Dat insulted
intended reading: Bill called John a republican, and then Bill insulted John.

5.4 Modifiers and Adjuncts

MODIFIER label is the most overloaded label in the Turkish dependency treebank. The lexicon induction algorithm defined relies mostly on dependency labels to predict CCG

categories. When assigning categories to “Modifiers”, the category of the head of the word which had been assigned in the previous recursive run is usually enough. For instance, a noun phrase modifier is assigned *NP/NP* or *NP\NP*.

To give a few examples, adjuncts are assigned *(NP/NP)/(NP/NP)* when they modify adjectives, *(S\NP)/(S\NP)* when they modify transitive verbs. Since there is no distinction between V modifiers and VP modifiers we predict the category of the adjunct depending on its position among the verb and its arguments (complements). If all the complements are in between the adjunct and the verb then we assign the adjunct *S/S* category. We also consider extraposed or extracted complements that may be on the other side of the verb. If the adjunct itself has an object this is also reflected in the category as an NP argument such as *(S/S)\NP* for cases like adverbs derived from verbs with derivational morphemes.

A predicate may have a sequence of modifiers and modifiers may modify other adjuncts, too. In this case, we may end up with categories like (5.7), and even more complex ones. CCG’s composition rule means that as long as adjuncts are adjacent and they have the same head, they can compose which means they all are assigned *S/S* for this case, and they compose to a single *S/S* at the end without compromising the semantics. This method eliminates many gigantic adjunct categories, that are especially unavoidable in lexemic lexicons, causing sparse counts from the lexicon, following Hockenmaier (2003a).

(5.5)

diferansiyel	eşitliklerinizdeki	matematiksel	tekillikleri	daha	iyi	
differential	equations+Poss+Rel	mathematical	singularites+Acc	more	good	

bir	modelle	ortadan_kaldırdığınızda	Tanrınız	da	tekillikle	birlikte
a	model+Ins	eliminate+P2pl+When	God+Ps2pl	too	singularity+Ins	together

ortadan_kalkar	.
disappear	.

When you eliminate the mathematical singularities in your differential equations with a better model, your god, too, disappears together with the singularities.

(5.6)

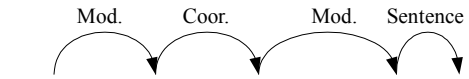
daha|Adv|((S/S)/(S/S))/((S/S)/(S/S))/((S/S)/(S/S))/((S/S)/(S/S)) – “more”
iyi|Adj|((S/S)/(S/S))/((S/S)/(S/S)) – “good”
bir|Det|NP/NP –“a”
modelle|Noun_Ins|(S/S)/(S/S) –“model+Ins”
ortadan_kaldırdığınızda|Noun_Verb|(S/S)\NP – “remove+When”
...when you eliminate it (singularity) with a better model...

(5.7) daha (Adv):= *(((S/S)/(S/S))/((S/S)/(S/S)))/(((S/S)/(S/S))/((S/S)/(S/S)))*
'more'

5.5 Coordination

The treebank annotation for a typical adverb coordination is shown in (5.8). The constituent which is directly dependent on the head of the sentence, "zıplayarak" in this case, takes its category according to the algorithm then all the other conjuncts are visited by the recursive algorithm. If one of the coordinating elements is involved in relativisation, the others are assumed to shared the extracted element if they have the same morphological clues such as relativisation particles. However, sharing of other arguments such as objects is not possible since this information is missing in the data.

(5.8)



Koşarak	ve	zıplayarak	geldi	.
run+While	*and*	*jump+While*	*come+Past*	

He came running and jumping.

In Çakıcı (2005), it is shown that verb phrase, verb or sentence coordination cannot be differentiated because of the lack of this information in the treebank. 800 sentences that had this kind of coordination that cannot be differentiated were removed from the treebank. We explain in Section 2.5 how we added secondary links connecting shared arguments to both conjuncts to solve this problem for coordination constructions involving matrix verbs. We use this information to predict the category of the head correctly. (5.9) is an example of this kind. Without this information predicate-argument structures of sentences with coordination are incomplete. These categories not only miss the predicate argument relations between the relativised verb *kusturan* and the extracted subject *kaptan*, but also the dependencies that implicitly state that kaptan is also the extracted subject of the first relativised verb *koşan*. This is because these verbs receive adjective categories determined by their parts of speech. This example may be compared with (6.9) in Section 6.4.4 where these dependencies are correctly captured with appropriate categories in a sentence of similar structure.

(5.9)

Kristof_Kolomb'un	*NP[nom]*	*– Kristof_Kolomb-Gen*
yeni	*(NP/NP)/(NP/NP)*	*– new*
zenginlikler	*NP/NP*	*– treasure-Plu*
peşinde	*(NP/NP)/(NP/NP)*	*– after*
koşan	*(NP/NP)\NP*	*– run-PresPart*
ve	*conj*	*–and*
tayfasına	*(NP/NP)/(NP/NP)*	*– crew-Poss3sg-Dat*
kan	*NP*	*– blood*
kusturan	*(NP/NP)\NP*	*– vomit-Caus-PresPart*
zalim	*NP/NP*	*– tyrannic*
bir	*NP/NP*	*– a*
kaptan	*NP*	*– captain*
olduğunu	*(NP\NP[nom])\NP*	*– be-PastPart-Agr-Acc*
fark_ediyorsunuz	*S\NP*	*– realise-Prog-P2pl*
.	.	.

You realise, that Christopher Colombus is a tyrannic captain who runs after new treasures and agonises his crew.

Verb ellipsis is considered to be one of the weaker points in dependency grammar (Oflazer et al., 2003) together with other headless constructions. Oflazer et al. (2003) describes inserting null elements as "dummy constituents" to overcome this problem in annotation. However, the final release of the treebank does not have these null elements. Therefore we chose to leave analysis of these sentences as future work.

5.6 Noun Phrases

There are no articles and thus no $NP/\bar{N}$ distribution in Turkish. All marked and unmarked instances of nouns are given category NP for the sake of simplicity. Object heads are given *NP* categories. Subject heads are given *NP[nom]*. The category for a modifier of a subject NP is *NP[nom]/NP[nom]* and the modifier for an object NP is *NP/NP* or *NP\NP* although NPs are almost always head-final.

5.6.1 Collocations

Collocations and some frequently co-occurring words are grouped into single entries in the treebank. However, the annotation of these constructions is not very consistent.

We assign the collocations their categories according to the dependency label and morphological features of the compound which are usually the features of the last word in the group.

5.7 Relativisation

Long-range dependencies, which are crucial for natural language understanding, are not modelled in the Turkish data. Hockenmaier handles them by making use of traces in the Penn Treebank (Hockenmaier, 2003a)[sec 3.9]. But Turkish treebank does not employ traces or any other means to represent long-distance dependencies or extractions of any type. For instance, the relativised verb is represented as a modifier of the head noun and there is no explicit or implicit relation between the extracted noun and the relativised verb as discussed in Section 2.5.3. However, because of lexemic nature of the lexicon that is described here, these secondary links added will not be of much use here. We will show CCG category set that predicts both surface and deep dependencies in cases of long distance dependencies in the next Chapter.

A typical subject extraction example is shown in (5.10). Relativised verb is considered as an adjective formed out of a verb phrase with the use of derivational morphology. Since its label is MODIFIER, it is assigned an adjective category. If it has arguments, these are also represented as backslashed NPs on the adjective category.

(5.10)

Kitabı	okuyan	adam	uyudu.
Book-Acc	*read-PresPart*	*man*	*sleep-Past.*
NP	$(NP_{\text{nom}}/NP_{\text{nom}})\backslash NP$	NP_{nom}	$S\backslash NP_{\text{nom}}$.

The man who read the book slept.

The lexicon in this form represents all the dependencies inherently in CCG categories. This is also consistent with the fact that the predicate-argument relations are lost in relativisation.. This means (5.11) and (5.12) will get exactly the same surface dependencies. This is enough for most of the state-of-the-art dependency parsers. However, we believe that recovery of predicate-argument relatins, and thus, semantic interpretation should be the basis of evaluation. We believe a system that is not capable of predicting *araba* as the adjunct in the first sentence and *kitap* as the object in the second sentence is not complete. Thus, treebanks should be designed to include this information and parsers should be able to predict them.

(5.11)

Benim	uyu-duğum	araba	yandı.
I-Gen	*sleep-PastPart*	*car*	*burn-Past*
$\overline{NP_{\text{nom}}}$	$\overline{(NP/NP)\backslash NP_{\text{nom}}}$	$\overline{NP_{\text{nom}}}$	$\overline{S\backslash NP_{\text{nom}}}$
	$\xrightarrow{\quad NP_{\text{nom}}/NP_{\text{nom}} \quad}$		
	$\xrightarrow{\quad NP_{\text{nom}} \quad}$		
	$\xrightarrow{\quad S \quad}$		

The car I slept in burned.

(5.12)

Benim	oku-duğum	kitap	yandı.
I-Gen	*read-PastPart*	*book*	*burn-Past*
$\overline{NP_{\text{nom}}}$	$\overline{(NP/NP)\backslash NP_{\text{nom}}}$	$\overline{NP_{\text{nom}}}$	$\overline{S\backslash NP_{\text{nom}}}$
	$\xrightarrow{\quad NP_{\text{nom}}/NP_{\text{nom}} \quad}$		
	$\xrightarrow{\quad NP_{\text{nom}} \quad}$		
	$\xrightarrow{\quad S \quad}$		

The book I read burned.

5.8 Punctuation

Punctuation marks can sometimes have dependents in METU-Sabancı Treebank. For instance, in a coordination structure, the first conjunct of a coordination has a dependency link to the comma that separates two conjuncts, and comma has a link to the head of the next conjunct. This case and the other types of inclusion of punctuation in the dependency structure is given in Section 2.3. Punctuation marks involved in a coordination are assigned the conjunction category.

Punctuation marks can also have different roles such as marking the sentential complements as shown in Figure 2.9. In these cases we change the dependency structure during the pre-processing stage to get the correct categories which are shown in Figure 5.12. Vocatives and words labelled SPEAKER may have punctuation as their "head" token.

5.9 Results

The most frequent words and their most frequent categories are given in Figure 5.13. The fact that the 7th most frequent word is the non-function word "dedi"(*said*) reveals the nature of the sources of the data —mostly newspapers and novels.

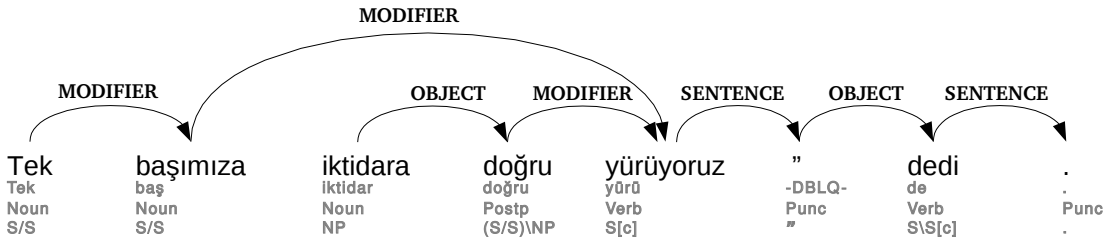


Figure 5.12: Use of punctuation in sentential complementation

token	**eng.**	**freq.**	**pos**	**most freq. cat**	**fwc***
,	Comma	2286	Conj	(NP/NP)\NP	159
bir	a	816	Det	NP/NP	373
-yAn	who	554	Rel. morph.	(NP/NP)\(S\NP)	554
ve	and	372	Conj	(NP/NP)\NP	100
de	too	335	Int	NP[nom]\NP[nom]	116
da	too	268	Int	NP[nom]\NP[nom]	86
dedi	said	188	Verb	S\NP	87
-DHk+AGR	which	163	Rel. morph.	(NP/NP)\(S\NP)	163
Bu	This	159	Det	NP/NP	38
gibi	like	148	Postp	(S/S)\NP	21
o	that	141	Det	NP/NP	37

***fwc** Frequency of the word occurring with the given category

Figure 5.13: The lexicon statistics

cattype	frequency	rank	type
NP	5384	1	noun phrase
NP/NP	3292	2	adjective,determiner, etc
NP[nom]	3264	3	subject NP
S/S	3212	4	sentential adjunct
S\NP	1883	5	transitive verb with pro-drop
S	1346	6	sentence
S\NP[nom]	1320	7	intransitive verb
(S\NP[nom])\NP	827	9	transitive verb

Figure 5.14: The most frequent category types

f	# categories
f>0	450
f>1	293
f>2	244
f>3	215
f>4	186
f>5	169
f>10	132
f>100	48

Figure 5.15: Distribution of category types

In Figure 5.14 the most frequent category types are shown. The distribution reflects the real usage of the language (some interesting categories are explained in the last column of the table). The category type distribution is shown in Figure 5.15. There are 450 distinct category types in total at the moment and 157 of them occur only once.

The English CCGbank lexicon contains 1286 types with around 439 occurring only once (Hockenmaier and Steedman, 2007) for about 1 million tokens with CCG categories. 556 categories appear more than 5 times.

German treebank has 50474 sentences and 900K tokens some of which were excluded in the lexicon induction process.[2] German lexicon contains 2506 lexical category types in which 1018 appear only once. 933 categories appear more than 5 times.

Numbers from all three treebnanks seem to have the same proportions, however, the Turkish treebank categories do not have features like German and English CCGBanks do. Therefore, lexical coverage and parsing coverage results obtained with these categories are the most precise estimates of the quality. However, we give these results here because too many infrequent categories mean irregularity and inconsistency in the data. The lexical coverage results are given in Section 5.9.1 and the parsing coverage measure is given in Chapter 9.

Figure 5.16 shows the growth of the category types as the sentence number increases. Different lines indicate the growth of category type frequencies greater than corresponding *n*. Note that even after 4500 sentences the curve for most frequent categories has not converged.[3] The data set is too small to give convergence and category

[2] About 8% of the sentences

[3] The slight increase after 3800 sentences may be because the data are not uniform. Relatively longer

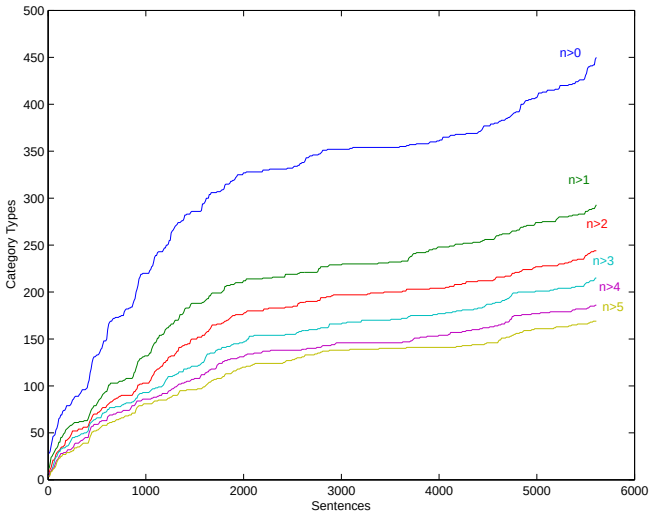


Figure 5.16: The growth of category types

types are still being added as unseen words appear. Hockenmaier (2003a) shows that the curve for categories with frequencies greater than 5 starts to converge only after 10K sentences in the Penn Treebank.

5.9.1 Coverage

The coverage of the lexicon is evaluated with a similar strategy Hockenmaier (2006) did for German CCG lexicon. She divided the lexicon into 10 parts and used 9 parts to extract a lexicon and the 10th part to test its coverage. Hockenmaier's test is a token-based comparison, and does not check if the token at hand appears in the lexicon more than one time. Thus, every occurrence is counted. This is one point that our strategy differs. We give the percentage of the unique token matches in the unique token set. The second difference is that we count the matches in test set given the 9 part-control set whereas Hockenmaier counts the matches in control set.

There are 5609 sentences in the set the lexicon is extracted from and there are 53796 tokens including puctuation. There are 450 category types and 19385 token types. There are 27895 unique word-category pairs. These numbers are very similar to the ones we had in Çakıcı (2005). The difference is that 800 sentences that had sentence or verb coordination had been removed from the original treebank in Çakıcı

sentences from a history article start after short sentences from a novel.

(2005).

We did a 10-fold cross evaluation on the data to test the coverage of the lexicon induced. We take out the 1/10 of sentences for test and take the rest as control set. We test the coverage by checking if the lexical entities and word-category pairs exist in the control set for each test set. Note that, when doing this we only compare the unique lexical entities and pairs, that way we do not count repetitive occurrences of the same entity in a test set.

We homogenise the data by taking 1 sentence of each 10 consecutive sentences in the test set of the fold everytime. In other words the first set has the $1^{st}, 11^{th}, 21^{st}$ etc. sentences in the data and the second set has $2^{nd}, 12^{th}, 22^{nd}$ and so on. This way we avoid the non-homogeneity problem we had in Figure 5.16. As seen in the figure, the number of distinct category types grow faster than the rest of the data after about 4300 sentences. This is due to the fact that the data is non-homogeneous and this portion of the treebank has much longer sentences and technical terms.

On the average 90.64% of the categories in the test set were seen in the training set. Standard deviation is 1.93. The number of unique words is close to the number of words in each evaluation set (3100 vs. 5400) which means not many words are frequent throughout each set which is a precursor to sparse data problems. Furthermore, the number of the words that were seen in training data is on the average 53.95% of all the unique words, and the ratio of seen word-category pairs is even less with an average of 37.06%.

5.9.2 Evaluation by sampling

As a small scale evaluation experiment we took a small sample of sentences (25) automatically selected throughout the data. These sentences were manually checked of their words are assinged correct CCG categories. They were also parsed using the CCG parser for Turkish described in Chapter 9. 14 of the sentences had at least one parse. Out of the 166 categories in these 25 sentences, 144 were correct and 22 wrong. Out of 25 sentences 9 had at least one wrong category. The rest of the sentences received a parse. Out of these 9 sentences only 1 still received a parse, since the mistake was an adjunct with wrong scope that was linked to some other element.

2 of the category errors were caused by annotation mistakes still present in the data. Of the 25 sentences 5 of the sentences did not receive a parse because at least one of the categories were not in the category dictionary.

There are 2 words that could not be assigned any category by the program. They are given NULL categories. The sentence with the most errors had 5 errors out of 9 words. A summary of this is shown in the following table. Note that this is not an evaluation of the linguistic quality of the lexicon. The categories that are right in word-based terms do not always yield the correct linguistic interpretation.

sentences	tokens	coverage	exceptions	no cat. assigned	cat. accuracy
25	166	14	5	2	86.7%

Chapter 6

A Morphemic CCG Lexicon

Turkish is a free word order language with very rich morphology. Not only case, tense and number, but also modality, polarity, voice and even relativisation are achieved through morphology. This means one Turkish word can translate into a full English sentence with several words. As discussed in Chapter 3, agglutinating languages may have very complex word forms. There are 231,818 morphemes including punctuation for 53,796 tokens in the treebank. This corresponds to 4.31 morphemes per word including the stem itself.[1] This means that for a fixed amount of data it is likely that some of the inflected (or derived) forms of the words will never be seen. When building language models, languages with complex morphology require either enormous amounts of data or generalisation of some sort. We show in this chapter how we generalise the CCG lexicon the induction of which is shown in the previous chapter using morphological clusters as lexical entities in Turkish. We attempt to create a wide-coverage morphemic lexicon using the IG-based dependencies in Turkish dependency treebank.

Our motivation is not limited to providing computational efficiency. There are linguistic constraints that make morphemic lexicons essential as well. Quite a few morphemes require semantic and syntactic scope greater than words. An example is what is called "suspended suffixation". In this type of phenomean, the morphological attachment characteristics of the affix contradicts with the semantic and syntactic scope it covers. The morpheme is affixed to the last conjunct in coordination and its morphophonemic, characteristics (such as vowel harmony) are determined by the word it is attached to, however the semantic scope it covers is the whole coordinating phrase.

Morphologically rich languages like Turkish, as well as other languages, suffer

[1] Some of these morphemes are zero morphemes, that do not correspond to surface forms. So, actual number of morphemes maybe smaller but because morph information is not included in the treebank, we do not know the exact number of surface morphemes per token.

	wb	IGb
tokens	53796	64992
token excl. punc.	43426	54662
Avg. sent. length excl. punc.	7.74	9.72
Avg. sent. length incl. punc.	9.57	11.56
Average number of tags per token	4.31	3.57

Figure 6.1: Numbers from Turkish treebank

from bracketing mismatches. A possible solution is to treat bound morphemes as separate lexical entities so that they can choose their own scope instead of being "bound" to the word they are attached to. We show in this chapter that morpheme or morpheme cluster based lexicons solve some of the mismatch problems mentioned.

In Section 6.1 we talk about the data we used to extract the grammar. Section 6.2 discusses the motivation for a morphemic approach. Section 6.3 discusses algorithm for inducing the morphemic lexicon. Section 6.4 gives analyses of frequent constructions. Section 6.4.7 discusses the evaluation of the resulting lexicon and compares it with the word-based lexicon in Chapter 5 and the final section is conclusion.

6.1 Data

We use the dependency treebank discussed in Chapter 2 to extract the morphemic CCG lexicon. The treebank consists of 5670 sentences [2]. The treebank contains dependencies, together with the morphological structure of each word. Morphological structure of a word consists of "inflectional groups" (IGs) that are divided by derivational boundaries. Relativisation and subordination are also represented as instances of morphological derivation of the involved relativised verbs in Turkish treebank. This means they exist in a different IG than the verb stem they are morphologically attached to. These IGs are the basic lexical entities for which we construct the CCG categories in this chapter as opposed to words in Chapter 5.

Figure 6.1 gives IG-based (IGb) and word-based (wb) statistics about the treebank. The average sentence length with and without punctuation are given. Punctuation constitutes about 10K of the tokens. The average number of IGs per word is 1.26.

[2]Note that this is the number after the correction of tokenisation errors

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Bu
lang. fam.	Sem.	Sin.	Sla.	Ger.	Ger.	Ger.	Jap.	Rom.	Sla.	Rom.	Ger.	Ura.	Sla.
genres	1:ne	6	3	8+	5+	1:ne	1:di	1: ne	1:no	9	4+	8	12
annotation	d	c+f	d	d	dc+f	dc+f	c+f	dc+f	d	c(+f)	dc+f/d	d	c+t
training data													
tokens (k)	54	337	1249	94	195	700	151	207	29	89	191	58	190
non-scor.	8.8	0.8	14.9	13.9	11.3	11.5	11.6	14.2	17.3	12.6	11.0	33.1	14.4
units (k)	1.5	57.0	72.7	5.2	13.3	39.2	17.0	9.1	1.5	3.3	11.0	5.0	12.8
tokens/unit	37.2	5.9	17.2	18.2	14.6	17.8	8.9	22.8	18.7	27.0	17.3	11.5	14.8
LEMMA	+	-	+	-	+	-	-	+	+	+	-	+	-
CPOSTAGs	14	22	12	10	13	52	20	15	11	15	37	14	11
POSTAGs	19	303	63	24	302	52	77	21	28	38	37	30	53
FEATS	19	-	61	47	81	-	4	146	51	33	-	82	50
DEPRELs	27	82	78	52	26	46	7	55	25	21	56	25	18
D.s H.=0	15	1	14	1	1	1	1	6	6	1	1	1	1
%HEAD=0	5.5	16.9	6.7	6.4	8.9	6.3	18.6	5.1	5.9	4.2	6.5	13.4	7.9
%H. preced.	82.9	24.8	50.9	75.0	46.5	50.9	8.9	60.3	47.2	60.8	52.8	6.2	62.9
%H. follow.	11.6	58.2	42.4	18.6	44.6	42.7	72.5	34.6	46.9	35.1	40.7	80.4	29.2
H.=0/unit	1.9	1.0	1.0	1.0	1.2	1.0	1.5	1.0	0.9	1.0	1.0	1.0	1.0
%n.p. arcs	0.4	0.0	1.9	1.0	5.4	2.3	1.1	1.3	1.9	0.1	1.0	1.5	0.4
%n.p. units	11.2	0.0	23.2	15.6	36.4	27.8	5.3	18.9	22.2	1.7	9.8	11.6	5.4
test data													
scor. tokens	4990	4970	5000	5010	4998	5008	5003	5009	5004	4991	5021	5021	5013
%new form	17.3	9.3	5.2	18.1	20.7	6.5	0.96	11.6	22.0	14.7	18.0	41.4	14.5
%new lem.	4.3	n/a	1.8	n/a	15.9	n/a	n/a	7.8	9.9	9.7	n/a	13.2	n/a

Table 6.1: CoNLL 2006 data

Characteristics of the data sets for the 13 languages (abbreviated by their first two letters): language family (Semitic, Sino-Tibetan, Slavic, Germanic, Japonic (or language isolate), Romance, Ural-Altaic); number of genres, and genre if only one (news, dialogue, novel); type of annotation (d=dependency, c=constituents, dc=discontinuous constituents, +f=with functions, +t=with types). For the training data: number of tokens (times 1000); percentage of non-scoring tokens; number of parse tree units (usually sentences, times 1000); average number of (scoring and non-scoring) tokens per parse tree unit; whether a lemma or stem is available; how many different CPOSTAG values, POSTAG values, FEATS components and DEPREL values occur for scoring tokens; how many different values for DEPREL scoring tokens with HEAD=0 can have (if that number is 1, there is one designated label (e.g. “ROOT”) for tokens with HEAD=0); percentage of scoring tokens with HEAD=0, a head that precedes or a head that follows the token (this nicely shows which languages are predominantly head-initial or head-final); the average number of scoring tokens with HEAD=0 per parse tree unit; the percentage of (scoring and non-scoring) non-projective relations and of parse tree units with at least one non-projective relation. For the test data: number of scoring tokens; percentage of scoring tokens with a FORM or a LEMMA that does not occur in the training data.

Tables 6.1[3] and 6.2[4] compare statistics across different dependency treebanks that were included in the CoNLL 2006 and 2007 shared tasks for dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). The percentages of new forms and new lemmas in the test sets give an idea about the inflectional properties of Turkish compared to other languages. 41% of all the tokens in the test data are *unseen* at least in that inflectional form. This number is 36.3% in Table 6.2 and is the highest among languages included. The fact that the percentage of unseen lemmas is also high means that the training set is too small for reasonable coverage but that only accentuates the importance of and the need for generalisation through morphology.

If we compare these numbers with Dutch data that has 15.9% of unseen lemmas in Table 6.1 which is slightly higher than Turkish (13.2%), we see that the main problem is not only unseen *words* but also unseen inflected forms.[5] Similarly, when we look at CoNLL 2007 figures, Arabic has the biggest ratio of unseen words to unseen lemmas which is about 6. But unseen words in Arabic are only 12% of the whole test set. This shows Arabic has very rich morphology as well, but because the percentage of unseen words is not as big as in Turkish it is expected to be less affected by sparse data problems at least for this particular test sample. This also means that Arabic would benefit from morphological generalisation, too.

6.2 Morphemic Lexicon

Marslen-Wilson (1999) gives a review of the experiments done to prove that the organisation of mental lexicon does not conform to Full Listing Hypothesis (FLH) (Butterworth, 1983). Marslen-Wilson (1999), furthermore, discusses experiments results of which suggest that compositional and productive derivational morphemes and inflectional morphemes are stored in the mental lexicon as separate entities. The issue of lexicon representation is also discussed in Hankamer (1989). Hankamer suggests that storing fully inflected forms of all the words in memory is not possible because of memory requirements.

[3]Data is taken from Buchholz and Marsi (2006).

[4]Data is taken from Nivre et al. (2007).

[5]Note that these statistics are from the version used for CoNLL 2006 shared task and thus might be slightly different from our statistics of the corrected treebank.

Multilingual										
	Ar	Ba	Ca	Ch	Cz	En	Gr	Hu	It	Tu
Language Family	Sem.	Isol.	Rom.	Sin.	Sla.	Ger.	Hel.	F.-U.	Rom.	Tur.
Annotation	d	d	c+f	c+f	d	c+f	d	c+f	c+f	d
Training Data										
Tokens (k)	112	51	431	337	432	447	65	132	71	65
Sentences (k)	2.9	3.2	15.0	57.0	25.4	18.6	2.7	6.0	3.1	5.6
Tokens/sentence	38.3	15.8	28.8	5.9	17.0	24.0	24.2	21.8	22.9	11.6
LEMMA	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
No. CPOSTAG	15	25	17	13	12	31	18	16	14	14
No. POSTAG	21	64	54	294	59	45	38	43	28	31
No. FEATS	21	359	33	0	71	0	31	50	21	78
No. DEPREL	29	35	42	69	46	20	46	49	22	25
No. DEPREL H=0	18	17	1	1	8	1	22	1	1	1
% HEAD=0	8.7	9.7	3.5	16.9	11.6	4.2	8.3	4.6	5.4	12.8
% HEAD left	79.2	44.5	60.0	24.7	46.9	49.0	44.8	27.4	65.0	3.8
% HEAD right	12.1	45.8	36.5	58.4	41.5	46.9	46.9	68.0	29.6	83.4
HEAD=0/sentence	3.3	1.5	1.0	1.0	2.0	1.0	2.0	1.0	1.2	1.5
% Non-proj. arcs	0.4	2.9	0.1	0.0	1.9	0.3	1.1	2.9	0.5	5.5
% Non-proj. sent.	10.1	26.2	2.9	0.0	23.2	6.7	20.3	26.4	7.4	33.3
Punc. attached	S	S	A	S	S	A	S	A	A	S
DEPRELS for punc.	10	13	6	29	16	13	15	1	10	12
Test Data										
Tokens	5124	5390	5016	5161	4724	5003	4804	7344	5096	4513
Sentences	131	334	167	690	286	214	197	390	249	300
Tokens/sentence	39.1	16.1	30.0	7.5	16.5	23.4	24.4	18.8	20.5	15.0
% New words	12.44	24.98	4.35	9.70	12.58	3.13	12.43	26.10	15.07	36.29
% New lemmas	2.82	11.13	3.36	n/a	5.28	n/a	5.82	14.80	8.24	9.95

Table 6.2: CoNLL 2007 data

10 languages of the multi-lingual track in CONLL'07 shared task for dependency parsing.

