NOUN PHRASE CHUNKER FOR TURKISH USING DEPENDENCY PARSER

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULLFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Mücahid Kutlu
July, 2010

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Özgür Ulusoy (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

_____

Dr. İlyas Çiçekli(Co-Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Ferda Nur Alpaslan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Fazlı Can

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Halil Altay Güvenir

Approved for the Institute of Engineering and Sciences:

_____

Prof. Dr. Levent Onural
Director of Institute of Engineering and Sciences

ABSTRACT

NOUN PHRASE CHUNKER FOR TURKISH USING DEPENDENCY PARSER

Mücahid Kutlu
M.S. in Computer Engineering
Supervisors
Prof. Dr. Özgür Ulusoy
Dr. İlyas Çiçekli
July, 2010

Noun phrase chunking is a sub-category of shallow parsing that can be used for many natural language processing tasks. In this thesis, we propose a noun phrase chunker system for Turkish texts. We use a weighted constraint dependency parser to represent the relationship between sentence components and to determine noun phrases.

The dependency parser uses a set of hand-crafted rules which can combine morphological and semantic information for constraints. The rules are suitable for handling complex noun phrase structures because of their flexibility. The developed dependency parser can be easily used for shallow parsing of all phrase types by changing the employed rule set.

The lack of reliable human tagged datasets is a significant problem for natural language studies about Turkish. Therefore, we constructed the first noun phrase dataset for Turkish. According to our evaluation results, our noun phrase chunker gives promising results on this dataset.

The correct morphological disambiguation of words is required for the correctness of the dependency parser. Therefore, in this thesis, we propose a hybrid morphological disambiguation technique which combines statistical information, hand-crafted grammar rules, and transformation based learning

rules. We have also constructed a dataset for testing the performance of our disambiguation system. According to tests, the disambiguation system is highly effective.

Keywords: Natural Language Processing, Noun Phrase Chunker, Turkish, Shallow Parsing, Morphological Disambiguation.

ÖZET

TÜRKÇE İÇİN BAĞIMLI ÇÖZÜMLEYİCİ KULLANARAK İSİM TAMLAMASI ÇIKARIMI

Mucahid Kutlu
Bilgisayar Mühendisliği Bölümü, Yüksek Lisans
Tez Yöneticileri
Prof. Dr. Özgür Ulusoy
Dr. İlyas Çiçekli
Temmuz, 2010

İsim tamlaması çıkarımı bir çok doğal dil işleme konusunda kullanılabilen yüzeysel çözümlemenin alt kategorisidir. Bu tezde, biz Türkçe metinler için bir isim tamlaması çıkarımı sistemi öneriyoruz. Cümle bileşkenleri arasındaki ilişkiyi göstermek ve isim tamlamalarını bulmak için ağırlıklı bir kısıtlayıcı bağımlı çözümleyici kullanıyoruz.

Bağımlı çözümleyici kısıtlamaları belirlemek için, manual olarak oluşturulan ve biçimbirimsel ve anlamsal bilgileri birleştirebilen etkili kurallar kullanır. Kurallar esnek yapıları gereği karmaşık isim tamlamaları yapılarını çözmek için uygundur. Kural dizisi değiştirilerek bağımlı çözümleyici diğer tüm cümle parçacığı çeşitlerini içeren bir yüzeysel çözümleyici olarak da kolaylıkla kullanılabilir.

Türkçe için insanlar tarafından oluşturulmuş güvenilir bir veri grubunun olmaması Türkçe ile ilgili olan doğal dil işleme çalışmalarını için önemli bir problemdir. Bu yüzden, Türkçe için ilk isim tamlaması veri gruplarını oluşturduk. Bu veri grupları üzerinde yaptığımız testlere göre, bizim önerdiğimiz isim tamlaması çıkarımı sistemimiz, umut verici sonuçlar vermektedir.

Kelimelerin biçimbirimsel bilgisini bilmek bağımlı çözümleyicinin doğru çalışması için önemlidir. Bu yüzden, bu tezde, istatistiksel bilgi ile, elle

oluşturulmuş gramer kuralları ile dönüşüm bazlı öğrenilmiş kuralları bir arada kullanan hibrit bir biçimbirimsel belirsizliği giderme tekniği önerdik. Ayrıca bizim belirsizlik giderici sistemimizin performansını ölçmek için bir veri grubu oluşturduk. Yaptığımız testlere göre, bizim önerdiğimiz sistem umut verici sonuçlar vermektedir.

Anahtar Kelimeler: Doğal Dil İşleme, İsim Tamlaması Çıkarımı, Türkçe, Yüzeysel Çözümleme, Dilbilimsel Belirsizliği Giderme.

Acknowledgement

I would like to express my deep gratitude to Dr. İlyas Çiçekli for his guidance, encouragement, and suggestions throughout the development of this thesis. I would like to thank Prof. Dr. Özgür Ulusoy for his support about the thesis.

I would like to thank Prof. Dr. Fazlı Can, Prof. Dr. Halil Altay Güvenir and Assoc. Prof. Dr. Ferda Nur Alpaslan for reading and commenting on the thesis.

I would like to thank all members of my family for their great moral support and patience especially during the development of my thesis. I also have to mention the help of my little nephew, Mehmet Emin Kara, because of his games that make me to have breaks during my studies and not allowing me to work too hard.

I would like to thank all of my friends who have helped during my master study. Especially, I would like to thank my friend Ibrahim Aydin for his help in writing the thesis and sharing his knowledge about the linguistics and Turkish grammar which was crucial for the development of the study. I also would like to thank my friend and neighbor living in "next", Abdullah Bulbul, for his support and patience to my linguistic jokes. In addition, I would like to thank my friend Cem Aksoy for his help in writing the thesis and for his hospitality. I also would like to thank my officemates, especially Enver Kayaaslan and Fahreddin Şükrü Torun.

I also would like to thank TUBITAK-BIDEB because of their financial support during my MS study.

Contents

List of Figures

List of Tables

List of Abbreviations

SOV	Subject object verb
POS	Part of speech tag
NP	Noun Phrase
VP	Verb Phrase
MP	Morphological Parse
SVM	Support Vector Machine
FSM	Finite State Machine
IR	Information Retrieval
NLP	Natural Language Processing
TBL	Transformation Based Learning
PCFG	Probabilistic Context Free Grammar
CDG	Constraint Dependency Grammar

Chapter 1

Introduction

The amount of digital resources increases every day and people can reach them easily via internet. Today, many people use news portals and read articles via Internet, instead of buying news papers. In addition, by the development of information retrieval (IR) technologies, people obtain information about almost anything via web sites like Wikipedia. By the developments in mobile technologies, people are also able to read books with their mobile devices. Furthermore, e-mails, chat programs play an important role for the communication among people.

Dealing with so many digital resources brings new problems, too. New solutions are required to handle these problems. For example, when people use a news portal, they want to read the news that they are interested, not the irrelevant ones. However, In IR technologies, finding related information becomes harder because of huge amount of resources. Detection of the information that the user really is looking for requires more complex operations.

Many researchers have been working on different natural language processing (NLP) tasks in order to efficiently process the digital resources easily for decades. Categorization, summarization, machine translation, information extraction, etc. are only a few of the topics that researchers work on. Since most of the resources are texts which are written in many different natural languages, understanding the meaning and having morphological analysis of texts are crucial for deeper analysis and performing further NLP tasks on them.

Most of the researchers made their studies based on a specific natural language since every natural language has many distinctive features. Most of the studies in the literature are based on commonly spoken languages such as English. On the other hand, a lot of studies have to be done for the languages which have relatively small amount of speakers. Turkish is one of these languages that require more research. In the literature, there are studies about morphological disambiguation (Daybelge and Cicekli, 2007; Oflazer and Tur, 1997), keyword extraction (Ozdemir and Cicekli, 2009), automatic text summarization (Kutlu et al. 2010; Altan, 2004) and sentence parsing (Istek, 2006; Oflazer, 2003) for Turkish. However, to the best of our knowledge, there is no previous study for Noun Phrase Chunking for Turkish. Therefore, in this thesis, we propose a Noun Phrase Chunker system which uses a dependency parser for Turkish.

Noun phrase chunking is a subset of shallow parsing (or text chunking). Shallow parsing consists of dividing sentences into non-overlapping phrases in such a way that syntactically related words are grouped in the same phrase. It can be considered as an intermediate step for full parsing. Each phrase has a name such as noun phrase (NP), verb phrase (VP), etc.

There are many motivations for shallow parsing. By an intuition, when we read a sentence, we read it chunk by chunk (Abney, 1991). In addition, as it is discussed in (Hammerton et al. 2002), some natural language tasks do not need full parsing. Furthermore, full parsing often provides too much information and sometimes does not provide enough information. For example, finding noun phrases and verb phrases may be enough for IR technologies. For question-answering, information extraction, text mining and automatic summarization, phrases that give us information about time, places, objects, etc. are more significant than the complete analysis of a sentence.

In this thesis, we concentrate on noun phrase chunking rather than handling all chunk types. As discussed in (Kuang-hua Chen et al., 1994), knowing noun phrases of a text makes us to understand the text to some extent. Therefore,

finding noun phrases is a significant step for further operations. In addition, being lack of reliable human tagged datasets for Turkish leaded us to concentrate only on noun phrases.

In this thesis, we propose a noun phrase chunker system for Turkish that consists of a morphological disambiguator and a dependency parser that uses hand-crafted constraining. The system takes a Turkish text as an input. The morphological analyses of words in the text are obtained by using SupervisedTagger tool (Daybelge and Cicekli, 2007). In order to find the correct morphological parse (MP) of the word, we propose a novel approach for morphological disambiguation for Turkish which is a hybrid system that combines statistical information and hand-crafted grammatical rules and transformation based learning rules. After disambiguation of the words, we use a dependency parser which uses hand-crafted rules in order to determine noun phrases. The dependency parser creates links between sentence components to represent the relationships between them. After creating links, we obtain noun phrases by processing them. The dependency parser we propose can be easily converted to a shallow parser that includes all types of phrases because of its generic structure.

1.1 Contribution

The contributions of this thesis are listed below:

- We propose a novel approach for morphological disambiguation for Turkish. According to our evaluation results, it gives promising results.
- The lack of reliable human tagged datasets is a significant problem for NLP studies about Turkish. We constructed a reliable manually tagged dataset which contains correct morphological analysis of 25098 words. The dataset can be used for researchers who want to work on Turkish in future.

- We used a dependency parser for noun phrase chunking of Turkish texts. This is the first noun phrase chunker system for Turkish which can be used in various NLP tasks such as summarization systems, machine translation, etc. The system we propose gives promising results according to our tests.

- In dependency parser, we use handcrafted rules which use constraints that allow rule designers to define specific situations. We defined many generic functions to be used in the creation of constraints. The system we propose can be easily converted into another shallow parsing system containing all phrase types by changing the rule set. The rule structure allows us to use semantic information and morphological information together.

- We have implemented a noun phrase tagger tool for constructing ground truth datasets. The tool eases the tagging process and will be helpful for overcoming being lack of dataset problem of Turkish in future. In addition, it can be used for other phrases.

- By using our noun phrase tagger tool, we have manually tagged noun phrases in three datasets which have different properties. The total number of main noun phrases in these datasets is 3941.

1.2 Linguistic background

In this section, some linguistics terminologies which are used in the rest of the thesis are given. Morphology is the study of the way that words are built up from morphemes. A word is the minimal free form in a language and a morpheme is often defined as minimal meaning-bearing unit in a language. For example, the word *school* consists of a single morpheme and the word *schools* consists of two morphemes: “school” and “–s”.

The morphemes can be divided into two categories: stems and affixes. The stem is the main morpheme of the word and contains the main meaning of the word. Affixes are attached to the stem in order to add additional meaning. The new meaning after attaching an affix is still related with the stem. In the example given above, school is the stem and –s is the affix which gives plural meaning to the word.

Attaching affixes to the words cause inflection or derivation. The difference between inflection and derivation is that the Part-Of-Speech (POS) tag of the new word remains the same with old one after inflection whereas POS tag can change after derivation. POS tag shows the main class of the word such as noun, verb, etc.

Words can have more than one affix. The number of affixes to be attached does not exceed 4 or 5 in English. However, in agglutinative languages, such as Turkish, 9 or more affixes can be attached to a word. In agglutinative languages, each affix represents a morpheme and affixes are stringed together. The restrictions for orderings of the morphemes are called *Morphotactics*.

1.3 Overview of the Thesis

The organization of the thesis is as follows:

- Chapter 2 summarizes previous studies on noun phrase chunking, shallow parsing and dependency parsers.
- Chapter 3 gives background information about Turkish and the scope of the study.
- Chapter 4 gives an overview of the system and explains the relationship between its components.

- Chapter 5 explains the morphological disambiguation system that we propose. The related work and test results for morphological disambiguation are also discussed in this chapter.
- Chapter 6 explains the dependency parser in detail. The link and rule structures and algorithms for constructing links and extracting noun phrases by using links are given in this chapter.
- Chapter 7 explains the experimental environment and discusses the evaluation results of the system.
- Chapter 8 gives the conclusion together with the future work for development of the noun phrase chunker system.

Chapter 2

Related Work

In this chapter, we explain previous studies on noun phrase chunkers, shallow parsers and dependency parsers.

2.1 Shallow Parsers

Shallow parsing has been attracting the researchers for decades. Church (1988) proposes a POS tagger and NP extractor using a stochastic model. However, noun phrases which are connected with conjunctions (i.e. and, or) are not within the scope of that study. That is to say, the scope of the study consisted of simple NPs.

Bourigault (1992) presents LEXTER which is a tool for extracting terminologies in French texts. LEXTER uses a cascaded approach. First, maximal length NP is extracted by using some heuristics. Then terminologies which are embedded to NPs are parsed by using grammar rules. An expert evaluates the results of LEXTER and reports that the recall is 95%. But the precision is not given. It is to be mentioned that Bourigault's goal is extracting terminological parses which is simpler than NP chunking.

Voutilainen (1993) presents NPTool for finding maximal length NP. The tool uses a lexicon combined with a constraint grammar. Although the reported recall is 98.5-100% and the precision is 95-98%, Chen et al. (1994) mention the existence of some inconsistencies in the sample output given in his Appendix

and claim that the recall is about 85%. Ramshaw and Marcus (1995) also point out the unreliable results and give some wrong outputs of NPTool.

Ramshaw and Marcus (1995) introduce NP chunking as a machine learning problem. They apply transformation based learning (TBL) by using lexical information. They use Wall Street Journal texts in Penn Treebank, F-Measure that they obtained in their evaluation is 92.0. However, the target NPs to be found are not recursively embedded (nested) NPs. That is to say, NPs in dataset are only base NPs.

Several groups worked with the same dataset and the same NP definition of Ramshaw and Marcus's pioneering study (1995). Argamon et al. (1998) use memory based sequence learning in order to determine NPs and VPs without using any lexical information (F-Measure = 91.6). Cardie and Pierce (1998) learn POS tag sequences that form a complete NP to find NPs that are not found in training set (F-Measure = 90.9). Veenstra (1998) uses a cascaded chunking that uses lexical information (F-Measure = 91.6). Daelemans et al. (1999) use memory based system and evaluates the system with a different dataset and reports a good performance. Tjong Kim Sang and Veenstra (1999) use a memory based system (F-Measure = 92.37), XTAG Research Group (1998) applies tree-adjoining grammar (F-Measure = 92.4) and Munoz et al. (1999) use a network of linear units for recognizing NP and SV phrases (F-Measure = 92.8). Although these three systems have better performance than Ramshaw and Marcus's study (1995), they are not feasible for implementing in an active learning framework, or are significantly most costly (Ngai and Yarowsky,2000).

Wojciech Skut and Brants (1998) propose a stochastic model for finding more complex phrases. NP, PP, AP and adverbials are in the scope and they try to recognize internal structure of them.

Most of the studies are performed on English texts. However, there are also some studies for other languages, too. Sobha et al. (2006) uses TBL for Tamil texts and obtain precision value of 97.4. Pattabhi et al. (2007) applies TBL for

three Indian languages: Hindi (73.80), Bengali (65.28), and Telugu (50.38) where accuracies are given in parenthesizes. Ravi Sastry et al. (2007) uses dynamic programming algorithm for finding best possible chunk sequences for the same three Indian languages: Hindi (F-Measure = 78.35), Bengali (F-Measure = 67.52), Telugu (F-Measure = 68.32). It is to be noted that, in their paper, there is an inconsistency with the result of Hindi language. In the abstract part of the paper, the F-Measure value is given as 69.98.

To the best of our knowledge, there is no previous study about shallow parsing or noun phrase chunking of Turkish texts.

2.2 Dependency Parsers

We use a dependency parser for noun phrase chunking. Therefore, looking at previous studies about dependency parsers is necessary. Dependency parsing is a technique that researchers are working on since 1960s for syntactic parsing. A theoretical discussion about dependency grammars can be found in (Nivre, 2005). The previous studies about dependency parsing can be categorized into two groups: Grammar-Driven and Data-Driven.

In grammar-driven approaches, very early studies use formalization technique which is very similar to context free grammars (Hays (1964); Gaifman (1965)). Another common technique is based on elimination of representations which are invalid according to constraints. That is to say, a dependency representation must satisfy all constraints in order to be accepted. Karlson (1990), Maruyama (1990), Harper and Helzerman (1995) Jarvinen and Tapanainen (1998) are some studies that use constraint grammars. Menzel and Schroder (1998) extend the framework of Maruyama (1990) by assigning a grade (between 0.0 and 1.0) to each constraint for representing the power of the constraint where 0.0 is the most powerful. Schroder (2002) also uses weighted constraint dependency grammar (WCDG) and applies a grading technique for

deciding the best analysis. The analysis having minimum total grade is considered as the best analysis.

Although link grammars are not considered as a dependency parser by its first developers (Sleator and Temperley, 1991), they can be classified under the dependency parsers because of having similar representations. Sleator and Temperley (1991, 1993) used dynamic programming algorithm with memorization in their link grammar.

The second approach for dependency parsers is data-driven approach. Carroll and Charniak (1992) use a probabilistic context free grammar (PCFG) model and test their system with an artificially created corpus. Eisner (1999a, 1999b) defines many several probabilistic approaches and tests them using supervised learning with Wall Street Journal section of Penn TreeBank. Collins et al. (1999) apply generative probabilistic models using Prague Dependency Tree as training data. Wang and Harper (2004) apply stochastic constraint dependency grammar (CDG) parser which is an extension of the CDG model with a generative probabilistic approach. Kudo and Matsumoto (2000, 2002) propose the deterministic discriminative approach which uses support vector machines (SVM) for Japanese dependency analysis.

A more detailed discussion about dependency parsers can be found in (Nivre, 2005) There are also studies for dependency parsing of Turkish sentences. Istek (2006) applies link grammar for parsing Turkish sentences. Oflazer (2003) uses extended finite-state approach. Oflazer uses violable constraints in order to prevent robustness problem and uses total link length for ranking the alternative analyses in case of ambiguity.

Our study is in the category of constraint dependency grammars. We manually define rules that are used by the dependency parser. The rules act as finite state machine and when all constraints of a rule are satisfied, we construct a link for representing relationship between corresponding sentence component. Therefore, our study can be considered as an FSM approach. We connect links

between sentence components called *Token* which are same with inflectional groups in (Oflazer, 2003). We also use a grading mechanism for eliminating alternative analyses. Therefore, our study can be considered as a weighted constraint dependency grammar.