6.2.1 Why morphemic Lexicon

The interaction of morphology with semantics and syntax has long been studied (Sproat, 1985; Sproat, 1998; Pesetsky, 1985; Spencer, 1988). There are so-called *bracketing paradoxes* that are taken as empirical evidence that morphological processes interfere with semantics and syntax. Bracketing paradoxes may be defined as the contradiction between morphological attachment characteristic of the affix and the semantic scope it covers and are covered in Chapter 3.

Relativisation is considered to be a deverbaliser that makes adjectives out of verbs by the relativiser morpheme. An example is shown below.

(6.1)

Ödev-i	bitir-en	çocuk	uyu-du.
homework-ACC	finish-PresPart	child	sleep-PAST.

The child who finished the homework slept.

We argue that the relativiser morpheme here should have scope over the whole VP. For instance in (6.1) the bracketing should be as shown in (6.2a) rather than (6.2b).

(6.2) a. [[[[Ödev-i bitir]-en] çocuk] uyu-du]
homework-Acc finish -PresPart child sleep-Past

b. [[[[Ödev-i] bitir-en] çocuk] uyu-du]
homework-Acc finish -PresPart child sleep-Past
The child who finished the homework slept.

There are many examples in Turkish like the relativisation example here, where morphemes have phrasal scope. Some of these are discussed in Chapter 6 and more examples will be given in the course of this chapter. Many studies argue that involving even coarse-grained morphological representation of lexicon in Turkish gives better results. For instance, Turkish treebank is designed on IG-based dependecies (Atalay, Oflazer, and Say, 2003). Bozşahin (2002) proposes a morphemic CCG lexicon together with attachment constraints to prevent overgeneration for Turkish. The parsing results by Çakıcı and Baldridge (2006) and Eryiğit and Oflazer (2006) show that taking at least derivational morphology into account improves the parser performance. Oflazer and Durgar El-Kahlout (2007) discusses the way morphemes or morpheme groups rather than words bind other morpheme groups in a machine translation system. A purely lexical approach would either be computationally expensive or deficient in performance because of the need to model all inflectional forms of a word when translating from

one language to the other in this particular study. A similar approach bringing the morphology to the scene is taken by Dyer (2007) for Czech which is a language with a similarly complex morphology.

6.2.2 Why morphemic CCG Lexicon

6.2.2.1 Generalisation and Computational constraints

Statistical parsers need as much data as possible to learn from. Especially lexicalised theories like CCG rely on words and their relations to each other. Learning these relations is not trivial when complexity of word forms in agglutinating languages are taken into account. Complex inflected forms make the distribution of word and category pairs sparse. The numbers in Figures 6.2 and 6.3 illustrate this for a relatively common verb in the Turkish data. There are 177 occurrences of the verb *gitmek (to go)* in the data. According to the output of the program there are 128 distinct category word pairs. This means the average frequency of a category-word pair is 1.38. When the inflections are ignored and only stems are taken into account, there are only 14 distinct category types assigned to this verb. 7 of these occur only once (Figure 6.2). This means the average frequency of a category-word pair for this verb is 177/14 = 12.64. Figure 6.3 shows the categories that the root *git* occurs with in the lexicon.

.....	frequency	cat/word
total occurrences	177	N/A
distinct cat-word pairs	128	1.38
distinct category types	14	12.64

Figure 6.2: The figures for the verb *git (go)*.

However, ignoring morphology in this sense usually hurts the performance let alone improving it. This is why a mid-way approach need to be taken where morphology could be used as features together with stemmed words. We show in Çakıcı and Baldridge (2006) and in Chapter 8 that the best results are obtained by using stems, morphological derivation information, and word minus stem (suffix) part of the word all together as features.

Figure 6.4 shows the CCG categories assigned to words with the lexicon induction process described in Çakıcı (2005). *Dar (narrow)* is an adjective, but it is assgined

f	word	cat
1	git	NP\NP[nom]
1	git	(NP\NP[nom])\NP
1	git	(S\NP)/(S\NP)
1	git	S/NP[nom]
1	git	(S\NP[nom])\NP
1	git	((S\NP)/(S\NP))\NP[nom]
1	git	((S\NP[nom])\NP)/((S\NP[nom])\NP)
2	git	S/S
2	git	NP\NP
3	git	NP
5	git	S/NP
25	git	S\NP
28	git	S\NP[nom]
106	git	S

Figure 6.3: CCG categories (cat) and frequencies (f) of entities of verb *git (go)* in morphemic lexicon.

(S/S)/(S/S) because the head word of the noun phrase is a LOCATIVE.ADJUNCT. It is unlikely that *Dar (narrow)* will have the same category unless we have a very large corpus. Alternatively, if we assign *NP/NP* to it, then we will not get a parse.

In the same sentence, a morphemic approach would provide categories shown in Figure 6.5. *Dar* will be assigned *NP/NP*, which is a legitimate adjective category. Similar categories will be induced for *yol (path)*, which will be *NP*, every time it is in a noun phrase, regardless of the type of its head, rather than getting different categories for every occurrence as an adjunct, derived verb, and so on. Morphologically rich languages like Turkish, rely on morphological analysis or at least some kind of generalisation. Turkish is one of the new languages gaining focus with the rise of multilingual language processing, and the amount of annotated data to train wide-coverage NLP systems is quite limited. Making morphological data more available to these systems will both improve the performances of these systems and provide a way to generalise over the data.

Lexical rules may also be employed to handle problems described above. A lexical rule that turns an NP that is locative marked into a locative adjunct is also legitimate

for the phrase *Dar yollarda – through narrow streets*. However, there are other situations such as many cases of derivational morphology in Turkish treebank that change almost all types to other types. Handling these with lexical rules will add unnecessary additional expressive power to the grammar. Solving these with assigning independent lexical status to morphemes would be both simpler and safer.

6.2.2.2 Linguistic Principles

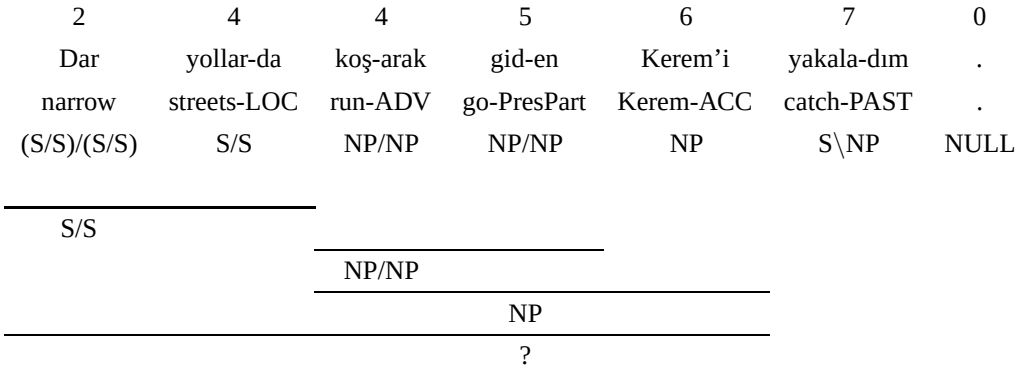
Acquiring all possible syntactic derivations representing all possible semantic interpretations is very important in wide coverage parsing. This is how the parse with the correct, or most likely interpretation is selected among many less likely or wrong parses. The categories in Figure 6.4 do not give the parse with the meaning that Kerem is running through narrow streets. This is one of the interpretations (the one we want) that this sentence has, together with another one with a different adjunct scope: "I saw, in narrow streets, Kerem who was running.". The adjunct modifies "run" in the first interpretation, where in the second interpretation it modifies "see".

The other concern is that the locative case marker *-da* has scope over the whole noun phrase *Dar yollar (narrow paths)* rather than just *yollar (paths)*. This means in order to get the correct semantic interpretation, we need to have *-da* have semantic scope over the whole phrase. This will yield the correct semantic interpretation much more simply and effectively than some other solutions such as "Quantifier Raising" (QR) and incorporating traces of morphemes (Pesetsky, 1985) to get the correct bracketing.

A similar problem is faced with *koşarak* and *giden* in the same sentence.[6] *-arak (-ing)* is a derivational suffix that makes adverbs out of verbs or verb phrases. Imagine we had a transitive verb instead of *run* here. In a CCG derivation, we would want the verb to combine with all of its arguments so the semantics is right. This would only be possible if we had a separate lexical entry for the suffix. The same is true for the relativised verb "git". The relativisation morpheme *-en* takes a VP, not a verb, as argument. This is achieved by having a separate lexical entry for it.

Figure 6.5 shows the morpheme based lexical entries and categories for this sentence.

[6]The morpheme *-en* is the surface realisation of -(y)An morpheme mentioned before.



I caught Kerem who left running through narrow streets.

Figure 6.4: This analysis of the sentence does not give the intended semantics.

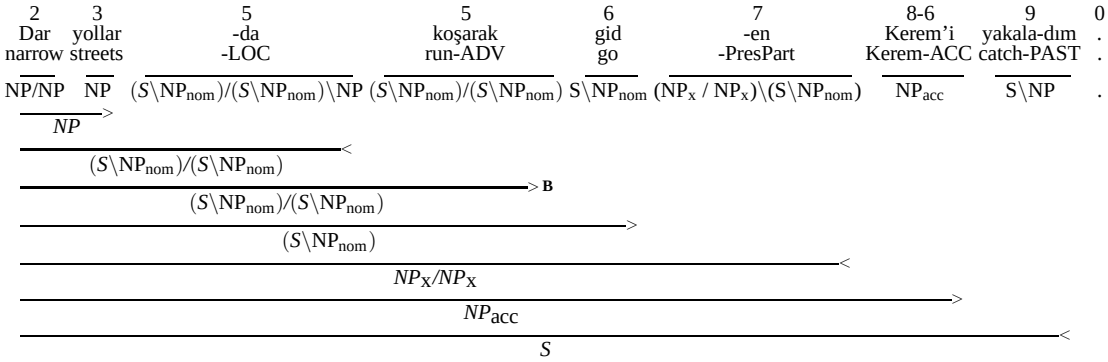


Figure 6.5: The morpheme-based categories for the sentence

6.3 Lexicon Induction

The algorithm is based on the one that is described in Çakıcı (2005). The dependencies are traversed from the head to the dependent in a recursive manner in order to construct the CCG categories. The results in Çakıcı (2005) imply that the Turkish CCG lexical categories are not complete after 53K tokens which is consistent with the results from CCGBank (Hockenmaier, 2003a). The ultimate aim is to generalise the lexicon with a morphological analyser and create a fully morphemic lexicon that will be helpful in overcoming the complexities that arise from the fact that Turkish is an agglutinative language with morphology heavily interlaced with syntax and semantics. We restrict ourselves to inflectional groups that are marked in the Turkish treebank for now, since analysis and automatic disambiguation of morphology at this level of detail remains a challenge.

The lexicon induction algorithm takes IG-based dependency structures as input and creates CCG categories for every token. Complement/adjunct distinction is important here. We take all the dependents that are called SUBJECT and OBJECT[7] as complements, modifiers (adverbs) and case adjuncts as adjuncts. The first step is translating the dependency graphs into IG based dependency graphs. After this step, these graphs go through pre-processing stages for regularisation. Finally, CCG categories are assigned to IGs. Details of these stages are explained in the following sections.

6.3.1 The morphemic dependency structure

The algorithm for the morphemic lexicon is very similar to the one described in Chapter 5. To preserve the connectedness of the dependency graph IGs must be involved in the dependency structure as well. There are several ways to represent IG-based dependencies of Turkish treebank including the approach taken in CoNNL 2006 shared task (Buchholz and Marsi, 2006) which replaces IGs with underscores because there is no lexical form for these in the treebank, and using the morphemes in them as features. The approach we used makes use of the IG tokens instead of representing them with an underscore. We believe a representation format that is as close to the lexical forms of the tokens as possible is the most appropriate approach.

In CoNLL 2006 shared task data set the IGs are represented by word-internal dependencies by replacing the inner IGs with underscore. Later in the evaluation process these dependencies are removed from the final score because they are trivial. The

[7] These are the main complement tags in the METU-Sabancı dependency treebank.

fact that morphemes are represented as underscore character means we cannot use the lexical information for morpheme groups. Eryiğit and Oflazer (2006) suggest that lexicalisation does not work for Turkish, however Eryiğit, Nivre, and Oflazer (2006) in a similar framework show that lexicalisation does improve the performance. Çakıcı and Baldridge (2006) also show that using both the stems and the rest of the word form as features give state-of-the-art results. However, since we don't have lexical information here, we will approximate these with using morpheme strings instead of completely ignoring them.

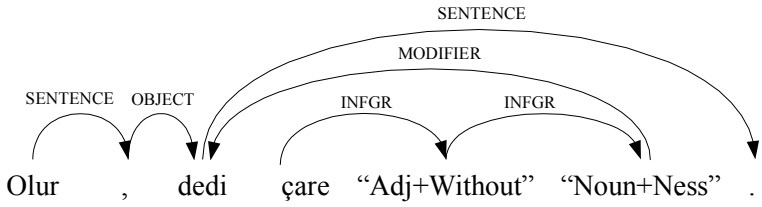
We introduce at least some degree of lexicalisation by assigning inflectional groups token names which are derived from the morpheme names since "morph" information is missing in the data. The token name for the IG representing an empty derivational morpheme that makes a verb out of a noun is shown below.[8]

(2,"Verb+Zero+Past+A3sg")– is *"Verb+Zero"*.

The lexicon induction algorithm requires a dependency graph that is connected. Only punctuation and sentence modifiers are allowed to be on top level, however, annotation inconsistencies and some tokenisation mistakes lead to disconnected graphs in some sentences. In Chapter 2 we show the cases where we connected these subgraphs and solution to some of the tokenisation mistakes. If the graph is still disconnected we treat each disconnected subgraph as a separate sentence.

When creating the morpheme based dependency graphs if there is more than one IG in a word, we make each of them depend on the IG immediately to the right and make the outward dependency emanate from the last IG to the IG that it depends on in the head word. The dependencies between the internal IGs of a word are labelled INFGR. So the implicit information that these are parts of one word is preserved. There is an example of this in (6.3).

(6.3)



T: Olur, dedi çaresizlikle

E: Alright , he said, hopelessly.

[8] As seen in the example, this is in fact a "Zero" morpheme, that does not have a surface lexical form. Zero morphemes are not rare (2022 instances) in the treebank.

6.3.2 Preprocessing

There were some inconsistencies in the treebank regarding the IGs involved in coordination. If a single-IG word coordinates with a word with multiple IGs and if the types of IGs (their part-of-speech tags) are different, it is usually the case that the link is annotated to go to the first IG of the second conjunct wrongly instead of the coordinating IG. This usually can be fixed by changing the dependency structure so that the link points to the last IG of the second conjunct. If the problem cannot be solved by the end of this step this means either there is an annotation mistake or IGs in the middle are involved in coordination or there is a suspended affixation case. However, this is very rare and most IG mismatch errors are solved by changing the link to point to the last IG.

All the sentences in Turkish treebank have at least one SENTENCE label, and if there is more than one, the one that depends on the final punctuation is considered to be the root. SENTENCE is assigned to the main predicate regardless of its part-of-speech tag. Fragments, for instance, are not differentiated. There are three types of copula in Turkish. One type is often considered as zero morpheme followed by person agreement shown in (6.4a). The other is the past tense that is shown by explicit morphemes *-yDH* and *-mHş* as shown in (6.4b-c). The last one is *-DHr* which is always in 3rd person. This is referred to as Epistemological copula.[9]

(6.4)

a.
Ev -de -yim
home -Loc -(CopZero) -Pers1sg
I am at home.

b.
Ev -de -ydi -m
home -Loc -CopPast -Pers1sg
I was at home.

c.
Ev -de -ymiş -im
home -Loc -CopNarr -Pers1sg
I was (supposedly) at home.

d.
Ev -de -dir
home -Loc -CopPres
He is (surely) at home.

The first case combined with 3rd person agreement (which is unmarked, too) in (6.5) shows no indication as to whether it is a copula sentence or a fragment. Since

[9] Can be translated as “It is” and implies definiteness.

intuitively copular sentences are more common, we add "Zero+Verb" IGs to the dependency structures during preprocessing to the end of non-verbal types that have SENTENCE label in the treebank. This is done because some copular sentences have this zero morpheme and some do not. For instance when the subject is in first person IG structure consists of 2 IGs, the first being the noun stem and the second being "Verb+Zero+..+A1sg" as in (6.6a). This preprocessing step is applied as a means of regularisation of copular sentences as shown in (6.6).

(6.5)
Ev -de
home -Loc -CopZero -Pers3sg(Zero)
(He) (is) at home.

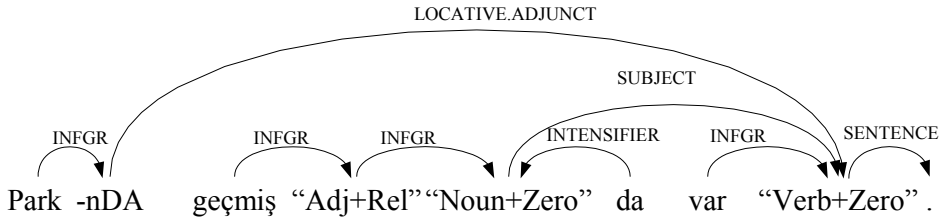
(6.6) a. Ev-de-yim

```
(1,"ev+Noun+A1sg+Pnon+Loc")(2,"Verb+Zero+Pres+A1sg")
```

b. Ev-de

```
(1,"ev+Noun+A1sg+Pnon+Loc")(2,"Verb+Zero+Pres+A3sg")
```

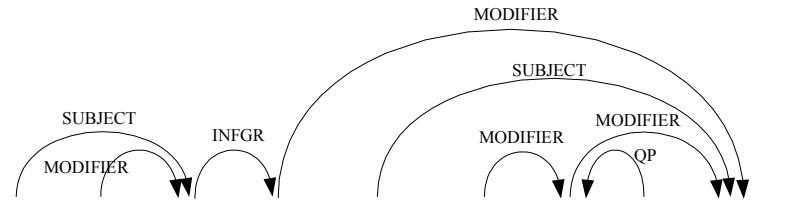
Case Adjuncts that are not objects or subjects are also split. One example is Figure 6.5 above where *-da* is assigned $(X/X)\backslash NP$ where $X = S\backslash NP_{nom}$, the category of the adjunct's head. The other example is (6.7).



(6.7)
Park -Loc past the one -Plu too there is

T: Parkta geçmiştekiler de var.
E: In the park, there are the ones from the past, too.

Conditionals are verbs that modify other verbs, so they have arguments of their own. This means they have phrasal scope. For this reason they are also split, even though this is considered as verbal inflection in the treebank. (6.8) is an example from the treebank. Conditional in this example serves as an adverb and it has its arguments. We believe it should be treated as other derived adverbs. Splitting conditionals will allow the verb stem to combine with its arguments before becoming an adverb.



(6.8) *He such die-Neg -Cond life-Poss2sg different way QP be-Aor-Past2*

T: O böyle ölmese hayatın başka türlü mü olurdu.

E: If he didn't die like this, would your life be different?

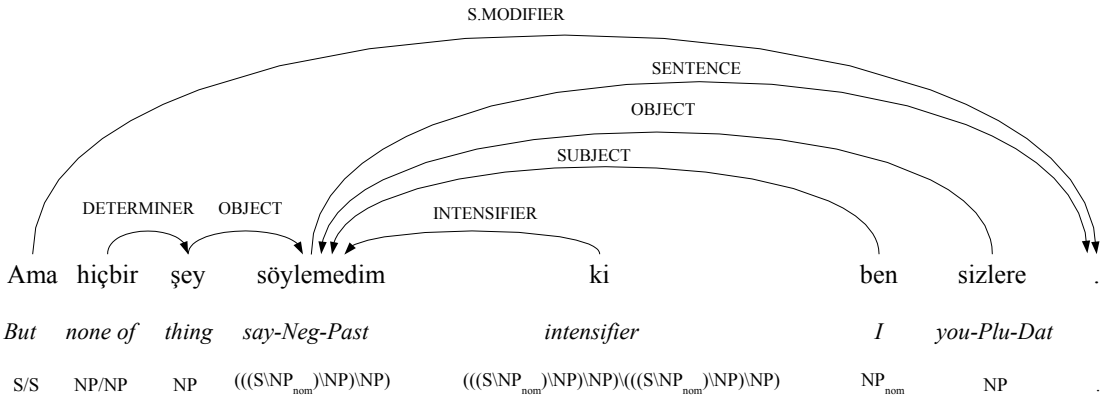
6.3.3 Algorithm

After translation of dependency graphs and preprocessing, the algorithm which is a modified version of the algorithm in Çakıcı (2005) is applied to the dependency structures.

If head is the stem of the word, we assign the category of the stem depending on the part-of-speech tags and give the (last) IG $X\backslash Cat$ where X is the result category of the stem after taking its arguments, if necessary, and Cat is the category it would have been assigned given its label. Note that in lexemic lexicon induction the category assigned to a word is usually determined by its relation to its surface-syntactic head in the treebank i.e. its dependency label. However, in morphemic lexicon induction, the stems are assigned categories dependending on their parts-of-speech, and the rest depending on their labels. Williams (1981) argues that the final IG acts as the head of the whole word or phrase if it has phrasal scope (Right Hand Rule). In a supporting view, particularly for the Turkish treebank, where IGs are mostly representations of segments separated by derivational morphemes derivational morphology changes lexical types of the items they are attached to.

Since we are dealing with IGs of a word here, the new lexicon consists of stems of words and their IGs instead of inflected words. The morphological structure is only given as morpheme names in the treebank, so we do not have lexical representation of morphemes, but we represent each IG with the first two morpheme names in it. This means we remove the rest of the morphemes which are always inflectional if they do not change the category types at this stage. We also add some IGs in preprocessing that are not based on the IG information in the treebank. This is explained in Section 6.3.2. The first case is the case markers that make a noun phrase an adjunct. Dative marker creates a dative adjunct, locative marker creates a locative adjunct, etc. These

correspond to prepositional phrases (PP) in English, but since they are named after the relevant case marker and the word adjunct in the treebank i.e LOCATIVE.ADJUNCT, INSTRUMENTAL.ADJUNCT, we will refer to them as adjuncts here. We treat these as we treat IGs and assign them categories of their own. Another case is the conditional modifier which in Turkish treebank was treated as inflection. This means it does not have an IG of itself. These are the two exceptions that needed to be handled separately by the algorithm.



But, I did not say anything to you.

Figure 6.6: The dependencies and the final CCG categories assigned to a sentence in the treebank

A simple example is given in Figure 6.6. This example does not contain any multiple IG words. The recursive algorithm applies to the 2 top level elements here, one by one. First it finds the S.MODIFIER and assigns it with category *S/S*, since the sentence it modifies is to the right. After that, it finds the other element at the top level, which is the main verb *söylemedim*. After counting its complements, the algorithm assigns ditransitive category to the verb. The complements receive their categories in a depth-first manner. This means *hiçbir* being *şey*'s determiner is assigned *NP/NP* right after *şey* is assigned NP. Objects are assigned *NP* and subjects are assigned *NP[nom]*. This example has argument scrambling to the right of the verb, as well. We take SOV as the canonical word order and assign categories according to this.

Coordination and extraction cases are handled differently. Examples of different linguistic structures are given in Section 6.4.

6.4 Results

We will focus on the outcome of the morphemic CCG lexicon approach for some specific constructions in the Turkish treebank. Taking morphemes as the smallest representational units has some advantages discussed previously. Some of the solutions of the problems discussed earlier are given in this section.

6.4.1 Passives and Causatives

As discussed in Chapter 2 and Chapter 3, voice is altered through morphology in Turkish. Voice morphemes are separated from the stems by IG boundaries, in a similar way derivational morphemes are. Surface dependency annotation of these construction lead to structures where all the complements of verbs depend on the IG containing voice morphemes. This leads to categories like the ones shown in the example (6.9).

Yerini	pek	değiş	-tirmez	.
place-Poss3sg-Acc	much	change	-Pass-Neg-Aor-A1sg	
NP	*(S\NP)/(S\NP)*	*S*	*(S\NP)\S*	
		——————— <		
		S\NP		
	——————————— >			
	S\NP			
——————————————— <				
S				

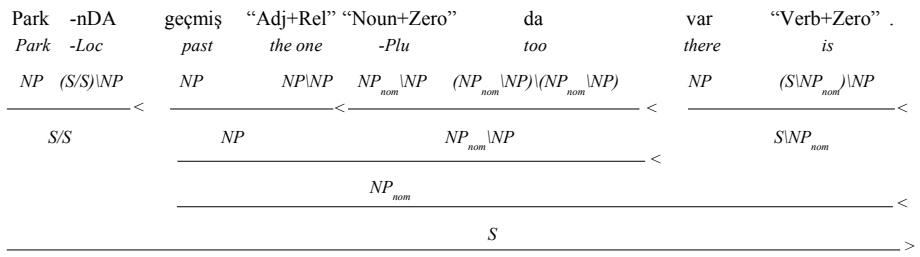
(6.9)

He does not change his place much.

$$\boxed{\begin{array}{l}\langle \textit{“Verb}+\textit{Caus}''_4, (S\backslash NP_1)\backslash S_2, 2, \textit{değiştirmez}_3, -\rangle \\ \langle \textit{pek}_2, (S\backslash NP)/(S\backslash NP)_1, 2, \textit{“Verb+Caus”}_4, -\rangle \\ \langle \textit{“Verb}+\textit{Caus}''_4, (S\backslash NP_1)\backslash S_2, 1, \textit{Yerini}_1, -\rangle \end{array}}$$

6.4.2 PPs or Adjuncts

As discussed earlier, the algorithm treats the case markers that are involved in PPs –or case “adjuncts” as they are annotated in Turkish treebank– as if they are IGs. This allows us to assign correct categories to the elements of the noun phrase the case marker is attached to. Note that in Figure 6.5, the CCG category of the locative case marker is *(S/S)\NP*. The categories assigned in the earlier example (6.7) are as in (6.10).



(6.10)

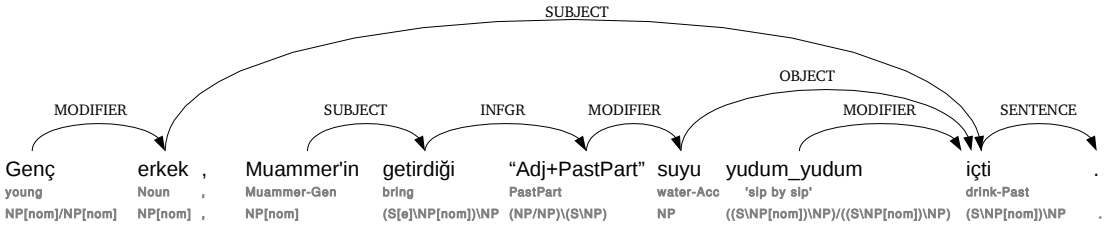
In the park, there are the ones from the past, too.

The category of the adjunct is determined by the number of complements there are in between itself and the head. This is especially practical since we do not have the information to differentiate between verbal adjuncts and VP adjuncts. This means, in this example, we assume that the locative adjunct is not scrambled and because the complements of its head are closer to the head than it is, we assume it is a VP modifier.

6.4.3 Relativisation

Object extraction and adjunct extraction examples are given together with the dependencies in the treebank that are used to create the categories in Figures 6.7 and 6.8.

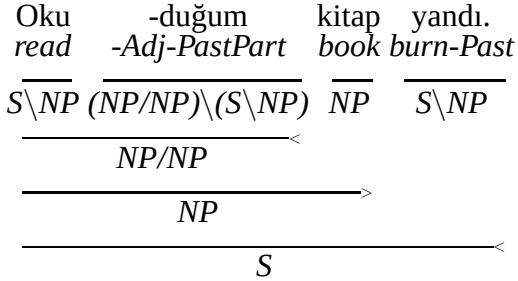
The fact that relative morphemes behave in a similar manner to relative pronouns in English provides the basis for the approach taken here for recovering long-range dependencies in extractions of this type.



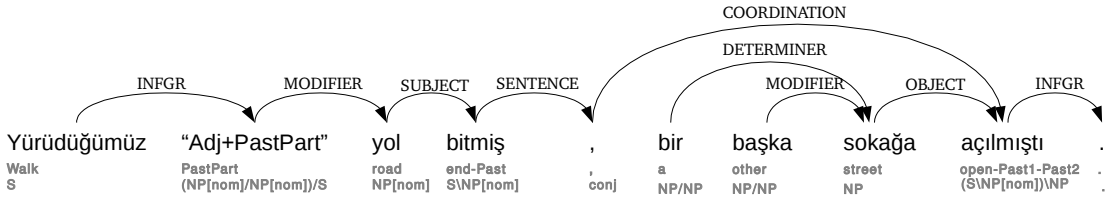
The young man drank the water Muammer brought sip by sip.

Figure 6.7: Categories for object extraction

(6.11)



The book I read burned.



The road we walked had ended and lead to another street.

Figure 6.8: Categories for adjunct extraction

(6.12)

Uyu	-duğum	araba	yandı.
sleep	*-Adj-PastPart*	*car*	*burn-Past*
$\overline{S}$	$\overline{(NP/NP)\backslash S}$	$\overline{NP}$	$\overline{S\backslash NP}$
	$\overline{NP/NP}\,<$		
	$\overline{NP}\,>$		
	$\overline{S}\,<$		

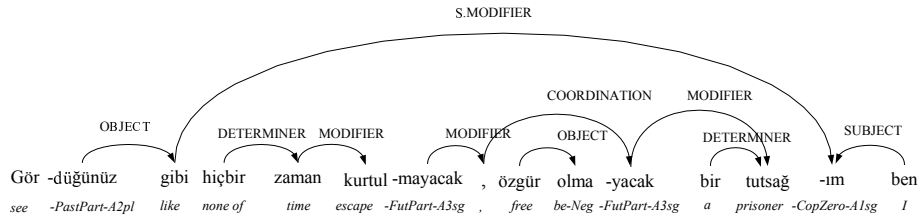
The car I slept in burned.

The relativised verb in (2.14b) is given a transitive verb category with pro-drop, *(S\NP)*, instead of *(NP/NP)\NP* that it would get otherwise. The relative pronoun-equivalent in Turkish, *-dHk+AGR*, now, has its own lexical category. A lexical entry with category *(NP/NP)\(S\NP)* is created and added to the lexicon to give the categories in (6.11) and in (6.12). This solves the problems described in Chapter 2.

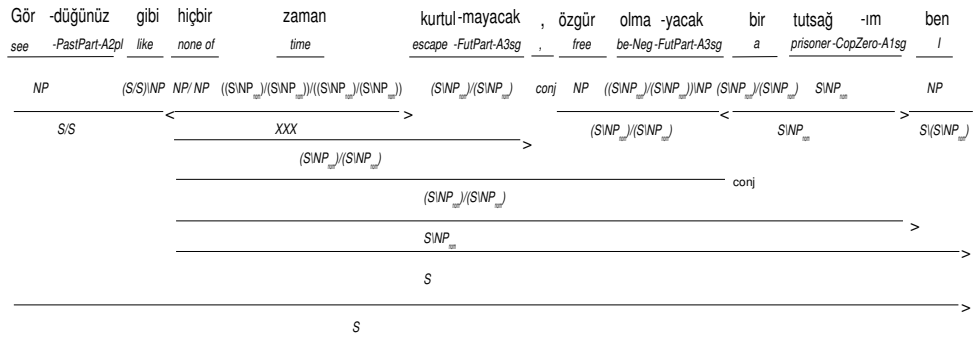
6.4.4 Long-distance dependencies

Long-distance dependencies as well as surface ones are recovered with the help of CCG categories. The dependencies that are derived using surface types of words are not sufficient to predict implicit predicate-argument relations. For instance, relativisation treated as adjectival noun phrase cannot recover predicate-argument relations of the relativised verb. Long-distance dependencies caused by coordination and extraction are also not trivial. (6.13) shows how morphemic lexicon created handles long distance dependencies of extraction and coordination.

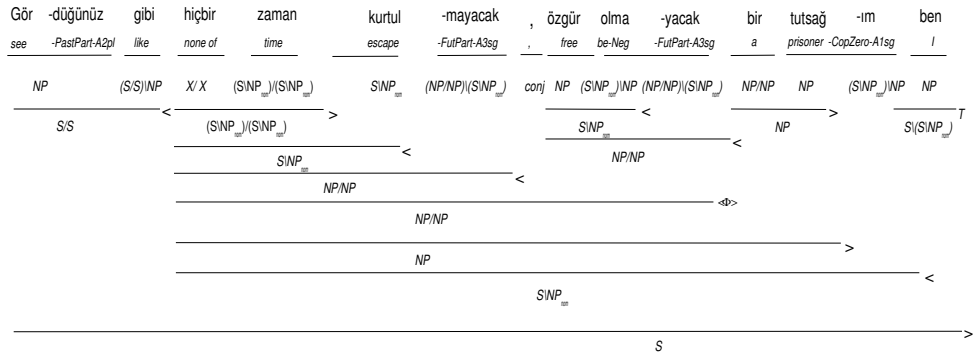
(6.13)



As you see, I am a prisoner who never will escape, be free.



(6.14)



(6.15)

The categories assigned to the words with the lexicon induction process are given to CCG parser to demonstrate the usefulness on a very small scale here. The dependencies that the CCG derivation in (6.15) yields are shown in Figure 6.9. In addition to the surface dependencies in the original dependency structure, long distance dependencies such as the ones that result from coordination are predicted, too. For instance, $\langle kurtulmayacak_7, S\backslash NP[nom]_1, 1, tutsağım_{14}, -\rangle$ and $\langle olmayacak_{11}, (S\backslash NP[nom]_1)\backslash NP_2, 1, tutsağım_{14}, -\rangle$ are not predicted with most dependency parsers, although they are crucial dependencies for semantic interpretation. These dependencies are obtained by giving the categories to the all-parse version of Clark and Curran (2007b)'s CCG parser. The output of the parser is shown in (6.16). Each line represents a dependency between two words. This example is a simplified version of a sentence in the treebank, and the full example is discussed in Chapter 9. The dependencies shown in blue dotted lines in Figure 6.9 are the ones that do not exist in the original dependency structure. These dependencies are recovered through coordination and extraction. When the dependencies shown in the figure are compared with the original treebank dependencies in (6.13), one can see that some dependency arcs point in opposite directions (red dotted arcs). This issue is also explained in Chapter 9.

The red arc in the figure indicates the dependency of the temporal adverb that is assigned to a different head than its correct head. With a trained model, these kinds of errors will be less frequent.

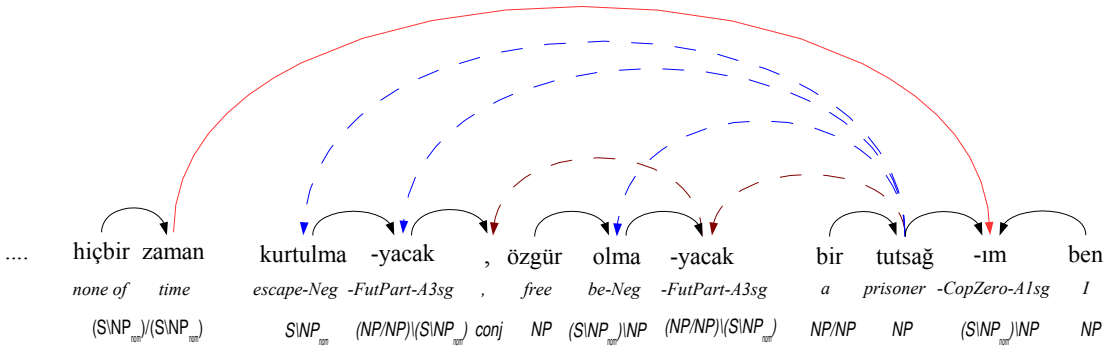


Figure 6.9: The dependencies recovered from the morphemic lexicon categories with a CCG parser.

(6.16)

$$
\begin{aligned}
&\langle -Noun + PastPart_2, NP\backslash S_1, 1, Gördüğünüz_1, - \rangle \\
&\langle gibi_3, (S/S_1)\backslash NP_2, 2, -Noun + PastPart_2, - \rangle \\
&\langle hiçbir_5, (S/S)/(S/S)_1, 2, zaman_6, - \rangle \\
&\langle -Adj + FutPart_8, (NP/NP_1)\backslash(S_2\backslash NP[nom]), 2, kurtulmayacak_7, - \rangle \\
&\langle olamayacak_1 1, (S\backslash NP[nom]_1)\backslash NP_2, 2, özgür_{10}, - \rangle \\
&\langle -Adj + FutPart_{12}, (NP/NP_1)\backslash(S_2\backslash NP[nom]), 2, olmayacak_{11}, - \rangle \\
&\langle ,_9, conj, 1, -Adj + FutPart_{12}, - \rangle \\
&\langle ,_9, conj, 1, -Adj + FutPart_8, - \rangle \\
&\langle bir_1 3, NP/NP_1, 1, tutsağım_{14}, - \rangle \\
&\langle kurtulmayacak_7, S\backslash NP[nom]_1, 1, tutsağım_{14}, - \rangle \\
&\langle olmayacak_{11}, (S\backslash NP[nom]_1)\backslash NP_2, 1, tutsağım_{14}, - \rangle \\
&\langle -Adj + FutPart_{12}, (NP/NP_1)\backslash(S_2\backslash NP[nom]), 1, tutsağım_{14}, - \rangle \\
&\langle -Adj + FutPart_8, (NP/NP_1)\backslash(S_2\backslash NP[nom]), 1, tutsağım_{14}, - \rangle \\
&\langle -Verb + Zero_{15}, (S\backslash NP[nom]_1)\backslash NP_2, 1, ben_{16}, - \rangle \\
&\langle -Verb + Zero_{15}, (S\backslash NP[nom]_1)\backslash NP_2, 2, tutsağım_{14}, - \rangle \\
&\langle zaman_6, S/S_1, 1, -Verb + Zero_{15}, - \rangle \\
&\langle gibi_3, (S/S_1)\backslash NP_2, 1, -Verb + Zero_{15}, - \rangle
\end{aligned}
$$

6.4.5 Copula Sentences and Fragments

There are a lot of sentences in the Turkish treebank without a verbal predicate. Copula sentences are sometimes identified with morphological markers such as *-DHr*, or past tense *-yDH*. When these markers do not exist, or when the sentence is simply a fragment (not a complete sentence) it is difficult to assign categories reliably.

We showed in Section 6.3.2 that the copular sentences missing the additional IG are fixed during preprocessing. The IG containing the zero morpheme is assigned a category X\Y where X is the category of the morpheme, the copular morpheme is attached to and Y is a verb category whose type depends on the number of arguments it has. The main predicate in (6.15) is an example of a copular sentence.

6.4.6 Coordination

In the example for adjunct extraction (Figure 6.8) there is also the issue of coordination. We mentioned in Chapter 5 that S V and VP type coordination is not differentiated in terms of annotation in the treebank. With the help of secondary links we added, we now get the correct categories for *bitmiş* and *açılmış* in the figure. The secondary links that are added are not shown in the figure here. A C.SUBJECT link from *yol* to *açılmıştı* makes sure that the information that the subject is shared among conjuncts is used, and *açılmıştı* takes a full transitive category and not a pro-drop one. This is especially important in cases of object sharing in order to get the correct valency of the verb.

6.4.7 Coverage

We have 27895 unique word-category pairs for 19385 distinct tokens in Chapter 5. The morphemic lexicon has 13016 distinct word-category pairs for 6315 distinct word stems and IG stem names. This is considerable improvement since we have now more than 69K tokens compared to about 54K word tokens for the lexemic lexicon. The average word-category pair frequency goes up from 1.97 to 5.32. Figure 6.10 demostrates the category distribution of one of the most frequent verbs in the treebank – *oku (read)* – with the lexemic approach in Chapter 5 and Figure 6.11 shows the category distribution with the morphemic approach.

There are 450 lexemic category types as compared to 311 morphemic category types. Although the number of category types is less in the morphemic lexicon, we believe that we have a more complete set of morphemic category types than lexemic category types. Figure 6.13 shows the distribution of morphemic categories with the data. The sudden rise that was observed in Figure 5.16 for the lexemic data is not seen in this figure. This means the part of the corpus after 4200 sentences have the same degree of growth as the part after about 1700 sentences where it starts a more linear growth curve of category types.

f	word	cat	f	word	cat
1	okuyan	(NP/NP)\NP	1	okuyup	(S/S)\NP
1	okuduğu	((S\NP[nom])/(S\NP[nom]))/((S\NP[nom])/(S\NP[nom]))	1	ok uyup	(NP/NP)/(NP/NP)
1	okuyorsunuz	S\NP	3	okuyor	S\NP
1	okumuşçasına	((S\NP[nom])/(S\NP[nom]))\NP	1	Okuyorum	S
1	okuyabilir	S	1	okurum	S\NP
1	okudunuz	S\NP	1	okurken	(S/S)\NP
1	okumadım	S	1	okuttum	NP\NP
1	okuyorum	S\NP[nom]	1	Okumayabilir	S/NP
1	okumuştunuz	(S\NP[nom])\NP	1	okudular	S
1	okunacağını	NP\NP[nom]	1	okurduk	S\NP
1	okuyun	S	1	okuyacaklar	S
1	okurdu	S/NP	1	okudu	S\NP
1	okurdu	S\NP	1	okuduk	S
2	okudu	S	1	okudun	S\NP
1	okursa	S/S	1	okumadım	S\NP[nom]
2	okur	S	1	okudum	S\NP
1	okurkenki	NP\NP	1	Okumuyorlar	S
1	okuyabilirim	(NP\NP[nom])/NP	1	okunamayan	NP/NP
1	okumalarını	NP\NP	1	Okudum	S
1	Okuyucunun	NP[nom]	1	okuyor	(S\NP[nom])\NP
1	okumaları	NP\NP	1	okuyayım	S\NP
1	okuyucudan	S/S	1	okudu	(S\NP[nom])\NP
1	okumasını	NP\NP			

Figure 6.10: CCG categories (cat) and frequencies (f) of all the derived and inflected forms of verb *oku (read)* in lexemic lexicon.

f	word	cat
2	oku	S\NP[nom]
4	oku	(S\NP[nom])\NP
20	oku	S
23	oku	S\NP

Figure 6.11: CCG categories (cat) and frequencies (f) of entities of verb *oku (read)* in morphemic lexicon.

	word match%	cat%|word	pair match%
mean	70.1	94	58.5
std. dev.	1.34	1.70	0.92

Figure 6.12: Results on the 10-fold evaluation of the morphemic lexicon

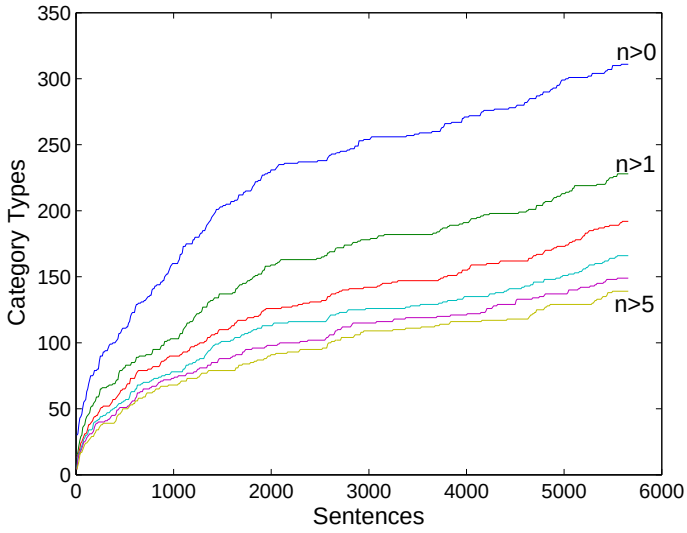


Figure 6.13: The growth of morphemic category types

Figure 6.12 gives the numbers of a 10-fold evaluation process to test the coverage of the lexicon. The method is the same as Section 5.9.1. We take out the 1/10 of sentences for test and take the rest as control set. We test the coverage by checking if the unique lexical entities and unique word-category pairs exist in the control set for each test set. Coverage is higher than the coverage in lexemic lexicon, implying a more complete set of categories and word-category pairs.

We also used an alternative evaluation method. We used CCG categories induced here as features to a dependency parser described in McDonald et al. (2005). The results are discussed in detail in Chapter 8. In summary, the boost in accuracy of the MST parser with the use of CCG categories was encouragingly high. Unlabelled and labelled dependency accuracies were 95.08% and 88.96% respectively. This shows that the use of CCG gold supertags boosts the performance of a dependency parser even when they are used as very simple features. We know that a supertagger is crucial in getting realistic results but we include this information as a means of evaluation of the gold standart CCG categories in the morphemic lexicon induced.

6.4.8 Evaluation by sampling

We perform a small evaluation with 25 sentences similar to the evaluation in Section 5.9.2. We provide the results of both lexicons for comparison purposes. 2 out of 202 words were assigned *NULL* categories by the lexicon induction process. All of the

sentences that were not parsed had at least one category error. On the other hand 3 sentences *with* errors (1 with a minor category feature error) were parsed. The category accuracy of the morphemic lexicon sample set is more than 6 points higher than the accuracy of the sample set from the lexemic lexicon. There were not any unseen categories in the morphemic evaluation set.

A comparison of lexemic and morphemic lexicon evaluations is given in the following table. Morphemic lexicon is seen to outperform the lexemic lexicon in both category accuracy and also parsing results with the CCG lexicon.

lexicon	# sent.	# tokens	accur.	correct	scorrect	cov.	unseen
morphemic	25	202	188	93.07%	17	20	0
lexemic	25	166	144	86.74%	16	14	5

6.5 Conclusion

We have explained why a morphemic approach is crucial in terms of theoretical and computational aspects. We induced a morphemic CCG lexicon from the Turkish treebank which is relatively small when compared to the treebanks for well-studied languages. These results show that especially for languages with complex morphology, generalisation is very important. CCG categories improve the performance of a dependency parser when they are used as features in the statistical model.

We achieved great improvement on word-category pair frequencies which we hope will help with training a supertagger. The supertagger needs a lot of data to be accurate (or as accurate as possible). The average category type per word goes up to 5.32 from 1.97 categories per word. In statistical terms this means we have less data sparseness. We have also shown in Section 6.4, the theoretical and linguistic advantages of the morphemic lexicon.

We used the inflectional groups to specify the lexical boundaries because we did not have a morphological analyser to obtain the actual morphemes. We have also included a few of the crucial inflectional morphemes, such as the ones involved in phrasal adjuncts, namely case markers. We are aiming to have a fully morphemic lexicon that takes into account of bracketing paradoxes and a more detailed study of the status of inflectional morphemes in a CCG lexicon in the future. We also plan to focus more on bracketing paradoxes that involve coordination that is similar to the ones discussed in (Fukushima, 1999) in Turkish.

The work explained in this chapter is important in both getting the correct cat-

egories out of supertagger because it improves the numbers and also predicting the dependencies out of linguistically more appropriate categories.

Chapter 7

Dependency Theory and Parsing

Dependency treebanks have emerged for many low-density languages in the last few years. The reasons for this among many others are that the dependency structures are intuitive, easy to implement and more appropriately flexible for representing phenomena in free word order langauges.

Dependencies are relations that are defined on words or smaller units where the sentences are divided into its elements called heads and arguments, e.g. verbs and objects. Dependency parsing aims to predict these dependency relations between lexical units to retrieve information, mostly in the form of semantic interpretation or syntactic structure.

Parsing is usually considered as the first step of natural language processing. To train supervised statistical parsers, a sample of data annotated with necessary information is required. There are different views on how informative or functional annotation should be. There are different constraints on the design process such as: 1) how intuitive (natural) the representation format is, 2) how easy it is to extract information from it, and 3) how appropriately and unambiguously it represents the phenomena that occur in natural languages.

Using phrase structure trees has been the de facto standard in annotation of linguistic data following the generative tradition. Syntactically annotated linguistic corpora are called treebanks, after tree representations of context-free derivations. Dependency representation, i.e. syntactic annotation of the sentences using the notion of heads, dependents and relations only, is becoming more popular especially with less studied languages. This does not mean that either phrase structure or dependency representation is deficient. They have respective advantages and disadvantages, which is out of the scope of this thesis. In this chapter, we will look at the dependency theory and gram-

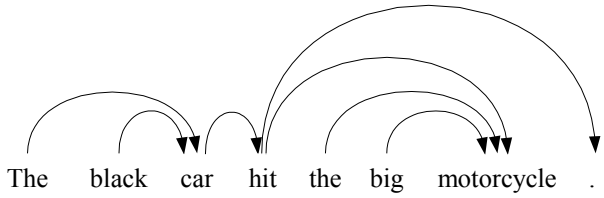


Figure 7.1: The dependency tree

mar from a historical perspective, give a brief summary of the existing dependency corpora designed with some version of the dependency theory, and the state-of-the art results of dependency parsers that use the data in this representation format.

7.1 Dependency Grammar

The concept of dependency grammar is usually attributed to Tesniére (1959) and Hays (1964). Tesniére's goal was to design a grammar that is useful in learning foreign languages. The result is a very intuitive and natural representation of grammatical structure. The dependency theory has since developed, especially with the works of Gross (1964), Gaifman (1965), Robinson (1970), Mel'čuk (1988), Starosta (1988), Hudson (1984), Hudson (1990), Sgall, Hajičová, and Panenová (1986), Barbero et al. (1998), Duchier (2001), Menzel and Schröder (1998), Kruijff (2001).

Dependencies are defined as links between lexical entities (words or morphemes) that connect heads and their dependents. Dependencies may have labels or be unlabelled. A dependency tree is often a directed (sometimes undirected (Sleator and Temperley, 1993)), acyclic (sometimes cyclic (Hudson, 1990)) graph of links that are defined between lexical entities in a sentence. Dependencies are usually represented as trees where the root of the tree is a distinct node.

An example dependency tree is in Figure 7.1. A phrase structure tree for the same sentence is shown in Figure 7.2. The ROOT of this tree is "hit".

Since Tesniére, much work has been done on dependency theory. Among many well known theories of dependency grammar are: Functional Generative Description (Sgall et al., 1969; Sgall, Hajičová, and Panenová, 1986; Petkevič, 1987; Petkevič, 1995), Dependency Unification Grammar (DUG) (Hellwig, 1986; Hellwig, 2003), Meaning Text Theory (Gladkij and Mel'čuk, 1975; Mel'čuk, 1988), Lexicase (Starosta, 1988), and Topological Dependency Grammar (Gerdes and Kahane, 2001). Kruijff (2001) also suggests a type of logic that is specific to dependency grammar, which is

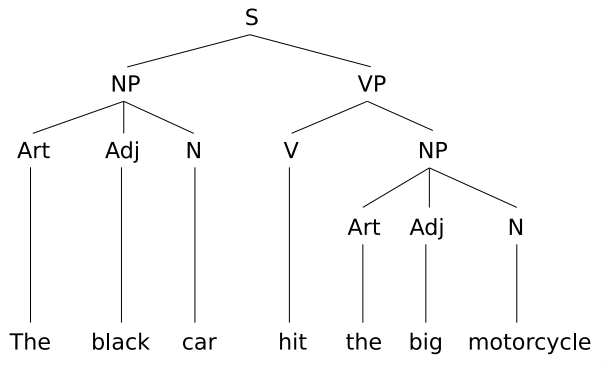


Figure 7.2: The phrase structure tree

called “Dependency Grammar Logic” (DGL). DGL aims transparent semantic interpretation during parsing.

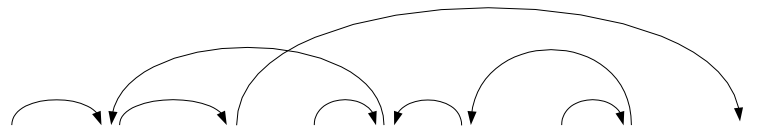
There are many open issues regarding the representation of dependency structure. Hays (1964) and Gaifman (1965) take dependency grammars as special cases of phrase structure grammars whereas Barbero et al. (1998), Menzel and Schröder (1998), Eisner (2000), Samuelsson (2000), Duchier (2001), Gerdes and Kahane (2001), Kruijff (2001) think they are completely different. Certain constrained versions of dependency grammars are, in fact, equivalent to context-free phrase structure grammars. Dependency languages were shown to be exactly context-free languages (Gross, 1964), just before Dependency grammars of single headed, projective dependency structures were shown to be weakly equivalent to context-free grammars by Gaifman (1965).

The dependencies are defined between lexical units. However, there are different views on what lexical units should be. The relations between constituents of phrase structure grammars are represented by relations between the head words of corresponding constituents in dependency grammars. However, there are alternatives to this. Mel’čuk (1988) allows morpheme based dependencies, whereas Tesniére (1959) defines groups of words resembling phrases as units between which the dependencies are defined.

One important issue in dependency theory is projectivity. Projectivity is beyond a representational preference. Most dependency grammars assume projectivity of dependencies. This means when dependencies are drawn on paper, with directed links above the sentence, these links should not cross. Many parsers force this constraint, as well as single-headedness for tractability concerns. For instance Eisner (2000), and Eisner (1996b) force this constraint with defining spans and allowing only words at

the edges of these spans to combine with other words. This, together with some other constraints brings the advantage of cubic time parsing. However, projectivity in dependency grammars does not allow representation of discontinous dependencies, for instance extraposed relative clauses like the one in (7.1).[1] It can be said that projectivity preserves the context freeness of dependency grammars. These structures can easily be translated into context free derivation trees. On the other hand, recognition of non-projective dependency grammmars, when unconstrained is NP-complete (Neuhaus and Bröker, 1997). However, the generative power can be restricted with gap-degreee and well-nestedness constraints. Gap-degree of a dependency structure is determined by the discontinuities it has. A dependency structure is well-nested if no two of its disjoint subtrees interleave. Recently, Kuhlmann and Möhl (2007) defined "regular dependency languages" and showed that applying different combinations of gap-degree and well-nestedness restrictions on non-projectivity in these languages gave a class of mildly context-sensitive grammars. This means it is possible to simulate the dependencies, that can be predicted with mildly-context sensitive grammars by adding extra generative power to otherwise context-free dependency grammars. They show that well-nested structures with a gap-degree of at most 1 are exactly the class of Lexicalised Tree Adjoining Grammars (LTAG).

(7.1)



We see in Figure 7.3 that the notion of non-projectivity is very common throughout languages although distribution of it is rare in any given language. The fact that it is rare does not make it less important because it is this kind of phenomena that defines the automata-theoretic class to which natural languages belong.

[1] This example is from Levy (2005). Although Levy (2005) drew the links between *woman* and *arrived*, and *woman* and *was* as crossing, these links can be drawn in a projective manner. However, Levy (2005) does not take into account the link between the root node, and the head of the sentence, which is necessary for most dependency grammars. This link brings in the non-projectivity to this sentence.

7.2 Dependency Treebanks

7.2.1 Why dependency trees?

Many dependency corpora have been designed and created in the past few years. These are mostly for languages other than English. Dependency representation is preferred when these corpora are designed. This can be argued by the following properties of dependency trees:

1 They are easier to annotate than some other representation types like phrase structure trees (PST). There are fewer tags and labels (only as many as words in a sentence) and no internal nodes.

2 They are much easier to extract information from, because the information is represented more directly. For example, predicate-argument structure can easily be extracted from dependency trees. On the other hand, the heads of phrases in phrase structure trees either need to be declared explicitly or could be found by heuristics and head finding algorithms as in Magerman (1994).

3 Worst-case complexity of a lexicalised PST parser is O(n^5) whereas non-projective dependency parsers without an underlying grammar module run in quadratic time (McDonald et al., 2005). Projective dependency parsers run in O(n^3) time (Eisner, 1996b; McDonald, Crammer, and Pereira, 2005; McDonald et al., 2005). McDonald, Crammer, and Pereira (2005) discuss why Eisner's projective parsing algorithm is slower than their non-projective parsing algorithm.

7.2.2 Dependency Treebanks

Annotation of dependency treebanks have accelerated in the past few years. As a result, there are numerous dependency treebanks that challenge multi-lingual parsing. Figure 7.3 compares dependency corpora of 19 languages that are used as the data sets in CoNLL 2006 and 2007 shared tasks for dependency parsing.[2] The reader is referred to Buchholz and Marsi (2006) and Nivre et al. (2007) for more information about these treebanks and the parsing results. Although, the underlying theory is the same in all of these treebanks there are major differences in the outcome that

[2] Some languages are not included in both tasks. The information in the first and second columns of each set belong to CoNLL 2006 and 2007 training data respectively.

Language	#T		#S		#T/#S		%NST		%NPR		%NPS		IR	
Arabic	54	112	1.5	2.9	37.2	38.3	8.8	-	0.4	0.4	11.2	10.1	Y	-
Basque	-	51	-	3.2	-	38.3	-	-	-	2.9	-	26.2	-	-
Bulgarian	190	-	12.8	-	14.8	-	14.4	-	0.4	-	5.4	-	N	-
Catalan	-	431	-	15	-	28.8	-	-	-	0.1	-	2.9	-	-
Chinese	337	337	57	57	5.9	5.9	0.8	-	0.0	0.0	0.0	0.0	N	-
Czech	1249	432	72.7	25.4	17.2	17.0	14.9	-	1.9	1.9	23.2	23.2	Y	-
Danish	94	-	5.2	-	18.2	-	13.9	-	1.0	-	15.6	-	N	-
Dutch	195	-	13.3	-	14.6	-	11.3	-	5.4	-	36.4	-	N	-
English	-	447	-	18.6	-	24.0	-	-	-	0.3	-	6.7	-	-
German	700	-	39.2	-	17.8	-	11.5	-	2.3	-	27.8	-	N	-
Greek	-	65	-	2.7	-	24.2	-	-	-	1.1	-	20.3	-	-
Hungarian	-	132	-	6.0	-	21.8	-	-	-	2.9	-	26.4	-	-
Italian	-	71	-	3.1	-	22.9	-	-	-	0.5	-	7.4	-	-
Japanese	151	-	17	-	8.9	-	11.6	-	1.1	-	5.3	-	N	-
Portuguese	207	-	9.1	-	22.8	-	14.2	-	1.3	-	18.9	-	Y	-
Slovene	29	-	1.5	-	18.7	-	17.3	-	1.9	-	22.2	-	Y	-
Spanish	89	-	3.3	-	27	-	12.6	-	0.1	-	1.7	-	N	-
Swedish	91	-	11	-	17.3	-	11.0	-	1.0	-	9.8	-	N	-
Turkish	58	65	5	5.6	11.5	11.6	33.1	-	1.5	5.5	11.6	33.3	N	-

Figure 7.3: Treebank information

#T = number of tokens * 1000, #S = number of sentences * 1000, #T/#S = tokens per sentence, %NST = % of non-scoring tokens, %NPR = % of non-projective relations, %NPS = % of non-projective sentences, IR = has informative root labels.[3]

originate from the questions like 1) how much information is needed to put in the dependency trees, 2) how strongly interlaced the different modules such as morphology syntax are in a language. Czech treebank (Böhmová et al., 2003) is a good example of a well-designed dependency treebank with 3 different levels of representation, namely, morphological, grammatical and tecto-grammatical layers as defined in Sgall, Hajičová, and Panenová (1986). In addition to the two obvious levels, the third level –tectogrammatical level– bears information such as valency of verbs, anaphora resolution, etc. The design of Turkish Treebank is a good example of an answer to the second question. Morphology-syntax interface makes word-based dependencies inappropriate for Turkish. Therefore, the dependencies are defined between inflectional groups in the Turkish treebank. These are two arguments among many on why it is very important to make a good feasibility study when designing a dependency treebank as different features of languages require different handling in terms of decisions such as word versus morphemic representation of the lexicon.

7.3 Dependency Parsing

Statistical or data-driven parsing methods have gained more focus with the continuous introduction of new linguistic data. Parsing was more focused on training and parsing with phrase structure trees and specifically English language because the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) was the main source available for a long time. With the introduction of treebanks of different languages, it is now possible to explore the bounds of multi-lingual parsing.