Chapter 3

Turkish Morphology and Noun Phrase Structures

In this chapter, some important features of Turkish syntax and morphology are explained in order to get familiar with the natural language that we work on. Later on, noun phrase structures in Turkish are discussed in detail and the boundaries of our study are given.

3.1 Distinctive Features of Turkish

Turkish is a member of the Altaic branch of the Ural-Altaic language family. It has many distinctive features than generally known languages (i.e. English). The features that are related to our study are listed as follows:

- Turkish is an agglutinative language which generates words by joining affixes together and each affix represents one unit of meaning, such as "single", "future tense", etc. This agglutinative property of Turkish loads many meanings to a single Turkish word. For example, the word: "uygarlaştıramadıklarımızdanmışsınızcasına" means "as if you are among those whom we could not civilize" and this single Turkish word has the semantic function of 11 words in English. In addition, the number of word forms that can be generated from a nominal or verbal root is theoretically infinite (Eryiğit and Oflazer, 2006). This feature makes morphological analysis of words harder in Turkish. Furthermore, suffixes can increase ambiguity by generating totally different words.

For example, when we add “m” suffix to word “ada” (island) we obtain word *adam* which means “my island” and also “man”.

- Basic word order of Turkish sentences is Subject-Object-Verb (SOV). However, Turkish grammar allows all constituent orders. Therefore, all 6 combinations of the order can be used. Changing order only changes the stress, not the meaning. For example, the sentence `I went to school` can be written as follows:

1.	Ben	okula	gittim.
	(I)	(to school)	(went)
2.	Okula	ben	gittim.
	(to school)	(I)	(went)
3.	Gittim	ben	okula.
	(went)	(I)	(to school)
4.	Ben	gittim	okula.
	(I)	(went)	(to school)
5.	Okula	gittim	ben.
	(to school)	(went)	(I)
6.	Gittim	okula	ben.
	(went)	(to school)	(I)

Although constituents at the sentence level can freely change, the parts of the constituents (such as noun phrases) do not change freely. For example, while *Kırmızı elma* (red apple) is legal, *elma kırmızı* (apple red) is not.

- Turkish has no grammatical gender. Nominal nouns do not have a gender as in German or Arabic, etc. For example, “die blume” (the flowers) and

"der tabelle" (the table) in German. "Die" is a determiner used for single female nouns and "der" is used for male nouns in German.

- In Turkish, the vowels of suffixes should have an agreement with the last vowel of that word. In the following example, suffix that gives the meaning of "my" is attached to different words and at each of them the vowel of the suffix is changed according to the last vowel of the word. The suffix is separated with '+' character from the word.

    - Kal**e**m+**i**m (My pencil)
    - Ok**u**l+**u**m (My school)
    - Müd**ü**r+**ü**m (My director)

  There are some suffixes that does not obey this rule such as "-yor".

3.2 Turkish Morphology and Morphotactics

Morphemes can be categorized as derivational morphemes and inflectional morphemes. Derivational morphemes are used to generate new words from another word with new meanings while inflectional morphemes are used to specify grammatical information (e.g. number, case, etc.). Derivational morphemes can change POS tag of the word while inflectional morphemes cannot. In the following subsections, we will explain the morphotactics for both morpheme groups in detail.

3.2.1 Inflectional Morphotactics

In this section, we give details inflectional morphotactics about nouns, and pronouns. Since the scope of our study is only noun phrases, morphotactics for verbal inflectional are not discussed.

Inflectional Morphotactics of Nouns

Nouns can take singular-plural suffixes, possessive markers and case markers in the order. Now let's see each suffix with an example. The sample root is *okul* (school) in the following examples and we add suffixes to this root word. The corresponding morpheme for that suffix is written in bold. The suffixes are also shown by separating them with '+' character, if needed.

1. Plural Suffixes: A noun can take a plural suffix as a first inflection suffix. If it does not take a plural suffix, it means that the word is singular.

    a. Singular: okul (school)

        Okul+Noun+**A3sg**+Pnon+Nom

    b. Plural: okul+lar (schools)

        Okul+Noun+**A3pl**+Pnon+Nom

2. Possessive Marker: The second inflectional morpheme is a possessive marker. A noun may not take a possessive marker or its possessive marker can be one of the six possessive markers.

    a. No Possessive marker: okul (school)

        Okul+Noun+A3sg+**Pnon**+Nom

    b. First Person-Singular: okul+um (my school)

        Okul+Noun+A3sg+**P1sg**+Nom

    c. Second Person-Singular: okul+un (your school)

        Okul+Noun+A3sg+**P2sg**+Nom

    d. Third Person-Singular: okul+u (his/her school)

        Okul+Noun+A3sg+**P3sg**+Nom

e. First Person-Plural: okul+umuz (our school)

   Okul+Noun+A3sg+**P1pl**+Nom

f. Second Person-Plural: okul+unuz (your school)

   Okul+Noun+A3sg+**P2pl**+Nom

g. Third Person-Plural: okul+ları (their school)

   Okul+Noun+A3sg+**P3pl**+Nom

3. Case Marker: Last inflectional morpheme is one of case markers which have functions of prepositions in English. There are seven case markers.

   a. Nominative: okul (school)

      Okul+Noun+A3sg+Pnon+**Nom**

   b. Locative: okul+da (at the school)

      Okul+Noun+A3sg+Pnon+**Loc**

   c. Dative: okul+a (to the school)

      Okul+Noun+A3sg+Pnon+**Dat**

   d. Ablative: okul+dan (from the school)

      Okul+Noun+A3sg+Pnon+**Abl**

   e. Accusative: okul+u (the school)

      Okul+Noun+A3sg+Pnon+**Acc**

   f. Instrumental: okul+la (with the school)

      Okul+Noun+A3sg+Pnon+**Ins**

g. Genitive: okul+un (the school's)

Okul+Noun+A3sg+Pnon+**Gen**

Inflectional Morphotactics of Pronouns

The morphemes for pronouns are more complicated than nouns. First and second person pronouns do not take singular-plural suffixes since the stem gives us that information, while third-person pronouns can take singular-plural suffixes. In addition, pronouns can take possessive markers and case markers in the order. However, a pronoun cannot have both of them together. There are two types of pronouns: Personal and demonstrative pronouns.

Personal Pronouns: A personal pronoun refers to a specific person or a thing. A list of personal pronouns is given below. Since the order of morphemes and suffixes are same as in nouns, we will give only singular-plural pronouns.

- Ben (I) : Ben+Pron+A1sg+Pnon+Nom
- Sen (You) : Sen+Pron+A2sg+Pnon+Nom
- O (He/She) : O+Pron+A3sg+Pnon+Nom
- Biz (We) : Biz+Pron+A1pl+Pnon+Nom
- Siz (You) : Ben+Pron+A2pl+Pnon+Nom
- O+nlar (They) : Ben+Pron+A3pl+Pnon+Nom

Demonstrative Pronouns: A demonstrative pronoun refers to a person or a thing by considering its distance in terms of time and space. A list of nominative demonstrative pronouns is given below.

- Bu (This) : Bu+Pron+A3sg+Pnon+Nom
- Şu (That) : Şu+Pron+A3sg+Pnon+Nom
- O(That): O+Pron+A3sg+Pnon+Nom

- Bu+nlar (These) : Bu+Pron+A3pl+Pnon+Nom

- Şu+nlar (Those) : Şu+Pron+A3pl+Pnon+Nom

We use *bu* (this) and *bunlar* (these) for the things that are near in terms of time or space and we use *şu* (that) and *şunlar* (those) for the further ones. We use *O* pronoun for the furthest objects. Single demonstrative pronouns can also be used as an adjective. Some examples are as follows: *Bu adam* (This man), o *gemi* (that ship).

3.2.2 Derivational Morphotactics

In Turkish, a word can take derivational suffixes that can change the POS tag of the word. Since there are many types of derivational suffixes we give only examples for some of them. A derivation morpheme in a morphological parse is represented as "^DB+POS+MorphName" where *POS* is the part of speech tag of the word after derivation and *MorphName* is the name of the derivational morpheme. The derivational suffixes are shown by separating them with '+' character when needed.

- A noun can be derived into a verb. For example, let's consider the following sentence: "Türkiye güzel bir ülkedir" (Turkey is a beautiful country). The word *ülkedir* (is a country) is a verb which is derived from the noun *ülke* (country) by taking *-dir* suffix. The morphemes and suffixes of the word *ülkedir* are represented as follows.

  Ülke+dir:
  Ülke+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+Cop+A3sg

- Every adjective in Turkish can be derived into a noun with a null morpheme. For example, let's consider the following sentence: "Galeridekilerin en ucuzunu aldı." (He bought the cheapest one in the gallery.) The word *ucuzunu* (the cheap one) is a noun having accusative

case marker which is derived from the adjective *ucuz* (cheap). The morphemes of the word *ucuzunu* are represented as follows:

Ucuz+u+nu: Ucuz+Adj^DB+Noun+Zero+A3sg+P3sg+Acc

- A number can also be derived into a noun with a null morpheme. For example, let's consider the following sentence: "30 Nisan 1986'da doğdu." (He was born in 30th April 1986.) The word *1986'da* (in 1986) is a noun having locative case marker is derived from the word 1986 which is a cardinal number. The morphemes of the word *1986'da* are represented as follows:

  1986+da: 1986+Num+Card^DB+Noun+Zero+A3sg+Pnon+Loc

- A verb can be derived into a noun. For example, let's consider the following sentence: "Okumayı seviyor" (He loves reading). The word *Okumayı* (reading) is a noun having accusative case marker which is derived from the verb *oku* (read) using *-ma* derivation suffix. The morphemes of the word *okumayı* are represented as follows:

  Oku+ma+yı: Oku+Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Acc

- Some derivational suffixes do not change the POS tag of the word. For example, the word *kitapçıdan* (from the book store) is a noun which is derived from the noun *kitap* (book) using *-çı* derivational morpheme. The morphemes of the word are represented as follows:

  Kitap+çı+dan:
  Kitap+Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Abl

- The number of derivational suffixes that a word can have can be more than one. For example, in the noun phrase *kitapçıdaki adam* (the man in the book store), the word *kitapçıdaki* (the one in the book store) is an adjective which is derived from the word *kitapçıda* (in the book store)

which is also derived from the noun *kitap* (book). The suffixes and morphemes of the word are represented as follows:

Kitap+çı+da+ki:
Kitap+Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Loc^DB
+Adj+Rel

3.3 Noun Phrase Structure in Turkish

Noun phrases are phrases that have pronoun or noun head word modified with a group of modifiers. Turkish is predominantly head-final, that is to say, modifiers precede the head. The phrases in which head precedes modifiers is out of the scope of this study. According to modifiers and number of head words, we can list the noun phrases as follows:

- A single noun, pronoun or proper noun can be a noun phrase without any modifier. For example, let's consider the following sentence: "Ben Ankara'ya otobüsle gittim" (I went to Ankara by bus.) The noun phrases are as follows: *Ben* (I) which is a pronoun, *Ankara'ya* (to Ankara) which is a proper noun and *otobüsle* (by bus) which is a noun.

- Modifiers can be adjectives when the head word is a noun.

red book:

Kırmızı	kitap
Kırmızı+Adj	kitap+Noun+A3sg+Pnon+Nom
(Red)	(book)

- Modifiers can be a number showing quantity of the head. In Turkish, the noun does not have to be plural when the number is bigger than one.

Three books:

Üç	kitap
Üç+Num+Card	kitap+Noun+A3sg+Pnon+Nom
(Three)	(book)

- The modifier can be also a pronoun with a genitive case marker. In this case, there should be an agreement between the possessive markers of the modifier and head word.

My book:

Benim	kitabım
(Ben+Pron+A3sg+**P1sg**+Gen)	(kitap+Noun+A3sg+**P1sg**+Nom)
(My)	(book+P1sg)

Your book:

Senin	kitabın
(Sen+Pron+A3sg+**P2sg**+Gen)	(kitap+Noun+A3sg+**P2sg**+Nom)
(Your)	(book+P2sg)

his book:

Onun	kitabı
(O+Pron+A3sg+**P3sg**+Gen)	(kitap+Noun+A3sg+**P3sg**+Nom)
His/her	(book+P3sg)

An incorrect example having no agreement on possessive markers is given below:

Benim	kitabın
(Ben+Pron+A3sg+**P1sg**+Gen)	(kitap+Noun+A3sg+**P2sg**+Nom)
(My)	(Book+P2sg)

- The modifier can be a noun with a genitive case marker. In this case, the head word should be a noun with third-person possessive marker.

friend's book:

Arkadaşın	kitabı
(Arkadaş+Noun+A3sg+Pnon+Gen)	(kitap+Noun+A3sg+**P3sg**+Nom)
(Friend's)	(book+P3sg)

- When the modifier is a noun, it does not have to be in genitive form. The modifier can also have nominative case marker and the head word can be a noun in any person possessive marker except non-person possessive marker (Pnon).

School book:

Okul	kitabı
(Okul+Noun+A3sg+Pnon+Nom)	(kitap+Noun+A3sg+P3sg+Nom)
(School)	(book+P3sg)

My school book:

Okul	kitabım
(Okul+Noun+A3sg+Pnon+Nom)	(kitap+Noun+A3sg+P1sg+Nom)
(School)	(my book)

Your school book:

Okul	kitabın
(Okul+Noun+A3sg+Pnon+Nom)	(kitab+Noun+A3sg+P2sg+Nom)
(School)	(your book)

- Modifiers can give information about what the head word is made of. In this type of NPs, modifier is a nominative noun with certain meanings and the head word is a noun with any possessive and case marker.

Gold watch:

Altın	saat
(altın+Noun+A3sg+Pnon+Nom)	(saat+Noun+A3sg+Pnon+Nom)
(gold)	(watch)

Wooden door:

Tahta	kapı
(tahta+Noun+A3sg+Pnon+Nom)	(kapı+Noun+A3sg+Pnon+Nom)
(Wooden)	(door)

- NPs can consist of more than one noun phrases. They are called as *nested NP*. A head can be modified with more than one modifiers or a modifier can also be modified by another word.

The friend of the man's son:

Adamın oğlunun arkadaşı

Morphological Parses:

- Adamın: Adam+Noun+A3sg+Pnon+Gen (man's)
- Oğlunun: Oğul+Noun+A3sg+P3sg+Gen (his son's)
- Arkadaşı: Arkadaş+Noun+A3sg+P3sg+Nom (his friend)

In this example, *Adamın* modifies *oğlunun* while *oğlunun* modifies *arkadaşı.* There is a sub-NP which is *Adamın oğlu (the man's son)* in main NP. Another sample nested NP is as follows:

Ten good students:

On	iyi	öğrenci
On+Num+Card	iyi+Adj	öğrenci+Noun+A3sg+Pnon+Nom
(Ten)	(good)	(student)

In the second example, the head word *öğrenci* has two modifiers which are *on* and *iyi*. *So, iyi öğrenci* (good student) is a sub-NP of the main NP. But we do not consider *on öğrenci* (ten student) as a sub-NP since it will destroy the linearity of the NP.

- NPs can contain more than one NP and each are connected with a conjunction words like and, or, etc. They are called as *conjunctive NP*. An example is as follows:

Ali and Veli went to school:

Ali	ve	Veli	okula	gitti.
(Ali)	(and)	(Veli)	(to school)	(went)

In this example, since both *Ali* and *Veli* are subjects of the sentence, *Ali ve Veli* can be considered as one whole NP. *Ali* and *Veli* are both sub-NPs of the main NP. Noun phrases can be connected with "le/la" suffix which is an instrumental case marker suffix. Same sentence can be written as follows: "Ali'yle Veli okula gitti." In this sentence the word *Ali'yle* is a proper noun having instrumental case marker and connected to the word *Veli*. Another example is as follows:

Red book or blue pencil:

Kırmızı	kitap	veya	mavi	kalem
(Red)	(book)	(or)	(blue)	(pencil)

In this example, there are two noun phrases which are *kırmızı kitap* (red book) and *mavi kalem* (blue pencil) and they are connected each other with a *veya* (or) conjuction.

Modifiers can also be connected with conjunctions and modify the same head. An example is as follows:

Red or blue pencil:

Kırmızı	veya	mavi	kalem
(Red)	(or)	(blue)	(pencil)

In this example, the words *Kırmızı* and *mavi* are connected with *veya* conjunction and modify the word *pencil* (kalem).

3.3 Scope of the Study

In this study, our aim is to find NPs which will be useful in information extraction studies and provide a better platform for fully parsing of sentences. All NP types mentioned above are in the scope of this study. However, in order to make fewer mistakes in this step, we do not want to work on sentence structure. Therefore, we ignored all noun phrases having a relative clause. In Turkish, there are no special words used for relative clause structures. But we can recognize them with words derived from verbs. Therefore, in our study, noun phrases containing a verb POS is ignored. For example, the following NP is out of the study:

Man who came from Ankara

Ankara'dan	gelen	adam
Ankara+Noun+Prop+A3sg+Pnon+Abl	gel+Verb^DB+Adj	adam+Noun
(From Ankara)	(who came)	(man)

The word *gelen* is an adjective which is derived from verb *gel* (come) and the verb is related to previous word *Ankara'dan*.

The words derived from a verb but are not related to sentence structure is exception of this concern. For example, in the following example, the word *yönetici* (manager) is a noun derived from the verb *yönet* (manage). Since it is

not directly related with the sentence structure, this NP is in the scope of the study.

Company's manager

Şirketin	yöneticisi
Şirket+Noun+A3sg+Pnon+Gen	yönet+Verb...^DB+Adj...^ DB+Noun+Zero+A3sg+P3sg+Nom
(Company's)	(manager)

In addition to noun phrases that are mentioned above, we also added the following phrases into the scope of the study.

- **Person Name:** A person can have more than one name or in a text, his/her name can be given with surname. Since all of those names belong to one person, finding person names is one of our aims. An example is given below. The noun phrases in English sentences are underlined.

  <u>Mucahid Kutlu</u> started to write his thesis:

  Mucahid Kutlu tezini yazmaya başladı.

  NP

  In addition, person names can contain the title or the word modifying the person. In the following sentences, two examples are given. In second example, *Teknik direktör* (Technique director) modifies the person name *Ertuğrul Sağlam*.

  <u>Technique director Ertuğrul Sağlam</u> went to Ankara

  Teknik direktör Ertuğrul Sağlam Ankara'ya gitti.

  NP NP

  NP

- **Institution Name:** Institutions can have long names and we consider all words of an institution name as an NP which can also have NPs in it. An example is as follows:

Where is <u>the Ankara City Health Center</u>?

Ankara İl Sağlık Merkezi nerede?

NP

- **Date:** Expressions of dates and times are also in the scope of this study although generally they are not acting as a noun phrase in a sentence. An example is given below.

<u>on Thursday 20 January 2008</u>

20 Ocak 2008 Perşembe günü

NP NP NP

- **Noun Phrases with Quotation Marks:** Quotation marks can be used for giving stress to some words. They can also be used in noun phrases, too. There are some NP samples below. In the first example, the NP is a single word, *Patron* (The boss) which is surrounded by quotation marks. In second example, the NP is *"Dur" ihtarını* (the "stop" warning) where the word *Dur* (stop) modifies the word *ihtarını*(warning)

<u>"The boss"</u> cannot manage the company well.

"Patron" şirketi iyi yönetemiyor.

NP

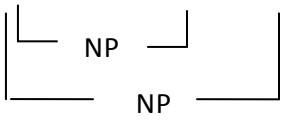
The driver didn't hear <u>the "Stop"warning</u>.

Sürücü "Dur" ihtarını duymadı.

NP

A noun phrase with more than one word can also be between two quotation marks. An example is given below. The main NP has a sub-NP which is “Robin Hood”.

The boy watched <u>the movie of “Robin Hood”</u>.
Çocuk “Robin Hood” filmini izledi.



However, it is to be mentioned that words between quotation marks and having no relationship does not form an NP. In the following sentence, *“Burada bekleme”* (“Don’t wait here”) is not an NP since *Burada bekleme* is not an NP.

The man said “Don’t wait here”.
Adam “Burada bekleme” dedi.

As we mentioned, Turkish is an agglutinative language and with the derivational suffixes, meaning and POS tag of the words can change with derivational suffixes. For this reason, we are going to construct token-based noun phrases. We split the word from the derivational boundaries and create a token for each derivation with its POS tag. A tokenization is given in the following example. The derivational suffixes are shown in bold.

Kitapçıdaki (the one in the book store)

Kitap+**çı**+da+**ki**

kitap+Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Loc^DB+Adj+Rel

Token$_1$ Token$_2$ Token$_3$

In this example, the word has three tokens. The last token is called head-token of the word. An example NP which does not end with a head-token is given in the following sentence.

Ceza mahkemelerince adam suçlu bulundu. (Man is found guilty by the Penalty Courts).

mahkemelerince (by the Courts) is an adverb which is derived from a noun *mahkemeleri* (courts) that is connected to the previous word *ceza* (Penalty). Therefore, we have to find the NP of *ceza mahkemeleri* (the Penalty Courts). In Turkish, modifiers should be a head-token. However, the modified token does not have to be, as seen in the example.

Chapter 4

System Architecture

In this chapter, we explain our system architecture. We propose a rule based dependency parser for extracting noun phrases in Turkish texts. General flow of the system can be seen in Figure 1.

Our system is composed of three main components which are Morphological Analyzer, Morphological Disambiguator and Dependency Parser which uses a set of hand-crafted constraining rules. As we get a Turkish text to extract its noun phrases, we first morphologically analyze the given Turkish text. In the morphological analysis, SupervisedTagger software (Daybelge and Cicekli, 2007) which uses a PC-Kimmo based morphological analyzer (Istek and Cicekli, 2007) is used. SupervisedTagger is consisted of a morphological analyzer, a collocation recognizer and a rule-based morphological disambiguation tool. We have updated some parts of SupervisedTagger. First, we corrected some wrong analysis of morphological analyzer that we observed. We also extended the morphological analyzer in SupervisedTagger using the official dictionary of the Turkish Language Council which contains nearly 33000 root words. Although this extension caused more ambiguity, it decreased number of unknown words that are morphologically analyzed with some heuristics. In addition, the morphological parsing capability of SupervisedTagger was improved by using an updated unknown word recognizer and new heuristics. Heuristics we added are as follows.

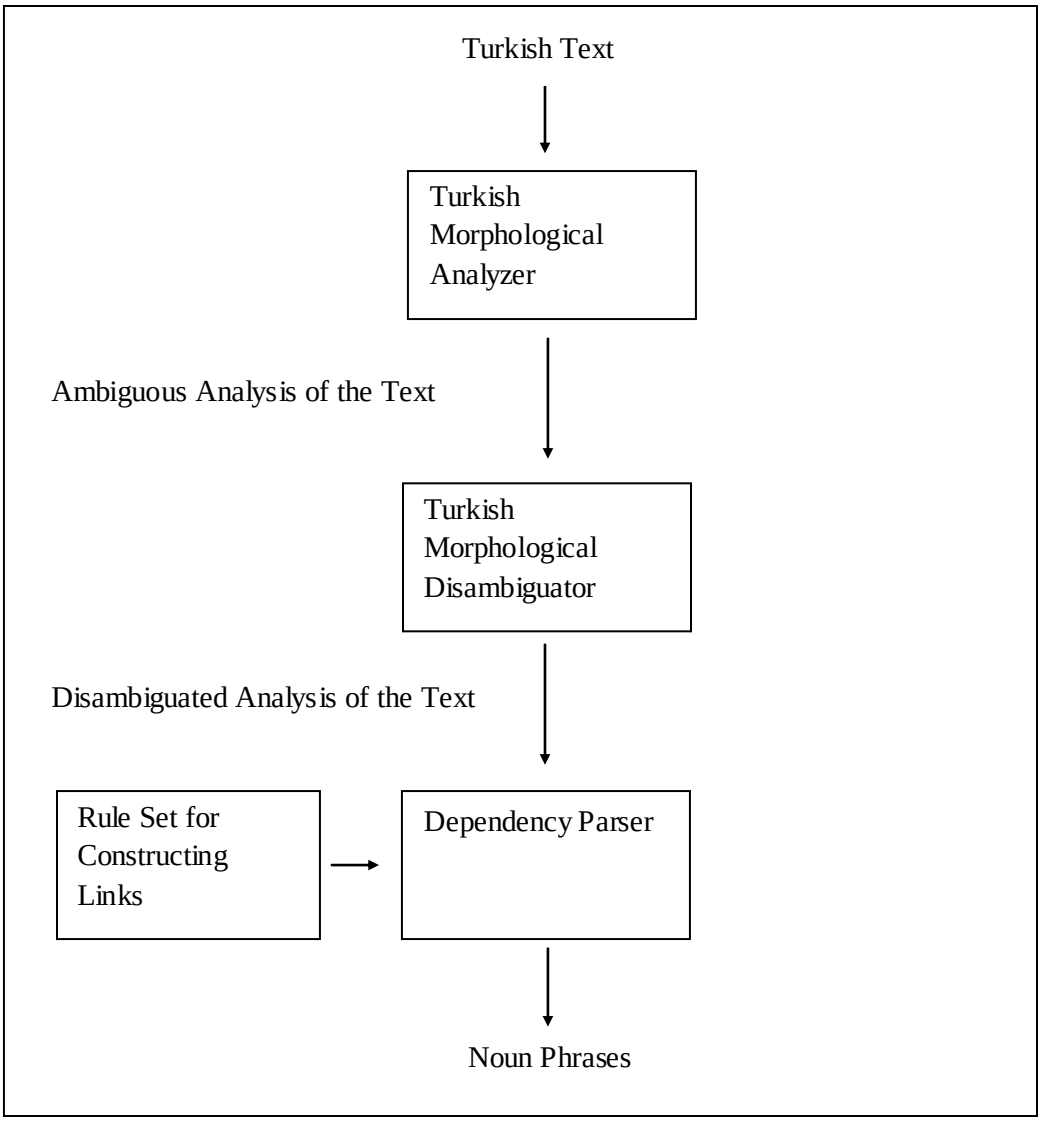


Figure 1. General Flow of the System.

- If a word begins with a capital letter and it is not the first word of the sentence, it is assumed that it has also a proper noun morphological parse even though it is not in the proper noun list.

- The words which are not in the dictionary and not correct according to the Turkish grammatical rules are assumed to be foreign words that have a proper noun morphological parse.

By enlarging dictionary and adding new heuristics, the average number of morphological parse per word increased from 1.8 to 2.0 which made morphological disambiguation task more difficult. After the morphological

analysis of words, the collocation recognizer of SupervisedTagger was also used to determine the collocations.

SupervisedTagger returns ambiguous results for most of the word. For example, the word *yakın* has 6 morphological parses which are:

- yakı+Noun+A3sg+P2sg+Nom (your plaster)
- yakın+Noun+A3sg+Pnon+Nom (near)
- yakın+Verb+Pos+Imp+A2sg (moan)
- yak+Verb+Pos+Imp+A2pl (burn)
- yakın+Adverb (near)
- yakın+Adj (near)

It is hard to perform correct calculations with ambiguous results since finding semantic meaning of the words and their grammatical functions in the sentence is crucial for our dependency parser. Therefore, we apply a morphological disambiguation technique that we propose in this study. We use a hybrid method which combines statistical information with hand-crafted grammar rules and transformation based learned rules. Five different steps are applied for disambiguation. In the first step, the most likely tags of words are selected. In the second step, we use hand crafted grammar rules to constrain possible parses or select the correct parse if we can. Next, the most likely morphological parses are selected according to the suffixes of the words that are unseen in the training corpus and still ambiguous. Then, we use transformation based rules that are learned by a variation of Brill tagger. If the word is still ambiguous, we use some heuristics for the disambiguation that strictly chooses a morphological parse. Detailed information and discussion of the morphological disambiguation is given in Chapter 5.

As we disambiguated the analysis results, we used hand-crafted rule based dependency parser to extract the noun phrases. The dependency parser constructs modifier links between tokens. The links have two important features which are type of the link and its score. The type of the link gives the

information about modification type; whether the modifier gives information about quantity of the modified token or its quality, etc. Score of the link shows the power of the link which is used in selection of rules when there is an ambiguity. That is to say, scores of the links represents how much it is the correct link when it is compared to others. Detailed information about link structures is given in Section 6.1.

Dependency parser connects links according to a rule set which consists of rules that determine the restrictions for constructing links between tokens. The rules also define scores and type of the link to be connected and put extra constraints when needed. Putting restrictions only to tokens that will be connected is not enough for handling complex structures since we need more information in analysis of a text, such as context or background information. Therefore, in our rule structure, we defined generic functions that can define constraints which use morphological and semantic information or even sentence structure. To the best of our knowledge, giving these type constraints to the rules that ease the job of rule designers and allow us to handle complex structures is the first study in the literature. Detailed information about rule structures is given in Section 6.2 and algorithm for constructing links is explained in Section 6.3. As we construct links between tokens, we extract noun phrases according to the links. Detailed information about algorithm for extracting noun phrases from constructed links is given in Section 6.4.

Chapter 5

Morphological Disambiguation

Morphological information of words is crucial for extracting noun phrases since we can obtain semantic information and grammatical function of the word in the sentence by using morphological information. However, ambiguity is the main problem of natural languages and a word can have many different meanings and morphological analyses. Reducing ambiguity is crucial for performing easy and correct operations on words. In this part, we propose a morphological disambiguation technique for Turkish words which we use in our dependency parser.

The rest of the chapter consists of three sections. Section 5.1 describes the related work in morphological disambiguation. Section 5.2 explains the proposed system. Section 5.3 describes the corpus and presents the performance results of the system.

5.1 Related Work

The related works about morphological disambiguation can be divided into three categories: statistical, rule based and hybrid which is the combination of the two approaches. Statistical approaches select the morphological parses using a probabilistic model that is built with the training set consisting of unambiguously tagged texts. There are various models described in the literature, such as, maximum entropy models (Ratnaparkhi, 1996; Toutanova and Manning, 2000), Markov Model (Church, 1988) and hidden Markov Model (Cutting et al, 1992). In rule based methods, hand crafted rules are applied in

order to eliminate some incorrect morphological parses or select correct parses (Daybelge and Cicekli, 2007; Oflazer and Tür, 1997; Voutilainen, 1995; Oflazer and Kuruöz, 1994). These rules can also be learned from a training set using a transformation based (Brill, 1995) or memory based (Daelemans, 1996) learning approaches. There are also studies that combine statistical knowledge and rule based approaches (Leech et al., 1994; Tapanainen and Voutilainen, 1994; Oflazer and Tür, 1997).

The disambiguation studies can also be divided according to the languages they are applied. Levinger et al. (1995) used morpho-lexical probabilities learned from an untagged corpus for morphological disambiguation of Hebrew texts. Hajic and Hladká (1998) used maximum entropy modeling for Czech which is an inflectional language. Morphological disambiguation of agglutinative languages, such as Turkish, Hungarian, Basque, etc., is harder than others because they have more morphological parses of words. Megyesi (1999) has used Brill's POS tagger with extended lexical templates to Hungarian. Hajic (2000) extended his work for Czech to five other languages including Hungarian. Ezeiza et al. (1998) combined statistical and rule based disambiguation methods for Basque. Rule based methods (Oflazer and Tür, 1997; Daybelge and Cicekli, 2007) and trigram-based statistical model (Hakkani Tür et al., 2002) are used for the disambiguation of Turkish words. Yüret and Türe (2006) propose a decision list induction algorithm for learning morphological disambiguation rules for Turkish. Sak et al. (2007) apply perception algorithm in disambiguation of Turkish Texts.

5.2 Disambiguation System

A Turkish word can have many morphological parses containing many morphemes that give us morphological information about the word. For example, the word *çiçekçi*(florist) has the following ***morphological parse*** (MP):

çiçek+Noun+A3sg+Pnon+Nom^DB+Noun +Agt+A3sg+Pnon+Nom. (1)

The first part gives us the stem which is *çiçek*(flower). Derivational boundaries are marked with ^*DB*. We define the part after the stem as the whole tag of the word. In parse, "^DB" shows that the word is derived from one type to another and its meaning has changed rather than its inflection. We define the final morphemes after the last derivation as the final tag of the word. For this example, the whole tag is:

Noun+A3sg+Pnon+Nom^DB+Noun+Agt+ A3sg+Pnon+Nom (2)

and the final tag is "Noun+A3sg+Pnon+Nom" where the type of derivation "Agt" is ignored. The rules that are learned by our system depend on the morphological parses, the whole tags or the final tags of words.

The general architecture of the system is given in Figure 2. Our disambiguation system consists of two main parts: training and disambiguation. The training corpus is used for the generation of the tables Most Likely Tag of Word Table (WordTbl) and Most Likely Tag of Suffix Table (SuffixTbl). WordTbl is used to retag the corpus by our Brill tagger in order to learn rules. WordTbl, SuffixTbl and the learned rules are used in the disambiguation process.

5.2.1 Generation of Tables

Two tables which contain statistical information about words and suffixes are generated using the training corpus. The first table (WordTbl) holds the frequencies of all morphological parses of words, and the second one (SuffixTbl) holds the frequencies of all possible morphological parses for suffixes.

Some morphological parses of words are rarely the correct parses of those words. For example, the word *kırmızı* has two meanings, one is "red" and the other one is the accusative form of word *kırmız* which is a bug name. However, most people do not know its second meaning because of its rare usage in daily life. In other words, a possible parse of a word can occur more than another

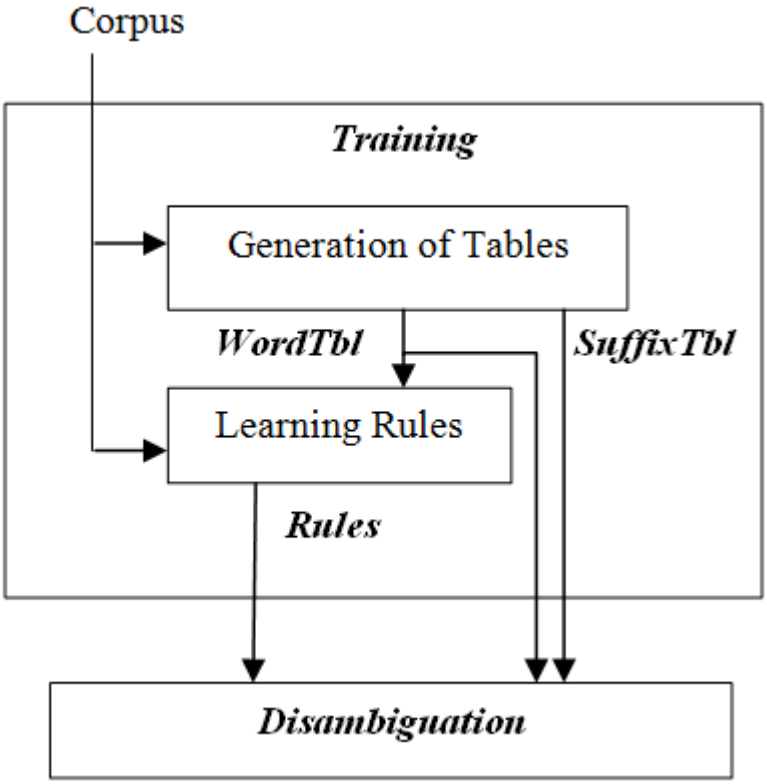


Figure 2. General Flow of Disambiguation System.

possible parse as the correct parse for that word in the corpus. For this reason, we generate WordTbl to contain the frequencies of all morphological parses for all words in our training corpus.

Since all possible Turkish words cannot be seen in a training corpus, WordTbl will not hold most likely parses for all words. In order to make an intelligent guess for the most likely parse of an unseen word, we use its suffix. For this purpose, we create SuffixTbl. In order to create SuffixTbl, we find suffixes of words according to their correct morphological parse and calculate the frequencies for tags corresponding to suffixes. For example, the suffix of the word *çiçekçi*(florist) whose morphological parse is given in (1) is "çi" and its corresponding whole tag is given in (2). We find frequencies of all corresponding whole tags for suffixes to store them in SuffixTbl.

5.2.2 Learning of Disambiguation Rules

In order to learn the disambiguation rules, we use a variation of Brill tagger. After all words in the corpus are initially tagged with their most likely parses using WordTbl, the disambiguation rules are learned by our Brill tagger.

The learned disambiguation rules are based on the morphological parses, the whole tags, or the final tags of the words. The general format of a disambiguation rule is as follows:

if conditions ***then***

select MPs containing TAG for $word_i$

The conditions of a rule depend on the possible MPs of the target word wordi, and the current selected MPs of the previous (or following) one or two words. Thus, the conditions of a rule can be one of the following:

- $wordC_i$ ***and*** $wordC_{i-1}$
- $wordC_i$ ***and*** $wordC_{i-1}$ ***and*** $wordC_{i-2}$
- $wordC_i$ ***and*** $wordC_{i+1}$
- $wordC_i$ ***and*** $wordC_{i+1}$ ***and*** $wordC_{i+2}$

If the condition of a rule r_i depends on more words than the condition of another rule r_j, we say that r_i is more specific than r_j. Each word condition $wordC_k$ is in the following form:

TAG of $word_k$ = TAG_a

The TAGs appearing in the conditions or MP selection part of a rule can be MPs, whole tags, or final tags. Thus, we call the rules based on MPs as MP Based (MPB) rules, the rules based on whole tags as Whole Tag Based (WTB) rules, and the rules based on final tags as Final Tag Based (FTB) rules. If two rules have the same number of condition words, the specificity relation among them depends on first the TAG in the selection part, then the tags appearing in

condition words in the order. MPB rules are more specific, WTB rules are more general than MPB rules, and FTB rules are the most general rules.

In the learning of disambiguation rules, we have used a variation of Brill tagger (Brill, 1995). We try all possible rules and select the rule which gives the best improvement. After applying the selected rule, we repeat the process in order to infer the other rules. These iterations end until there is no progress or the improvement is below a threshold. In the selection of the best rule, we differ from the original Brill tagger. We select the rule with the highest precision as the best rule in iterations. For example, if rule A causes 100 correct tags and 1 wrong tag and rule B causes only 10 correct tags without any wrong tags, the original Brill tagger may choose the rule A for that iteration. However, we select rule B because it has higher precision. The reason for this approach is that we want to increase the correctness of the condition words in the rule applications.