We give a review of Dependency parsing in this section. Most of these parsers assume dependency structures as single-headed, projective structures, and they create surface syntactic dependencies which is less expressive than what some dependency grammars express. Deep linguistic analysis does not get enough attention when focused on quantitative improvement. Fortunately, there are also studies that focus on recovering deep dependencies as well as surface ones such as Clark, Hockenmaier, and Steedman (2002), Hockenmaier (2003a), Levy and Manning (2004), Riezler et al. (2002),Cahill et al. (2008) and so on.

7.3.1 Collins' Czech Parser

Some of the early efforts of data-driven dependency parsing were focused on translating dependency structures to phrase structure trees because phrase structure parsers already existed. Translating dependency structures with crossing dependencies is not trivial if the surface order needs to be preserved. Thus, the incompatible translation of dependency structures to phrase structure trees results in varying degrees of loss of information.

Collins et al. (1999) translate the Prague Treebank dependency trees to phrase structure trees in the flattest way possible and name the internal nodes after part-of-speech tags of the head words of nodes. They use Model 2 of Collins (1999) and evaluate the attachment score on the dependencies extracted from the resulting phrase structure trees of the parser. However, crossing dependencies cannot be trivially translated into phrase structure trees with the surface order of the words unchanged (Çakıcı and Baldridge, 2006). But, Collins et al. (1999) do not mention non-projective (crossing) dependencies, therefore, it is not clear what they did for these cases in the Czech treebank.

7.3.2 Deterministic Parsers

Deterministic parsers need only one pass over the data. thus, they are potentially time-efficient as long as the constant factors are small. Nivre (2003) gives a deterministic parsing algorithm for projective dependency trees. He uses a probabilistic framework to guess the parser actions. The parser makes only one pass on the input and assigns a dependency structure at once, without any redundancy or backtracking thus the algorithm runs in linear time (Nivre, 2003). Nivre, Hall, and Nilsson (2004) uses memory-based learning (Daelemans, 1999), and achieves better accuracy. A similar approach is followed by Yamada and Matsumoto (2003). They look at the pairs of words to decide whether they should be linked or not. *right* decides that the dependency is right-to-left and *left* decides it is left-to-right. They use support vector machines to decide on the next action which is one of 3 actions: *shift*, *left*, and *right*. 3 different SVM classifiers trained for each distinct action were used to decide which action to take next based on the two words in question. The parsing algorithm runs in $O(n^2)$. Kudo and Matsumoto (2000), and Kudo and Matsumoto (2002) use support vector machines to parse Japanese. MaltParser of Nivre et al. (2007) provides results for multilingual dependency parsing using both memory-based learning and SVM classifiers. Oflazer (2003) uses an extended finite state automaton to parse Turkish dependency structures.

7.3.3 Eisner's Dependency Parsers

One of the most influential statistical systems for parsing dependency structures directly is due to Eisner (1996b). He proposes 3 different models for direct dependency parsing. He evaluates them on dependencies derived from the Penn Treebank. His generative model which is similar to the Model 1 in Collins (1999) achieves the best performance on a 400 sentence test sample from WSJ. Eisner's parser is a projective parser thus it cannot predict crossing dependencies.

Eisner's parser relies on structures called spans. He effectively designs an $O(n^3)$ algorithm for parsing projective dependency structures, with the use of these spans. Classical lexicalised parsers run in $O(n^5)$ time because, the head of a phrase can be anywhere in the phrase. The same holds for the head of the dependent, thus n times n possibilities are also stored in the chart (Eisner, 2000). However, in Eisner's parser, the head of phrase is only allowed at two ends of the spans. Among three models described in Eisner (1996b), generative one, that generates left and right dependents recursively has the best performance. The generative probabilities in the generative

model are calculated by the given formula.

$$Pr(\text{words, tags, links}) = \prod_{1\leq i\leq n}\Big(\prod_{c=-(1+size(\text{lwl}(i))),c\neq 0}^{1+size(\text{rwl}(i))} Pr(tword(kid_c(i))|tag\big(kid_{c-1}(i)\big),tword(i)\Big) \tag{7.2}$$

Here, *size*lwl*(i)* and *size*rwlemph(i) give the number of left and right dependents of the word *i* respectively. *tword* denotes tagged word.

Eisner (2000) defines bilexical grammars based on this formalism, and shows that cubic time parsing is possible with lexicalised dependency grammars. He defines two automata lw_i and rw_i for the left dependents and right dependents of a head word w_i, and generates these. This is similar in ways to Collins' generation of left and right daughters independently.

Sleator and Temperley (1993) gives a similar algorithm to Eisner (1996b) independenctly (Eisner, 2000). Their dependency structures, are called *linkages* instead of dependency trees, and are made up of labelled, undirected links. However, they use the same decomposition, which Eisner (1996b) calls spans (Eisner, 2000).

7.3.4 Nivre's Parsers

Nivre (2003) defines a deterministic parser. Nivre parses a small Swedish dependency corpus with this parser. The parser runs in a greedy deterministic mode that goes over the input string once, assigning dependencies to each word. It is deterministic in the sense that only one analysis is available for a given sentence. Parser actions consist of : **Shift**, **Reduce**, **Left-Arc**, and **Right-Arc**. The input string is accepted if the dependency structure the parser suggests is well-formed when the final state is reached. Well-formedness conditions are: acyclicity, projectivity and single-headedness together with connectivity.

Nivre, Hall, and Nilsson (2004) use the same parser together with Memory based learning and achieve better performance than the original system that uses hand-written grammar rules. Nivre and Scholz (2004) apply deterministic parsing to English text.

Recently, mostly with introduction of multi-lingual dependency data, non-projective parsing methods became more popular than before. Parsing non-projective dependency graphs were shown to be NP-complete. However, this does not stop the attempts to create approximations for non-projective parsing. Nivre and Nilsson (2005) do pseudo-projective parsing which is first defined in Nasr (1998) and later implemented by Kahane, Nasr, and Rambow (1998). Non-projective dependencies are encoded in the

projective link labels to be decoded after projective parsing to create non-projective structures by post-processing.

7.3.5 Graph based algorithms (McDonald *et al.*'s Parsers)

McDonald et al. (2005) apply graph spanning algorithms to dependency parsing. They formalise dependency parsing as the problem of finding a maximum spanning tree in a directed graph. Chu-Liu-Edmonds algorithm is used to parse non-projective dependency structures. This algorithm has two major advantages: it runs in $O(n^2)$ time and it handles non-projective dependencies directly. McDonald et al. (2005) show that this algorithm significantly improves performance on dependency parsing for Czech, especially on sentences that contain at least one crossed dependency. McDonald, Crammer, and Pereira (2005) report results for strictly projective parsing. For this, they use Eisner's algorithm mentioned in the previous section.

Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) is used to determine the weights of dependency links as part of this computation. Variations of this parser have been used in CoNLL 2006 shared task and received the highest ranking among the participants averaged over the results of all of the 13 languages (Buchholz and Marsi, 2006). The use of morphological features are shown to improve the overall performance of the multilingual system. This parser runs in two stages. The first stage assigns the unlabelled dependencies and the second stage decides on the labels afterwards.

When no linguistic or global constraints are applied, this parser may yield impossible dependency sequences such as assigning two subjects to a verb (Riedel, Çakıcı, and Meza-Ruiz, 2006). This is because McDonald et al. (2005) does not have an underlying grammar formalism and relies on first order edge decisions. However, they explain that second-order non-projective dependency parsing is NP-complete. McDonald and Pereira (2006) present an approximation algorithm to solve this problem, and achieve state-of-the-art performance.

7.3.6 Deep dependency parsers

Since most dependency parsers are projective, there is a misconception that dependency grammars are projective and thus context-free. Various dependency theories allow non-projective structures as well as multiple-headed words (Hudson, 1984), (Mel'čuk, 1988).

Lombardo and Lezmo (2000) follow a "GPSG" style approach in handling long-distance dependencies in dependency grammar. Thet introduce empty elements (gaps) for handling long-distance dependencies resulting from extraction, coordination and control. They give a formal theory of dependency syntax with non-lexical units, however, they did not implement a parser, so they do not provide experimental results.

CCG parsers inherently predict long-distance dependencies. Generative models of Hockenmaier (2003a) and discriminative models of Clark, Hockenmaier, and Steedman (2002), Clark and Curran (2007b) give state-of-the-art results in recovery of deep linguistic information. We will give a more detailed review of CCG parsers in Chapter 9.

Levy and Manning (2004) induce long-distance dependencies from a context free framework. Dienes and Dubey (2003) do deep syntactic processing by post-processing the output of CFG parsers. Cahill et al. (2008) also induces a more expressive LFG grammar from the Penn treebank with similar techniques.

7.4 Discussion

There is growing body of work on creating new treebanks for different languages. Requirements for the design of these treebanks are at least as diverse as these natural languages themselves. For instance, some languages have a much more strong morphological component or freer word order than others. There are challenges both for dependency parsing and for different dependency theories. For instance, modelling morpho-syntactic relations in dependency representation for morphologically complex languages. Representing "deep" dependencies needs more attention as well. Although these constitute a fraction of all the phenomena in natural languages, they are important for semantic interpretation.

This chapter has reviewed dependency grammar theories together with recent advances in statistical dependency parsing for different languages. Some current challenges in building dependency treebanks and dependency parsing have also been discussed. Dependency theory and practical applications of dependency representations have advantages and disadvantages. The fact that dependency parsing is easy to adapt to new languages, and is well-adapted to representing free word-order, makes it the preferred representation for many new linguistic corpora. Dependency parsing is also developing in the direction of multi-lingual parsing where a single system is required to be successful with different languages. This research may bring us closer to under-

standing the universals of language, and thus to building better NLP systems.

Chapter 8

Turkish Dependency Parsing

In this chapter, parsing models for Turkish using the METU-Sabancı dependency treebank are described. Different representation schemes for dependency structures are investigated and different styles of projective and non-projective parsing models are compared. We show, in particular, that representing distinctions about derivational morphology and case marking provides large improvements in the accuracy of recovering word-word and IG-IG surface dependencies.

We aim to give a comparison of direct versus indirect approaches to dependency parsing. By indirect, we refer to approaches that translate dependency structures to a different representation format –usually phrase structure trees– for which the parsing models used to be more popular presumably as a result of the dominance of Penn Treebank. We also explore the effectiveness of projective versus non-projective algorithms in direct dependency parsing. We use the principles of Collins' parser which is a generative parser and Maximum Spanning Tree parser of McDonald et al. (2005) which is a discriminative parser.

Section 8.1 gives an overview of the relevant work on Turkish dependency parsing. Section 8.2 outlines our procedure for mapping dependency structures to tree structures as input for training the Collins parser, and discusses different POS tag sets with varying levels of sensitivity to Turkish morphology. Section 8.4 introduces the parsing models used, and Section 8.6 reports and discusses the performance of the various configurations tested. In particular, Section 8.6.3 reports the results of experiments testing the effect of using CCG supertags with MST parser. and Section 8.6.4 reports the results of experiments in which the inflectional groups in the treebank are directly represented in the dependency structures.

8.1 Literature Review

Since the creation of the Turkish treebank (Atalay, Oflazer, and Say, 2003; Oflazer et al., 2003) several parsers were created to parse it. Turkish dependency structures are non-projective, which is expected given its free word order nature. Parsers that parse projective and non-projective structures are implemented by various authors. A review of dependency parsers that use Turkish treebank is given in the following sections.

8.1.1 Eryiğit and Oflazer (2006)

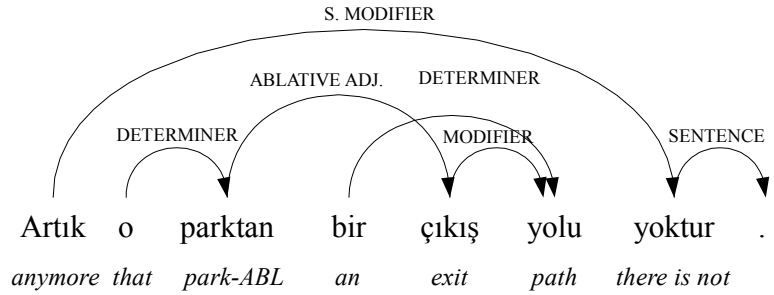
Eryiğit and Oflazer (2006) present a statistical dependency parser for Turkish evaluated on a subset of the Turkish dependency corpus. They use a backward beam search algorithm of Sekine, Uchimoto, and Isahara (2000) with a probabilistic classifier. Since the parser cannot handle non-projective or leftward dependencies, they take only a subset of the Turkish treebank that consists only of the sentences with non-crossing rightward links.

Eryiğit and Oflazer (2006) argue that lexicalisation does not have any effect on the training process. However, we do not think it is trivial to argue this since there is not enough information for lexicalisation in an IG-based system like theirs. IGs in Turkish treebank are only defined as components of words and they consist of a string of morphological tag names. Therefore, an IG does not have a corresponding surface form.

Their best-performing model recovers 72.3% of the rightward dependencies. This is 2.7 percentage points higher than their highest baseline and 12.4 points higher than the baseline that is achieved by linking every word to its rightward neighbour. They give a word-based accuracy of 81.3% together with results of a “pure word based” model. They describe word-based dependency as correct if the current word is dependent on any of the IGs of the target word. They also say that 90% of the words had their dependents in the range of next 3 words and 4 IGs in case of IG based dependencies. In a word-based system the above claim about the word-based dependencies would be similar to saying that a word is dependent on the head in the n^{th} position or n+1 or n+2 which makes it very likely that one of these are the correct head especially in an only rightward linking system. An assumption about the IGs being independent entities in the dependency structure is initially made by Eryiğit and Oflazer (2006). Therefore, it could be argued that calculating word-based dependency accuracy in this way is flawed. Also, the word boundaries are not known when parsing (because of the

IG independence assumptions), thus we cannot predict whether any given two IGs are in the same word or not.

If we look at the statistics of the languages given in Section 7.3, we see that the projective links constitute most of the dependencies in Turkish data. The percentage of sentences with at least one non-projective link is 33.6% reflecting the fact that the non-projective dependencies these sentences contain constitute 5.5% of all dependencies.[1] Looking at the numbers for the other data sets in 2007 shared task in Figure 7.3, it is obvious that Turkish, in fact, has the greatest number of non-projective links. Thus, excluding such a vital part of data compromises both the coverage and interpretability of the results. Figure 8.1 shows an example non-projective dependency structure in the Turkish treebank.



E: There is no exit from that park, anymore.

Figure 8.1: Crossing dependencies in Turkish treebank

Our experiments explained in this chapter are different from Eryiğit and Oflazer (2006) in four ways. First, we focus on word-word dependencies *as well as* IG-IG dependencies. Second, we parse *all* sentences rather than just projective ones with only rightward links. Third, we provide results for parsing with automatic POS tagging as well as for gold-standard POS tags. Finally, we provide results for labeled as well as unlabeled dependencies.

8.1.2 Eryiğit, Nivre and Oflazer (2006)

Eryiğit, Nivre, and Oflazer (2006) use Nivre's parsing algorithm (Nivre, 2006; Nivre, 2003) with support vector machines to predict the next action of the parser. They incre-

[1] These numbers are different than the ones given in CoNLL 2006, which are 11.6 and 1.5 respectively. Given that a corrected version of the treebank was used in CoNLL 2007, we take the more recent numbers as correct. However, in both versions punctuation marks are connected to a top-level root node which is artifically created. This causes extra non-projective links in the data-sets for non-sentence-final punctuation.

mentally add morphological and lexical information and show that lexicalisation and the use of morphology improves the parsing performance, contrary to what Eryiğit and Oflazer (2006) suggest. One other finding is that IG-based dependencies perform better than word-based. Eryiğit, Nivre, and Oflazer (2006) do not follow Eryiğit and Oflazer (2006) in calculating word-based dependencies and their findings are compatible with the results we provide in this chapter.

Results for word-based dependencies in Eryiğit, Nivre, and Oflazer (2006) are surprisingly low (67.2% unlexicalised and 70.2% lexicalised) compared to the "pure word-based" score in Eryiğit and Oflazer (2006) (77.7%). This may be because even though they use the same configuration of labels and tags as Eryiğit and Oflazer (2006), the evaluation is not compatible because Eryiğit and Oflazer (2006) train only on projective and rightward linking sentences and evaluate on the same set while Eryiğit, Nivre, and Oflazer (2006) train on projective sentences and use the entire treebank for evaluation. Neither of these parsers predict the non-projective dependencies. This is similar to the approach we chose when evaluating Collins' parser. We trained on sentences with different surface order caused by projectivisation and evaluated on the whole treebank. Since Collins' parser is a phrase structure parser it does not predict crossing dependencies, either.

Results of Eryiğit, Nivre, and Oflazer (2006) also show that word-based dependency results are worse than IG-based dependencies and that lexicalisation improves the accuracy. However, the effect of lexicalisation is not uniform over the set of syntactic categories. They also parse the data set of all projective rightward linking dependency subset and observe improvement for both the lexicalised 78.3% unlabelled and unlexicalised (76.1% unlabelled) systems, lexicalised being a few percentage points more in all results.

Their results are obtained by ten fold cross-validation on the data. They also give results on the CoNLL 2006 Shared task on multilingual dependency parsing data set and report 75.82% unlabelled and 65.68% labelled attachment scores.

8.1.3 CoNLL 2006 Shared Task on Dependency Parsing

The data set used in CoNLL 2006 shared task (Buchholz and Marsi, 2006) for Turkish treebank is a split set of roughly 5000 training sentences and 600 test sentences. While converting the treebank to CoNLL format, several rather unconventional decisions were made about the way to translate the dependencies into the target format.

The punctuation is not evaluated in CoNLL but Turkish dependency treebank uses some commas as conjunctions. This brings in a problem of incompatibility among coordinations with comma and coordinations with conjunctions.

Coordination with commas were translated as shown in Figure 8.2 and Figure 8.3. Commas which were heads of conjunct heads in the original version were made to depend on these heads and conjunct heads were linked directly from right to left.

We believe, the way one conjunct was linked to the next one to the right proves to be problematic, since losing the constituency information might change the probability distribution and the consistency in the treebank. The constituency information is lost when the label of the first conjunct is changed. This is illustrated in Figure 8.3 and Figure 8.4. Therefore, in terms of statistical parsing, the numbers for different types of coordination changes, will make it difficult to for any decoder to differentiate between these. Also inconsistency between sentences with punctuation versus sentences with clitics as conjunctions will arise because the same configuration is not applied to conjunctions such as *and* and *but*.

1	Hülya	Hülya	Noun	Prop	A3sg|Pnon|Nom	13	SUBJECT	_	_
2	,	,	Punc	Punc	_	13	PUNC	_	_
3	Onu	o	Pron	PersP	A3sg|Pnon|Acc	5	OBJECT	_	_
4	seri	seri	Adj	Adj	_	5	MODIFIER	_	_
5	düşün	düşün	Verb	Verb	Pos|Imp|A2sg	8	COORDINATION	_	_
6	,	,	Punc	Punc	_	8	PUNC	_	_
7	hadi	hadi	Interj	Interj	_	8	VOCATIVE	_	_
8	gel	gel	Verb	Verb	Pos|Imp|A2sg	10	OBJECT	_	_
9	,	,	Punc	Punc	_	10	PUNC	_	_
10	dedi	de	Verb	Verb	Pos|Past|A3sg	11	SENTENCE	_	_
11	ve	ve	Conj	Conj	_	13	COORDINATION	_	_
12	kolumdan	kol	Noun	Noun	A3sg|P1sg|Abl	13	OBJECT	_	_
13	çekti	çek	Verb	Verb	Pos|Past|A3sg	0	ROOT	_	_
14	.	.	Punc	Punc	_	13	PUNC	_	_

Hülya , Onu seri düşün , hadi , gel , dedi ve kolumdan çekti .
Hulya , it-Acc quick think , let's , come , say-Past and arm-Abl pull .

E: Hulya said "think that quickly, come on", and she pulled my arm.

Figure 8.2: A sentence from CoNLL test set

One third of Turkish data set consists of non-scoring tokens. This is mostly caused by the representation style of IG-based dependencies. IGs have their own lexical representation, independent from the word they belong to. However, some LEMMA and FORM fields are changed into "_" character to differentiate between an internal IG and

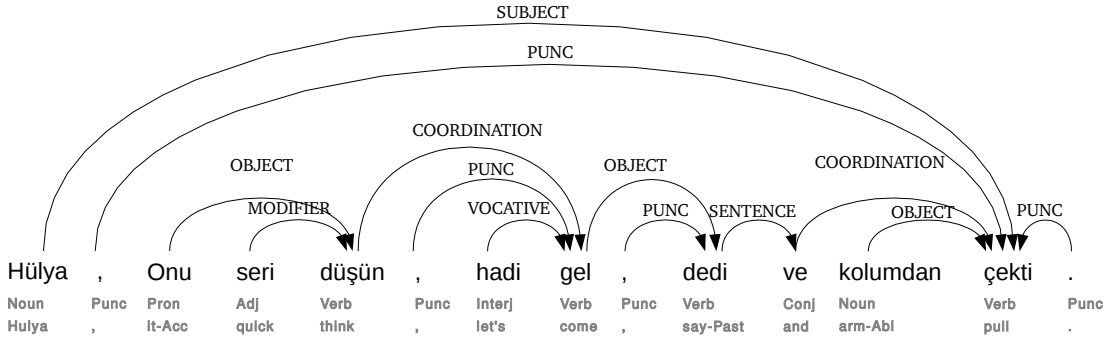


Figure 8.3: Graphical representation of the sentence in Figure 8.2

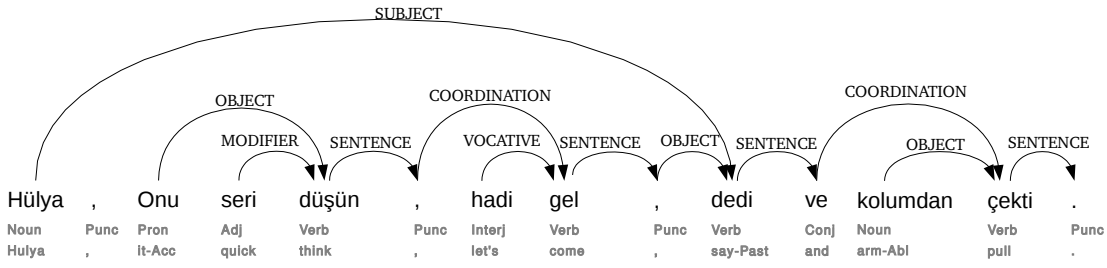


Figure 8.4: The original dependency structure of the sentence

a final IG. Internal IGs always depend on the next lexical element. The dependency emanating from the final IG of a word goes to the head IG of that word. An example of this is in (8.1). Word-internal dependencies were treated like punctuation, and were excluded from scoring. However, we believe, and show in Section 6.4.7 that this representation causes loss of information and may hurt the performance of the parsers unnecessarily.

(8.1)

1	-	-	Punc	Punc	_	9	PUNC	_ _
2	Galiba	galiba	Adv	Adv	_	9	S.MODIFIER	_ _
3	siz	siz	Pron	PersP	A2pl|Pnon|Nom	8	SUBJECT	_ _
4	insanları	insan	Noun	Noun	A3pl|Pnon|Acc	6	OBJECT	_ _
5	_	yönlen	Verb	Verb	_	6	DERIV	_ _
6	_	_	Verb	Verb	Caus|Pos	7	DERIV	_ _
7	yönlendiren	_	Adj	APresPart	_	8	MODIFIER	_ _
8	_	takım	Noun	Noun	A3sg|P3sg|Abl	9	DERIV	_ _
9	takımındansınız	_	Verb	Verb	Zero|Pres|A2pl	0	ROOT	_ _
10	.	.	Punc	Punc	_	9	PUNC	_ _

Turkish data is the most difficult to parse looking at the average parse score among all languages. This is attributed to 8 different genres in Turkish data and high percentage of unseen LEMMA and FORM values in the Turkish test set by Buchholz and Marsi (2006). One of the best average scores was by McDonald, Lerman, and Pereira (2006). Their results show that the use of morphological features for treebanks

	LA	UA	data
Eryiğit and Oflazer (2006)	-	77.7	part
Eryiğit et al. (2006)	62.0	70.7	all
Çakıcı and Baldridge(2006)-MST	72.3	84.9	all
Çakıcı and Baldridge (2006)-MST	72.6	85.6	part

Table 8.1: Word-based Turkish parsing

	LA	UA	data
Eryiğit and Oflazer (2006)	-	73.5	part
Eryiğit et al. (2006)	64.9	73.8	all
Eryiğit et al. (2006)	68.9	78.3	part
CoNLL 06 (by Nivre et al.)	65.7	75.8	conll
CoNLL 07 (by Titov et al.)	79.8	86.2	conll

Table 8.2: IG-based Turkish parsing

that have this information improves the average performance by nearly 1 point in both the unlabelled and the labelled scores. McDonald (2006)[pp.100] reports that Turkish parsing scores lose about 2 points for unlabelled and 2.6 points for labelled accuracy when morphological features are not included. Turkish is the most affected with inclusion of these features among languages.

8.1.4 CoNNL 2007 Shared Task on Dependency Parsing

A comparison of the parsers for Turkish is given in Tables 8.1 and 8.2. Only the numbers that are comparable with each other are included, such as the “pure word based” results for word based dependency recovery in Eryiğit and Oflazer (2006). Also note that all punctuation is included in evaluation in Eryiğit and Oflazer (2006) and CoNLL 2007 whereas trivial punctuation (marking the root node) are not included in results of Çakıcı and Baldridge (2006) and punctuation was shown to make a difference of 8 points in Nivre et al. (2007). The best score achieved by Nivre et al. (2006) in CoNLL 2006 and by Titov and Henderson (2007) in CoNLL 2007 shared tasks.

The quantitative difference between CoNNL 2006 shared task and 2007 shared task Turkish parsing results is significant. It is seen that CoNLL 2007 results are

considerably better. This could be because of several reasons. First of all, punctuation was included in CoNLL 2007 evaluation. Secondly, training set grew in size since a new development set was annotated. Also this development set is a different set and changes the genre ratios in training and test. We believe that the biggest contribution to this improvement is the fact that a subset of the corrections we explained in this thesis were applied to the Turkish treebank in between 2006 and 2007 shared tasks. With or without punctuation 6 percentage points of improvement was observed in the overall result for Turkish (Nivre et al., 2007).

8.2 Tree-Based Models

In order to use a phrase-structure parser with the treebank, it is necessary to create trees out of the annotated dependency structures. We chose a method similar to the one suggested by Collins et al. (1999) for Prague Dependency treebank for Czech. In this section, we describe a few simple strategies that dramatically improve the performance of phrase-structure parsers for dependency recovery.

8.2.1 Mapping Dependencies to Trees

Collins et al. (1999) outline three choices when creating trees from dependencies: (a) branching factor, (b) choice of non-terminal labels, and (c) the set of POS tags to be used. A fourth choice, which they do not mention, is how to handle non-projective dependencies.

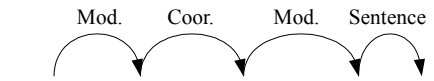
We create the flattest possible trees and use the POS tags to create non-terminal labels as explained in Collins et al. (1999). Tags are derived from the morphological analyses in the treebank; there are 15 tag types in the most basic tag set. Dubey and Keller (2003) also use flat trees because of the nature of the German treebank and find that sister-head dependencies are much more useful than head-head dependencies that are dominant in Collins parsers.

344 of the 5620 sentences have at least one crossed dependency. This makes the mapping process non-trivial. There are ways of faithfully representing crossed dependencies using mechanisms such as traces. However, this would involve considerable effort. Our goal is to compare the *straightforward* application of a phrase-structure approach to the MST model, which handles crossed dependencies natively. Under our approach for mapping dependency structures to phrase structure trees, dependents

appear immediately adjacent to their heads. This gives the correct head-daughter dependencies but changes the word order from that in the original sentence. Collins' parser is seen as a phrase structure baseline for Turkish dependency parsing.

8.2.1.1 Coordination

Coordination is represented as a sequence of head labels and the COORDINATION label in the Turkish Treebank. If the modifier coordination is considered in (8.2), the first conjunct is linked to the conjunctive word *ve* with a MODIFIER dependency link and then a COORDINATION link goes from *ve* to the second conjunct, and finally, the last conjunct links to the head of the sentence with a MODIFIER link. This translates to a phrase structure model as in Figure 8.5 with the translation method used here.



(8.2) Koşarak ve zıplayarak geldi .
Running and jumping come+Past

He came running and jumping.

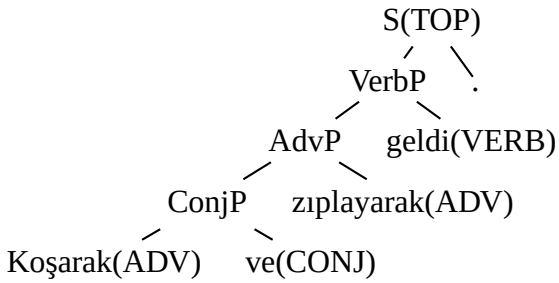
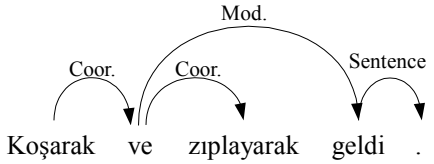


Figure 8.5: The parse tree for the sentence in (8.2).

A more phrase-structure friendly way of representing coordination dependencies is used in some other treebanks. This can simly expained as making the conjunctive *ve* the head of coordinating phrases and then linking it to *geldi*, as shown in (8.3). This would ensure that coordinating elements are on the same level and would form a tree as in Figure 8.6. For now, we choose to make the minimal changes to the baseline trees in this study, but the impact of such tree-transformations could be the focus of future experiments.



(8.3) Koşarak ve zıplayarak geldi .

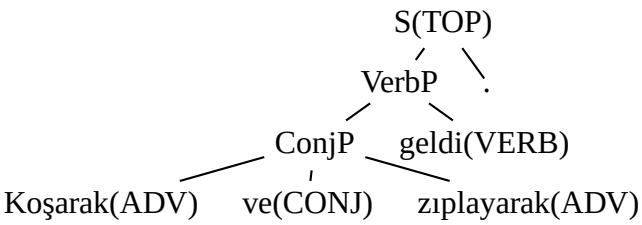


Figure 8.6: The parse tree for the sentence in (8.3).

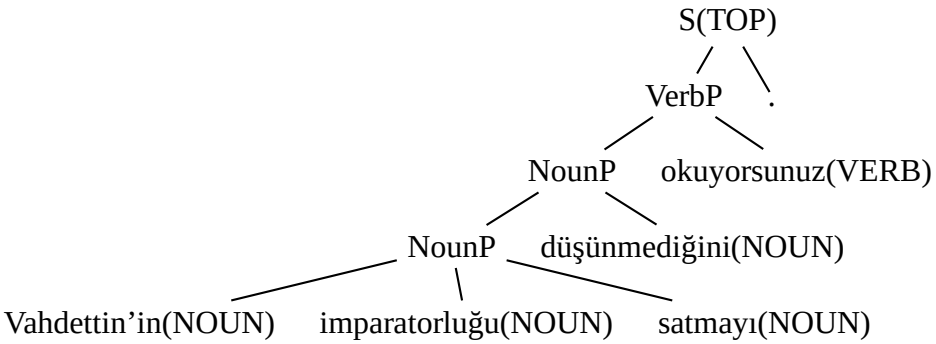


Figure 8.7: The basic mapping does not distinguish NPs from the subordinate clauses.

8.2.1.2 Punctuation

Apart from the sentence final punctuation that the sentence head is dependent on, all other punctuation is ignored by the translation process, unless a dependency link emanates from them (e.g commas in coordination, sentential complementation).

In our experiments, we have excluded punctuation without dependency links from all scoring.

8.2.2 Modifications to the Baseline Trees

The morphological structure of each word is represented in inflectional groups (IGs) in the treebank. The POS tags we use for the parser are derived from these inflectional groups, which we use to create four distinct tag sets.

Our basic tag set uses only the POS tags in a word's last IG. For example the POS tag is *Verb* for (8.4), and *Noun* for (8.5).

(8.4) *istemiyorum "I don't want..."*
IG='[(1,"iste+Verb+Neg+Prog1+A1sg")]'

(8.5) *kurtulmak "to escape"*
IG='[(1,"kurtul+Verb+Pos")(2,"Noun+Inf+A3sg+Pnon+Nom")]'

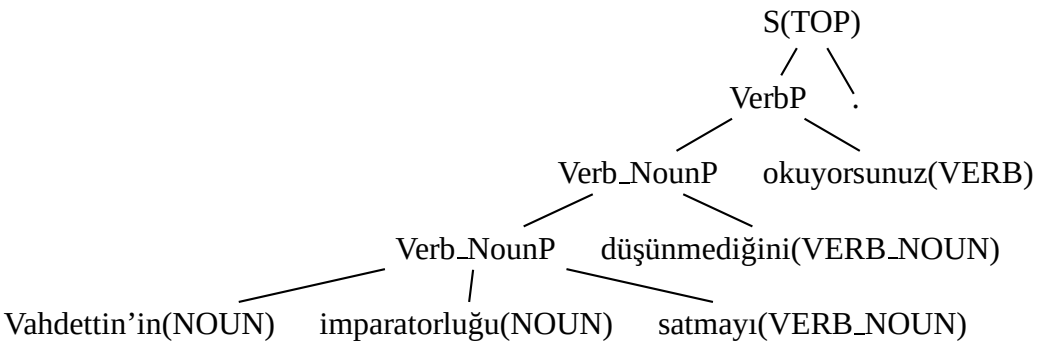


Figure 8.8: The improved mapping with extended pos tags.

However, this causes some problems regarding the way subordination and extraction are represented in the treebank for sentences like (8.6). Use of basic tags results in the tree in Figure 8.7. *satmayı* and *düşünmediğini* are both subordinate verbs, Howevet, there is no way to discriminate between a subordinated clause and a *NounP* with the flat tree structure we derive from the dependencies since they have the same basic part-of-speech tag.

(8.6)

Vahdettin'in	imparatorluğu	satmayı	düşünmediğini	okuyorsunuz
Vahdettin-Gen	empire-Acc	sell-Inf-Acc	think-Neg-PastPart-Agr	read-Prog

"You read that Vahdettin was not planning to sell the empire".

We create enriched POS tags for our second configuration by concatenating the original tag of the morphological stem and the final tag. Words with only one inflectional group are not affected by this change. This gives *Verb* for (8.4) and *Verb_Noun* for (8.5). This kind of information is expected to help subcategorisation choices for some words such as subordinated verbs and thereby help with predicting the relation between such words and their dependents. This means that *Vahdettin'in* and *imparatorluğu* will be correctly identified as a dependent of the subordinated verb *satmayı* instead of being clustered as a noun group. The same holds for the subordinated verb *düşünmediğini* and the rest as shown in Figure 8.8. A detailed description of different tags sets created by including varying levels of lexical and morphological information is given in Section 8.3.

8.3 POS tag sets

There are 102 morpheme names in the Turkish treebank. Not all of them can co-occur being restricted by the rules of morphotactics. In a real-life application, however, even

only the ones that *can* co-occur together with the possibility of the permutation of IGs containing them will give rise to sparse data problem for morphological (or part-of-speech) tagging. This means we cannot use the combination of all the morphemes as they appear in the data which are as sparse as the inflected forms of the words themselves. We need to find means to represent all the necessary information while preserving the compactness of the part-of-speech tags. For this we suggest several means of representation.

Turkish treebank does not have separate POS tag information. POS tags we mention here are drawn from the morphological parses of the words in the treebank. We tested the effect of using different POS tag sets on performance of the dependency parsers. We added the derivational boundary information up to a level of two in the POS tags, and then we added case information to nouns. All these are morphological information. However, the trade-off between data sparsity caused by using very rich POS tags and performance gain by using more informative tags starts to become biased against the performance gain at some point in between using 2 levels of derivational tags and case and using full IG tags that are given in the treebank data. This does not mean that when category types are diverse then we will always lose performance. For instance, although the CCG category types derived from the data are 450 compared to 15 basic POS tag types, using CCG categories as tags boosts the performance of unlabelled results for MST parser to about 95% from about 85%. This is because CCG categories carry even more information than morphological categories. It has been argued and shown that when given supertags that contain as much information as CCG categories contain parsing is much easier and very accurate (Bangalore and Joshi, 1999; Clark and Curran, 2006). Hockenmaier (2003a) makes a similar comparison of CCG tags with Penn Treebank style part-of-speech tags.

We used different types of POS tags that are enriched with morphological information that is present in the treebank data. The first set is the control set that only contains the actual part-of-speech information of the inflected (or derived) word. The other tagsets are obtained by gradually adding more morphological information on these basic tags.

In a typical morphological structure of a derived word (See Figure 8.5) that is made up of IGs, there are as many POS tags as there are derivational boundaries. In (8.5) the verb root *kurtul* goes through nominalisation. The next more informative POS tag set is obtained by concatenating the POS tags of the first IG, which is the POS tag of the root word to the POS tag from the last IG which is the actual POS tag of that

name	info	example
BAS	last IG tag	*Noun*
EXT	first& last IG tag	*Verb_Noun*
CAS	BAS+case for nouns	*Noun_Nom*
EC	EXT+case for nouns	*Noun_Nom*
CRYP	1^{st} letter of each tag in the last IG	*2NIAPN*

Table 8.3: Different tag sets

(derived) word. This way, we differentiate between the root words and the derived words. The POS tag of the nominalised verb in the figure will be Verb_Noun instead of Noun with this schema. We suspect that given that some morphosyntactic phenomena such as relativisation and causativisation are only marked with derivational morphology and IG-based dependencies, making this distinction will improve the performance on sentences with these phenomena.

The third tag set includes the basic tag information which is same as the first set. In addition, this tag set has grammatical case information included for nouns. None of the other grammatical features are included in this basic tag group.

The fourth set is obtained by extending the second set to have case information for surface nouns. All the POS tags apart from *Noun*s are the same as type 2 POS tags. Nouns, regardless of whether they are derived or not have POS tags: *Noun*_Case, Case being the case information of the noun (or nominalised form).

The last set is created from the last IG of the word. The number of the IG and the first letters of all tags in that IG are concatenated to form a cryptic tag. This tag only contains derivation information implicitly, since a number bigger than 1 means that the word has more than 1 IG, thus derived.

Table 8.3 shows different POS tag sets used in the experiments and the information they contain. The example column shows what the tags are in each tag set for the example in (8.5).

Eryiğit, Nivre, and Oflazer (2008) used a similar approach of combining information in POS tags in unlexicalised parsing. They also show that lexicalisation and using similar morphological information to ours improves parsing performance.

8.4 Parsing models

8.4.1 Head-driven generative parsing

Collins (1997) describes several lexicalised head-driven generative parsing models that are now widely known and used. They incorporate varying levels of structural information, such as distance features, the complement/adjunct distinction, subcategorisation and gaps. The core idea is to decompose the calculation of context-free rule probabilities by first generating a head and then generating its left and right modifiers independently.

We use Dan Bikel's multi-lingual parsing engine (Bikel, 2002) to train such models for parsing Turkish. We use Collins' model 1, so the features are standard ones: words, tags and distance over heads and modifiers. We also use the first-order bigram dependencies described in (Collins et al., 1999). With this extension, the generation of a modifier is dependent on the previous modifier as well as the parent and the head:

$$\prod_{i=1\ldots n+1} \mathcal{P}_l(L_i(l_i)|L_{i-1},P,h,H)$$

We use Bikel's default approximation of the previous modifier, where it is either the (a) START symbol (no previous modifiers), (b) a coordinating conjunction, (c) a punctuation mark, or (d) MISC for all other modifiers.

We train the parser on the trees mapped from the dependencies, as described in section 8.2.1, and then parse unseen sentences with and without their POS tags. Dependencies are then recovered from the trees derived by the parser by reversing the dependency structure to tree mapping.

Context-free structures are capable of only representing projective dependencies. To represent ill-nested dependencies on such structures, mechanisms such as traces must be utilised, and the parser must be made aware of them. Not doing this will mean that the parser is simply unable to recover such dependencies.

We train a parsing model on trees which have been "uncrossed" and then apply the model to sentences which may have crossed dependencies in them. In such cases, we will fail to get the crossed dependencies. We can use this to explore the effectiveness of different configurations and compare them to the non-projective parsing models.

8.4.2 Discriminative dependency parsing

While phrase-structure parsers such as that of Collins are widely-used for recovering dependencies as well as syntactic structures, there are many others which solve the dependency parsing task directly, beginning with Eisner's cubic generative dependency algorithm (Eisner, 1996b). McDonald, Crammer, and Pereira (2005) provide a discriminative version of Eisner's dependency parser that scores alternative analyses using large-margin constraints determined with the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003). For English, this parser performs on par with using a Collins model to recover dependencies, yet is far more efficient.

McDonald et al. (2005) define a new algorithm that formalises dependency parsing as the problem of finding a maximum spanning tree in a directed graph. Again, MIRA is used to determine the weights of dependency links as part of this computation. This algorithm has two major advantages: it runs in $O(n^2)$ time and it handles non-projective dependencies directly. McDonald et al. (2005) show that this algorithm significantly improves performance on dependency parsing for Czech, especially on sentences which contain at least one crossed dependency. Given this, it is natural to expect that the algorithm will be similarly useful for Turkish, and our experiments confirm this.

We use McDonald's MSTParser implementation with the same four tag sets as we do with the Bikel parser. It comes ready with an extensive set of features (McDonald, Crammer, and Pereira, 2005). These features incorporate almost all the different ways in which the words and POS tags of a head and dependent (and words/tags in between them) can be related. These out-of-the-box features prove to be extremely effective.

Because MSTParser uses a discriminative criterion, many more features can be included without running into problems due to independence assumptions. We thus extended the parser to optionally use a wider range of features; specifically, we use word stems and suffixes to create many new features that contain the relationships of these to each other, to full words, and to POS tags. We obtain the stems from the treebank itself, and as suffixes we use the remainder of the word after removing the stem.[2] Performance with these features should indicate whether even such a rough morphological analysis is useful for parsing morphologically rich languages.

[2]More precisely, we remove a prefix with the same number of characters as the stem in order to handle sound changes. For example, the word *tutsağım* has the stem *tutsak*; from this we get the suffix *ım*.

8.5 Experiments

We report a number of experiments that compare various configurations which vary the parser, the tagger, and the tag-sets. We use three parser configurations: Collins' phrase-structural model, MSTParser's projective model, and MSTParser's non-projective model. There are four different tag sets (see Section 8.3): (a) the basic ones [BAS], (b) extended tags (tag of the stem plus tag of final inflectional group) [EXT], (c) case for nouns [CAS] and (d) the combination of (b) and (c) [EC]. Furthermore, we consider an enriched feature set for MSTParser that incorporates stems and suffixes. For tagging, we use either tags produced by a tagger[3] [TT] or gold tags from the treebank [GT].

We perform 10-fold cross-validation over all the sentences in the treebank. Model performance is given for both word-level and sentence-level dependency accuracy.[4] We provide unlabelled scores for the Collins parser, we give both labelled and unlabelled for MSTParser. Unlabelled word and sentence accuracy are abbreviated as UA and SUA, respectively. LA and SLA are likewise used for labelled accuracy. Scores are globally determined rather than averaged over all individual folds. We take word-final punctuation in the Turkish treebank to constitute the *root* symbol (familiar from other work on dependency parsing) in our evaluation. We do this because the word-final punctuation is given a dependency link to a dummy root symbol, but this happens unambiguously for all sentences. This link is thus trivial to identify, so we exclude it from consideration for scoring all our models.

Turkish is a predominantly head-final language, so the toughest baseline is one for which all words depend on the word immediately to the right. This baseline correctly captures 63.2% of the unlabelled dependencies. The left branching baseline simply highlights how few *adjacent* leftward links are in the treebank – just 6.1%.

8.6 Results

Table 8.9 shows the performance of the Collins model under the various configurations. We see that even with the most basic tag set, the parser easily beats the right-linking baseline. Using the richer tag sets helps considerably, mirroring the results of Collins et al. (1999) for Czech based on similar strategies. Because many Turkish words convey what would require several words in English, it is too crude to just label them

[3] We use the OpenNLP tagger (`opennlp.sf.net`).

[4] We score our models with the evaluation script used for the CoNLL-X dependency parsing shared task, and evaluate significance with Dan Bikel's significance tester.

Model	UA	SUA
LEFT	6.1	4.3
RIGHT	63.2	16.2
BAS-TT	71.9	35.6
BAS-GT	73.6	37.5
EXT-TT	74.0	36.3
EXT-GT	76.2	39.0
CAS-TT	76.2	38.5
CAS-GT	77.8	40.7
EC-TT	77.4	38.9
EC-GT	79.3	41.5

LEFT = left branching baseline RIGHT = right branching baseline BAS= basic tag set EXT= extended tag set (ex:Noun_Verb) CAS= case information for nouns EC= extended tag + case TT= tagged data GT= gold tags

Figure 8.9: Performance of Collins model with different tag sets.

with simple tags like *Noun*. The extended tags (EXT), such as *Noun_Verb*, are crucial for getting the syntactic distribution of such words correct. Case information on tags (CAS) is also fundamental; for example, nominative and genitive nouns appear in very different contexts, so collapsing them as in the basic tag set keeps the parser from being able to handle them appropriately. From the basic tag set BAS to the most complete EC, performance is improved by 6%.

Unsurprisingly, performance suffers when using tags from the tagger rather than the gold standard tags. However, the drop is not great, and the 77.4% accuracy achieved by the model using the EC tags is well above the 63.2% baseline — and it is obtained with only access to the raw words. Note that the parser is capable of tagging for itself – for the same configuration using parser tags instead of the tagger's, the performance is 74.6%. This is actually *not* in line with many previous results, where it is often found to be better to let the parser tag for itself than to use a POS tagger. This is probably due to the fact that both the corpus and the tag sets are small, so the maximum entropy tagger is able to model the tags themselves more effectively than the parser, which obtains its probabilities directly from frequence counts and is thus more reliant on large amounts of data.

Table 8.10 shows the results for the non-projective maximum spanning tree parser.

Model	UA	SUA	LA	SLA
BAS-TT	79.0	39.1	61.5	19.6
BAS-GT	81.5	42.7	65.9	22.9
EXT-TT	80.0	40.5	62.6	20.1
EXT-GT	83.2	44.6	67.9	23.5
CAS-TT	79.8	40.7	64.7	21.7
CAS-GT	82.5	44.5	70.0	26.6
EC-TT	80.6	41.7	65.3	22.3
EC-GT	84.5	46.9	72.1	28.2

Figure 8.10: Performance of MST non-projective model with different tag sets.

Across the board, this parser clearly beats the Collins parser on recovering unlabelled dependencies. When given gold tags, the MST parser given access to the same tag set beats the Collins parser by over 5%. It also shows less variance to the choice of tag set, with only a 3% difference between BAS and EC, compared to 6% for the Collins parser. However, its performance when using tags from the tagger rather than gold tags is relatively more affected than the Collins parser. Nonetheless, its absolute performance even with tagger tags is still well above that of the Collins parser.[5]

The labelled scores for the MST parser also show some interesting patterns. Most obvious is that labelled performance is more heavily affected than unlabelled when the parser is given tags from the tagger. This is unsurprising since some tags correlate closely with some labels, such as the tag *Noun_Nom* (nominative-case noun) and the label SUBJECT. On a similar note, we see that the CAS tag set (where case is given) improves labelled accuracy from 65.9% for the basic set to 70.0%, a more significant jump than the 67.9% provided by the EXT tag set.

Table 8.11 provides the results for when the MST parser is given the stems and suffix features in addition to the word and tag features that come out of the box. The additional features provide a significant boost in performance ($p < 0.05$) for all configurations. Most interestingly, the performance when using the tagger tags is a more marked improvement over the model with stems and suffixes. The stems and suffixes essentially provide a means to lexicalise the model with less sensitivity to data sparsity

[5] Performance would presumably not be as degraded if the parser was *trained* on tags from the tagger rather than gold tags. That way, the material that the parser trains on is deficient in similar ways to the material it is tested on.

Model	UA	SUA	LA	SLA
BAS-TT	80.6	41.2	62.3	19.4
BAS-GT	83.3	45.5	66.7	22.9
EC-TT	81.3	42.6	65.8	22.5
EC-GT	84.9	47.7	72.3	28.2

Figure 8.11: Performance of MST non-projective model with the BAS and EC tag sets using stem and suffix features.

than full words on their own. They thus help keep the model from choosing poorly when it is given an incorrect tag from the tagger. This indicates that lexical information is both useful and sufficient despite the small size of the treebank, contra Eryiğit and Oflazer (2006), whose statistical dependency model pays attention only to tags and distance measures.

The projective dependency parser (Eisner's algorithm) actually performs very similarly to the non-projective one. For example, with gold tags, the EC tag set and the stems and suffixes features, it achieves 84.8%/48.1% UA/SUA and 72.2%/28.5% LA/SLA, not significantly different from the performance attained by the non-projective parser (see EC-GT in 8.11). This is actually not very surprising, given that only 5% of the dependencies in the treebank are crossed. Nonetheless, we can see the importance of the non-projective algorithm more clearly by scoring both models on just the 344 sentences that had at least one crossed dependency. For these, the non-projective parser with the EC tag set and the stem and suffix features achieves 76.3%/64.0% unlabelled/labelled accuracy. The projective parser with the same tags and features obtains 75.1%/62.9%. This mirrors what McDonald et al. (2005) found for Czech, though the difference they found was greater: 81.5% for the non-projective versus 74.8% for the projective.

8.6.1 Rightward and Non-crossing Dependencies

In the treebank, there are 3501 sentences which have only rightward links.[6] These are the sentences Eryiğit and Oflazer (2006) used in their evaluation. Their best model achieved 77.2% word-word UA on these sentences. Our best *projective* model (EC-

[6]This is a slightly different number than that given by Eryiğit and Oflazer (2006) (3398) This might be because we do not count crossed dependencies on the IG level.

GT) gets 85.6%/53.1% UA/SUA and 72.6%/31.6% LA/SLA on these sentences. This large improvement should be considered in the light that our model can posit leftward links, a degree of freedom that is not granted to Eryiğit and Oflazer's model. Unsurprisingly, on these test sentences, the projective parser is slightly, but significantly, better than the non-projective one's performance of 85.4%/52.3% UA/SUA and 72.2%/30.9% LA/SLA. The Collins model achieves 81.6%/47.8% UA/SUA on the rightward-linking sentences. Please note that on the sentence level, we assign correct dependency structures to almost half of the sentences.

Model	UA	SUA
BAS-TT	72.6	39.6
BAS-GT	73.6	41.5
EC-TT	79.7	44.8
EC-GT	81.6	47.8

Figure 8.12: Performance of Collins model with different tag sets on sentences with *only* rightward links.

Model	UA	SUA	LA	SLA
Eisner BAS-GT	82.0	47.4	64.9	25.0
MST BAS-GT	83.3	45.8	66.7	23.1
Eisner EC-GT	85.6	53.1	72.6	31.6
MST EC-GT	85.4	52.3	72.2	30.9

Figure 8.13: Performance of discriminative parsers with different tag sets on sentences with *only* rightward links.

8.6.2 Part of Speech Tagging

To contextualise our results, we feel it important to stress that most work on dependency parsing uses gold standard POS tags as input to the parser. Our absolute best result of 84.9% by EC-GT with stems and suffixes can thus be compared to other work which assumes information beyond just the word string, including the results presented in Eryiğit and Oflazer (2006) for Turkish. Yet for a parser to be useful outside the context of the experimental sandbox, it needs to be able to deal with untagged text. Our

best configuration for this more stringent criterion is the MST parser with the EC tag set, tags from the tagger, and without features based on stems and suffixes (the former of which we obtain from the treebank, not automatically). This model, shown as EC-TT in Figure 8.10, obtains 80.6% UA.

8.6.3 CCG categories as supertags

We show that even with a parser that is not able to use the structural information in CCG categories, including them improves the performance of parsing. We integrated CCG categories as simple features to MST parser. Using categories as unanalysed identifiers boosted the performance of the parser. This could be thought of as an upper bound since we use gold standard tags that we induced from the data automatically (Chapter 5 and Chapter 6).

Foth, By, and Menzel (2006) describe a recent study. They use supertags to guide dependency parsing of German. The supertags they use are not CCG categories but they are surprisingly similar to CCG categories in terms of the kind of information they contain. They have directionality information and head and argument distinctions. Foth, By, and Menzel (2006) also integrate the relation names and local context of the word into the supertags. They use a different parser from the one that is used here. Foth, By, and Menzel (2006) achieve 24% error reduction in parsing performance over their baseline which they say is already competitive.

Model	UA	SUA	LA	SLA
as POS TAGS	93.66	68.70	87.10	46.92
as FT - BAS	93.92	69.52	87.31	47.42
as FT - EC	94.60	71.36	87.80	47.92
as FT - CRYP	93.90	68.02	87.84	47.86
ST - EC	66.60	16.29	35.12	0.62

Figure 8.14: Performance of MST non-projective parser with CCG categories

Our first experiment acts as a baseline performed to see if using CCG categories would help the parser at all. Thus, we replaced all POS tags with the CCG categories of the words. The boost in performance was encouraging (93.66% unlabeled and 87.10% labeled). As shown in the previous section and in (Hockenmaier, 2003a) using more informative part-of-speech tags boosts the performance. CCG categories are very infor-

mative tags since they contain information pertaining to predicate-argument structure and local and global dependencies.

We explored with a few more experiments whether this information could be better used. MST parser uses a feature called "coarse tag" which is the first letter of the POS tag. In our second set of experiments we used POS tags as coarse tags and CCG supertags as fine-grained tags. The results of these are shown in [FT-BAS] and [FT-EC] rows. [FT-BAS] uses our basic pos tags as coarse tags and [FT-EC] uses the extended tags. This shows that using the extended POS tags always yields better results with 94.60% unlabelled and 87.80% labelled accuracy. It is also important to note that 71% of the sentences have completely correct unlabelled dependencies.

In the last experiment in Figure 8.14 we used the supertagger described in Clark and Curran (2004b) as a front-end to the parser [ST]. The performance of the supertagger was very low. This result was expected given the size of the data and uninformativeness of the POS tags in Turkish treebank. This had a big impact on the MST parser decisions and the performance radically dropped when given mostly wrong CCG supertags by the supertagger.

8.6.4 Inflectional Groups as lexical entities

The results were better when the IGs were included in the dependency structure but dependency links were normalised. This means the dependency links were always from the dependent to the stem (the first IG) of the head and they always emanated from the last IG of the word. We have 64992 tokens (IGs) in morphemic data compared to 53826 tokens (words). The IGs could further be decomposed into smaller morphological units. Here, only derivational morphology is taken into consideration when forming the IG boundaries.

The first two rows in Figure 8.15 shows the IG to IG dependency recovery with no additional features used apart from the ones MSTParser uses. The first row is the attachment score for the exact linking corresponding to the ones in the Treebank. The second row is the score obtained by using only the stems of the full words and second morpheme tags for IGs.

The third and fourth rows show the attachment score with CCG categories given as features. When the dependent is always connected to the first IG of the head word the scores are higher than the "correct" IG dependency score given in the fourth row. This difference may be caused by annotation inconsistencies which we encountered many

Model	UA	SUA	LA	SLA
correct IG lexicalised	81.08	33.21	72.0	18.45
correct IG stems only	80.66	32.5	70.7	16.65
first-IG, gold CCG cats	95.08	69.74	88.96	46.14
correct IG, gold CCG cats	93.59	65.10	87.46	43.41
correct IG, supertagged	74.41	21.09	58.17	9.68
CoNLL set with conll eval (non-stem igs removed)	89.72	-	84.39	-
CoNLL set with conll eval	92.37	-	88.37	-
CoNLL set with mst_eval	93.03	54.98	89.63	43.92

Figure 8.15: Performance of MST non-projective parser with inflectional groups as entities

times in the data or simply because there was not enough data to train on for non-initial IG dependencies. 88% of the non-null dependencies were dependencies to the first IG of the head word. Only 10.2% were to the second IG and the rest were distributed among 3rd, 4th and 5th IG dependencies.

Gold standard tags give an upper bound on the parsing results. The fifth row shows the results with a supertagger front-end to the parser. Although supertagging accuracy is higher than it is for word-based results, it is still very low. The system is trained with gold CCG categories and the test set was supertagged before parsing. This results in a dependency accuracy which is lower than the results when no CCG categories are used which is shown in the first row. Labelled dependency accuracy seems to be more affected by supertagging accuracy. This is expected because CCG categories are closely related to relation types in dependencies. For instance a noun will be labelled as OBJECT if it has category NP, but if it is assigned NP[nom] by the supertagger, the label will be SUBJECT instead. Taggers are affected by the training data size as all other statistical systems. We show by comparing word-based and IG-based supertagged results that morphological smoothing improves both the performance of the supertagger and the dependency parser. However, an increase in training data size will undoubtedly result in higher accuracy in supertagging which is about 53% in the experiments here.

The last 3 rows show the performance of MSTParser on a different evaluation set, namely, CoNLL 2006 split set of Turkish. The results are evaluated with the CoNNL

evaluation script The conversion to CoNLL data set is discussed in Section 8.1.3. We treat internal IGs just the same as other elements in the dependency structure. The row labelled *CoNLL set with conll eval non-stem IGs removed* shows the accuracy when words are turned into "_" character if they are not final IGs. This is done to make the results comparable to the results in CoNLL 2006. The last row in the table shows the evaluation that is done on all IGs, which, we believe, is more appropriate for the treebank.

8.7 Conclusion

We have demonstrated a range of dependency parsing regimes for Turkish. All our models perform well above a right-linking baseline, even when using tags from a tagger rather than gold standard ones, although it is a very high baseline consistent with head-final nature of Turkish. Simple extensions to the tag set provided large improvements to parsing accuracy for all models. The discriminative dependency parsers of McDonald et al. (2005) easily outperform the Collins-style phrase-structure parser. The difference can be attributed to the fact that the dependency parsers attack the problem directly and do not need the extra level of indirection of phrase structure trees, which can have complications such as training on phrase structures with uncrossed orders and then testing on sentences with crossed dependencies.

We also extended the MST parser with features based on word stems and suffixes in addition to full words and tags. These features infuse morphology into the parsing model, which would be expected to be important in a morphologically rich language like Turkish. Our results show significant improvements with these features, especially when the parser was supplied with tags from a tagger.

Even though the non-projective MST algorithm and the projective Eisner algorithm (both using MIRA) achieve similar performance overall, we showed that the former is significantly better on the subset of sentences in the Turkish treebank which have at least one crossed dependency.

Our experiments with CCG categories as supertags proves once more that "supertagging is almost parsing" (Bangalore and Joshi, 1999) even when used as features for a discriminative dependency parser that computes the probabilities of the dependencies directly regardless of whether they give a correct CCG derivation or not. CCG categories are very informative supertags and using gold standard supertags gives us the upper bound results for parsing. However, the importance of high accuracy su-

pertagging is significant. We will focus on this issue in the next chapter.

Experiments with a morphologically sensitive model show that there is great potential in using morphological information in parsing because 1) morphological smoothing is inherently a good way to overcome sparse data problem for relatively small corpora 2) use of this extra information proves to affect the parsers decisions in a positive and more linguistically accurate way. McDonald, Lerman, and Pereira (2006) also show that the use of morphological features improves the performance in multilingual parsing experiments. The next step in this direction is to use a full morphological analyser and a disambiguator as a front end to the parsers.

Using stems of the words together with morphological information does not give the expected improvement. Using inflected forms instead of IGs in the morphemic models leads to higher accuracy. This is possibly because the morphological information is organised with regards to derivational boundaries, which means the stem of a word does not share the part-of-speech tag as the word in most cases. This possibly causes confusion as the dependency is very closely related to the amount of information in POS tags, as shown in previous sections. Exploring the effects of including more information on inflectional morphology is the focus of future research.