The rules are learned by using the dataset of 25098 hand-tagged words. After tagging all words in the training set with their most likely tags, we infer the best rule at each iteration of the algorithm. We generate all possible rules from all the words in the dataset. After generating all rules, we select the rule with the highest precision as the best rule. If there is more than one rule with the highest precision, we select the one which affects more words. When there is more than one rule with the highest precision and they affect the same number of words, we have the option to select the most specific one or the most general one, and we select the most specific one. We made this decision as a result of our empirical tests. In our empirical tests, when the most specific rules are selected 0.999 accuracy is obtained in training set by learning 395 rules. When the most general rules are selected, 0.995 accuracy obtained by learning 345 rules. As a result, we observed that using the specific rules is more preferable than using general rules.

5.3 Morphological Disambiguation

In the morphological disambiguation of Turkish words, we have used a hybrid disambiguation system which uses statistical techniques, rule based techniques and some heuristics. After the given Turkish text is morphologically analyzed by a Turkish morphological analyzer, the hybrid disambiguation steps are applied.

In our hybrid disambiguation tool, we use the statistical information in tables WordTbl and SuffixTbl, hand-crafted rules of SupervisedTagger, rules learned by our Brill tagger and some heuristics. The disambiguation algorithm consists of five major components:

- *Selection of the Most Likely Tag of Word*
- *SupervisedTagger Disambiguation*
- *Selection of the Most Likely Tag of Suffix*
- *Application of the Learned Rules*
- *Selection with Fall-Back Heuristics.*

The system tries to find the correct morphological parses step by step using the components in the given order. After the last step *Selection with Fall-Back Heuristics*, a single morphological parse will be selected for each word.

5.3.1 Selection of the Most Likely Tag of Word (MW)

The statistical information in the table WordTbl helps us to find the most likely parses of words appearing in the training set. If the word exists in WordTbl, the most frequent parse of that word is selected. Since not all words appear in the training set, some words will be still ambiguous at the end of this step. WordTbl may not contain all words because our training data set is small, and the number of unique Turkish words is huge. In one of our experiments, we determined that the number of unique words is 870.000 in a 6 billion word Turkish corpus. In

fact, this is one of the reasons that we decided to use a hybrid approach for the morphological disambiguation.

5.3.2 SupervisedTagger Disambiguation (ST)

In this step, the words are tried to be disambiguated by the SupervisedTagger software. SupervisedTagger uses 342 hand-coded disambiguation rules of two types: *selection* and *elimination* rules. The selection rules select a morphological parse directly. The elimination rules eliminate the wrong ones as much as it can. In other words the selection rules completely disambiguate words, and the elimination rules reduce the ambiguity levels of words. SupervisedTagger is applied only to ambiguous words. At the end of this step, there can still be ambiguous words.

5.3.3 Selection of the Most Likely Tag of Suffix (MS)

If the word is not disambiguated by the first two steps, we try to disambiguate using the statistical information in the table SuffixTbl. The possible suffixes of a word are determined according to its morphological parses, and the most likely morphological parse corresponding to those suffixes is selected if the suffixes appear in SuffixTbl. The word may not be disambiguated at this step because of the huge number of possible suffixes. In one of our experiments, we also determined that the number of unique suffixes is 40.000 in a 6 billion word Turkish corpus.

5.3.4 Application of the Learned Rules (LR)

The rules that are learned by our Brill tagger are applied in this step. The order of rule application is the order of learning. The conditions part of a rule contain a condition depending on the target word of the rule, and one or two more

conditions depending on other condition words. A rule can be applicable to a target word if all of the following are satisfied:

- Its condition words are completely disambiguated and satisfy their conditions.
- The target word is disambiguated and satisfies its condition or the target word is ambiguous and one of its still possible parses satisfies its condition.
- At least one of the possible parses of the target word contains the correct tag given in the selection part of the rule.

When a rule is applied, the target word can be completely disambiguated, or some of its parses are selected as its possible parses. If the target word contains only one morphological parse satisfying the correct tag, it is disambiguated; otherwise its parses satisfying the correct tag are selected as possible parses for the next step. For example, the following rule is applicable under the given conditions:

if final TAG of $word_i$ = *Adverb* ***and***
whole TAG of $word_{i-1}$=*Noun+A3sg+P3sg+Nom*
then select MP containing whole tag *Adjective* for $word_i$

If the whole tag of the selected MP of $word_{i-1}$ is Noun+A3sg+P3sg+Nom, the final tag of at least one of possible MPs for $word_i$ is Adverb and $word_i$ contains at least one possible MP having the whole tag Adjective, those MPs are selected by this rule for $word_i$.

5.3.5 Selection with Fall-Back Heuristics (SH)

At this last step, a small number of words can still be ambiguous. In this step, we perform the selection with fall-back heuristics in order to disambiguate the remaining ambiguous words. We have determined the following four heuristics

and applied them in the given order. The application order is determined empirically.

5.3.5.1 Selection of Non-Derived (SND)

Since Turkish is an agglutinative language, we can change the part of speech tags or the inflections of words by adding suffixes. SND heuristic selects the parses containing no derivation suffixes since non-derived words are more common than derived words.

5.3.5.2 Selection of Proper Noun (SP)

SP heuristics selects the proper noun senses of the words if their possible parses contain proper noun senses.

5.3.5.3 Selection of Noun (SN)

Nouns exist in texts more than other part of speech tags. SN heuristic selects the parses that their part of speech tags are noun.

5.3.5.4 Selection of Shortest (SS)

After applying all techniques and heuristics, if the word is still not disambiguated, we select the shortest parse in terms of the character length. Since this is selecting randomly, we hope that the number of words remains at this step will be a small amount.

5.4 Evaluation

We have constructed a data corpus consisting of 25098 tagged words. In the preparation of the corpus, we used Turkish texts from different news portals. The texts are analyzed by SupervisedTagger and ten graduate students tagged words with correct morphological parses. The statistical information about our dataset is given in Table 1. There are 12 different part of speech tags which are noun, proper noun, conjunction, pronoun, adjective, question, interjection, verb,

adverb, post-position, number and punctuation. The 47.7% of the corpus is unambiguous. The most ambiguous word has 16 different parses. There are 2063 distinct whole tags which show the ambiguity problem of Turkish.

Our disambiguation system uses five different techniques (MW, ST, MS, LR, and SH) step by step. It is obvious that the order of the techniques is crucial for the performance of the system. In order to see which order gives the best accuracy, we have applied each technique separately and obtained the average accuracies by using 10 fold cross validation. In Table 2, the second column shows the average number of words having more than one parse and they are processed by the corresponding technique. The accuracy of the technique for the applied words is given in the third column. The fourth column shows the accuracy for disambiguated words so far (disambiguated words by the technique plus unambiguous words (UW)).

Table 1. Statistics of Data Corpus

Number of words	25098
Number of distinct words	8493
Average number of parses per word	2.007
Number of words with one parse	11979
Maximum number of parses in one word	16
Number of distinct parses	18203
Number of distinct whole tags	2063
Number of distinct final tags	373
Number of proper nouns	3305
Number of non-proper nouns	9670
Number of derived words	4772

Since MW gives the highest accuracy, it is reasonable to choose MW for the first step. Applying the learned rules at the first step is not reasonable since there are not enough disambiguated words yet. The reason for having high accuracy so far is that we have few words that are disambiguated in the first step, and unambiguous words in the corpus increase the accuracy.

Table 2. Results of Techniques for the First Step

Technique	# of words applied	Acc. of Tech.	Acc. of (UW+Tech.)
MW	822.8	0.918	0.966
ST	802.4	0.798	0.919
MS	872.4	0.700	0.874
LR	31.3	0.690	0.992

For the second step, we tried MS, ST, and LR. The results are given in Table 3. As we compare Table 2 and Table 3, we can see that the accuracy of techniques increased, meaning that using more reliable techniques in the early steps causes an increase in the accuracy of other techniques by eliminating words that they cannot disambiguate correctly. Applying the learned rules (LR) at this step is again the worst technique. Using ST in the second step gives a higher accuracy than using MS. Disambiguating more words in earlier steps with higher accuracy is better since the ambiguous words will be disambiguated with less reliable heuristics unless we disambiguate them at earlier steps. Thus,

Table 3. Results of Techniques for the Second Step

Technique	# of words applied	Acc. of Tech.	Acc. of (UW+MW+ Tech)
LR	24.9	0.687	0.967
ST	260.3	0.874	0.961
MS	369.4	0.761	0.927

we select ST as the second, and MS as the third. Since ST and MS are better than LR, LR is chosen as the 4th component. The accuracy at the end of the 4th step is 0.942.

After the applications of the first four components, there still exist some ambiguous words, and the number of ambiguous words after applying MW, ST, MS and LR is 71.3 on average (2.4%). In order to disambiguate the remaining ambiguous words, we use fall-back heuristics. The order of heuristics can also

be important for the correct disambiguation of the rest. Therefore, we tried heuristics in different order to see their effects.

Table 4. Results for Fall-Back Heuristics

Heuristic	# of words applied	Acc. of Technique	Acc. of (UW+MW+SW+ST+LR+Tech.)
SP	27.1	0.679	0.9379
SND	6.8	0.809	0.9404
SN	9.4	0.787	0.9401
SS	71.3	0.497	0.9280

We first tried all of them separately in the first order to determine the most reliable one. In Table 4, the results for heuristics are given. From Table 4, we see that SND performs better than others but disambiguate less words while SN is the second in terms of accuracy. SS is the worst heuristic which is actually like selecting randomly. In addition, after SS, there are no words to be disambiguated anymore. Therefore, we tried all combinations where the first one can be SN or SND and the last one is SS to get the best accuracy. Since SND-SP-SN-SS order produced the best accuracy, we use that order for heuristics. Finally, we disambiguated all words having the accuracy of 0.935 by using the order of MW-ST-MS-LR-SH.

SupervisedTagger uses hand-coded disambiguation rules. In order to measure the performance of the statistical components of our system, we removed SupervisedTagger component from the system. The accuracy of the overall system is dropped to 0.924 from 0.935. This means that hand-coded rules help to improve the performance of the system. We believe that the importance of hand-coded rules would reduce significantly if we would train our system with a huge tagged corpus.

In the calculation of the accuracy, we consider the whole morphological parse. However, in some words, all parses have same inflections after their last

derivations so that they have the same grammatical function in the sentence. For example, the word “kaldır” has the following parses:

- kal+Verb^DB+Verb+Caus+Pos+Aor+A3sg (to make stay)
- kaldır+Verb+Pos+Aor+A3sg (to lift)

Although they have different meanings, they have the same final tag (Verb+Pos+Aor+A3sg), and they have the same grammatical function in the sentence. Therefore, choosing one of them does not make a difference unless the

Table 5. The Distribution of Wrong Disambiguation

Our Selection / True	Prop	Adj	Adverb	Noun	Verb	Sum
Prop	51	55	10	126	24	266
Adj	10	42	16	58	15	141
Adverb	1	32	14	26	1	74
Noun	187	92	15	583	42	919
Verb	9	14	0	22	51	96
Sum	258	235	55	815	133	1496

meanings are important. In the calculation of the accuracy, if we consider only the final inflections (the final tags), the accuracy of the overall system becomes 0.940. When only the final part of speech tags are considered, the accuracy becomes 0.976.

When we examined the mistakes of our system, we observed that most of the mistakes are in nouns, proper nouns and adjectives, verbs and adverbs. Since the rest of the mistakes are 7.7% of all mistakes, we will focus on only the mistakes of these 5 POS tags. In Table 5, the distribution of wrong disambiguation is given. In the calculation, the sum of mistakes in every fold is used. The left column shows the true POS while the above row shows our selection for the corresponding POS. We can see that adjectives are mostly confused with nouns. This is reasonable, since every adjective can also be used as noun in Turkish.

Adverbs are also mostly confused with adjectives. Nouns are mostly confused with nouns. This is an expected result since Turkish is an agglutinative language and there can be many different inflections from a stem. Verbs are most confused with verbs with different inflections. In addition, we can say that nouns are the POS tags mostly confused while adverbs are the least.

Chapter 6

Dependency Parser

In this chapter, we explain our proposed dependency parser approach for extracting noun phrases. After we get the results from morphological disambiguator, our dependency parser uses hand-crafted rules for constructing links between tokens in order to determine modifiers and modified tokens. Once we constructed links, we extract noun phrases by using them. In the following sections, we explain our link and rule structures, and algorithms for constructing links and extracting noun phrases from constructed links, respectively.

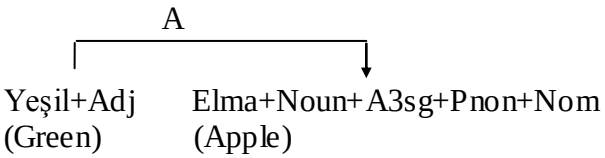
6.1 Link Structure

We use links to represent relationship between tokens. In order to understand the structure, features of links and specifications for constructing links are listed as below.

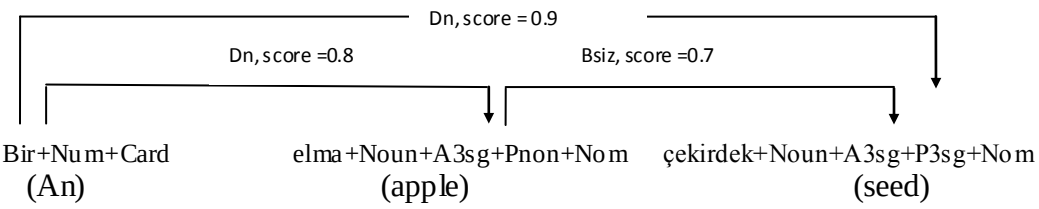
- A token can modify only one token however, a token can be modified by more than one tokens.
- A link can be constructed between only two tokens.
- There can be at most one link between two tokens.
- The modifier token is called *Source* of the Link and the modified token is called *Target*.
- The links have one direction which is from *Source* to *Target*. Since Turkish is head-final, the links are normally from left to right.
- Only head tokens of words can modify a token.

- Each link has a name representing the relation type between tokens (See Appendix A for the list of all link names).
- The non-head tokens are connected to the tokens at their right with a *DB* link.

A link example between two tokens is given below. The token *Yeşil* (green) modifies the token *Elma* (apple) and the link name is *A* which shows that the modifier is an adjective.



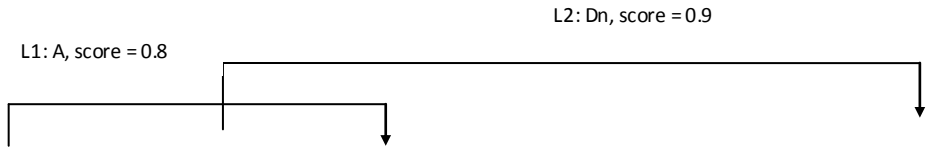
Ambiguity is the main problem in natural language processing studies. This fact is also true in determining the modifiers and modified token. A modifier must modify only one token as mentioned above, however, in some cases; we have to decide which link is the true one since there may be more than one possible link that can be constructed. In order to overcome this problem, we give every link a priority score showing which link is more likely to be. In the following example, the token *Bir* (An) can modify two tokens which are *elma* (apple) and *çekirdeği* (seed) with same link name *Dn*. *Dn* means that modifier gives quantity of the modified token. By giving different priority scores, we can determine the correct link by selecting the link with higher score. In the example, we select the upper link and so *Bir* (An) modifies *çekirdeği* (seed).



In addition, we do not allow crossing links to occur. There are two link examples, one is forbidden in our design and the other one is its correct form

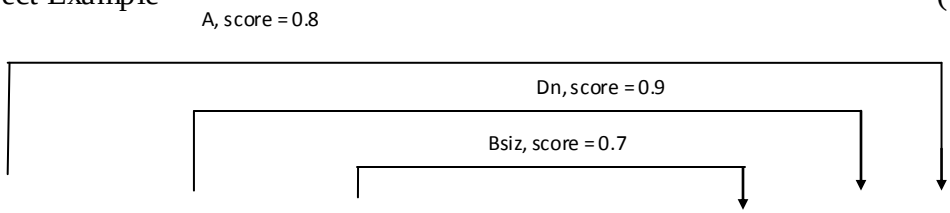
below. In the first example, L1 is not constructed since it causes crossing with L2.

Wrong Example (3)



Kırmızı+Adj Bir+Num+Card elma+Noun+A3sg+Pnon+Nom çekirdek+Noun+A3sg+P3sg+Nom
(Red) (an) (apple) (seed)

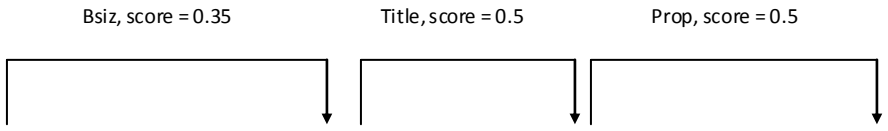
Correct Example (4)



Kırmızı+Adj Bir+Num+Card elma+Noun+A3sg+Pnon+Nom çekirdek+Noun+A3sg+P3sg+Nom
(Red) (an) (apple) (seed)

There is a list of example link structures that we represent for certain noun phrase types to understand our structure better:

- Person Names: Computer Engineer Ali Kaya



Bilgisayar Mühendisi Ali Kaya
Computer (Engineer) (Ali) (Kaya)

Morphological Parses:

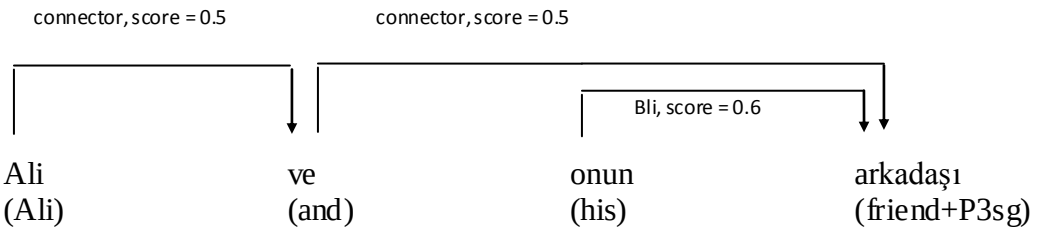
-Bilgisayar+Noun+A3sg+Pnon+Nom

-Mühendis+Noun+A3sg+P3sg+Nom

-Ali+Noun+Prop+A3sg+Pnon+Nom

-Kaya+Noun+Prop+A3sg+Pnon+Nom

- Conjunctive Noun Phrases: Ali and his friend



Morphological Parses:

-Ali+Noun+Prop+A3sg+Pnon+Nom

-ve+Conj

-o+Pron+A3sg+Pnon+Gen

-arkadaş+Noun+A3sg+P3sg+Nom

6.2 Rule Structure

In our system, we use hand-crafted rules for connecting links between two tokens. A rule consisted of 5 parts which *are Source, Target, Constraints, Priority and Link Name*. These parts specify beginning and end tokens of links, constraints for rule application, priority of the link and its name. A template of a rule is as follows:

CONSTRUCT a link between $Token_i$ and $Token_j$ with name of ***Link Name*** with a ***Priority*** IF $Token_i$ can be a ***Source*** and $Token_j$ can be a ***Target*** and all ***Constraints*** are satisfied.

For a sample rule structure, let's consider the following sentence:

“Bir öğrenci bu yaz üç yüz kitap okudu.”(A student read three hundred books in this summer)

In order to connect the words *üç*(three) and *yüz*(hundred), we can handle it by using the following rule.