We have given state-of-the-art results for Turkish dependency parsing and showed that there is great potential in using morphological information. We used 10-fold cross validation in all our experiments and trained and evaluated the parsers on the corrected version of the treebank which proves to be important especially when the results for CoNNL 2006 and 2007 shared task results for Turkish are compared. CoNNL 2007 uses a corrected version of the data. Turkish parsing results on the average are significantly higher with this data set than CoNLL 2006 data set.

Chapter 9

Parsing with Combinatory Categorial Grammar

Dependencies that are not explicitly represented in the syntactic representation are as important for semantic interpretation as the ones that are. They are sometimes referred to as hidden or "deep" dependencies, and are usually found in extraction, wh-movement, gapping and so on. Predicting deep dependencies has been the focus of research for some authors of parsers. Some of them tried to solve this problem by post-processing the context-free parse trees (Levy and Manning, 2004; Dienes and Dubey, 2003; Johnson, 2002), while others used more powerful grammars such as CCG to predict these automatically (Clark, Hockenmaier, and Steedman, 2002; Hockenmaier, 2003a; Clark and Curran, 2007b). In dependency theory, deep dependency parsing is usually ignored and sometimes resisted. It is claimed that parsing directed graphs that allow multiple-heads is intractable (McDonald, 2006). However, these dependencies in natural languages are not arbitrary and, most of the phenomena that cause these and the crossing dependencies were shown to only require generative power that is slightly more than context-free. Most non-projective dependency parsers assume arbitrary non-projective graphs which are difficult to parse. CCG is able to represent and predict surface and deep dependencies without any extra effort theoretically. It was shown that efficient and high-accuracy parsing to predict these non-conventional dependencies is possible (Clark and Curran, 2007b; Hockenmaier, 2003a).

Another argument is that the deep dependencies are semantic dependencies rather than syntactic ones, thus they do not need to be present in the syntactic representation of dependency graphs (McDonald, 2006). However, dependencies involved in control, coordination and other phenomena have been included in syntactic representation in

various Dependency theories and multiple-heads and deep dependencies are allowed in some theories such as Hudson's (2007) Word Grammar.

In this chapter, we will give a review of wide-coverage statistical CCG parsers. We will also cover the information regarding adaptation of a state-of-the-art CCG parser for parsing Turkish. In the final part, we will give the results of parsing Turkish and compare the outcome with the earlier results of dependency parsing.

9.1 Why CCG parsing

Combinatory Categorial Grammars explain phenomena such as coordination and extraction elegantly as discussed earlier. CCG parsers recover local and long-distance dependencies without any post-processing or extra effort. CCG can be parsed in polynomial time with a worst case complexity of $O(n^6)$ (Vijay-Shanker and Weir, 1990; Vijay-Shanker and Weir, 1993). Successful adaptation of the "spurious ambiguity" removing methods such as Eisner's constraints, and chart parsing algorithms that facilitate dynamic programming made creating very fast wide-coverage statistical CCG parsers possible. To summarise, CCG is a formalism that is expressive enough, and the output of CCG parsing is highly informative. In addition, all this can be done very efficiently.

CCGBank is the biggest corpus of CCG derivations created semi-automatically by transforming Penn Treebank into a treebank of CCG derivation trees (Hockenmaier and Steedman, 2007). The wide-coverage CCG parsers can be divided into groups regarding the methods of statistical models they use and the representation structures they model on. Hockenmaier's generative model on normal form derivation trees trains on CCGBank trees whereas Clark, Hockenmaier, and Steedman (2002) model the dependencies derived from these derivation trees. Clark, Hockenmaier, and Steedman (2002) create a similar model to that of Collins (1996) modeling dependency structures. Clark and Curran apply log-linear training methods and they model on both derivation trees and on dependencies (Clark and Curran, 2003; Clark and Curran, 2004b; Clark and Curran, 2007a; Clark and Curran, 2007b). All these parsers use data derived from CCGbank (Hockenmaier and Steedman, 2007).

Clark and Curran (2006) shows that a parser can be trained with only words and CCG categories. The dependencies that are used to train the data are extracted from the category sequences rather than full derivations. The performance of this model is 1.3 points of F-score less than the model trained on the full data. This shows the amount

of information in the CCG categories.

9.2 Hockenmaier's (2003) parser

Hockenmaier's (2003a) parsing models are generative models that are trained on a CCG-derivation database that is induced semi-automatically from the Penn Treebank. The translation procedure is explained in Section 5.1. Hockenmaier justifies using derivation trees with theoretical and efficiency arguments with regards to the claim by Steedman (2000) that derivation structures are regarded merely as a record of construction.

Generative models of parsing estimate the probability of a parse given a sentence depending on the probability of their parts. On the other hand, in a conditional model the probability is estimated directly. Hockenmaier uses CKY chart parsing algorithm with beam search to narrow the search space. Hockenmaier's model on normal-form CCG derivations is an adaptation of Collins (1997). Hockenmaier differentiates between 4 different kinds of expansion, namely, *leaf*, *unary*, *left*, and *right*. The last two depend on the information if the head is on the left or the right one of the two sub-trees.

Hockenmaier explains that even though the model does not assign zero-probability to non-normal-form derivations, in practice, it will always prefer normal-form derivations. Additional features such as distance do not improve the performance of the parser, however, Hockenmaier argues that this is because of inherent characteristics of generative models such as the trade-off between bias and variance problem described by Geman, Bienenstock, and Doursat (1992).

Hockenmaier (2003b) proposes a generative model for predicate-argument relations. This outperforms the earlier results by Clark, Hockenmaier, and Steedman (2002). Hockenmaier claims that this is the first model to recover all local and long-range word-word dependencies. It also solves the problems caused by the multiple heads reported by Abney (1997).

The parsers described in Hockenmaier (2003a) could not be used for parsing Turkish simply because the parser needs CCG derivation trees to train on. Translating the Turkish dependency treebank to CCGBank representation is problematic because of crossing dependencies and the lack of markers for non-surface dependencies in the Turkish treebank (Çakıcı and Baldridge, 2006).

9.3 Clark, Hockenmaier and Steedman (2002) parser

This parser uses the dependency structures that are derived from the CCG derivations in CCGBank. This decision is motivated by several factors. First, the evaluation of CCG parsers with dependencies is much more informative than the PARSEVAL evaluation based on constituency information. Second, is the convenience of not having to deal with problems caused by non-standard derivations, spurious ambiguity and such.

In Clark, Hockenmaier, and Steedman (2002), a sentence S consists of word-POS-tag pairs $S = \langle w_1,t_1\rangle, \langle w_2,t_2\rangle, ...\langle w_n,t_n\rangle$, and a dependency structure π is a $\langle C,D\rangle$ pair. C is a list of categories $C = c_1,c_2,c_3,..c_n$ assigned to each word and $D = \{\langle h_{f_i}, f_i, s_i, h_{a_i}\rangle | i = 1,...m\}$ is a list of dependency relations represented by a 4-tuple including the head word h_{f_i}, the functor category, f_i, the argument slot s_i and the argument head h_{a_i}.

Note that this representation does not make any assumptions on how many heads a word can have, thus allows multiple heads unlike the standard dependency approach discussed in previous chapters. The probability of a dependency structure is defined as in (9.1).

(9.1)
$$P(\pi) = P(C,D|S) = P(C|S)P(D|C,S)$$

The $P(C|S)$ part can be approximated as :

(9.2)
$$P(C|S) \approx \prod_{i=1}^{n} P(c_i|X_i)$$

where X_i is the local context. The category-word pair probabilities are calculated using maximum entropy techniques following Ratnaparkhi (1996) as described in Clark (2002) and later in Clark and Curran (2004a). $P(D|C,S)$ is written as follows:

(9.3)
$$P(D|C,S) = \prod_{i=1}^{m} P(h_{a_i}|C,S)$$

where h_{a_i} is the head word filling the argument slot (*dependent word* in the terminology used here) in the i^{th} dependency in a list of m dependencies. The estimation method used here is taken from Collins (1996). Since the dependency number changes over different category sequences, they used a geometric mean of $p(\pi)$ as the ranking function averaged by the number of dependencies in D. They use a CKY chart parsing

algorithm similar to the one described in Steedman (2000) and claim to deal with the redundant analysis in a similar way to what Komagata (1997) did.

Their results were 90.1% precision and 89.9% recall on unlabelled dependencies, and 81.8% and 81.9% on labelled dependencies on Section 23 of WSJ Penn Treebank. Labelled dependencies are regarded as correct when the 4-tuples exactly match with the gold standard. Unlabelled dependencies are regarded as correct when there is some kind of dependency between two heads regardless of the argument slot the argument fills. They also give the results on object extraction dependency recovery as a preliminary perspective on long-distance dependency recovery.

9.4 Clark and Curran (C&C) Parser

Clark and Curran (2007b) describe a number of log-linear parsing models trained on CCGBank to parse the Penn Treebank. With log-linear models the parse space can be represented in terms of features, and adding new features is relatively easy. Clark and Curran make considerable use of optimisation techniques and parallelized programming to account for the performace requirement of the estimation task. The memory requirements vary between 25-30GB of memory and an 18-node cluster was used in the experiments.

There is a dependency model and a normal form model, but they are both evaluated by the amount of correct dependencies they recover. The output of the parser was also evaluated on DepBank (King et al., 2003), in order to make a cross-formalism comparison and outperformed the RASP parser (Briscoe and Carroll, 2006) even though the dependency evaluation experiment had certain disadvantages for the CCG system.

The basics of the log-linear model is explained in the rest of this Section. The reader is referred to Clark and Curran (2007b) for a detailed introduction to the concept and the implementation details of all the parsers covered here.

9.4.1 Clark and Curran's (2004a) parser

Clark and Curran (2004b) apply log-linear models that are described in Clark and Curran (2003) to wide-coverage CCG parsing. They give results for both a dependency model trained using all-derivations including the non-standard ones and a normal-form model trained on normal-form derivation trees. They use the same evaluation method as Clark, Hockenmaier, and Steedman (2002).

The dependencies that are used in training the dependency model and evaluating both models are derived from the CCGBank predicate-argument relations. These dependencies are 5-tuples, examples of which are shown in 9.4. The first element is the head representing the dependency relation, the second element is the lexical category. The third element is the argument slot of the word that constitutes the remaining part of the dependency (the argument, the dependent etc.) which is given in the fourth slot. The fifth slot is allocated for the locality feature of the dependency. The subscripts on the word forms show their positions in the sentence. The following dependencies are derived from the sentence "*IBM bought the company*".

(9.4) $\langle bought_2, (S\backslash NP_1)/NP_2, 2, company_4, -\rangle$
$\langle bought_2, (S\backslash NP_1)/NP_2, 1, IBM_1, -\rangle$
$\langle the_2, NP/N_1, 1, company_4, -\rangle$

In a sentence with extraction , for instance in *The company which IBM bought*, the dependency between *bought* and *company* is as represented in (9.5). This dependency is not a local one. The final field in the tuple shows that the dependency is a long-range dependency created by the object extraction category $(NP\backslash NP)/(S/NP)$.

(9.5) $\langle bought_2, (S\backslash NP_1)/NP_2, 2, company_4, (NP\backslash NP)/(S/NP)\rangle$

The probability of a dependency structure π given a sentence S is defined as follows:

(9.6)
$$P(\pi|S) = \sum_{d\in\Delta(\pi)} P(d,\pi|S)$$

Here, $\Delta(\pi)$ is the set of all derivations that lead to the dependency structure π. A conditional log-linear model of a parse $\omega \in \Omega$ for a sentence is.

(9.7)
$$P(\omega|S) = \frac{1}{Z_S} e^{\lambda.\, f(\omega)}$$

ω means different things for normal-form model and the dependency model. For the normal form model it is simply a derivation, however for the dependency model it is a $\langle d,\pi\rangle$ pair.

They use of efficient numerical algorithms such as Limited-Memory BFGS algorithm and clusters for keeping the packed charts. They estimate the feature weights by using simple log-likelihood estimation.

The performance of the two models are very close. The dependency model gives labelled precision and recall scores of 86.7 and 85.6 and the best performing normal-form model gives 87.0 and 86.8. However, it is interesting to note that Clark and Curran find that adding long-range dependencies as features had no impact on parsing accuracy. They say it may be because these dependencies are very rare in the linguistic data. An evaluation on long-range dependencies is also not given in Clark and Curran (2004b) as the focus is more on efficient parsing. They apply Eisner constraints and *seen rules* constraints, which ensures only seen rules are used in parsing. Both methods considerably increase the speed. They also put a 300.000 limit on charts when training to derive the dependencies. However, the limit is 1million during testing to get the maximum coverage.

The parsers described in Clark and Curran (2007b) do not account for the type of ambiguity that Komagata (1997) categorises as the second type of "genuine ambiguity": "lexico-semantic ambiguity" which means categories may be assigned multiple semantic types. For instance, in C&C parsers two different types of control – object control and subject control – cannot be differentiated because a lexical category can only return a certain set of dependencies corresponding to a single semantic interpretation.[1] The following category is used for both object and subject control (as in *promise*), but it does not make the semantic distinction between two.

(9.8) persuade := $((S[dcl]_{persuade}\backslash NP_1)/(S[to]_2\backslash NP_X))/NP_{X,3}$

Head passing is performed with variables as shown in the relative pronoun category (9.9). When we change the direction of the slashes we obtain the object relative particle in Turkish.

(9.9) who := $(NP_X\backslash NP_{X,1})/(S[dcl]_2/NP_X)$

(9.10) -diği := $(NP_X/NP_{X,1})\backslash(S_2\backslash NP_X)$

9.4.2 Partial Training (Clark and Curran, 2006)

Clark and Curran (2006) show that only CCG categories are enough to train a parser

[1] Note that this is a design choice, not a property inherent to CCG.

and still obtain results within a certain percentage of the results obtained by using fully annotated data for training.

In contrast to previous CCG parsers that train on the derivation trees of CCGbank, partial training requires only CCG supertags (lexical categories) to create the training data for the parser. This is a very important step in efficient parsing, especially when Clark and Curran (2004b) show that CCG supertagging is very accurate when there is enough data. This agrees with the finding of Bangalore and Joshi (1999) who show that supertagging does most of the work when parsing. Training using the derivation extracted from only category sequences (partially annotated data) also attempts to solve the annotation bottleneck, since annotation of CCG categories is easier than annotation with syntactic trees.

Clark and Curran (2006) take the sentences decorated with CCG categories and attempt to parse these training sentences with a CCG parser that creates many derivations. They then take the dependency structures that are created by a certain percentage of all the derivations and assume that these are most likely the correct dependencies.

The gold-standard dependencies are extracted with the help of the parser described by Clark and Curran (2004b). The parser returns the dependencies occuring in k% of all the derivations licenced by that correct CCG category sequence. Setting k=100 has an effect of returning 100% precision but low recall. This is because a dependency that is in all of the derivations is bound to be in the correct derivation. Decreasing k increases the recall but decreases precision . They show that when k is 70% the recall increases to 85.87 while precision is as high as 99.09%.

The F-score they got from the parser in the end is only 1.3 % less than the parser trained on full dependency structures. This is a significant result. This shows the amount of linguistic information there is in CCG categories.

9.4.3 Supertagging

The supertagger is the crucial component of the C&C parser. Supertagging step boosts the speed of the parser, because in most cases the most likely category is the correct one. The memory requirement would be impossible to meet without the supertagger, if for example, the words were assigned all existing lexical category types as Hockenmaier (2003a) does (Clark and Curran, 2007b).

A variant of the Viterbi algorithm for HMM taggers is used to find the most likely category sequence for a sentence. The features used are the words and part-of-speech

tags in a 5-word window and the two previously assigned categories. Clark and Curran use a tag-dictionary for the word-category pairs occuring together in the data and have a k=20 limit for the word and the category occuring together. If this constraint is not satisfied, the POS tag of the word is used instead. Supertagger only assigns the 425 category types that occur more than 10 times in the data.

Multitagging (Curran, Clark, and Vadas, 2006) leaves some lexical category ambiguity to the parsing level by assigning a list of categories instead of a single category to each word. It assigns a list of categories that are within a probability threshold (β) to the word. This increases the possibility of assigning the correct category to the word, while limiting the entries in the chart.

Clark, Steedman, and Curran (2004) show that parsers can easily be adjusted to adapt to different domains by simply annoatating a number of sentences to improve supertagging accuracy on those sentences. Clark, Steedman, and Curran (2004) annotate a relatively small number of sentences by hand with CCG categories. They showed that this made a significant difference in the accuracy of the parser on questions and object extraction cases. These are two cases that modern parsers usually fail on. This shows that supertagging plays an important role in not only improving the accuracy but also adapting to other domains or hard-to-solve problems in NLP.

9.5 C&C and Turkish

We use the dependency model of the C&C Parser which is described in Clark and Curran (2007b) for parsing the Turkish treebank. Using CCG categories as features in MSTParser improved the performance. In the rest of this chapter, we demonstrate how the C&C parser is modified for creating a parsing model for Turkish and give the results of the parser. We explore the degree of information the lexicons (morphemic and lexemic) provide to the CCG parser, and how well the lexicon and the parser perform on deep dependencies that other parsers cannot handle.

The Turkish word-based (lexemic) CCG lexicon is compatible with the surface syntactic approach. We expect to recover only surface dependencies with this lexicon. This is because long-range dependencies arising from relativisation, control and such cannot be represented in this version of the lexicon. Relativisation is treated as simple nominal modification with the lexical category of a relativised verb as in: *NP/NP*. Thus, the dependencies captured with the gold-standard tags and with the lexemic lexicon will be directly comparable to MST parser results. However, the treatment

of coordination is internal to the parser and we still hope to get some of the long-distance dependencies resulting from coordination. Morphemic lexicon is expected to overcome some of the problems that are inherent in the lexemic approach.

The C&C parser has a dictionary cutoff of 20 which is hardwired into the parser. However, only 174 words out of 19385 unique words (including punctuation) appear more than 20 times in the lexemic lexicon. Even some function words are not in this list of 174 most frequent words. In the morphemic lexicon there are 6280 distint tokens in total, out of which 380 occur more than 20 times. The numbers of the distinct words are significantly different because all the words are either stem forms or morpheme-cluster names rather than inflected forms in the morphemic lexicon. A dictionary cutoff of k=20 is clearly too high because of the small size of the Turkish dataset. For the experiments here, we changed this to k=10, although we plan to optimise this value empirically in the future.

9.5.1 Turkish Markedup File

C&C uses a marked-up file that is integral to parsing as it is used for extracting the dependencies, and controls the head passing mechanism. Each lexicon requires a marked up file for the categories in the tag dictionary. The markedup files for the morphemic and the lexemic lexicons were annotated manually. Markedup entries look like the following:

(9.11)

```
(NP/NP)\(S\NP[nom])
  2 ((NP{Y}/NP{Y}<1>){_}\(S{Z}<2>\NP[nom]{Y}){Z}){_}
  1 ignore
  2 ignore
```

This is the category for the subject relative particle. The number 2 on line 2 indicates the number of dependencies this category licences. The lines following the annotation indicate the DepBank representation and names of the dependencies. We ignore these as the word ignore indicates.

9.5.2 Turkish Rules

Order changing rules are needed in some cases of scrambling including scrambling out of possessive constructions. In the following example, the genitive marked noun is scrambled out of the NP. This example is taken from Hoffman (1995). This approach considers genitive noun to be the subject of the genitive clause, following Szabolcsi (1983). In the other example, which is a genitive extraction case, we solve the problem with a similar approach.

Preserving directionality is important. However, some order changing rules are allowed in the Turkish lexicon in exchange for a more compact lexicon to explain scrambling, such as the extraposition rule that is included in the Turkish grammar by Bozşahin (2002). This rule is needed to explain some phenomena of extraposition of genitive nouns and such. However, it is restricted to NP and only the backward one is allowed. Bozşahin (2002) shows that the two "legal" word orders in Turkish are SOV and OSV and post-verbal scrambling is a case of extraposition, thus is handled by the following rule. However, he also suggests that the S in the rule is not discourse equivalent to S with the unscrambled order, in a way this restricts the use of this rule.

(9.12) **Extraposition** NP → S\(S\NP)

(9.13)

Ben	kapısını	boyadım	evin.
I	door-Poss3sg-Acc	paint-Past-Pers1sg	house-Gen
$S/(S\backslash NP_{nom})$ T	$NP_{acc}\backslash NP_{gen}$	$(S\backslash NP_{nom})\backslash NP_{acc}$	NP_{gen}
	$(S\backslash NP_{nom})\backslash NP_{gen}$ <B		
$S\backslash NP_{gen}$ >B×			$S\backslash(S\backslash NP_{gen})$ T
S <			

Turkish is mainly head-final. Extraposition rule is only needed to account for arguments scrambling to the right of the head. This is integrated into the type-raising rules in the parser. It is a direction changing rule, however it is necessary to handle some cases of post-verbal scrambling and cases like above. This rule was used in Baldridge (2002) in Turkish CCG grammar but was not discussed extensively, as it interacts with discourse and information structure.

Type Raising Rules

We have only added type-raising rules for NPs. These are:

$T/(T\backslash NP)$ where $T \in \{S, S\backslash NP, (S\backslash NP)\backslash NP, ((S\backslash NP)\backslash NP)\backslash NP\}$

Forward crossing composition was not implemented in the current release of the C&C parser. However, forward crossing composition is necessary for handling some

word-orders and other phenomena in Turkish. We lose some coverage because of this, and we will implement this rule in the future.

9.5.3 Training

Only about 65% of the sentences in the lexemic lexicon give a parse. This number is just above 70% for the morphemic lexicon. Only the sentences that are assigned at least one parse is included in the training. We use the same set of features for the dependency model as described in Clark and Curran (2007b), however, since we do not have normal-form derivations, the features that are extracted from normal-form derivations are excluded.

9.6 Evaluation

An important part of the evaluation process is extracting the gold-standard dependencies to which the output of the parser will be compared. Surface dependencies in Turkish dependency treebank is different from the output of the C&C parser. To make them compatible, a series of transformations are applied to the surface dependencies in the treebank.

1. Coordination is different in two formalisms. The way coordination is annotated is explained in Section 2.5.4. It is different from the dependencies output by the C&C parser. C&C outputs coordination dependencies in a way that the conjunction is the head of all conjuncts, and the coordination dependencies are always linked to the head of conjuncts. METU-Sabancı Treebank notation which has a sequential order, treats the conjunction as the modifier of the conjunct to the right, in a left to right manner. Therefore, some dependencies that C&C outputs do not match the dependencies of the treebank.

2. Predicate-argument relations are not equivalent to word-word dependencies in terms of directionality. In the case of verbs the order for predicate and argument is the same as the dependent and head order in C&C, however, a verb modifier, for instance, has the opposite order. A modifier is a dependent of the verb in dependency grammar terms, but since it has the functor category, the dependency related to the modifier category in C&C dependencies is represented as if the modifier is the head (in dependency terms). The order in the gold-standard dependencies was adjusted according to the dependency label between each dependent and head in the treebank during the

translation.

3. Punctuation is included in the dependency structure in some cases as explained in Section 2.3. However, C&C dependencies do not include any punctuation. Therefore, we excluded the punctuation that does not have a conj category from evaluation. This means we will lose points for some dependencies as we show in the next section.

Unlabelled evaluation in C&C checks if the dependency between two words is correctly predicted regardless of the argumet slot. Labelled evaluation takes into account the argument slot as well. All 5 fields need to be exactly the same in labelled evaluation. We provide the unlabelled dependency evaluation in this dissertation.

There are also the case of long-range dependencies that are produced in addition to the standard dependencies. These are extra dependencies that other parsers described in the previous chapters are not able to recover. These dependencies are very important in recovering the predicate-argument information in semantic analysis. Examples of these are, dependencies in extraction, coordination. The performance of the parser on these extra dependencies will be discussed in a limited fashion in Section 9.7.[2]

10-fold cross validation is applied to the data that consists only of the sentences that are assigned at least one parse by the parser given the lexical categories. Precision and recall figures are provided because the dependencies predicted by the parser and the gold-standard dependencies may differ. F-score is the is the balanced harmonic mean of precision (P) and recall (R): $F = 2PR/(P+R)$.

9.7 Turkish Results

The coverage of the morphemic lexicon 70.1%. The most common reason for a sentence to not be parsed is that one or more of its CCG categories are wrong. Another reason is the ambiguous markedup annotations of the categories. The Turkish categories do not have features apart from [nom] feature on NPs. This causes a great amount of ambiguity especially in the NP specifier categories. For instance, the head in the genitive construction is assigned NP\NP by the lexicon induction algorithm. On the other hand specifiers in some scrambled NP constructions are also assigned the same category. The former has the first NP as head whereas the in the latter the second NP is the head. However, it is only possible to annotate the markedup category in the

[2]The reason for this is that we do not have gold-standard deep dependency information originally in the data. Therefore, these dependencies can only be evaluated partly by comparing them to the secondary links that were added to the treebank data. These secondary links are explained in Sections 2.5.3 and 2.5.4.

Model	coverage	cats	UPrec	URec	F
morphemic	70.1	99.46	72.57	81.18	**76.63**
lexemic	65.3	99.43	65.31	72.72	**68.82**
no-oracle(morphemic)	97.3	71.55	55.64	63.1	**59.13**

Table 9.1: Dependency recovery with C&C parser

parser in one way. This costs some coverage points in either choice of annotation.

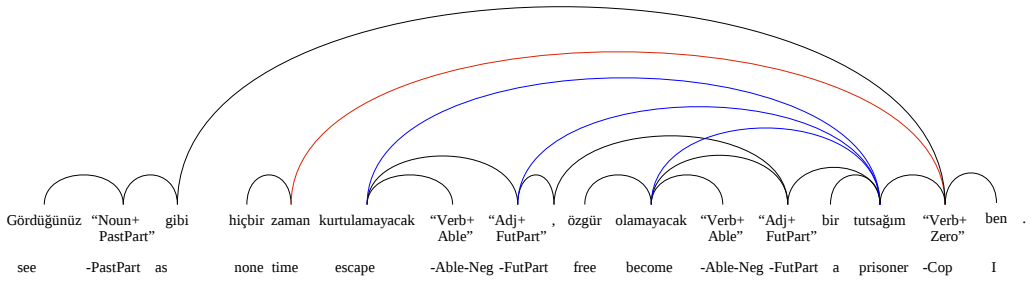
Table 9.1 shows the performance of the parsers trained with the lexemic and morphemic lexicons. The first and the second line shows performances with the gold-standard tags. The coverage is lower than the case where the supertagger is utilised which is shown in the last line. This is because the parser runs in multitagging mode and the words are assigned less probable categories until a parse is found. Note that the accuracy of the systems with the gold-standard tags are higher because they are evaluated on only the sentences that is assigned at least one parse which is 70.1% of the sentences in the morphemic case. However, with supertagging the correct dependencies are evaluated against almost all the sentences with 97.3% coverage.

9.7.1 Long distance dependencies

The following examples are included to emphasize the different kinds of dependencies each of the two lexicons can predict. As seen in the examples below, a morphemic lexicon has the ability to facilitate the correct predicate argument relation assignment by letting the verb take its arguments and then combine it with the extracted noun while creating the long-distance dependencies licenced.

This example demonstrates a few linguistic phenomena. This is the output of the parser with the input shown at the bottom. Each line shows a dependency the format of which is explained in Section 9.4.1. This example demonstrates the solutions to various phenomena such as subject extraction, coordination and post-verbal scrambling. Example 9.14 visualises the parser output that is given below in the form of dependencies. Long distance dependencies created by coordination are also extracted. The parser tends to connect adjuncts to the farthest head, usually the main predicate in the sentence. If there is an S/S, the parser makes it connect to the last verb.

(9.14)



Input:

Gördüğünüz|Verb|S –*see*

"Noun+PastPart"|PastPart|NP\S –*PastPart*

gibi|Postp|(S/S)\NP –*as*

,|Punc|,

hiçbir|Det|(S/S)/(S/S) *none of* —nominal modifiers

zaman|Noun|S/S *time*

kurtulamayacak|Verb|S\NP[nom] –*escape* —verb with subject extraction

"Verb+Able"|Able|(S\NP[nom])\(S\NP[nom]) –-*able*—derivation from verb to verb

"Adj+FutPart"|FutPart|(NP/NP)\(S\NP[nom]) –-*FutPart*—conjunction

,|Punc|conj

özgür|Adj|NP –*free*

olamayacak|Verb|(S\NP[nom])\NP –*become*

"Verb+Able"|Able|(S\NP[nom])\(S\NP[nom]) –-*Able*

"Adj+FutPart"|FutPart|(NP/NP)\(S\NP[nom]) –-*FutPart*

bir|Det|NP/NP –*a*

tutsağım|Noun|NP – *prisoner*

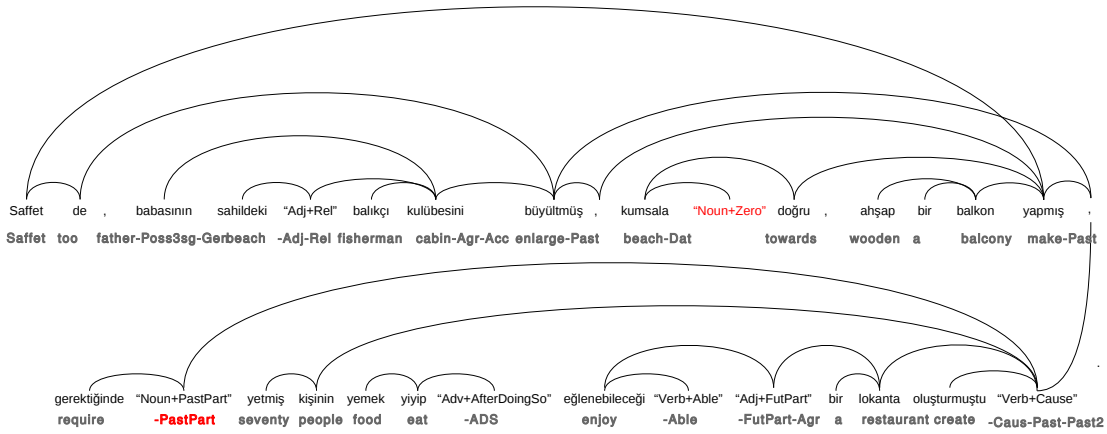
"Verb+Zero"|Zero|(S\NP)/NP[nom] –-*Aor*

ben|Pron|NP[nom] –*I* — post-verbal argument (word order scrambling)

.|Punc|.