CONSTRUCT a link between $Token_i$ and $Token_j$ with name of ***NN*** with a ***1.0*** IF $Token_i$ is a ***Number*** and $Token_j$ is a ***Number*** and ***there is no another token between them***.

In this example, the rule can be applied only when $Token_i$ is *üç* and $Token_j$ is *yüz*. When $Token_i$ is *Bir*(One) and $Token_j$ is *üç* (three), although both of them can be a Source and Target tokens, the rule will not be applied since the constraint (there is no another token between them) does not satisfy. Now, let’s see each part of the rule in detail.

6.2.1 Link Name

The name of the link is given in this part of the rule. By defining names to links, we can distinguish the types of modification. For example, if the modifier gives a quantity information about the modified token, we use the *Dn* link name and if the modifier gives information about quality or color of the modified token, we use the *A* link name. By naming links that helps us to distinguish them, we can use links names in writing constraints of rules which is explained in Section 6.2.5.

Some links can de directly used for noun phrases, such as, *A* or *Dn*. However, some links like NN do not define a noun phrase (See the example above, *üç bin* (three thousand) is not an NP). Therefore, another advantage of naming links is that it helps us to group them.

It should be noted that more than one rule can use same link name.

6.2.2 Priority

The priority of the link that is constructed should be a positive number where a bigger number means a higher priority. It should be noted that two rules with same link names can have different priorities.

The priorities of the rules are determined manually by the rule designer. As we have a large dataset, this property of our system allows us to have learning for determining real priorities of rules. This is left as future work because of lack of a large dataset.

6.2.3 Source & Target

In this section, we discuss *Source* and *Target* token properties. We have discussed both of them at once, since their specifications are same. Before applying a rule to a token, we first check whether that token satisfies the source/target properties of the rule. We can check many different properties of a token in this process which can vary in each rule. Therefore, we have defined two components for specifying *Source/Target*, one of them is *Source/Target* type that specifies which parts of the token are taken account and the other one gives the value to be checked. The types can be categorized in four groups: Token Based, Word Based, Context Based and Special Types. Now, let's see the types which are used in our system.

6.2.3.1 Token Based

We consider only the properties of the corresponding tokens in token-based types. There are 4 types in this category:

Part of Speech (POS): In this type, we consider only the POS tag of the token and ignore all other inflectional morphemes.

Part of Speech & Partial Inflections (POSPI): We consider POS tag of the token and look for existence of inflections defined in the rule.

Part of Speech & All Inflections (POSFI): We consider POS tag and all inflections of the token. Only stems can vary in this type.

Full Token (FT): We check exact match with the token's morphological parse. For example, if the value in the rule is "^DB+Adj+Rel", we will select the tokens which is not the first token of the word and having those morphemes.

6.2.3.2 Word Based

In this category, if the corresponding token belongs to a word which has more than one token, we consider also the other tokens of the word which are at the left side of the corresponding token. In the following example, if *Token*$_2$ is considered as a *Source/Target* token by a rule with this type of specification, we also consider the values of *Token*$_1$.

Kitapçıdaki (the one in the book store) (5)

Kitap+**çı**+da+**ki**

kitap+Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Loc^DB+Adj+Rel

Token$_1$ Token$_2$ Token$_3$

There are 6 types in this category:

Surface Form (SF): In this type, we look for exact match with the surface form of the token and value defined in rule. Let's consider the example (5). If the value in the rule is *kitapçıda* (in the bookstore), then the rule will be applicable for Token$_2$, since the morphological parse of *Token*$_1$ and *Token*$_2$ is as follows:

kitap+Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Loc

Since the surface form of this morphological parse is *kitapçıda*, the rule is valid only for Token2.

Starting with Surface Form (SSF): We check whether the surface form of the token starts with the value defined in the rule or not. Since Turkish is an agglutinative language, many words can be derived from a stem and this type is used for specifying tokens derived from a certain surface form. For example, if the value in the rule is *kitapçı* (bookstore), we accept all the tokens which have a surface form that starts with *kitapçı*, such as, *kitapçılar* (bookstores), *kitapçıdakiler* (the ones in the bookstore), etc. For (5), the rule is valid for $Token_2$ and $Token_3$ and is not valid for $Token_1$.

Starting with a Surface Form & Partial Inflections (SSFPI): We add inflection requirement to SSF type which allows us to use morphological information. For example, if the value in the rule is *"kitapçı, Loc",* then we accept all the token of the words which have a surface form that starts with *kitapçı* (bookstore) and having locative case marker, such as *kitapçıdakiler* (the ones at the bookstores). However, the tokens that does not have a locative case marker, like *kitapçıdan* (from the bookstore), *kitapçılarımız* (our bookstores), won't be accepted as *Source*/*Target*. For (5), the rule will be valid only for $Token_2$.

Lexical Form (LF): We check exact match with the value defined in the rule and morphological parse of corresponding token. We consider the morphological parses of the tokens at the left, too. For (5), if the value in the rule is "kitap+Noun+A3sg+Pnon+Nom^DB+Noun+Agt+A3sg+Pnon+Loc", the rule is valid for only $Token_2$.

Starting with Lexical Form (SLF): We check whether the morphological parse of the word that the token belongs to starts with the value defined in the rule. In this case, we can put a restriction to the previous tokens or we can define the stem with its POS tag which can be useful because of ambiguity. For example, the word *yakın* has the following morphological parses:

- yakı+Noun+A3sg+P2sg+Nom (your plaster)
- yakın+Verb+Pos+Imp+A2sg (burn)
- yak+Verb+Pos+Imp+A2pl (moan)

- yakın+Adj (near)
- yakın+Adverb (near)
- yakın+Noun+A3sg+Pnon+Nom (near)

If the value in the rule is "yakın+Adj", we will select all the tokens which have a morphological parse of "yakın+Adj" or tokens derived from them like *yakınlaş* (get close), *yakınlaştı* (he got close).

Stem with partial inflections (SPI): In this type, we specify the token with only stem of the token and with its desired inflections. For example, the value in the rule is "yak, Verb", we will select the tokens which have a stem of verb *yak* (burn), like *yaktı* (he burned), *yaksaydı* (what if he burned). When a word has a derivation suffix, its meaning changes but it still has a related meaning with that word. So we can group words having related meanings with this type specification. If the word has more than one token with given POS tag, then the rule will be valid for all that kind rules. For example, if the value in the rule is "kitap, Noun", the rule will be valid for $Token_1$ and $Token_2$ for (5).

6.2.3.3 Context Based

Some grammatical functions of words can change according to context. In order to handle situations where context information is important, we have defined 2 context based types:

Surface Form of a group of words at left (SFWL): We check the context and look at the words at left side of the corresponding token. This type is needed for handling some collocations and noun phrases that act as adjectives. For example, in the following word group, *insanlık dışı davranış* (inhumane behavior), *insanlık dışı* means inhumane and this group act as adjective although the word *dışı* (out of) is a noun. By using SFWL, if the value in the rule is *insanlık dışı*, then the rule is valid for tokens having surface form of *dışı* (out of) where surface form of the previous word is *insanlık* (humanity). However, it has to be mentioned that this does not connect the words *insanlık* (humanity) and *dışı* (out of). For connecting them, another rule should be written.

Surface Form of a group of words at right (SFWR): The only difference with SFWL is that, this time, we check the words at right side of the token.

6.2.3.4 Special Types

We have two special types which cannot be categorized as the others.

LIST: A link can be applicable for many different sources/targets. Therefore, grouping them will be helpful so that we do not need to write same rule for different sources/targets again and again. We write all the sources we want to use for a rule in a file by using the source types explained in this section and use this list when needed. A list file can contain different source/target definition types.

ANY: ANY is a reserved word which means that every token can be a *Source/Target* for the corresponding rule. We use this type for handling noun phrases with quotation marks. Quotation marks can be used for giving stress for words between them. So there can be any words with any type between two quotations. Therefore, we cannot put any restriction to them. For example, in the following sentence, *"Dur" uyarısına* ("Stop" warning) is an NP and the first quotation mark should be connected to the token *Dur* (Stop) which is a verb.

Sürücü "Dur" uyarısına uymadı. (The driver didn't obey the "Stop" warning.)

It is a fact that we can define Source and Target tokens with more than one definition type. However, having so many types will ease the job of rule designers and can be useful for handling situations that we can face in future.

6.2.4 Constraints

Although source and target can be considered as constraints, we may need more constraints for handling more complex noun phrase structures. Therefore, in each rule, we can also put as much as constraints we want by using generic functions we defined. Each constraint is connected each other with AND operation, that is to say, if one of the constraints does not hold, the rule cannot

be applied. In addition, a constraint can include more than one constraints connected with AND, OR and NOT logic operations. In the Figure 3, we give a template of rule constraints. We assume that capital letters are functions returning a boolean result. The rule in the Figure 3 has 5 constraints and failure in any of them will prevent the construction of the link. In constraints we use '!' character for NOT operation. So in 2nd constraint, the function B must return false for the success of the rule. Additionally, we use '|' character for OR operation and '^' character for AND operation. In 3rd constraint, C or D functions should return true for the success of the rule. We use brackets to group the functions which allow us to write long and complex constraints. In 4th constraint, E should return true and F must return false. In the last constraint, we can see that we have two groups connected with OR operation. G and H should return true or I and J should return true for the success of the rule.

```
<Rule>
....
        <Constraint> A </Constraint>
        <Constraint> !B </Constraint>
        <Constraint> [C|D] </Constraint>
        <Constraint> [E^ [!F]] </Constraint>
        <Constraint> [[G^H] | [I^J]] </Constraint>
</Rule>
```

Figure 3. Rule Constraint Template.

Defining boolean functions is also an important part of our system. We have defined some flexible functions that can take parameters and perform a specific operation. The functions can be categorized into three groups: Only-One Token functions, Range functions and Both-Token Functions. Now let's see them in detail.

6.2.4.1 Only-One-Token

These functions are applied to a single token. The template of a function is as follows:

Token: FUNCTION NAME (Parameter1, Parameter2,…)

For defining the token that function will be applied, we use indexes of source and target tokens of the link to be constructed. Following notations are used in our rule formats for defining indexes:

- S -> Source Token
- T -> Target Token
- S+x -> The x^{th} token beginning from source token in the right direction.
- S-x -> The x^{th} token beginning from source token in the left direction.
- T+x -> The x^{th} token beginning from target token in the right direction.
- T-x -> The x^{th} token beginning from target token in the left direction.
- B -> First token of the sentence
- E-> Last token of the sentence

This type definition of tokens allows us defining specific and powerful rules that cannot be defined in link grammars. We can exactly determine the token and use context information. There is a list of defined only-one-token functions below.

- **InList (LISTNAME):** This function searches the token whether it is in the given list or not. In order to use semantic information, this function can be used. For example, if we do not want the source token to be a country name, we can write the following constraint :
    - !S:inList(CountryNameList)
- **Contains(Morpheme):** This functions checks whether the morphological parse of the token contains "Morpheme" or not. For example, if we do not want the target token to have `Pnon` morpheme. Then, we can use the following constraint :

    - !T:contains(+Pnon)
- **hasEndLink(LinkName):** This function checks whether the token is modified by a token with the given link name
- **hasBeginLink(LinkName):** This function checks whether the token modifies a token with the given link.
- **derivedFrom(POS,ExceptList):** This function checks whether the previous tokens of the word that target token belongs to has the POS tag given as parameter. Second parameter is not mandatory. If it is also given and if the word is an element of the exception list, function returns true. Since the scope of our study does not include words derived from verb and affecting sentence structure, this function play an important role. An example constraint can be as follows:
    - !T:derivedFrom(Verb)
- **isPossible(Morpheme):** Disambiguation can sometimes eliminate correct morphological parses and select the wrong one. In some situations, we may want to consider eliminated parses, too. In order to do this, we use *isPossible* function that checks whether there is a parse containing Morpheme given as parameter. For example, "İlyas Çiçekli" is a person name and surname, however, *Çiçekli* can be considered as an adjective by the disambiguator, too. In case a wrong disambiguation is performed, we can handle it with this function. An example can be as follows:
    - T:isPossible(Prop)
- **isNPBeginning():** This function takes no parameter and checks whether the token is a beginning token of a noun phrase or not. We can understand this by looking the modifier link of the token. If the token has a modifier link and the link is one of the links that can start an NP, we can say that token is a beginning token of an NP.
- **isNPHead():** This function takes no parameter and checks whether the token is a head token of a noun phrase or not. We can understand this by

looking the links that modifies the token. If one of the links can start an NP, we can say that the token is a head of an NP.

- **isConnectedToLink(Link Name):** This function checks whether the token is somehow connected to the link type given as parameter.
- **isEndToken():** This function checks whether the token is a head token of a word or not.
- **isLastWord():** This function checks whether the word that token belongs to is last word of the sentence.
- **turnsInto(POS):**This function checks whether the preceding tokens of the word that target token belongs to has the POS tag given as parameter. An example constraint can be as follows:
    - !T:turnsInto(Verb)
- **isFirstCharacterCapital():** This function checks whether the word that target token belongs to starts with a capital letter or not. This function is useful to determine proper nouns since proper nouns always start with a capital letter.

6.2.4.2 Range Functions

This type functions are applied to all tokens in a defined range. Template of a range function is as follows:

Beginning Token->Ending Token: FUNCTION NAME (Parameter1, Parameter2,…)

For defining beginning and ending token, we use same notation explained above. An example is as follows.

S+1->T-1: FUNCTION NAME ()

In this example, we apply the function to the tokens from the token next to source to previous token of target token. There is a list of defined range functions below.

- **numberOfWords(Integer):** If number of words between beginning and ending tokens is equal to the value given as parameter, function returns true; otherwise, returns false. For example, if we do not want to have any word between modifier and modified word, we can use the following constraint :
    - S->T:numberOfWords(0)
- **containsUnconnectedToken():** This function checks whether there is a token that has no relation with the tokens in given range. For example, let's consider the following sentence: "Küçük çocuk güzel okula gitti."(The small child went to a beautiful school). If we want to create links between adjectives and nouns, we will have two links from the token *küçük*(small), which are *küçük-çocuk*(small child) and *küçük-okula*(to small school). In order to prevent second option, we can use this function since there is no relation between *çocuk* and *okula*.
    - !S->T:containsUnconnectedToken()

There are also functions like constainsPOS, containsList, etc. which are similar functions explained in Only-One-Token section. The only difference is we apply the functions to all tokens in range and return false if any of the tokens does not satisfy the constraint.

6.2.4.3 Both-Token Functions

This type functions are applied two tokens for finding agreements. Template of a range function is as follows:

Token1~Token2: FUNCTION NAME (Parameter1, Parameter2,…)

For defining Token1 and Token2, we use same technique mentioned before. There is a list of defined Both-Token functions below.

- **agree(type):** This function has two types. One of them is agreement on person and other one is agreement on plurality of the token. The function checks whether there is agreement on given type. By this constraint, we

prevent occurrence of noun phrases like *bir adamlar* (one men) that has no agreement on plurality. An example statement is as follows:

  - S~T:agree(plurality)

- **hasSameInflectionAtEnd():** This function is used for conjunctive noun phrases and prevents conjunctive noun phrases that has different case markers. For example, *evler, arabalar ve çocuklar* (houses, cars and children) is a valid noun phrase, however, *evde, arabalarda ve çocuklar* (at home, in cars and the children) is not a valid noun phrase because of having different inflections. It is to be mentioned that last noun phrase can have different case markers if the others are in nominative form. For example, *Ankara, İstanbul ve İzmir'de* (Ankara, Istanbul and in Izmir) is considered as a valid noun phrase.

Now, let's see a sample rule in order to be clearer. Rules are written in XML format. The following rule connects adjectives to nouns and proper nouns.

```
<NounPhraseRule>
    <ID>1</ID>
    <sourceType>POS</sourceType>
    <source>Adj</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>A</linkType>
    <priority>0.5</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]
    </constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!S:hasEndLink(Proper)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
    <constraint>[![T:isPossible(Verb)]^[S:isPossible(Adverb)]]
    </constraint>
</NounPhraseRule>
```

Since the source type is POS and source is ADJ, the rule will consider all tokens having adjective as POS tag. The target type is LIST which will check the given list which is in format as follows:

POS Noun
POS Prop

So the rule will consider all tokens having Noun or Prop as POS tag. By using this list, we do not have to rewrite the same rule for different targets. Link type is given as 'A' and priority is given as 0.5. Then the constraints part comes which consisted of 8 constraints. First and third constraint check tokens between source and target whether there is any token having verb as POS tag or not. Second constraint checks whether there is a token that cannot be between modifier and modified tokens (i.e. semicolon). The reason of 4th constraint is as follows. There can be punctuations between the modifier and modified token. For example, let's consider the following NP:

hızlı, ucuz ve yeni bir araba(a fast, cheap and new car)

The adjective *hızlı* (fast) modifies *araba* (car) and there is a punctuation between them. However, when the number of tokens between source and target is 1, we cannot allow this (i.e. *hızlı, araba* (fast, car)). This constraint shows the power of rules for defining complex situations.

Some nouns can act as an adjective and modify a noun. 5th and 7th constraints prevents building links between adjectives and nouns acting as adjective. In the following phrase, the word *güzel* (beautiful) can modify the word *altın* (gold) since *altın* can have a noun POS tag. 5th constraint prevents this kind modification.

Güzel altın saat (beautiful gold watch)

Some surnames or names can also be an adjective. For example, let's consider the following sentence.

İlyas Çiçekli öğrencisiyle konuşuyor. (İlyas Çiçekli is talking to his student.)

Çiçekli is a surname but also it has another MP which is an adjective meaning "with-flower". Therefore, without sixth constraint, the token *Çiçekli* can be connected to the token *öğrencisiyle* (to his student) when there is a mistake in disambiguation. This constraint prevents connecting links if the source is modified with a Proper link meaning that the modified token is a proper name.

Morphological disambiguation does not work with 100% success and adverbs and adjectives can be confused in this process. By using 8th constraint, we do not let build links if the source has an adjective MP and target has a verb MP and both are eliminated in disambiguation process. Of course, this constraint has some risk, too, since it can prevent some correct NPs.

6.3 Dependency Parser Algorithm for Connecting Links

In this section, we explain our algorithm for connecting links between tokens by using rules explained in previous section. We start processing every token of a sentence from the end to the beginning of the sentence and we repeat this iteration if a new link is constructed during the current iteration. A formal representation of the algorithm can be seen in Algorithm 1.

We first create empty lists of links for every token in the sentences (Line 1). These lists store all possible links that can be constructed from corresponding token where the token is the source. As mentioned above, our algorithm runs in a loop (line 2-42) and if any new link is constructed in the iteration, we pass every word again to find a new link. The reason for this is that there are some rules that depend on existence of certain link types and these new links can cause those rules to be applied.

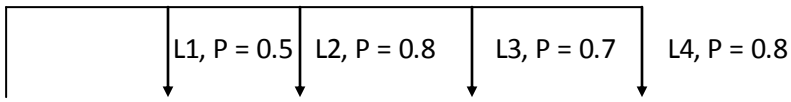
```
1.  Create empty Link lists for each token in the sentence
2.  while new links have been contructed
3.  {
4.     Token sourceToken = last token of Sentence S
5.     while S has more tokens
6.     {
7.        if sourceToken is an head token
8.        {
9.           for each Rule R
10.          {
11.             if sourceToken is a suitable source for R
12.             {
13.                for  i = index of sourceToken +1, i ≤ number of Tokens in
14.                     S,i++
15.                {
16.                   targetToken = ith Token of S
17.                   if targetToken is a suitable target for R
18.                       AND all constraints of R are satisfied
19.                   {
20.                      if possible link has contradiction with previous
21.                         built links AND  have smaller precison
22.                            do not built the link
23.                      else
24.                      {
25.                         if there is a contradiction but
26.                            it has a higher priority
27.                                Delete the previously built link
28.                         connect Link L from sourceToken to targetToken
29.                         add L to link list of the sourceToken
30.                      }// end else
31.                   }// end if
32.                }// end for
33.             }// end if
34.          }// end for
35.          sort the links the link list of the sourceToken
36.          according to their priority
37.       }// end if
38.       sourceToken = previous token of sourceToken
39.    }// end while
40.    if no new link is constructed
41.       exit
42. }// end while
```

Algorithm 1. Connecting Links

We start each iteration from at the end of the sentence (line 4) and move to left token when a process of a token is finished (line 38). When we select a token as a possible source token, we first check whether it is a head token of a word or not (line 7). If it is not a head-token, we do not process that token and skip it and move to next one (line 38). Otherwise, we try to apply each rule R in the rule set to sourceToken. If the token can be a source for that rule, we look for target token for that rule. We consider every token that is at right side of the sourceToken as a possible target token (line 13). If a token can be a target token according to rule R and all constraints of rule R are satisfied (line 17-18), we check cross link situation. If there will be a cross when this possible link is

constructed, we prevent one of the links that cause the cross. In preventing one of these links, priority of them plays an important rule. In (3), we do not let the link L1 to be constructed since L1 has a less priority than L2's priority. However, if the priority of L1 were higher than priority of L2, then we would construct the link L1 and delete L2. If there is a cross and priority of R is smaller, we do not build a link (Line 20-22). If there is a cross but priority of R is higher, we delete the link that cause cross (Line 24-25), we connect our link (line 28) and add the link to link list of the sourceToken (line 29). As we are done with all rules and targets, we sort the links in corresponding list according to their priority (Line 35-36). Since a token can modify only one token, we select the link with the highest priority value as the modifier link of the token. If there are more than one links with highest priority value, we select the one that the distance between target and source tokens is smallest. An example for selecting the link with highest priority is given below where P shows the priority value of the corresponding link. We select L2 since it has the highest priority value. Although L4 is has same priority, we eliminate it since target token of L2 (uçağında) is nearer than target token of L4 (çantaları).

Bags of people in American passenger plane are lost



Amerikan yolcu uçağında+ki insanların çantaları kayboldu.

As we processed all tokens and couldn't construct a new link, we end the process. Otherwise, we repeat the all process again to connect new links. The algorithm does not have an infinite-loop problem. In order to have an infinite loop, we have to build and delete same link with same priority again and again. However, links are deleted according to the priority of the rules. The link with the highest priority will be definitely built and once it has been build, other links are built without having a cross situation with it. The links will be built

according to their priority, respectively, as long as they do not have a cross situation with a link with a higher priority. Therefore, our algorithm definitely comes to a stable situation and ends the iterations.

6.3.1 Sample Link Construction

In this section, we give an example link construction. A representation is given for the trace of the algorithm. For construction of links we use an abstract rule set which is consisted of three rules. The properties of rules are given in Table 6. For example, Rule1 connect links between T2 and T3 with a 0.2 priority.

Table 6. Abstract Rule Set for the Sample Link Construction

Rule ID	Source	Target	Priority
1	T2	T3	0,2
2	T2	T4	0.5
3	T1	T3	0.9

We use an abstract sentence which has 4 tokens in the order: T1, T2, T3 and T4. The trace of the algorithm is given in Table 7 where constructed links are given at each step.

The algorithm starts from the last token, T4. Since T4 and T3 cannot be a source for our rules, we do not build any link starting from them. When the sourceToken is T2, we build two links: T2-T3 and T2-T4. Since priority of Rule 2 is higher than priority of Rule 1, we build a link between T2 and T4. However, when the sourceToken is T1, we build a link between T1 and T3 which causes a cross with T2-T4 link. Since T1-T3 link has a higher priority, we delete T2-T4 link. Since we build new links in this iteration, we start a new iteration. Again no link is built for T3 and T4. When sourceToken is T2, we can build again two links as before. Since T2-T4 link has a contradiction, we do not build it and build a link between T2 and T3. When sourceToken is T1, since we already build a link and cannot build another link with a higher priority, we keep the

current link. In 3rd iteration, we process every token again. Since there is no new link in this iteration, we end the process at the end of this iteration.

Table 7. Trace of the Algorithm for the Sample Sentence and the Rule Set

Iteration Number	Source Token	Links
1	T4	Empty
1	T3	Empty
1	T2	T2-T4
1	T1	T1-T3 (T2-T4 is deleted)
2	T4	T1-T3
2	T3	T1-T3
2	T2	T1-T3, T2-T3
2	T1	T1-T3, T2-T3
3	No more link is constructed and after this iteration, algorithm ends	

6.4 Algorithm for Obtaining Noun Phrases from Links

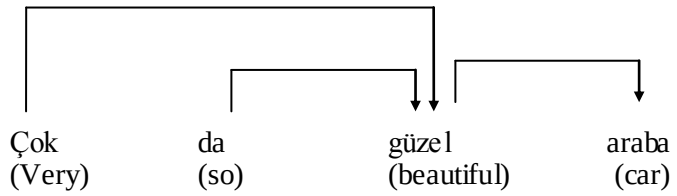
Although we can find every relationship between tokens of a sentence with our system, we will focus on only noun phrases because of the scope of our study. In our scope, as we mentioned, we try to find nested noun phrases, meaning that every noun phrase can contain sub-noun phrases. While main noun phrases can be extracted easily, detecting boundaries of sub-noun phrases requires more complicated process. In addition, noun phrases like person and institution names are to be handled specifically. Therefore, we developed an algorithm for extracting noun phrases which can be easily used for other sentence parsing operations. Formal representation of our algorithm is shown in Algorithm 2.

In the algorithm, NounPhrases is the list where we store all main noun phrases of the sentence (Line 1). We process every word W of sentence S, from left to right. A noun phrase cannot start with a conjunction, like *ve* (and), or a punctuation, etc. Therefore, we process a word W if it can be a starting word of

```
1.  NounPhrases = empty list
2.  for each word W of Sentence S
3.  {
4.       if  W can be a starting word of an NP
5.       {
6.           np is an empty string
7.          Token T = head token of W
8.          Link L = modifier link of T
9.          if  L can be a starting link of an NP
10.         {
11.             while L != null
12.            {
13.                Token B = source token of  L
14.                Token E = target token of L
15.                for each token C between B and E
16.               {
17.                    if  C has no modifier link
18.                        get the nex word and go to line  4
19.                    np += surfaceform of current token
20.                }
21.                if nounPhrase is a proper nounPhrase AND  L is an NP-Link
22.                    Add np to nounPhrases
23.                L = modifier link of E
24.                }// end while
25.           }// end if
26.    }// end if
27. }
28. add single noun phrases
```

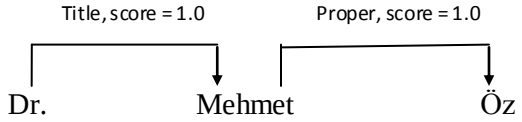
Algorithm 2. Extracting Noun Phrases from connected links

a noun phrase (Line 4). We get the head-token of W, since no links can be constructed beginning from a non-head token (Line 7). In some noun phrases, some tokens give stress only, instead of extra semantic information. For example, let's consider the following example.



In the noun phrase, the word *da* gives only stress and we cannot have a noun phrase "da güzel araba" since it will be meaningless without the previous word *Çok*. That is to say, it loses its function with previous word *Çok*. Therefore, we use a list that contains link names which cannot be a starting link of a noun phrase. When we faced with a token with a modifier link in this list, or a token that has no modifier link, we skip that word and move to next one (Line 9 - 11). We start moving with links as much as we can to obtain longest noun phrase and add sub-noun phrases in path, if exists. We get beginning and ending tokens of L

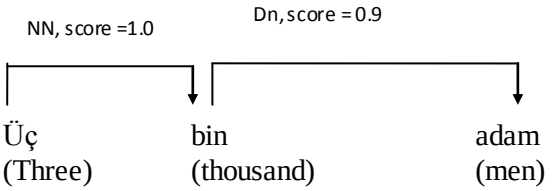
(Line 13-14) and for each token between these tokens we check whether they are connected to any tokens between B and E. If any token is not connected, we end our job with that word W and get the next word and go to Line 4 (Line17-18). Otherwise, we update the surface form of noun phrase to be extracted (Line 19). As we come to the end of the link, we need more checks to be sure that the noun phrase is valid or not. The last token of current noun phrase can have a modifier link which connects a person name with his/her surname and name and surnames shouldn't be in different NPs. Therefore, if the last token has that kind of link, we do not form a noun phrase which ends with this token. For example, in the noun phrase below, we shouldn't have a sub-NP like "Dr. Mehmet" since "Öz" is the person's surname.



The last token cannot also be a conjunction word, like *ve*(and), *veya*(or), etc. which cannot be head word of a noun phrase.

In addition, we grouped links as noun phrase links and non-noun phrase links since some links may not represent a noun phrase. For example, let's consider the following example.

Three thousand men: (6)



When we start extracting from word *üç*(three), we do not extract a noun phrase *üç bin* (three thousand) since NN link is not defined as noun phrase link. At line 21, we perform these checks and add the noun phrase in case of success. Then, we get the modifier link of E and repeat the process until L is null. When L is null, we reach the largest NP that we can have by starting from word W.

Later on, we select the next word and apply same process. When adding a noun phrase, we check its starting and ending points. If another noun phrase covers it, we add the noun phrase as sub-NP of that noun phrase. As all words are processed and we are done with all links, we need to form single nouns in conjunctive noun phrases as sub-NPs and also generate noun phrases with single nouns having no links (line 28) since if a noun is not modified by another noun or does not modify another noun, than it can be considered as a single NP.

6.5 Sample Noun Phrase Extraction

In this part, we will give a sample noun phrase extraction of a sentence by giving results of each step explicitly. Let's consider the following sentence.

Kırmızı başlıklı kız evin büyük kapısını açtı. (The girl with red cap opened the big door of the house)

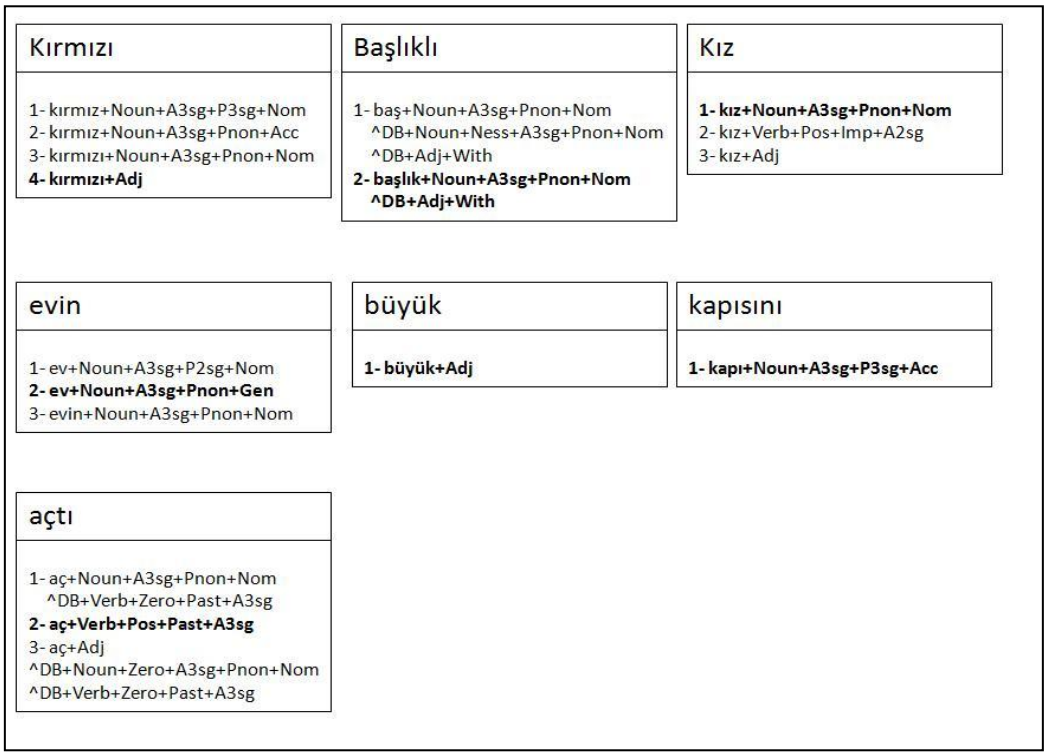


Figure 4. Morphological Parses of Words of Sample Sentence.

First, we will obtain the morphological parses of each word. The morphological parses can be seen in Figure 4. We can see that there are some ambiguous words. Therefore, we perform our disambiguation technique and select only one MP for each word. The bold MPs show the selected ones in Figure 4.

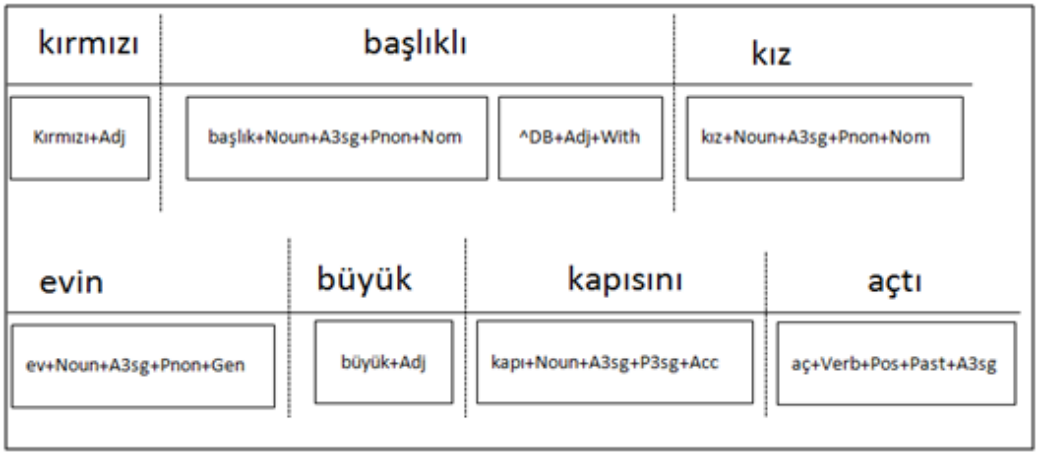


Figure 5. Tokenized Sentence According to Selected MPs.

Then we construct the sentence with using tokens. The sentence with its selected MPs can be seen in Figure 5. We obtain the tokens according to words' selected MP.

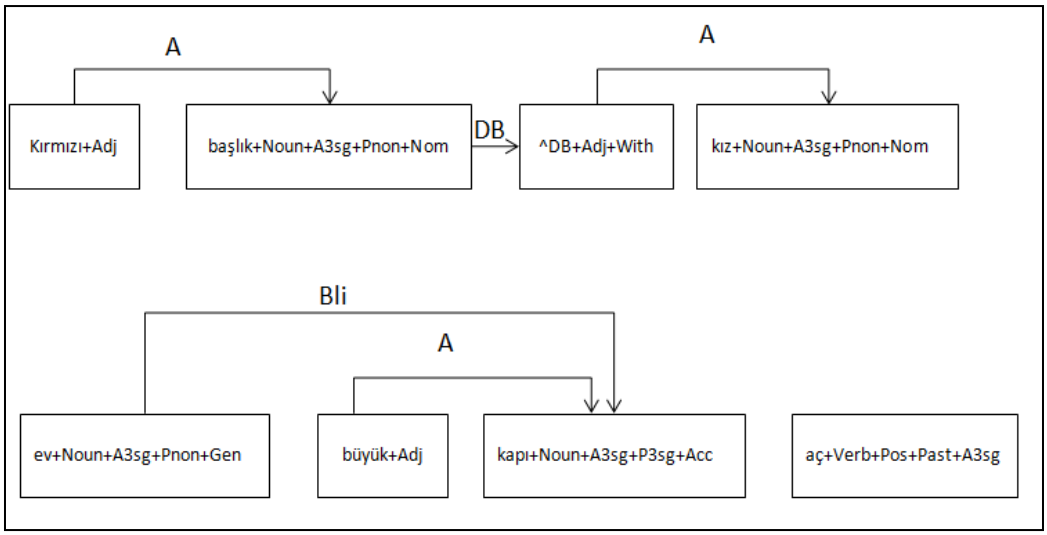


Figure 6. Sample Sentence with Links.

After disambiguation, we apply the rules to construct links. Constructed links can be seen in the following Figure 6. As we construct the links, we extract the noun phrases. There are two nested NPs in the sentence which are:

- *Kırmızı başlıklı kız* (girl with red cap). This NP contains also another NP which is *Kırmızı başlık* (red cap)
- *evin büyük kapısı* (the big door of the house). This NP contains also another NP which is *büyük kapısı* (the big door)

Chapter 7

Evaluation

In this chapter, we give details about the datasets for testing our system and give results of test and discuss them.

7.1 Experimental Setup

One of the biggest problems in NLP studies about Turkish is lack of ground truth datasets. There had been no dataset for testing noun phrase chunkers for Turkish before our study. Therefore, we had to construct our small datasets. The noun phrases in datasets are tagged manually by a native speaker. In order to ease tagging process, we have implemented a tagger tool, NPTagger, which can be also used by other researchers. The tool has the following features:

- A text file or TXT output file of SupervisedTagger can be given as an input to the NPTagger
- Save and load operations are available for editing.
- Surface form, type and head word of NPs are to be decided by the user.
- NPTagger automatically generates nested NPs.
- The types can be changed by user. So the tool can be used for tagging other phrases, too.

We have constructed three manually tagged datasets (D1, D2, D3) having different properties by using NPTagger. D1 and D3 are generated by using news

articles while stories which are consisted of short sentences are used in construction of D2. Morphological analysis of words in D3 is tagged by humans, that is to say, we can use it in order to see effect of morphological disambiguation.

Some statistical information about datasets is given in Table 8. Main NPs are directly components of sentence and are not a sub-NP of any NP. In forth row, number of main NPs is given while in fifth row, number of all NPs including sub-NPs is given.

Table 8. Statistical Information about Datasets: D1, D2 and D3

Features	D1	D2	D3
Number of Words	5695	3854	3152
Average Length of Main NPs	2.04	1.59	2.05
Total Number of Main NPs	1743	1208	990
Total Number of NPs including sub-NPs	2511	1398	1424

7.2 Results

We have written 98 rules and applied it to datasets mentioned above (See Appendix B for full list of rules). Most of the rules are designed by considering D1 and D3. After a pre-testing with D2, we have made only small changes. In designing rules, we didn't write rules that would cause over-fitting problem.

Noun Phrases can be at different sizes in terms of words and it is obvious that finding longer NPs is more difficult than shorter ones. In Table 9, detailed information about NPs in D1 and ones we found and results are presented separately for each different size of NPs. The second row shows the number of main NPs in D1 according to size of NPs. In third row, total number of main and sub-NPs in D1 is given. In forth row, number of main NPs we found is given while fifth row gives the total number of NPs we found. In sixth row, we show the number of main NPs found correctly. In this calculation, if a main NP we found is a main NP in dataset, we count it as a hit, otherwise it is counted as a

miss. In seventh row, we give the performance of the system with another evaluation called Total NP Match. We find how many of the NPs in dataset including sub-NPs are found. This is a simple calculation that calculates number of correct NPs ignoring whether it is a sub-NP or a main NP.