A coordination example

This is a relatively long sentence in the corpus. The sentences here are much longer than the average length in the corpus. This is to show that coordination dependencies are correctly recovered with CCG in this example despite the sentence length. In addition to the coordination dependencies represented in the treebank, the dependency representing subject relation between *Saffet* and *yapmış* is captured correctly. However, this could not be carried to the last conjunct because only one C.SUBJECT could be added to the dependency structure in the treebank due to design restrictions. Thus the dependency between *Saffet* and *oluşturmuştu* could not be captured. The input that



Saffet, too, had extended his father’s fishermen’s cabin at the coast, built a wooden balcony towards the beach, created a restaurant where seventy people can eat and have fun if needed.

Figure 9.1: A coordination example with more than two conjuncts

consists of words and categories for this example is given below. More examples of this sort are given in Appendix C.

Input:

Saffet|Noun_Nom|NP[nom]
,|Punc|,
sahildeki|Noun_Loc|NP
balıkçı|Noun_Nom|NP/NP
büyültmüş|Verb|(S\NP[nom])\NP
kumsala|Adj|NP
doğru|Postp|((S\NP[nom])/(S\NP[nom]))\NP
ahşap|Adj|NP/NP
balkon|Noun_Nom|NP
,|Punc|conj
“Noun+PastPart”|Noun_Loc|(S/S)\S
kişinin|Noun_Gen|NP[nom]
yiyip|Verb|(S\NP[nom])\NP
elenebileceği|Verb|S
“Adj+FutPart”|Adj|(NP/NP)\S
lokanta|Noun_Nom|NP
“Verb+Caus”|Verb|((S\NP[nom])\NP)\S
de|Conj|NP[nom]\NP[nom]
babasının|Noun_Gen|NP/NP
“Adj+Rel”|Adj|(NP/NP)\NP
kulübesini|Noun_Acc|NP
,|Punc|conj
“Noun+Zero”|Noun_Dat|NP\NP
,|Punc|,
bir|Det|NP/NP
yapmış|Verb|(S\NP[nom])\NP
gerektiğinde|Verb|S
yetmiş|Num|NP[nom]/NP[nom]
yemek|Noun_Nom|NP
“Adv+AfterDoingSo”|Adv|(S/S)\(S\NP[nom])
“Verb+Able”|Verb|S\S
bir|Det|NP/NP
oluşturmuştu|Verb|S
.|Punc|.

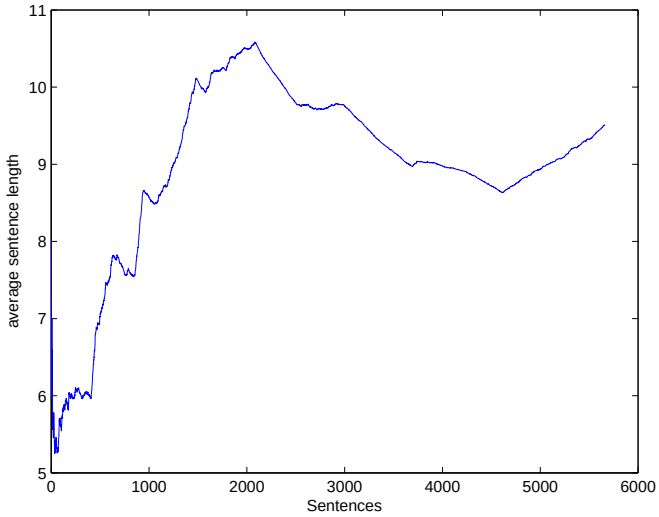


Figure 9.2: Average sentence length throughout the corpus

9.7.2 Sentence Length

The average sentence length for the parsed sentences is slightly lower than overall average sentence length. This indicates that short sentences are easier to parse and a bigger proportion of short sentences are parsed than that of long sentences. However, Figures 9.3 and 9.2 show that after 4500 sentences, there is a steep rise in the length of the sentences, but the coverage is not affected much by this. That part of the treebank is composed of news articles. The sentences in this genre are the longest, however, they show regular behaviour in the sense that they do not contain as much word order variation. This means the coverage is affected more by the quality of the lexicon than the factors like sentence length. We claim this because of the following reasons.

1) The difference between the average sentence lengths for the parsed and unparsed parts of the data is not too big. The average sentence length for the parsed sentences is 10.27 tokens as opposed to 11.23 overall average. This means the average sentence length for unparsed sentences is about 13.3. Therefore, the parser does not only chose the easiest (shortest) sentences but it also takes on the more difficult ones.

2) It is more likely there will be at least one wrong category when the sentences are long. In addition to this, long sentences are often ignored when annotation errors are being checked because of psychological reasons. One tends to correct

the short, easily comprehensible, sentences first.

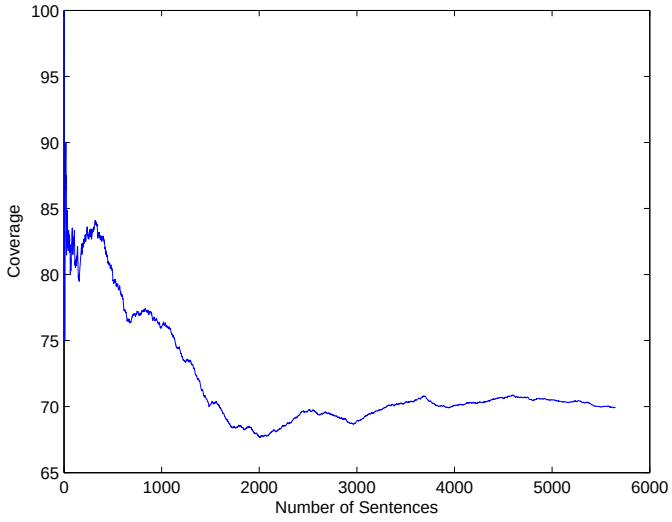


Figure 9.3: Coverage among data

9.8 Supertagging Turkish

Supertagging results are given in this section. The model for the morphemic lexicon performs better than the one for the lexemic lexicon. It is obvious that the data is too small since the supertagger does not benefit from taking only stems of the words as tokens despite the expectation that this would reduce the number of unseen words and hence improve the performance. The lack of improvement indicates that the classifier usually backs off to pos-tags even with the stems as tokens to guess the supertag which results in no improvement on the supertagging accuracy. For the first time, we have used case information for an IG-based experiment. Using case information for IGs with noun and pronoun first tags improved the performance by 11 points. The results in word based supertagging are as expected. Once again extended part-of-speech tags with case information proves to be the best performing tag set.

There are only 114 category types that occur more than 10 times out of 311. This threshold of 10 also needs to be lowered to match the Turkish data which is smaller because when only categories that occur more than 10 times are taken the coverage drops significantly. An experiment was performed with lexemic lexicon to investigate

Morpheme-based

tag-set	accuracy	sentence
BAS	57.84	17.51
CAS(noun+pron)	68.75	23.25
CAS(noun)	68.6	23.05
Stems	57.1	17.01

Word-based

tag-set	accuracy	sentence
BAS	45.46	17.48
CAS	53.43	21.37
EC	56.7	21.96
CRYP	56.44	21.71
CRYP-2	56.34	20.48

Table 9.2: Supertagging results for Turkish (10-fold validation)

this. The best single tagging result on lexemic lexicon is obtained by a category cutoff of 5, and the latest version of the corrected treebank. The category accuracy is 62.21% and the sentence accuracy is 24.93% with this configuration. It is considerably higher than the closest result on lexemic supertagging with category cutoff 10 which is 56.7%. However, reducing this number any further does not give more improvement. We use this model in the lexemic parsing experiments.

Because supertagging is a very important stage in CCG parsing with C&C parser we put a lot of effort in improving the accuracy of the supertagger which was 45% for lexemic lexicon with 10-fold cross validation in the earlier stages. This is a very low accuracy for the supertagger and it cannot be expected to be informative of the category types of the words. For instance for MST paser which only uses the supertags as features, it reduces the dependency accuracy significantly from about 94% to around 71% for unlabelled dependencies. Continuous effort is being put into improving the supertagger by us by including different kinds of morphological information as well as cleaning the data.

Clark and Curran (2004a) use a lexical category dictionary. This includes all lexical categories which appear at least 10 times in Sections 02-21 of CCGbank, resulting in a set of 425 categories. Clark and Curran (2004a) show this set to have very high coverage on unseen data. The supertagger trained on Turkish data was trained with this restriction of frequency on lexical categories in effect in most of the experiments. The coverage on Turkish data is quite high as well (about 97%) with the supertagger assigned categories. However, the recall of the dependencies recovered are not very high with this restriction as shown in Table 9.3.

With all these in effect the multitagger performs above 88% with the most restricted

tagdict cutoff	accuracy	Sentence correct	tag set
1	53.16	15.24	EC
3	53.11	15.20	EC
5	53.13	15.20	EC
5	53.13	15.20	CRYP

Table 9.3: Tag dictionary cutoff variation for morphemic lexicon

lexicon	tag set	accuracy	sentence	t/w
LEX	CRYP	79.64	37.33	2.65
LEX	EC	82.04	41.65	2.82
MORPH	BAS	88.17	45.80	2.40
MORPH	$CAS_{(noun+pron)}$	88.15	45.88	**2.20**
MORPH	$CAS_{(noun)}$	**88.19**	45.75	2.22

Table 9.4: Multitagging results

($\beta = 0.1$) configuration for the morphemic lexicon. Similar results were observed in multitagging with regard to the use of morphological information. Once again, morphemic lexicon beats the lexemic lexicon in supertagging performance. Using both nominal or pronominal case seems to bring the supertagging accuracy by about .02 points. However, it shoul be noted that the tag per category rate is 2.20 for this configuration compared to 2.40 for the basic tags and 2.22 for the best performing configuration. This provides less ambiguity and a faster system when efficiency is important. The parsing results for morphemic lexicon were obtained with the supertagger results of which are shown in the last line.

9.9 Conclusion

We provided results for parsing with two lexicons created from the same set of data annotated with surface syntactic dependencies. We demonstrated that morpheme based lexicons outperform the word-based counterparts in all platforms from parsing to training supertaggers. But more importantly, morphemic lexicon establishes semantically relevant long-range dependencies like those discussed in Example 9.14 in Section

9.7.1, which was taken from from the METU/Sabancı treebank, and which the parser gets completely correct.

Chapter 10

Conclusion and Future Work

A much cleaner Turkish dependency treebank is provided with the improvements discussed in Chapter 2. Annotation mistakes that were in the order of thousands were corrected aiming at a consistent and more reliable treebank. These corrections were applied to the CoNLL 2007 shared task data and were proved to be very important comparing the average parsing performances between CoNLL 2006 and 2007 shared tasks on dependency parsing (Nivre et al., 2007).

The limitations of the annotation style in the Turkish treebank is explored (Chapters 2 and 6). Two very important problems in the treebank were discussed and solutions were proposed. One is recovering the predicate-argument structure in certain forms of extraction, and the other is the coordination scope problem. We show that these problems can be solved effectively to provide a lexicon for a highly representative grammar theory that handles deep linguistic phenomena that some other theories fail to capture (Chapter 6).

The heavy interaction between morphology and syntax in Turkish requires unorthodox methods in parsing. The data need to be generalised and morphological information need to be put into use. This is motivated not only because of sparse data problems but also because of linguistic concerns. The wider scope of morphemes, their interactions with syntax and semantics is explored in various parts of this thesis mostly in Chapters 2, 3 and 6. We argue that a smaller-than-word representation for the lexicon is necessary to explain the bracketing mismatches and morphosyntactic phenomena that are abundant in Turkish data. We show that morphemic lexicon performs qualitatively and quantitatively better than the lexemic lexicon in recovering surface and deep dependencies (Chapters 8 and 9).

Wide coverage parsing results with C&C parser for Turkish are given for both

lexemic and morphemic lexicons in Chapter 9. We show that using morphological information and inherent information in CCG categories both improve the performance of the parser and a morphemic lexicon outperforms the lexemic results in supertagger training models and the parser models.

CCG categories are also used as simple features in MSTParser (Section 8.6.3) which parses dependencies directly using the MST technique. State-of the art dependency parsing results were obtained when CCG categories were used. The aim here is to take this as a lower bound in comparison to results with CCG parsing which provides more informative dependencies. CCG captures "deep" dependencies, in addition to the ones captured by the MSTParser.

In this thesis, we have utilised methods to make the best use of morphological information in a range of parsing systems. We have also provided algorithms to create linguistically meaningful lexicons aiming to model long-distance dependencies and solve linguistic problems such as bracketing paradoxes based on smaller-than-word elements. Making the best use of the information at hand is crucial in supervised systems of NLP for low density languages like Turkish. The methods described here to achieve this are expected to be adaptable to other low-density languages.

10.1 Future Work

- The representational status of the inflectional morphemes other than the ones included in the morphemic lexicon here will also be studied. A complete and correct CCG lexicon for Turkish is our target. Issues like pro-drop in genitive constructions and such are also to be researched in the near future.

- Only coordination involving SENTENCE labels were reannotated with long-distance dependencies in this dissertation. We aim to cover all types of coordination dependencies, and also design a practical and linguistically sound way of annotating gapping constructions in the treebank. This comes under the heading of lexicon improvement.

- We aim to apply morphological analysis and tag disambiguation and integrate this to the parsers to provide some degree of freedom without overcrowding the parser with morphological ambiguity. An existing two-level morphological analyser may be used for this purpose (Çakıcı, 2002).

- Increasing the coverage of the CCG parser to 100% is one of the priorities. This process includes both the correction of the treebank dependencies and the perfection of the CCG grammar induction process. Disambiguation of the marked up categories in the CCG parser should also be considered as some of the loss of coverage is caused by this as explained in the previous chapter.

  There are many English specific constraints hardwired in the C&C parser. We aim to optimise the parameters of the parser according to the needs of Turkish data.

  Forward crossing composition rule in CCG was not implemented in the C&C parser because CCGBank did not have this rule. However, certain word orders and extraposition constructions require this rule to be present. We aim to implement this rule to be able to handle these cases.

- We aim to reseach bracketing paradoxes that involve coordination and such in the future. Inflectional morphemes such as tense participate in suspended affixation, but these are not handled in the current lexicon as they are seen as part of the last IG in the current schema. These can be handled in a similar way to the case markers that have phrasal scope over adjunct phrases.

- In the long term we aim to create a lexicon with semantic representations following Bos et al. (2004). We also would like to explore the ways to bootstrapping the lexicon with the use of unsupervised methods.

Appendix A

Turkish Morpheme Glosses

The following table shows the descriptions of the morphemes that are used throughout the examples in the thesis. Some of these do not correcpond to a surface morpheme. These ones are included in the treebank for information on some aspects. Part-of-speech tags such as *Noun, Verb* are example of this.

tag	desctiption	tag	description
A1pl	1st person pl	NotState	noun-noun derivation
A1sg	1st person singular	Noun	Noun
A2pl	2nd person plural	Num	Numeral
A2sg	2nd person singular	Opt	Opt
Ord	Ordinal number	A3pl	3rd person plural
A3sg	3rd person singular	P1pl	Possesive 1st person plural (our)
Abl	Ablative	P1sg	Possessive 1st person singular
Able	Abilitative	P2pl	Possessive 2nd person plural
Acc	Accusative	P2sg	Possessive 2nd person singular
Acquire	derivational morph.	P3pl	Possessive 3rd person plural
Adj	Adjective	P3sg	Possessive 3rd person singular
Adv	Adverb	Pass	Passive
AfterDoingSo	verb-adverb derivation	Past	Past tense
Agt	Agentative	PastPart	Relativisation morpheme
Aor	Aorist	PCAbl	Particle requiring Ablative
As	adjective-adverb derivation	PCAcc	Particle requiring accusative
AsIf	adjective-adverb derivation	PCDat	Particle requiring dative
Become	noun-verb derivation	PCGen	Particle requiring genitive
ByDoingSo	verb-adverb derivation	PCIns	Particle requiring instrumental

Table A.1: Morpheme descriptions

tag	desctiption	tag	description
Card	Cardinal number	PCNom	Particle requiring nominative
Caus	Causative	PersP	Personal pronoun
Cond	Conditional	Pnon	No possessive marker
Conj	Conjunctive	Pos	Positive
Cop	Copular	Postp	Postposition
Dat	Dative	Pres	Present
DemonsP	DemonsP	PresPart	Relativiser morpheme
Desr	Deontic modality	Prog1	Progressive type 1
Det	Determiner	Prog2	Progressive type 2
Distrib	Distributive	Pron	Pronoun
Dup	Duplicate	Prop	Proper noun
Equ	Equative	Punc	Punctuation
FitFor	noun-noun derivation	Ques	Question particle
Fut	Future	QuesP	Question pronoun
FutPart	Relativiser morpheme	Range	Range
Gen	Genitive	Real	Real number
Hastily	verb-adverb derivation	Recip	Reciprocal
Imp	Implicative	Reflex	Reflexive
InBetween	noun-noun derivation	ReflexP	Reflexive Pronoun
Inf	Infinitive	Rel	Relativiser morpheme (nouns)
Ins	Instrumental	Related	noun-adj derivation
Interj	Interjection	Since	noun-adverb derivation
JustLike	JustLike	SinceDoingSo	verb-adverb derivation
Loc	Locative	Stay	verb-verb derivation
Ly	adjective-adverb derivation	Time	time
Narr	Narrative	Verb	Verb
Neces	Epistemic modality	When	verb-adverb derivation
Neg	Negative	While	verb-adverb derivation
With	noun-noun derivation	Without	noun-noun derivation
Ness	noun-noun derivation	WithoutHavingDoneSo	verb-adverb derivation
Nom	Nominative	Zero	zero morpheme

Table A.2: Morpheme descriptions Continued

Appendix B

IG types in METU-Sabancı Treebank

“Adj+Agt” –HcH (Noun-Adj) (Verb-Adj)
“Adj+AsIf” – CAsHnA – cA –esiye (olduresiye) (Verb-Adj and Adj-Adj)
“Adj+FitFor” –lHk (Noun-Adj)
“Adj+FutPart” –AcAk (Verb-Adj)
“Adj+InBetween” – arasI (Noun-Adj)
“Adj+JustLike” –HmsH (Noun-Adj)
“Adj+PastPart” –dHk (Verb-Adj) These two morphemes are not named according to their semantic role. They are named after their phonological properties. They (Pres-part) can be used for both subject extraction and other type of extractions, thus they are not named in a different way.
“Adj+PresPart” –yAn (Verb-Adj)
“Adj+Rel” –ki (Noun-Adj)
“Adj+Related” –Hk (ex: antropolojik) –sAl (ex: ulusal) (Noun-Adj)
“Adj+With” –lH (Noun-Adj)
“Adj+Without” –sHz (Noun-Adj)
“Adj+Zero”
“Adv+AfterDoingSo” -Hp (Verb-Adv)
“Adv+As” –CA (Verb-Adv) “Adv+AsIf” –CAsHnA (Verb-Adv)
“Adv+ByDoingSo” –ArAk (Verb-Adv)
“Adv+Ly” –CA (Adj-Adv)
“Adv+Since” –DHr (Noun-Adv) – makes temporal adverbs such as yıllardir–*for years*
“Adv+SinceDoingSo” –AlH (Verb-Adv)
“Adv+When” –HncA (Verb-Adv)
“Adv+While” –ken (Verb-Adv)

“Adv+WithoutHavingDoneSo” –mAdAn (Verb-Adv)
“Noun+Agt” –HcH (Noun-Noun) (Verb-Noun)
“Noun+FutPart” –AcAk (Verb-Noun)
“Noun+Inf” –mAk –yHş –mA (Verb-Noun)
“Noun+Ness” –lHk (Noun-Noun) (Adj-Noun)
“Noun+NotState” –mEzlHk (Verb-Noun)
“Noun+PastPart” –DHGH (Verb-Noun)
“Noun+Zero”
“Pron+A3pl”
“Pron+A3sg”
“Verb+Able” –AbHl (Verb-Verb)
“Verb+Acquire” –lAn (Noun-Verb)
“Verb+Become” –lAş (Noun-Verb)
“Verb+Caus” –DHr -Hr -Ht -t (Verb-Verb)
“Verb+Hastily” –Hver (Verb-Verb)
“Verb+Pass” –Hl –Hn (Verb-Verb)
“Verb+Recip” –Hş (Verb-Verb)
“Verb+Reflex” –Hn (Verb-Verb)
“Verb+Stay” –(y)Akal (Verb-Verb)
“Verb+Zero

Appendix C

Some C&C parsing examples

Parsing results of various sentences are given here. The results are given in the form of output text followed by the parser input. The parser is run in the dependency output mode, thus each line represents a dependency between two words.

This is an example of an extraction from the adjunct clause. With the word-based lexicon the categories induced do not give the correct parse. By separating the relativisation morpheme *-dIGI* and giving it the correct category we get all the predicate argument relations.[1]

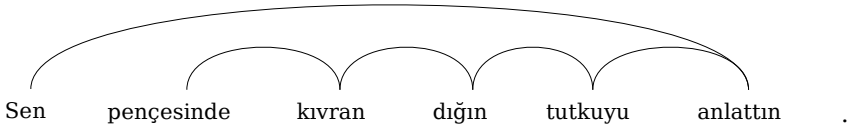
```
pencesinde_2 (S{Y}/S{Y}<1>){_} 1 kvran_3 0
digin_4 ((NP{Y}/NP{Y}<1>){_}\S{Z}<2>){_} 2 kvran_3 0
digin_4 ((NP{Y}/NP{Y}<1>){_}\S{Z}<2>){_} 1 tutkuyu_5 0
anlattin_6 ((S{_}\NP[nom]{Y}<1>){_}\NP{Z}<2>){_} 2 tutkuyu_5 0
anlattin_6 ((S{_}\NP[nom]{Y}<1>){_}\NP{Z}<2>){_} 1 Sen_1 0
```

Input:

Sen|NP[nom] *–you*
pençesinde|S/S *–claw-Poss3sg-Loc*
kıvran|S *–struggle*
dığın|(NP/NP)\S *–PastPart*
tutkuyu|NP *passion-Acc*
anlattın|(S\NP[nom])\NP *–tell-Past-P2sg*
.|.

You spoke about the passion in claws of which you struggle.

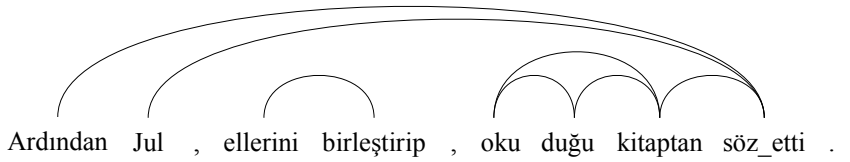
[1] Turkish characters were changed into English counterparts in the parser outputs for readibility.



Another example with an object relative is:

```
birlestirip_5 (((S{Y}\NP[nom]{Z}){Y}/(S{Y}\NP[nom]{Z}){Y}){_}\NP{V}<1>){_} 3 ellerini_4 0
dugu_8 ((NP{Y}/NP{Y}<1>){_}\(S{Z}<2>\NP{Y})){_} 2 oku_7 0
oku_7 (S{_}\NP{Y}<1>){_} 1 kitaptan_9 0
dugu_8 ((NP{Y}/NP{Y}<1>){_}\(S{Z}<2>\NP{Y})){_} 1 kitaptan_9 0
soz_etti_10 ((S{_}\NP[nom]{Y}<1>){_}\NP{Z}<2>){_} 2 kitaptan_9 0
soz_etti_10 ((S{_}\NP[nom]{Y}<1>){_}\NP{Z}<2>){_} 1 Jul_2 0
Ardindan_1 (S{Y}/S{Y}<1>){_} 1 soz_etti_10 0
```

Input:
Ardından|S/S
Jul|NP[nom]
,|,
ellerini|NP
birleştirip|((S\NP[nom])/(S\NP[nom]))\NP
,|,
oku|S\NP
duğu|(NP/NP)\(S\NP)
kitaptan|NP
söz_etti|(S\NP[nom])\NP
.|.



In the above example, *birleştirip* is not connected to the rest of the dependency structure. This is because the category assigned to it is not informative enough. It is unable to represent all the dependencies this adverb licences. Because it is a derived adverb, it needs to take its arguments before it is turned into an adverb. Note that the morphemic lexicon solves this problem.

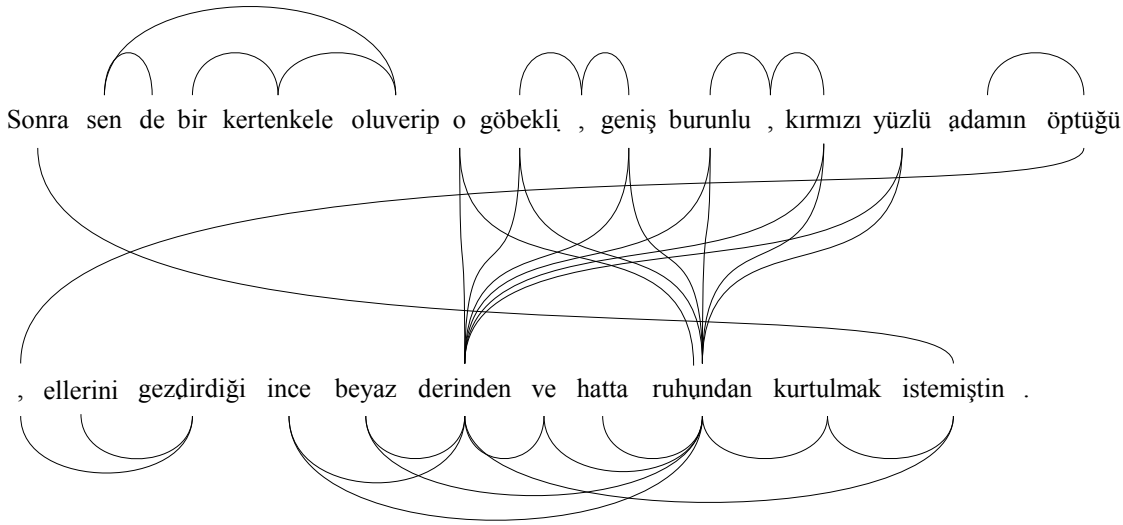
Long-distance dependencies that cannot be recovered with the lexemic lexicon are also recovered in the morphemic lexicon. The morpheme that controls the object extraction (*-dugu*) ensures the correct predicate-argument structure is recovered between *okuduğu* and *kitap* as shown below.

```
kucuk_5 ((NP{Y}/NP{Y}){Z}/(NP{Y}/NP{Y}){Z}<1>){_} 2 masanin_6 0
masanin_6 (NP{Y}/NP{Y}<1>){_} 1 uzerinde_7 0
"-nDA"_8 (((S{Y}\NP{Z}){Y}/(S{Y}\NP{Z}){Y}<1>){_}\NP{W}<2>){_} 3 uzerinde_7 0
"Verb+Caus"_10 (S{Y}\S{Y}<1>){_} 1 birlestirip_9 0
"-nDA"_8 (((S{Y}\NP{Z}){Y}/(S{Y}\NP{Z}){Y}<1>){_}\NP{W}<2>){_} 2 birlestirip_9 0
birlestirip_9 (S{_}\NP{Y}<1>){_} 1 ellerini_4 0
birlestirip_9 (S{_}\NP{Y}<1>){_} 1 Jul_2 0
"Adv+AfterDoingSo"_11 ((S{Y}/S{Y}<1>){_}\S{Z}<2>){_} 2 birlestirip_9 0
"Adj+PastPart"_14 ((NP{Y}/NP{Y}<1>){_}\(S{Z}<2>\NP{Y}){Z}){_} 2 okudugu_13 0
okudugu_13 (S{_}\NP{Y}<1>){_} 1 kitaptan_15 0
"Adj+PastPart"_14 ((NP{Y}/NP{Y}<1>){_}\(S{Z}<2>\NP{Y}){Z}){_} 1 kitaptan_15 0
soz_etti_16 (S{_}\NP[nom]{Y}<1>){_} 1 kitaptan_15 0
"Adv+AfterDoingSo"_11 ((S{Y}/S{Y}<1>){_}\S{Z}<2>){_} 1 soz_etti_16 0
Ardindan_1 (S{Y}/S{Y}<1>){_} 1 soz_etti_16 0
```

Input:

Ardından|Adv|S/S
Jul|Noun_Nom|NP[nom]
,|Punc|,
ellerini|Noun_Acc|NP
küçük|Adj|(NP/NP)/(NP/NP)
masanın|Noun_Gen|NP/NP
üzerinde|Noun_Loc|NP
”-nDA”|Loc|((S\NP)/(S\NP))\NP
birleştirip|Verb|S\NP
”Verb+Caus”|Verb|S\S
”Adv+AfterDoingSo”|Adv|(S/S)\S
,|Punc|,
okuduğu|Verb|S\NP
”Adj+PastPart”|Adj|(NP/NP)\(S\NP)
kitaptan|Noun_Abl|NP
söz_etti|Verb|S\NP[nom]
.|Punc|.