Table 9. Results of D1 According to NP Length

Length of Noun Phrase	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***≥10***	***Sum***
# of Main NPs in D1	775	548	211	93	58	23	20	9	2	4	1743
# of Total NPs in D1	851	1005	347	149	77	39	25	11	2	5	2511
# of Main NPs we found	882	536	225	86	51	18	21	7	2	4	1832
# of Total NPs we found	939	965	344	132	75	32	28	10	4	4	2533
# of Main NPs found correctly	720	492	184	73	45	13	16	5	1	3	1552
# of Total NPs found correctly	778	880	293	108	59	24	20	7	1	3	2173

In finding main NPs, the system performs best for NPs of size 2. NPs with only one words is the runner up while correctness in NPs size of 3 is the third. From the table, we can see that our system can correctly find NPs which are longer than 10 words.

By using information in Table 10 we can calculate recall, precision and F-Measure Value for Total NP Match, which are showed in Table 8. As it is explained in our algorithm, we first find noun phrases according to links and then we mark as nouns and pronouns that have no links, as noun phrases. Therefore, NPs with size of more than one word shows the performance of our rules. Because of this reason, we show the results when single NPs are ignored in third row.

Now let's analyze the reasons of the system's mistakes in D1. According to our observations over the results, we can categorize the reasons of the mistakes as follows:

Table 10. Exact Match Results for D1

Measure Type	F-Measure Value	Recall	Precision
Exact Match with all NPs	0.862	0.865	0.858
Exact Match with non-single NPs	0.857	0.840	0.875

- ***Wrong Disambiguation***: A mistake in disambiguation step affects our rule constraints, sources and targets. It can cause mistakes for other NPs, too. The following example couldn't be found by our system since the word has 6 meanings and two of them are `burn` as a verb and 'close' as an adjective. Since our disambiguation system chooses the verb sense for *yakın* (burn) one, we couldn't find that NP.

  Yakın kaynaklar (close sources)

  In addition, after finding NPs according to links, we mark nouns and pronouns that have no links as single noun phrases. Mistakes in wrong disambiguation directly affect these kind mistakes.

- ***Sentence Structure:*** In Turkish, the sentence order is SOV, as we mentioned. This structure causes mistakes like in the following example.

  Adam arabasını tamir etti. (The man fixed his car) (6)

  In this sentence, there are actually two NPs which are *Adam* (The man) and *arabasını* (his car). As we mentioned before, in Turkish, a nominative noun can be a modifier of a third-person possessive noun. When we put a rule for handling these kind NPs, we find also wrong NPs like *Adam arabasını* (Accusative form of "man car"). It is very easy to confuse the objects and subjects in Turkish since their position in sentence is so close. These kind mistakes cannot be seen in languages having order of Subject-Verb-Object like English

- ***Wrong Head Selection:*** In nested NPs, although we find the main NP correctly, there is a common mistake that we do in determining head token of the sub-NPs. For example, there are two sample NPs below and each has a sub-NP. In first example, the sub-NP is *eğitimsiz insane* (uneducated person) where the adjective *eğitimsiz* (uneducated) modifies the nominative noun. However, in second example, the sub-NP is *yarış arabası* (race car) where the adjective *Yavaş* (slow) modifies the noun *arabası* (car). We cannot decide the correct head noun by using only morphological information. We need semantic information about this which can be obtained by a learning system or a WordNet for Turkish.

  Eğitimsiz insan davranışı(uneducated person behavior)

  Yavaş yarış arabası (slow race car)

- ***Collocation Recognizer:*** We used collocation analyzer before disambiguation as mentioned. However, in some situation, collocation analyzer caused mistakes. For example, in the following sentence, *rol oynadı* (he played role) is recognized as a collocation and the morphological parse of the word is "rol oyna+Verb+Pos+Past+A3sg" where stem is "play role". However, the adjective *büyük* (big) modifies the word *rol* (role). Because of collocation analyzer, we lose the token of *rol* and cannot find the NP *büyük rol* (big role).

  Maçı kazanmasında büyük rol oynadı.(He played big role in winning the match)

- ***Conjunction Rule Mistakes:*** Conjunction NPs are the most complex NPs because of being consisted of more than one NPs. When we ignored mistakes in conjunction NPs which comes from wrong disambiguation, there are still some problems. Context information, semantics of the words are crucial for determining them correctly. Therefore, a deeper analysis must be done and more complex rules must be written for this

kind NPs. Being lack of a good WordNet for Turkish is the bottleneck of this problem.

- ***Entity Name Mistakes:*** Entity names can be consisted of many names which are not morphologically connected. For example, in the following entity name, there is not any morphological connection between *Bilkent Üniversitesi* (Bilkent University) and *Bilgisayar Mühendisliği Bölümü* (Computer Engineering Department)

  Bilkent Üniversitesi Bilgisayar Mühendisliği Bölümü (Computer Engineering Department of Bilkent University)

  This kind NPs can be found by using semantic information or checking the proper morphological parse of the words. However, sometimes there can be no semantically relationship between words since the entity name can contain a sponsor's name like *Ziraat Türkiye Kupası* (Ziraat Turkey Cup) or city names can be appended to the institution names like *Ankara Cumhuriyet Başsavcılığı* (Ankara Chief Public Prosecutor's office). Having no semantic relation between words makes the process harder. In addition, finding sub-NPs correctly is still a problem when there is no semantic relationship. Furthermore, if the word is the first word of the sentence, then the morphological analyzer may not give a proper sense for the word. Therefore, checking proper sense may also not work in some cases.

The ratios of mistakes in D1 for each reason we have mentioned above is given in Table 11. Since the ratios are obtained by controlling the system's output manually, there can be some insignificant mistakes about ratios. As we see from the table, wrong disambiguation is the main problem of the study. In addition, we can see that conjunctive noun phrases are hardly to find because of their complex structure. A deeper analysis about them can decrease amount of the mistakes. Another important problem is wrong head selection. Semantic analysis of words

Table 11. Ratios of Mistakes in D1 According to Reasons We Observed

Reason of The Mistake	Ratio of Wrong NPs we found	Ratio of NPs not found in D1
Wrong Disambiguation	0.180	0.189
Sentence Structure	0.050	0.044
Wrong Head Selection	0.100	0.109
Collocation Recognizer	0.003	0.012
Conjunction Rule Mistakes	0.119	0.189
Entity Name Mistakes	0.044	0.083
Wrongly Disambiguated Single Main NPs	0.183	0.071
Miscellaneous	0.321	0.302

or using learning systems and giving links priorities according to the probabilistic result of the learning can handle this problem. Ratio of Collocation Recognizer mistakes is very little but in a larger datasets, the importance of this kind mistake will increase. The mistakes about entity names can be handled by a deeper analysis of an expert about Turkish Language. In addition, using a list name of generally known institution names will be helpful.

Table 12. Results of D2 According to NP Length

Length of Noun Phrase	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***Sum***
# of Main NPs in D2	706	353	111	24	7	2	2	2	0	1	1208
# of Total NPs in D2	725	487	136	30	9	4	3	2	1	1	1398
# of Main NPs we found	723	366	122	21	9	1	0	1	0	0	1243
# of Total NPs we found	734	507	143	26	9	1	1	1	0	0	1422
# of Main NPs found correctly	614	310	92	14	6	1	0	1	0	0	1038
# of Total NPs found correctly	628	443	109	17	7	1	0	1	0	0	1206

When we have tested our system with D2, then we get the results given in Table 12. As it is seen from the table, NPs are shorter in D2 and the system

success decrease as the length of NPs increases. The precision, recall and F-Measure value for D2 are given in Table 13.

Table 13. Exact Match Results for D2

Measure Type	F-Measure Value	Recall	Precision
Exact Match with all NPs	0.855	0,863	0.848
Exact Match with non-single NPs	0.849	0.859	0.840

As we analyze the mistakes for D2, the ratios of mistakes in D2 for each reason we have mentioned above are given in Table 14.

Table 14. Ratios of Mistakes in D2 According to Reasons We Observed

Reason of The Mistake	Ratio of Wrong NPs we found	Ratio of NPs not found in D1
Wrong Disambiguation	0.236	0,249
Sentence Structure	0.083	0.114
Wrong Head Selection	0,060	0.052
Collocation Recognizer	0.000	0.010
Conjunction Rule Mistakes	0.037	0.124
Wrongly Disambiguated Single Main NPs	0.241	0.155
Miscellaneous	0.343	0.295

From the table, we can see that wrong disambiguation is the main problem of this dataset. An increase in performance of wrong disambiguation will affect the performance highly. Second important problem is sentence structure. We can say that in shorter sentences, the mistakes which are caused by sentence structure increases. Conjunction noun phrases are still hard to define because of having complex structure. There is no mistake with entity names since the texts belong to stories which do not contain any entity names.

In some applications only main NPs can be enough. Therefore, we have checked correctness of main NPs we have found. If a main NP we have found is exactly same with a main NP in dataset, then we counted them as Full Match, which is same evaluation in 6^{th} rows in Table 7 and 11. However, if the NP we

found is a sub-NP in dataset, we counted them as Partial Match. According to this evaluation, the results are given in Table 15 for D1 & D2.

Table 15. F-Measure Values for D1 and D2 According to Correctness of Main NPs

Dataset	Full Match F-Measure Value	Full Match + Partial Match F-Measure Value
D1	0.868	0.895
D2	0.847	0.867

From the full match value, we can see how many of the main noun phrases we found are main noun phrase in datasets. The result of D1 is higher than the result of D2. The reason of this is that we have written the rules by considering D1. By adding new rules that will handle problems of D2 will increase the results. By the results in third row, we can see that how many of main noun phrases we found are correct noun phrases. The results are promising since they can be improved by adding new rules or using more semantic information. In addition, working with specialists on Turkish language can increase the performance of the system.

7.3 Effect of Morphological Disambiguation

We propose two new systems, one for morphological disambiguation and one for noun phrase chunking. As mentioned, our noun phrase chunking system uses morphological disambiguation. The mistakes in disambiguation can affect the noun phrase chunker. In order to see the performance of noun phrase chunker, we performed test with D3 which is a manually tagged noun phrase dataset and the morphological parses of words in D3 is also tagged by humans. So, we can assume that all given morphological parses are true. In testing with D3, we used these given morphological parses, instead of output of our system. The results according to noun phrase length are given in Table 16.

Table 16. Results of D3 According to NP Length

Length of Noun Phrase	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***≥10***	***Sum***
# of Main NPs in D3	451	297	127	47	35	13	8	7	0	5	990
# of Total NPs in D3	489	560	203	80	50	18	9	8	0	7	1424
# of Main NPs we found	457	303	126	49	33	15	8	3	0	5	999
# of Total NPs we found	493	545	197	78	46	20	9	4	1	7	1403
# of Main NPs found correctly	435	284	118	43	28	11	8	3	0	4	934
# of Total NPs found correctly	472	519	179	71	40	15	8	3	0	6	1313

From the table, we can see that best precision and recall results are obtained in single noun phrases. This is an expected result since most of the single noun phrases are found by the given morphological parses. The precision and recall decrease as the length of the noun phrase increases. In addition, we see that our rules are capable of correctly finding noun phrases which have a size of bigger than 10. The precision, recall and F-Measure values for all noun phrases are given in Table 17.

Table 17. Exact Match Results for D3

Measure Type	F-Measure	Recall	Precision
Exact Match with all NPs	0.929	0.922	0.936
Exact Match with non-single NPs	0.913	0.899	0.927

We can see that there is a huge increase in precision and recall results, when compared with tests with D1 and D2. When the single noun phrases are ignored, the performance of the system decreases as expected. However, the results are still better than the previous tests. Therefore, we can say that as the disambiguation system's performance increase, success in noun phrase chunking

will increase. The reasons of the mistakes of our system based on our observations and their ratios in all mistakes are given in Table 18.

As we can see from the table, the main problems of our rules are wrong head selection and conjunction noun phrases. These mistakes can be handled by our system with a deeper analysis by linguistics. Third main problem is about entity names. These can also be handled by providing a list of entity names and some linguistic features about them. The reason of not handling these mistakes was to avoid over-fitting for datasets.

Table 18. Ratios of Mistakes in D3 According to Reasons We Observed

Reason of The Mistake	Ratio of Wrong NPs we found	Ratio of NPs not found in D3
Sentence Structure	0.066	0.045
Wrong Head Selection	0.231	0.196
Conjunction Rule Mistakes	0.187	0.196
Entity Name Mistakes	0.132	0.223
Others	0.385	0.339

We calculated F-Measure for main NPs with D3 to compare results in Table 15. The F-Measure value of Full-Matches is 0.939 and F-Measure value of Full Matches and Partial Matches is 0.965. The high difference between results in Table 15 and results for D3 shows the importance of disambiguation.

The F-Measure value of Full & Partial Matches is higher than result in exact match comparison. Therefore, we can say that our system words better in finding main noun phrases. The reason for this is that finding sub-noun phrases requires more semantic information than main noun phrases while finding main noun phrases can be mostly handled with morphological information of words. The high F-Measure of Full & Partial matches shows that as long as disambiguation is performed well, the main noun phrases we found with only 98 rules are very reliable.

Chapter 8

Conclusion

In this thesis, we have developed an NP Chunker for Turkish. To the best of our knowledge, this is the first NP Chunker for Turkish. We have implemented a dependency parser which uses constraint based handcrafted rules for NP chunking. Our dependency parser has many distinctive features. It uses a scoring algorithm in constructing links for overcoming ambiguity problem. In addition, we have defined powerful constraining rules that allow rule designers to use semantic and morphological information together and handle very complex structures. By changing rules, links for shallow parsing of other phrases and full sentence parsing can be constructed, too.

Our dependency parser connects links between tokens rather than words. Therefore, morphological analysis is crucial for our NP Chunker in order to determine tokens. However, ambiguity is a significant problem that has to be handled for more reliable and easy computations and correct tokenization. In this thesis, we proposed a new morphological disambiguation technique which combines statistical information, hand-crafted constraining rules and transformation based learned rules. We constructed a dataset consisted of 25098 manually tagged word for testing and training the disambiguation system. This dataset is also significant for researchers who want to work on Turkish. Our tests with the dataset give promising results.

We have also constructed three small datasets for testing our NP Chunker. These datasets consisted of 3941 main noun phrases at total. We implemented a tool for easing tagging process. This tool can be used for increasing size of

dataset in future. Our NP Chunker gives promising results and the performance can increase by getting help from experts on Turkish language. It is also to be mentioned that the scope of this thesis includes NPs with complex structures like nested NP, entity names and conjunctive NPs. We cannot compare our results with previous studies because of using different datasets.

Enlarging training dataset of disambiguation process and testing dataset of NP chunker is left as future work. Our dependency parser structure can also be used for parsing other phrases, too. Therefore, shallow parsing of NPs, VPs, etc. is also left as future work.

BIBLIOGRAPHY

[1] Voutilainen, A., Heikkilä, J., Anttila, A. (1992). Constraint Grammar of English. University of Helsinki.

[2] Voutilainen, A. (1993). NPTool, a detector of English noun phrases. In Proceedings of the Workshop on Very Large Corpora (pp. 48-57). Association for Computational Linguistics.

[3] Abney, S. (1991). Parsing by Chunks. Principle-Based Parsing , pp. 257-278.

[4] Altan, Z. (2004). A Turkish Automatic Text Summarization System. In Proceedings of IASTED International Conference on Artificial Intelligience and Applications, (pp. 311-316). Innsbruck, Austria.

[5] Argamon, S., Dagan, I., & Krymolowski, Y. (1998). A memory-based approach to learning shallow. In Proceedings of COLING-ACL '98. Association for Computational Linguistics.

[6] Bourigault, D. (1992). Surface grammatical analysis for the extraction of terminological noun phrases. In Proceedings of the 14th International Conference on Computational Linguistics, (pp. 977-981).

[7] Brill, E. (1992). A simple-rule based part-of-speech tagger. In Proceedings of the Third Conference on Applied Natural Language Processing. Trento,Italy.

[8] Brill, E. (1994). Some advances in rule-based part of speech tagging. In Proceedings of the Twelfth National Conference on Artificial Intelligence. Seattle, Washington.

[9] Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* , 21(4):543-566.

[10] Cardie, C., & Pierce, D. (1998). Error-driven pruning of treebank grammars for base noun phrase identification. In Proceedings of COLING-ACL '98. Association for Computational Linguistics.

[11] Carrol, G., & Charniak, E. (1992). Two experiments on learning probabilistic dependency grammars from corpora, Technical Report TR-92. Department of Computer Science, Brown University.

[12] Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted. In Proceedings of the Second Conference on Applied Natural Language Processing. Austin, Texas.

[13] Collins, M., Hajic, J., Brill, E., Ramshaw, L., & Tillmann, C. (1999). A statistical parser for Czech. In Proceedings of the 37th Meeting of the Association for Computational Linguistics (ACL), (pp. 505–512).

[14] Cutting, D., Kupiec, J., Pealersen, J., Sibun, P. (1992). A practical part-of-speech tagger. In Proceedings of the Third Conference on Applied Natural Language Processing. Trento,Italy.

[15] Daelemans, W. (1996). MBT: A memory-based part of speech tagger-generator. In Proceedings of the Fourth Workshop on very Large Corpora, (pp. 14-27).

[16] Daelemans, W., Bosch, A. v., & Zavrel, J. (1999). Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11-14 .

[17] Eisner, J. M. (1996a). An empirical comparison of probability models for dependency grammar, Technical Report IRCS-96-11. Institute for Research in Cognitive Science, University of Pennsylvania.

[18] Eisner, J. M. (1996b). Three new probabilistic models for dependency parsing: An exploration. In Proceedings of the 16th International Conference on ComputationalLinguistics (COLING), (pp. 340–345).

[19] Ercan, G., & Cicekli, I. (2007). Using Lexical Chains for Keyword Extraction. *Information Processing & Management*, , Vol 43, No. 6, pp. 1705-1714.

[20] Eryiğit, G., & Oflazer, K. (2006). Statistical Dependency Parsing of Turkish. In Proceedings of EACL 2006 11th Conference of the European Chapter of the Association for Computational Linguistics. Trento,Italy.

[21] Ezeiza, N., I. Alegria, J.M. Arriola, R. Urizar and I. Aduriz. (1998). Combining Stochastic and Rule based Methods for Disambiguation in Agglutinative Languages. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, (pp. 379-384). Montreal, Quebec, Canada.

[22] F Karlsson, A Anttila. (1995). Constraint Grammar- A Language Independent System for Parsing Unrestricted Text. Mouton de Gruyter.

[23] Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and Control* , 8:304–337.

[24] Group, T. X. (1998). A Lexicalized Tree Adjoining Grammar for English. University of Pennsylvania. IRCS Tech Report 98-18.

[25] H. Sak, T.Güngör, and M. Saraçlar. (2007). Morphological disambiguation of Turkish text with perceptron algorithm. In Proceedings of CICLing 2007, (pp. 107-118).

[26] Hajiˇc, J. and B. Hladká. (1998). Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In Proceedings

of the 36th Annual Meeting of the Association for, (pp. 483-490). Montreal, Canada.