This is an example from the lexemic lexicon. The problem with this parse is the word-word dependencies are wrong because relativisation relations are not modeled.




```
de_3 (NP[nom]{Y}\NP[nom]{Y}<1>){_} 1 sen_2 0
bir_4 (NP{Y}/NP{Y}<1>){_} 1 kertenkele_5 0
oluverip_6 (((NP{Y}/NP{Y}){_}\NP[nom]{W}<1>){_}\NP{V}<2>){_} 3 kertenkele_5 0
oluverip_6 (((NP{Y}/NP{Y}){_}\NP[nom]{W}<1>){_}\NP{V}<2>){_} 2 sen_2 0
,_9 conj 1 genis_10 0
,_9 conj 1 gobekli_8 0
,_12 conj 1 kirmizi_13 0
,_12 conj 1 burunlu_11 0
optugu_16 ((NP{Y}/NP{Y}){_}\NP{Z}<1>){_} 2 adamin_15 0
gezdirdigi_19 ((NP{Y}/NP{Y}){_}\NP{Z}<1>){_} 2 ellerini_18 0
,_17 conj 1 gezdirdigi_19 0
,_17 conj 1 optugu_16 0
kurtulmak_26 (NP{Y}\NP{Y}<1>){_} 1 ruhundan_25 0
hatta_24 (NP{Y}/NP{Y}<1>){_} 1 ruhundan_25 0
ve_23 conj 1 ruhundan_25 0
ve_23 conj 1 derinden_22 0
beyaz_21 (NP{Y}/NP{Y}<1>){_} 1 ruhundan_25 0
beyaz_21 (NP{Y}/NP{Y}<1>){_} 1 derinden_22 0
ince_20 (NP{Y}/NP{Y}<1>){_} 1 ruhundan_25 0
ince_20 (NP{Y}/NP{Y}<1>){_} 1 derinden_22 0
yuzlu_14 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 ruhundan_25 0
yuzlu_14 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 derinden_22 0
kirmzi_13 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 ruhundan_25 0
kirmizi_13 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 derinden_22 0
burunlu_11 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 ruhundan_25 0
burunlu_11 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 derinden_22 0
genis_10 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 ruhundan_25 0
genis_10 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 derinden_22 0
gobekli_8 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 ruhundan_25 0
```

```
gobekli_8 (NP[nom]{Y}/NP[nom]{Y}<1>){_} 1 derinden_22 0
o_7 (NP{Y}/NP{Y}<1>){_} 1 ruhundan_25 0
o_7 (NP{Y}/NP{Y}<1>){_} 1 derinden_22 0
istemistin_27 (S{_}\NP{Y}<1>){_} 1 ruhundan_25 0
istemistin_27 (S{_}\NP{Y}<1>){_} 1 derinden_22 0
Sonra_1 (S{Y}/S{Y}<1>){_} 1 istemistin_27 0
```

Input:

Sonra|Adv|S/S –*then*

sen|Pron|NP[nom]–*you*

de|Conj|NP[nom]\NP[nom] – *too*

bir|Det|NP/NP – *a*

kertenkele|Noun_Nom|NP –*lizard*

oluverip|Adv_Verb|((NP/NP)\NP[nom])\NP –*be-Hastily-AfterDoingSo*

o|Det|NP/NP *that*

göbekli|Adj_Noun|NP[nom]/NP[nom] –*belly-With*

,|Punc|conj

geniş|Adj|NP[nom]/NP[nom] – *wide*

burunlu|Adj_Noun|NP[nom]/NP[nom] –*nose-With*

,|Punc|conj

kırmızı|Adj|NP[nom]/NP[nom] –*red*

yüzlü|Adj_Noun|NP[nom]/NP[nom] – *face-With*

adamın|Noun_Gen|NP[nom] – *man-Gen*

öptüğü|Adj_Verb|(NP/NP)\NP –*kiss-PastPart*

,|Punc|conj

ellerini|Noun_Acc|NP –*hands-Acc*

gezdirdiği|Adj_Verb|(NP/NP)\NP – *move-PastPart*

ince|Adj|NP/NP –*thin*

beyaz|Adj|NP/NP –*white*

derinden|Noun_Abl|NP –*skin-Poss2sg-Abl*

ve|Conj|conj –*and*

hatta|Conj|NP/NP – *even*

ruhundan|Noun_Abl|NP – *soul-Poss2sg-Abl*

kurtulmak|Noun_Verb|NP\NP – *escape-Inf*

istemiştin|Verb|S\NP – *want-Narr-Past-P2sg*

.|Punc|.

References

Abeillé, A., editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

Abney, Steven P. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

Ades, A.E. and Mark Steedman. 1982. On the order of words. *Linguistics and Philosophy*, 4:517–558.

Ajdukiewicz, Kazimierz. 1935. Die syntaktische konnexitat. In *Polish Logic, ed. Storrs McCall, Oxford University Press*, pages 207–231.

Antworth, Ewan L. 1990. *PC-KIMMO: A two-level Processor for Morphological Analysis*. Summer Institure of Linguistics, Dallas.

Atalay, Nart B., Kemal Oflazer, and Bilge Say. 2003. The annotation process in the Turkish Treebank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

Bach, Emmon. 1983. On the relationship between word-grammar and phrase-grammar. *Natural Language and Linguistics Theory*, 1:65–89.

Bach, Emmon. 1984. Some generalizations of categorial grammars. In F. Landman and F. Veltman, editors, *Varieties of Formal Semantics*. Foris, Dordrecht, pages 1–23.

Baldridge, Jason and Geert-Jan M. Kruijff. 2003. Multi-modal Combinatory Categorial Grammar. In *Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics*, pages 211–218, Budapest, Hungary.

Baldridge, Jason M. 2000. Strong equivalence of CCG and Set-CCG. ms. University of Edinburgh, 2000.

Baldridge, Jason M. 2002. *Lexically Specified Derivation Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

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

Bar-Hillel, Yehoshua. 1953. A quasi-arithmetic description for syntactic description. *Language*, 29:47–58.

Bar-Hillel, Yehoshua. 1964. *Language and Information*. Addison-Wesley, Reading.

Bar-Hillel, Yehoshua, C. Gaifman, and E. Shamir. 1964. On categorial and phrase structure grammars. In *Language and Information ed. Bar-Hillel, Addison-Wesley*, pages 99–115.

Barbero, Christina, Leonardo Lesmo, Vincenzo Lombardo, and Paola Merlo. 1998. Integration of syntactic and lexical information in a hierarchical dependency grammar. In *Proceedings of the Workshop on Processing of Dependency-Based Grammars (COLING-ACL)*, pages 58–67, Montreal, Canada.

Bikel, Daniel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, San Francisco.

Bird, Steven. 1990. *Constraint-based Phonology*. Ph.D. thesis, University of Edinburgh.

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, pages 1240–1246, Geneva, Switzerland.

Bozşahin, Cem. 1998. Deriving the predicate-argument structure for a free word order language. In *Proceedings of COLING-ACL'98, Montreal*, pages 167–173.

Bozşahin, Cem. 2002. The Combinatory morphemic lexicon. *Computational Linguistics*, 28(2):145–186.

Bozşahin, Cem. to appear. Word order, word order flexibility and the lexicon. In Sumru Özsoy, editor, *Theoretical Issues in Word Order*. Kluwer.

Brants, Sabine, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra1, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2:597–619.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In E. Hinrichs and K. Simov, editors, *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

Bresnan, Joan. 1982. *The mental representation of grammatical relations*. MIT Press, Cambridge, MA.

Briscoe, Ted and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the parc depbank. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the 10th Conf. on Computational Natural Language Learning (CoNLL-X)*. SIGNLL.

Butterworth, Brian. 1983. Lexical representation. In *Language production Vol.2: Development, writing and other language processes*. Academic Press, London, pages 257–294.

Cahill, Aoife, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics.*, 34(1):81–124.

Cahill, Aoife, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage pcfg-based LFG approximations. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 320–327, Barcelona, Spain. Association for Computational Linguistics.

Cahill, Aoife, Mairéad McCarthy, Josef van Genabith, and Andy Way. 2002. Parsing-with PCFGs and automatic f-structureannotation. In *Proc. of the Seventh International Conference on LFG*, pages 76–95, Palo Alto, CA.

Carpenter, Bob. 1991. The generative power of Categorial grammars and Head-driven Phrase Structure Grammars with lexical rules. *Computational Linguistics*, 17(3):301–314.

Carpenter, Bob. 1992. Categorial grammars, lexical rules and the English predicative. In R. Levine, editor, *Formal Grammar:Theory and Implementation*. Oxford Press, page chapter 3.

Carpenter, Bob. 1999. The Turing-completeness of Multimodal Categorial Grammars. In *Papers presented to Johan van Benthem in Honor of his 50th Birthday, ESSLLI, Utrecht*.

Carroll, John, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.

Çakıcı, Ruken. 2002. A computational interface for syntax and morphemic lexicons. Master's thesis, Middle East Technical University.

Çakıcı, Ruken. 2005. Automatic induction of a CCG grammar for Turkish. In *ACL Student Research Workshop*, pages 73–78, Ann Arbor, MI, USA.

Çakıcı, Ruket and Jason Baldridge. 2006. Projective and non-projective Turkish parsing. In *Proceedings of the TLT 2006*, pages 19–30, Prague, Czech Republic.

Çetinoğlu, Özlem and Kemal Oflazer. 2006. Morphology-syntax interface for Turkish LFG. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 153–160, Sydney, Australia.

Cha, Jeongwon and Geunbae Lee. 2000. Structural disambiguation of morpho-syntactic Categorial parsing for Korean. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 1002–1006, Saarbrucken. CoLing, ACL.

Cha, Jeongwon, Geunbae Lee, and JongHyeok Lee. 2002. Korean Combinatory Categorial Grammar and statistical parsing. *Computers and the Humanities*, 36(4):431–453.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. of the AAAI*, pages 598–603, Providence, RI. AAAI Press/MIT Press.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139. Morgan Kaufmann Publishers Inc.

Chen, John and K. Vijay-shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, Trento, Italy.

Chomsky, Noam. 1970. Remarks on nominalization. In R. Jacobs and P. Rosenbaum, editors, *Readings in Transformational Grammar*. Ginn and Co., Waltham, MA, pages 184–221.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. Studies in Language. Harper & Row, New York,, New York.

Clark, Stephen. 2002. A supertagger for Combinatory Categorial Grammar. In *Proceedings of the 6th International TAG+ Workshop*, pages 19–24, Venice, Italy.

Clark, Stephen and James R. Curran. 2003. Log-Linear models for wide-coverage CCG parsing. In *In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 97–104, Sapporo, Japan.

Clark, Stephen and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, pages 282–288, Geneva, Switzerland.

Clark, Stephen and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 104–111, Barcelona, Spain.

Clark, Stephen and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the*

NAACL, Main Conference, pages 144–151, New York City, USA, June. Association for Computational Linguistics.

Clark, Stephen and James R. Curran. 2007a. Formalism-independent parser evaluation with CCG and Depbank. In *Proceedings of the 45th meeting of the ACL*, pages 248–255, Prague, Czech Republic, June.

Clark, Stephen and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

Clark, Stephen, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Clark, Stephen, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP-04*, pages 111–118, Barcelona, Spain.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th Annual Meeting of the ACL*, pages 16–23, Madrid, Spain.

Collins, Michael. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, Michael, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, Maryland, USA.

Collins, Michael J. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting of the Association for Computational Linguistics*, pages 184–191.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*.

Şehitoğlu, Onur and Cem Bozşahin. 1999. Lexical rules and lexical organization: Productivity in the lexicon. In Evelyn Viegas, editor, *Breadth and Depth of Semantic Lexicons*. Kluwer, pages 39–57.

Curran, James R., Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 44th Meeting of the ACL*, pages 697–704, Sydney, Australia, July.

Curry, Haskell B. and Robert Feys. 1958. *Combinatory Logic: Vol.1*. North Holland.

Daelemans, Walter. 1999. Memory-based language processing. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:287–292.

Dienes, Péter and Amit Dubey. 2003. Deep syntactic processing by combining shallow methods. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 431–438, Sapporo, Japan.

Dowty, David R. 1979. *Word Meaning and Montague Grammar*. Reidel, Dordrect.

Dubey, Amit and Frank Keller. 2003. Probabilistic parsing for German using sisterhead dependencies. In *In ACL '03: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan.

Duchier, Denys. 2001. Lexicalized syntax and topology for non-projective dependency grammar. In *Proceedings of the Joint Conference on Formal Grammars and Mathematics of Language FGMOL*, Helsinki.

Dyer, Christopher J. 2007. The "noisier channel": Translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 207–211, Prague, Czech Republic, June. Association for Computational Linguistics.

Eisner, Jason. 1996a. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 79–86, Santa Cruz, California.

Eisner, Jason. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.

Eisner, Jason. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H Bunt and A Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*. Kluwer, Dordrect, pages 29–62.

Erguvanlı, Eser. 1979. *The Function of Word Order in Turkish*. Ph.D. thesis, University of California.

Eryiğit, Gülşen, Joakim Nivre, and Kemal Oflazer. 2006. The incremental use of morphological information and lexicalization in data-driven dependency parsing. In *ICCPOL*, pages 498–508.

Eryiğit, Gülşen, Joakim Nivre, and Kemal Oflazer. 2008. Dependency parsing of Turkish. *Computational Linguistics*, 34(3).

Eryiğit, Gülşen and Kemal Oflazer. 2006. Statistical dependency parsing of Turkish. In *Proceedings of the* 11^{th} *Annual Meeting of the EACL*, pages 89–96, Trento, Italy.

Foth, Kilian, Tomas By, and Wolfgang Menzel. 2006. Guiding a constraint dependency parser with supertags. In *Proc. 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 289–296.

Fukushima, Kazuhiko. 1999. Morphology and logical form. *Journal of Linguistics*, 35:297–320.

Gaifman, Haim. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–345.

Gazdar, Gerald. 1988. Applicability of indexed grammars to natural languages. In Uwe Reyle and Christian Rohrer, editors, *Natural Language Parsing and Linguistic Theory*. Reidel, Dordrecht, pages 69–94.

Geman, S., E. Bienenstock, and R. Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural COmputation*, 4:1–58.

Gerdes, Kim and Sylvain Kahane. 2001. Word order in German: A formal dependency grammar using a topological hierarchy. In *Proc. of the 39th Annual Meeting of the ACL*, pages 220–227, Toulouse, France.

Gladkij, Alexej V. and Igor A. Mel'čuk. 1975. Tree grammars: I. a formalism for syntactic transformations in natural languages. *Linguistics*, 50:47–82.

Göçmen, Elvan, Onur T. Şehitoğlu, and Cem Bozşahin. 1995. An outline of Turkish syntax. Technical Report 95-2, Department of Computer Engineering, METU.

Gross, Maurice. 1964. On the equivalence of models of language used in the fields of mechanical translation and information retrieval. *Information Storage and Retrieval*, 2(1):43–57.

Güngör, Tunga. 1995. *Computer Processing of Turkish: Morhological and Lexical*. Ph.D. thesis, Bogazici Unoversity, Istanbul, Turkey.

Hajič, Jan. 1998. Building a syntactically annotated corpus: Prague dependency treebank. In E. Hajicová and B. Hladká, editors, *Issues of Valency and Meaning, Studies in Honour of Jarmila Panevová*. Charles University Press, Karolinum, Prague, pages 106–132.

Hakkani-Tür, Dilek, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36:381–410.

Hankamer, Jorge. 1986. Finite state morphology and left to right phonology. In *Proceedings of the West Coast Conference on Formal Linguistics*, volume 5.

Hankamer, Jorge. 1989. Morphological parsing and the lexicon. In William Marslen-Wilson, editor, *Lexical representation and process*. MIT Press, Cambridge, MA, USA, pages 392–408.

Hays, David G. 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525.

Hellwig, Peter. 1986. Dependency unification grammar. In *Proceedings of the 11th coference on Computational linguistics*, pages 195–198, Bonn, Germany.

Hellwig, Peter. 2003. Dependency unification grammar. In Eichinger Agel, V., Eroms L.M., Hellwig H.-W., Heringer P., H. J., and H. Lobin, editors, *Dependency and Valency*. Walter de Gruyter, Berlin, Germany, pages 593–635.

Hendriks, Herman. 1993. *Studied Flexibility: Categories and Types in Syntax and Semantics*. Ph.D. thesis, University of Amsterdam.

Hepple, Mark. 1987. Methods for parsing Combinatory Categorial Grammar and the spurious ambiguity problem. Master's thesis, University of Edinburgh.

Hepple, Mark. 1990. *The Grammar and Processing Order and Dependency: A Categorial approach*. Ph.D. thesis, University of Edinburgh.

Hepple, Mark and Glyn Morrill. 1989. Parsing and derivational equivalence. In *Proceedings of the 4th Conference of the European chapter of the Association for Computational Linguistics*, pages 1–10, Manchester.

Hockenmaier, Julia. 2003a. *Data Models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Hockenmaier, Julia. 2003b. Parsing with generative models of predicate-argument structure. In *In Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, pages 359–366.

Hockenmaier, Julia. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, Sydney, Australia, July. Association for Computational Linguistics.

Hockenmaier, Julia and Mark Steedman. 2005. CCGbank manual. Technical Report MS-CIS-05-09, Department of Computer and Information Science, University of Pennsylvania.

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

Hockett, Charles F. 1954. Two models of grammatical description. *Word*, 10:210–231.

Hoeksema, Jack. 1984. *Categorial Morphology*. Ph.D. thesis, Rijksuniversiteit Groningen, New York.

Hoeksema, Jack and R. Janda. 1988. Implications of process morphology for categorial grammar. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*. Reidel, Dordrecht, pages 199–248.

Hoekstra, H., M. Moortgat, I. Schuurman, and T. Wouden. 2001. Syntactic annotation for the spoken Dutch corpus project (CGN). In Walter Daelemans, editor, *Proceedings of CLIN2000*.

Hoffman, Beryl. 1995. *The Computational Analysis of the Syntax and Interpretation of "Free" Word Order in Turkish*. Ph.D. thesis, University of Pennsylvania.

Hudson, Richard A. 1984. *Word Grammar*. Blackwell, Oxford.

Hudson, Richard A. 1990. *English Word Grammar*. Blackwell, Oxford.

Hudson, Richard A. 2007. *Language Networks. The New Word Grammar*. Oxford University Press, Oxford.

Johnson, Mark. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting on ACL*, pages 136–143, Philadelphia, Pennsylvania.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, Cambridge, UK, pages 206–250.

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall.

Kabak, Barış. 2007. Turkish suspended affixation. *Linguistics*, 45(2):311–348.

Kahane, Sylvain, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proceedings of COLING '98-ACL*, pages 646–652.

Karttunen, Lauri. 1989. Radical lexicalism. In Mark Baltin and Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, pages 43–65.

Karttunen, Lauri and Kenneth R. Beesley. 2001. A short history of two-level morphology. In *ESSLLI 2001 Lecture Notes*, August.

Kay, Martin and Ronald Kaplan. 1983. Word recognition. ms. Xerox Palo Alto Research Center.

King, Tracy H., Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. Parc 700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

Köksal, A. 1973. *Automatic Morphological Analysis of Turkish*. Ph.D. thesis, Hacettepe University, Ankara, Turkey.

Komagata, Nobo. 1997. Efficient parsing for CCGs with generalized type raised categories. In *Proceedings of the 5th International Workshop in Parsing Technologies*, pages 135–146, Boston, MA.

Komagata, Nobo. 2004. A solution to the spurious ambiguity problem for practical Combinatory Categorial Grammar parsers. *Computer Speech and Language*, 18:91–103.

König, Esther. 1989. Parsing as natural deduction. In *Proceedings of the 27th Annual meeting of the Association for Computational Linguistics*, pages 272–279, Vancouver.

Kornfilt, Jaklin. 1997. *Turkish*. Routledge.

Koskenniemi, Kimmo. 1983. Two-level model for morphological analysis. In *International Joint Conference on Artificial Intelligence,IJCAI-83*, pages 683–685.

Kruijff, Geert-Jan. 2001. *Categorial-Modal Architecture of Informativity: Dependency Grammar Logic and Information Structure*. Ph.D. thesis, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic.

Kruijff, Geert-Jan and Jason M. Baldridge. 2000. Relating Categorial Type Logics and CCG through simulation. ms. University of Edinburgh, 2000.

Kudo, T. and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*.

Kudo, T. and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of the 6th Conf. on Computational Natural Language Learning (CoNLL)*.

Kuhlmann, Marco and Mathias Möhl. 2007. Mildly context-sensitive dependency languages. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Prague, Czech Republic.

Levy, Roger. 2005. *Probabilistic models of Word Order and Syntactic Discontinuity*. Ph.D. thesis, Stanford University.

Levy, Roger and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of ACL*.

Lewis, G. L. 1967. *Turkish Grammar*. Clarendon, Oxford.

Lieber, Rochelle. 1980. *On the Organization of the Lexicon*. Ph.D. thesis, MIT.

Lieber, Rochelle. 1992. *Deconstructing Morphology: Word formation in Syntactic Theory*. University of Chicago Press, Chicago.

Lin, Dekang. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

Lombardo, Vincenzo and Leonardo Lezmo. 2000. A formal theory of dependency syntax with non lexical units. *Journal de Traitment Automatique des Langues*, 41(1):179–210.

Magerman, David M. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283.

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

Marslen-Wilson, William. 1999. Abstractness and combination: The morphemic lexicon. In Simon Garrod and Martin J. Pickering, editors, *Language Processing*. Psychology Press, East Sussex, UK, pages 101–119.

McDonald, Ryan. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, Ann Arbor, MI, USA.

McDonald, Ryan, K. Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of the 10th Conf. on Computational Natural Language Learning (CoNLL-X)*.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the* 11^{th} *Annual Meeting of the EACL*.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP 2005*, Vancouver, B.C.

Mel'čuk, Igor A. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press, New York.

Menzel, Wolfgang and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In *Proc. Workshop on Processing of Dependency-Based Grammars*, pages 78–87.

Miyao, Yusuke, Takashi Nonomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of RANLP*, pages 285–291.

Moortgat, Michael. 1997. Categorial Type Logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Langauge*. Elsevier Science B.V., New York.

Moortgat, Michael and Richard Moot. 2001. CGN to Grail: Extracting a type-logical lexicon from the CGN annotation. In Walter Daelemans, editor, *Proc. of CLIN 2000*, pages 126–143.

Moortgat, Michael and Richard Moot. 2002. Using the spoken Dutch corpus for Type-Logical grammar induction. In *Proc. of the 3rd International Language Resources and Evaluation Conference*, Las Palmas.

Morrill, Glyn. 1994. *Type-logical Grammar*. Kluwer, Dordrecht.

Müller, Stefan. 2003. Solving the bracketing paradox: an analysis of the morphology of German particle verbs. *Journal of Linguistics*, 39:275–325.

Nasr, Alexis. 1998. A formalism and a parser for lexicalised dependency grammars. In *Proceedings of International Workshop on Parsing Technologies*, pages 186–195, Prague.

Neuhaus, Peter and Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proceedings of ACL/EACL*, pages 337–343, Madrid.

Neumann, G. 1998. Automated extraction of stochastic lexicalized tree grammars from treebanks. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammar and Related Frameworks*, pages 120–123.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of 8th International Workshop on Parsing Technologies*, pages 149–160.

Nivre, Joakim. 2006. *Inductive Dependency Parsing*. Springer.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Deniz Yüret, Jens Nilsson, and Sebastian Riedel. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *CoNNL 2004*, pages 49–56, Boston, Massachusetts, USA.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. natural language engineering. *Natural Language Engineering*, 13(2):95–135.

Nivre, Joakim, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *CoNLLX*, pages 221–225, New York City.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 99–106.

Nivre, Joakim and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING)*, pages 64–70.

Oehrle, Richard T. to appear. Multi-modal type-logical grammar. In Robert D. Borsley and Kersti Borjars, editors, *Non-transformational Syntax: A Guide to Current Debate*. Blackwell, Oxford, UK.

Oflazer, Kemal. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 6(2).

Oflazer, Kemal. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544.

Oflazer, Kemal and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Oflazer, Kemal, Elvan Göçmen, and Cem Bozşahin. 1994. An outline of Turkish morphology. Technical Report TU-LANGUAGE, NATO Science Division SfS III, Brussels.

Oflazer, Kemal, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gokhan Tür. 2003. Building a Turkish Treebank. In Abeillé (Abeillé, 2003), pages 261–277.

Oflazer, Kemal and Gökhan Tür. 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 69–81.

Oflazer, Kemal and Gökhan Tür. 1997. Morphological disambiguation by voting constraints. In *Proceedings of the 35th Annual Meeting of the ACL*, pages 222–229.

Öztaner, Serdar Murat. 1996. A word grammar of Turkish with morphophonemic rules. Master's thesis, Computer Engineering Department, Middle East Technical University.

Pareschi, Remo and Mark Steedman. 1987. A lazy way to chart-parse with Categorial Grammars. In *Proceedings of the 25th Annual Meeting of the ACL*, pages 289–296.

Pembeci, Izzet. 1998. A unification-based tool for learning of Turkish morphology. Master's thesis, Computer Engineering Department, Middle East Technical University.

Pesetsky, D. 1979. Russian morphology and lexical theory. Master's thesis, MIT.

Pesetsky, David. 1985. Morphology and logical form. *Linguistic Inquiry*, 16:193–246.

Petkevič, Vladimír. 1987. A new dependency based specication of underlying representations of sentences. *Theoretical Linguistics*, 14:143–172.

Petkevič, Vladimír. 1995. A new formal specication of underlying structures. *Theoretical Linguistics*, 21:7–61.

Pollard, Carl and Ivan Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA.

Rambow, Owen and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory, Volume 39 of Studies in Language, Companion Series*. John Bejamins, Amsterdam, pages 167–190.

Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *In Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

Riedel, Sebastian, Ruket Çakıcı, and Ivan Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming. In *Proceedings of the Tenth Conference on Computational Natural Language Learning CoNLL-X*, pages 226–230, New York City.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-

Functional Grammar and discriminative estimation techniques. In *Proc. of the 40th Annual Meeting of the ACL*, pages 271–278, Philadelphia, PA.

Robinson, Jane J. 1970. Dependency structures and transformational rules. *Language*, 46(2):259–285.

Russell, Kevin. 1993. *A Constraint-based approach to Phonology and Morphology*. Ph.D. thesis, University of Southern California, Los Angeles.

Sagan, Carl. 1985. *Cosmos*. Ballentine, New York.

Samuelsson, Christer. 2000. A statistical theory of dependency syntax. In *Proceedings of the 18th conference on Computational linguistics*, pages 684–690, Saarbrücken, Germany.

Schmerling, S. F. 1983. Montague morphophonemics. In J. F. Richardson, M. Marks, and A. Chukerman, editors, *Papers from the Parasession on the Interplay of Phonology, Morphology and Syntax*. Chicago Linguistic Society, Chicago, pages 222–237.

Sekine, Satoshi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2000. Backward beam search algorithm for dependency analysis of Japanese. In *17th International Conference on Computational Linguistics*, pages 754–760.

Selkirk, E. O. 1982. *The Syntax of Words*. MIT Press, Cambridge, Mass.

Sgall, Petr, Eva Hajičová, and Jarmila Panenová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel and Academia, Dordrecht and Prague.

Sgall, Petr, Ladislav Nebeský, Alla Goralčíkova, and Eva Hajičová. 1969. *A Functional Approach to Syntax in Generative Description of Language*. Elsevier, New York.

Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the fifth conference on Applied natural language processing*, pages 88–95, Washington, DC.

Sleator, D. and D. Temperley. 1993. Parsing English with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technologies (IWPT)*, pages 277–292.

Spencer, Andrew. 1988. Bracketing paradoxes and the English lexicon. *Language*, 64(4):663–682.

Spencer, Andrew. 1991. *Morphological Theory*. Blackwell.

Sproat, Richard. 1985. *On Deriving the Lexicon*. Ph.D. thesis, MIT.

Sproat, Richard. 1992. *Morphology and Computation*. The MIT Press.

Sproat, Richard. 1998. Morphology as component or module. In Andrew Spencer and Arnold M.Zwicky, editors, *The Handbook of Morphology*. Blackwell, pages 335–348.

Starosta, S. 1988. *The Case for Lexicase: An Outline of Lexicase Grammatical Theory*. Pinter.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Steedman, Mark and Jason Baldridge. to appear. Combinatory Categorial Grammar. In Robert D. Borsley and Kersti Borjars, editors, *Non-transformational Syntax: A Guide to Current Debate*. Blackwell, Oxford, UK.

Stump, Gregory T. 1991. A paradigm-based theory of morphosemantic mismatches. *Language*, 67:675–725.

Szabolcsi, Anna. 1983. The possessor that ran away from home. *The Linguistics Review*, 3:89–102.

Tesniére, Lucien. 1959. *Elements de Syntaxe Structurale*. Klinsieck, Paris.

Titov, I. and J. Henderson. 2007. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proc. of the 10th Conf. on Computational Natural Language Learning (CoNLL-X) Shared Task*, Prague.

Trask, Robert Lawrence. 1993. *A Dictionary of Grammatical Terms in Linguistics*. Routledge, London.

Underhill, Robert. 1972. Turkish participles. *Linguistic Inquiry*, 3(1):87–99.

Underhill, Robert. 1986. Turkish. In Slobin and Zimmer, editors, *Studies in Turkish Linguistics*. John Benjamins Pubishing Company, pages 2–23.

Vijay-Shanker, K. and David Weir. 1990. Polynomial parsing of Combinatory Categorial Grammars. In *Proceedings of the 28th Annual Meeting of the ACL*, pages 1–8, Pittsburgh, PA.

Vijay-Shanker, K. and David Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Vijay-Shanker, K. and David Weir. 1994. The equivalence of four extensions of Context-Free Grammar. *Mathematical Systems Theory*, 27:511–546.

Wall, Robert E. 1972. *Introduction to Mathematical Linguistics*. Prentice Hall, Englewood Cliffs, NJ.

White, Michael and Jason Baldridge. 2003. Adapting chart realization to CCG. In *Proceedings of European Workshop on Natural Language Generation*, pages 119–126, Budapest, Hungary.

Williams, E. 1981. On the notions 'lexically related' and 'head of a word'. *Linguistic Inquiry*, 12:245–274.

Wittenburg, Kent. 1986. *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification-Based formalism*. Ph.D. thesis, University of Texas.

Wittenburg, Kent. 1987. Predictive combinators: A method for efficient parsing of Combinatory Categorial Grammars. In *Proceedings of the 25th Annual Meeting of the ACL*, Stanford, CA.

Xia, Fei. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium*.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies*, pages 195–206.

Yüret, Deniz and Ferhan Türe. 2006. Learning morphological disambiguation rules for Turkish. In *Proceedings of the Human Language Technology Conference of the*

NAACL, Main Conference, pages 328–334, New York City, USA, June. Association for Computational Linguistics.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666.

Zettlemoyer, Luke S. and Michael Collins. 2007. Online learning of CCG grammars for parsing to logical form. In *Proceedings of the 27th Joint Conference on Empirical Methods in Natural Language Processing*, pages 678–687, Prague.