[27] Hajiˇc, J. (2000). Morphological Tagging: Data vs. Dictionaries. In Proceedings of the Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL). Seattle.

[28] Hakkani-Tür, D. Z.,Oflazer, K.,Tür, G. (2002). Statistical Morphological Disambiguation for Agglutinative Languages. *Computers and the Humanities,* 36(4) .

[29] Harper, M. P., & Helzerman, R. A. (1995). Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language* , 9: 187–234.

[30] Hays, D. G. (1964). Dependency theory: A formalism and some observations. *Language* , 40:511–525.

[31] Istek, O. (2006). A Link Grammar For Turkish. MS Thesis . Bilkent University.

[32] Istek, O., & Cicekli, I. (2007). A Link Grammar for an Agglutinative Language. in Proceedings of Recent Advances in Natural Language Processing (RANLP 2007), (pp. 285-290). Borovets, Bulgaria.

[33] J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. (2002). Introduction to Special Issue on Machine Learning Approaches to Shallow Parsing. *Journal of Machine Learning Research* , 551-558.

[34] Jarvinen, T., & Tapanainen, P. (1998). Towards an implementable dependency grammar. In Proceedings of the Workshop on Processing of Dependency-Based Grammars , (pp. 1-10).

[35] Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In Proceedings of the 13th International Conference on Computational Linguistics, (pp. 168–173).

[36] Korkmaz, T. (1996). Turkish Text Generation with Systemic-Functional Grammar . MS Thesis . Ankara,Turkey: Bilkent University.

[37] Kuang-hua Chen, Hsin-Hsi Chen. (1994). Extracting noun phrases from large-scale texts: a hybrid approach and its automatic evaluation. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, (pp. 234-241).

[38] Kudo, T., & Matsumoto, Y. (2002). Japanese dependency analysis using cascaded chunking. In Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL), (pp. 63-69).

[39] Kudo, T., & Matsumoto, Y. (2000). Japanese dependency structure analysis based on support vector machines. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC), (pp. 18–25).

[40] Kupiec, J. (1992). Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language* , 6:225-242.

[41] Kutlu, M., Cigir, C., & Cicekli, I. (2010). Generic Text Summarization for Turkish. *The Computer Journal* .

[42] L.A. Ramshaw and M.P. Marcus. (1995). Text chunking using transformation-based learning. In Proceedings of the Third Workshop on Very Large Corpora. ACL.

[43] Leech G.,Garside R., Bryan M. (1994). CLAWS4: The tagging of the British National Corpus. In Proceedings of COLING'94, (pp. 622-628).

[44] Levinger, M., U. Oman and A. Itai. (1995). Learning Morpho-lexical Probabilities from an Untagged Corpus with an Application to Hebrew. *Computational Linguistics* 21(3) , 383-404.

[45] Maruyama, H. (1990). Structural disambiguation with constraint propagation. In Proceedings of the 28th Meeting of the Association for Computational Linguistics, (pp. 31–38). Pittsburgh.

[46] Megyesi, B. (1999). Improving Brill's POS Tagger for an Agglutinative Language. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, (pp. 275-284). College Park, Maryland, USA.

[47] Menzel, W., & Schroder, I. (1998). Decision procedures for dependency parsing using graded constraints. In Proceedings of the Workshop on Processing of Dependency-Based Grammars, (pp. 78-87).

[48] Munoz, M., Punyakanok, V., Roth, D., & Zimak, D. (1999). A learning approach to shallow parsing. In Proceedings of EMNLP-WVLC'99. Association for Computational Linguistics.

[49] Ngai, G., & Yarowsky, D. (2000). Rule writing or annotation: cost-efficient resource usage for base noun phrase chunking. In Proceedings of ACL'02, (pp. 117-125).

[50] Nivre, J. (2005). Dependency grammar and dependency parsing. Vaxjö University: School of Mathematics and Systems Engineering: Technical Report MSI report 05133.

[51] Oflazer K., Tür G. (1997). Morphological Disambiguation by Voting Constraints. In Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics, (pp. 222-229).

[52] Oflazer, K., Kuruöz I. . (1994). Tagging and morphological disambiguation of Turkish text. In Proceedings of the 4th Applied Natural Language Processing Conference, (pp. 144-149). ACL.

[53] Oflazer, K., Tür, G. (1996). Combining Hand-crafted Rules and Unsupervised Learning in Constraint-based Morphological Disambiguation. In Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing. Philadelphia, USA.

[54] Oflazer, K., Tür, G. (1997). Morphological Disambiguation by Voting Constraints. In Proceedings of ACL/EACL, The 35th Annual Meeting of the Association for Computational Linguistics. Madrid, Spain.

[55] Oflazer, K. (2003). Dependency parsing with an extended finite-state approach. *Computational Linguistics* , 29: 515–544.

[56] Ozdemir, B., & Cicekli, I. (2009). Turkish Keyphrase Extraction Using Multi-Criterion Ranking. In Proceedings of the 24rd International Symposium on Computer and Information Sciences (ISCIS 2009). Northern Cyprus.

[57] Pattabhi, R. K., Vijay, S. R., Vijayakrishna, R., & Sobha, L. (2007). A Text Chunker and Hybrid POS Tagger for Indian Languages. In Proceedings of IJCAI-07 workshop on Shallow Parsing for South Asian Languages.

[58] Sang, E. F. (2000). Noun phrase recognition by system combination. In Proceedings of the Language Technology Joint Conference ANLP-NAACL2000, (pp. 50-55). Seattle, Washington, USA.

[59] Sang, E. F., & Buchholz, S. (2000). Introduction to the CoNLL-2000 Shared Task: Chunking. In Proceedings of CoNLL-2000 and LLL-2000, (pp. 127-132). Lisbon, Portugal.

[60] Sastry, G. R., Chaudhuri, S., & Reddy, P. N. (2007). An HMM based Part-Of-Speech tagger and statistical chunker for 3 Indian languages. In Proceedings of IJCAI-07 workshop on Shallow Parsing for South Asian Languages.

[61] Schroder, I. (2002). Natural Language Parsing with Graded Constraints. PhD thesis . Hamburg University.

[62] Skut, W., & Brants, T. (1998). Chunk tagger – statistical recognition of noun phrases. In Proceedings of the ESSLLI Workshop on Automated Acquisition of Syntax and Parsing. Saarbrücken.

[63] Sleator, D., & Temperley, D. (1993). Parsing English with a link grammar. In Proceedings of International Workshop on Parsing Technologies (IWPT), (pp. 277–292).

[64] Sleator, D., & Temperley, D. (1991). Parsing English with a link grammar, Technical Report CMU-CS-91-196. Carnegie Mellon University, Computer Science.

[65] Sobha, L., & Vijay, S. R. (2006). Noun Phrase Chunking in Tamil. In Proceedings of the MSPIL-06, (pp. 194-198). Bombay.

[66] T. Daybelge, I. Cicekli. (2007). A Rule-Based Morphological Disambiguator for Turkish. In Proceedings of Recent Advances in Natural Language Processing, (pp. 145-149). Borovets, Bulgaria.

[67] Tapanainen P., Voutilainen A. (1994). Tagging Accurately - Don't guess if you know. In Proceedings of ANLP'94, (pp. 47-52).

[68] Tjong Kim Sang, E. F., & Veenstra, J. (1999). Representing Text Chunks. In Proceedings of EACL '99. Association for Computational Linguistics.

[69] Veenstra, J. (1998). Fast NP chunking using memory-based learning techniques. In Proceedings of the Eighth Belgian-Dutch Conference on Machine Learning.

[70] Voutilainen, A. (1995). A syntax-based part-of-speech analyzer. In Proceedings of the Seventh Conference of the European Chapter of the Association of Computational Linguistics. Dublin, Ireland.

[71] Voutilainen, Atro and Pasi Tapanainen. (1993). Ambiguity resolution in a reduetionistic parser. In Proceedings of EACL'93. Utrecht, Holland.

[72] Wang, W., & Harper, M. P. (2004). A statistical constraint dependency grammar (CDG) parser. In Proceedings of the Workshop in Incremental Parsing: Bringing Engineering andCognition Together (ACL), (pp. 42–29).

[73] Yüret, D., Türe, F. (2006). Learning Morphological Disambiguation Rules for Turkish. In Proceedings of HLT-NAACL .

APPENDIX A

Links and Their Explanations

Link Name	**Explanation**
A	The source is an adjective. i.e. *Küçük adam*(Small man)
Dn	The source is a number. i.e. *3 araba*(3 cars)
Time	The noun phrase is related to time. i.e. *Bu sabah* (This morning)
NumberPossesive	The source is a number and the target is a noun with third person possessive marker. i.e.*1990 senesi* (year of 1990)
ofProp	The target is a proper noun and target is a member of a group. i.e. Lig *Şampiyonlarından Beşiktaş* (Beşiktaş, from the league champions.)
Bsiz	The source is a nominative noun and target has a possessive marker rather than Pnon. i.e. *ders notu*(Lecture note)
Bli	The source is a noun or a pronoun with a genitive case marker and target is a noun with a third person possessive marker. *Adamın arabası*(The man's car)
Prop	Source and target has a proper noun sense but the disambiguator found them as noun. The link is used to handle Entity Names. Its priority is smaller than the link *Proper*.

AA	The source has ablative case marker and target is adjective. i.e. *konudan bağımsız*(free from the subject)
MadeOf	The source shows what is made of target. i.e. *Altın saat* (golden watch)
Title	The target is a proper noun and the source is its job or caption. i.e. *Prof. Ali*
toQ	The target is a quotation mark and source is anything.
fromQ	The source is a quotation mark and target is anything.
Quotation-Possessive	The quotation mark is connected to a noun with a third person possessive marker. i.e. *"dur" uyarısı*("stop" warning)
Dash	The source or target is a dash. i.e. *sarı-kırmızı*(yellow-red)
Connector	The source or the target is a conjunction word. i.e. *kitap ve kalem*(book and pencil)
Proper	The source and the target is a proper noun.
NA	The source is a noun that acts as an adjective and the target is a noun or a proper noun.
ANP	The source is a noun and a part of an NP which acts as adjective. i.e. *insanlık dışı davranış*(inhumane behaviour)
NominativeModifier	Same with link *Bsiz*. The reliability of this link is less.

QuotationAdj	Source is a quotation mark and the target is a noun. The previous word of quotation mark should be an adjective. i.e. *"iyi" adam*("good" man)
AdverbConj	Connects conjunction to an adjective where the adjective modified with an adverb. For example, the link is constructed between "da" and "güzel" in the following NP: *Çok da güzel araba*(very good car)
numberConnector	It is used for connecting numbers that define a range. For example, the link is constructed between "30" and "ile" in the following NP : *30 ile 40 kadar araba*(nearly 30 or 40 cars)
NN	It is used for connecting numbers that are defining a unique number. For example, The link is constructed between "Üç" and "bin" in the following NP: *3 bin adam*(three thousand man)
withProp	It is used for connecting a special type of noun phrases. The source is nominative proper noun and target is a one of the words with certain semantic. For example, the link is constructed in the following word phrases : *Hint asıllı*(Indian origined), *John isimli*(with name of John)
postNumber	It is used for connecting number and conjunctions in a certain conjunction type. For example, *400 veya üstünde adam*(400 or more men)
Distance	It is used for connecting distance units. For example, the link is constructed between "metre" and

	"santim" in the following word phrases : *3 metre 10 santim*(3 meters and 10 centimeters)
AToA	Connects two adjectives where both modify same word and there is a conjunction word between them. i.e. *iyi ve güzel araba*(good and beautiful car)

APPENDIX B

List of Rules

In implementation of the rules, NumberPossesive link is named as YearN, AA is named as From, madeOf is named as Takisiz, toQ is named as toTirnak, fromQ is named as fromTirnak, Quotation-Possesive is named as QuatationBsiz, Dash is named as TireLink, NominativeModifier is named as Belirtisiz, Distance is named as uzunluk and AToA is named as SifatToSifat, by the rule designer.

```
<!--  küçük adam - small man  -->
<NounPhraseRule>
        <ID>1</ID>
        <sourceType>POS</sourceType>
        <source>Adj</source>
        <targetType>LIST</targetType>
        <target>NounPropList.txt</target>
        <linkType>A</linkType>
        <priority>100</priority>
        <constraint>!T:hasBeginLink(Takisiz)</constraint>
        <constraint>!S:hasEndLink(Proper)</constraint>
        <constraint>!T:hasBeginLink(NA)</constraint>
        <constraint>[![T:isPossible(Verb)]^[S:isPossible(Adverb)]]</constraint>
        <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
        <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
        <constraint>[![S->T:numberOfWords(1)]^[S-
        >T:containsPOS(Punc)]]</constraint>
        <constraint>!S:inList(Rule1ExceptList.txt)</constraint>
        <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
</NounPhraseRule>

<!—Hızlı yarış arabası - fast race car - connects adj to the noun with Bsiz Link
        -->
<NounPhraseRule>
        <ID>2</ID>
        <sourceType>POS</sourceType>
        <source>Adj</source>
        <targetType>POS</targetType>
        <target>Noun</target>
        <linkType>A</linkType>
        <priority>120</priority>
        <constraint>T:hasEndLink(Bsiz)</constraint>
        <constraint>!S->T:containsUnconnectedToken()</constraint>
        <constraint>!S->T:containsLink(About)</constraint>
        <constraint>!S->T:containsMorpheme(+Gen)</constraint>
        <constraint>!S->T:containsMorpheme(Adj+Rel)</constraint>
        <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
        <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
        <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
        <constraint>!S:inList(Rule2ExceptList.txt)</constraint>
        <constraint>[![S->T:numberOfWords(1)]^[S-
        >T:containsPOS(Punc)]]</constraint>
 </NounPhraseRule>

<!-- 5  bir adam - one man -->
<NounPhraseRule>
    <ID>3</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
```

```
    <targetType>LIST</targetType>
    <target>NounPropPronList.txt</target>
    <linkType>Dn</linkType>
    <priority>100</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
    <constraint>!T:turnsInto(Adj)</constraint><!-- see rule 64-->
    <constraint>!S->T:containsUnconnectedToken()</constraint>
</NounPhraseRule>

<!--  Güzei ve iyi car - beautiful and good car - connects sifat to sifat  -->
<NounPhraseRule>
    <ID>4</ID>
    <sourceType>POS</sourceType>
    <source>Adj</source>
    <targetType>POS</targetType>
    <target>Adj</target>
    <linkType>SifatToSifat</linkType>
    <priority>95</priority>
    <constraint>[![S->T:numberOfWords(1)]^[S+1:inList(ConnectorList.txt)]]
    </constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>!T:derivedFrom(Verb)</constraint>
</NounPhraseRule>

<!-- bu akşam - This night -->
<NounPhraseRule>
    <ID>5</ID>
    <sourceType>POS</sourceType>
    <source>Adj</source>
    <targetType>LIST</targetType>
    <target>ZamanList.txt</target>
    <linkType>Time</linkType>
    <priority>110</priority>
    <constraint>[![[S->T:containsPOS(Noun)]|
    [S->T:containsPOS(Prop)]]]</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
 </NounPhraseRule>

  <!--     bir akşam - a night       -->
  <NounPhraseRule>
    <ID>6</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>ZamanList.txt</target>
    <linkType>Time</linkType>
    <priority>110</priority>
    <constraint>[![[S->T:containsPOS(Noun)]|
    [S->T:containsPOS(Prop)]]]</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
</NounPhraseRule>

<!--    4 yüz - 4 hundred -->
<NounPhraseRule>
    <ID>7</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>POS</targetType>
    <target>Num</target>
    <linkType>NN</linkType>
    <priority>200</priority>
```

```
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  1990 senesi - year of 1990 -->
<NounPhraseRule>
    <ID>8</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>SeneList.txt</target>
    <linkType>YearN</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- Liderlerinden Richard - Richard, one of the leaders, -->
<NounPhraseRule>
    <ID>9</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+Abl,+A3pl</source>
    <targetType>POS</targetType>
    <target>Prop</target>
    <linkType>ofProp</linkType>
    <priority>80</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!S:contains(Pnon)</constraint>
</NounPhraseRule>

<!-- Okul arkadaşı - school friend -->
<NounPhraseRule>
    <ID>10</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropNomList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Bsiz</linkType>
    <priority>100</priority>
    <constraint>!T:contains(Pnon)</constraint>
    <constraint>S~T:agree(Plural)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!T:hasEndLink(Belirtisiz)</constraint>
    <constraint>[![S+1->T-1:containsPOS(Noun)]|
    [S+1->T-1:containsPOS(Prop)]]^[!S->T:numberOfWords(0)]</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S->T:containsList(Rule10BlockerList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T:inList(BirlesikIsimList.txt)</constraint>

  </NounPhraseRule>

<!-- For connecting third person possesive nouns to previous nomivative words if
still they are not connected-->
<NounPhraseRule>
    <ID>11</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropNomList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Belirtisiz</linkType>
    <priority>40</priority>
    <constraint>!T:contains(Pnon)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsPOS(Punc)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
```

```
    <constraint>!S->T:containsList(Rule10BlockerList.txt)</constraint>
    <constraint>!T:hasEndLink(Bsiz)</constraint>
    <constraint>!T:hasEndLink(Belirtisiz)</constraint>
    <constraint>!T:hasEndLink(Bli)</constraint>
    <constraint>!T:inList(BirlesikIsimList.txt)</constraint>
</NounPhraseRule>

<!--  Connects person name to a proper noun found as a noun -->
<NounPhraseRule>
    <ID>12</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+A3sg,+Pnon,+Nom</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Proper</linkType>
    <priority>120</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>!S:endsWith(oğlu)</constraint>
    <constraint>S:inList(person.first_name.txt)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
</NounPhraseRule>

<!-- Adamın işi - the job of the man, the man's job -->
<NounPhraseRule>
    <ID>13</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropGenList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Bli</linkType>
    <priority>90</priority>
    <constraint>T:contains(+P3)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T->E:hasP3SGNounDerivedFromVerb()</constraint>
    <constraint>!T:inList(SifatIsimTamlamari.txt)</constraint>
</NounPhraseRule>

<!-- For connecting third person possesive nouns to previous genitive words if
still they are not connected -->
  <NounPhraseRule>
    <ID>14</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropGenList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Bsiz</linkType>
    <priority>40</priority>
    <constraint>T:contains(+P3)</constraint>
    <constraint>S~T:agree(Plural)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsPOS(Punc)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T:hasEndLink(Bsiz)</constraint>
    <constraint>!T:hasEndLink(Bli)</constraint>
    <constraint>!T:inList(BirlesikIsimList.txt)</constraint>

  </NounPhraseRule>

<!-- en ucuz - the cheapest -->
<NounPhraseRule>
```

```
    <ID>15</ID>
    <sourceType>POSFI</sourceType>
    <source>Adverb+AdjMdfy</source>
    <targetType>POS</targetType>
    <target>Adj</target>
    <linkType>Eap</linkType>
    <priority>110</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsPOS(Punc)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
</NounPhraseRule>

<!--  Connects nominative proper noun to a proper noun. ie. Necip Fazıl -->
<NounPhraseRule>
    <ID>16</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+A3sg,+Pnon,+Nom</source>
    <targetType>POSPI</targetType>
    <target>Prop,+A3sg,+Pnon</target>
    <linkType>Proper</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
    <constraint>!S:endsWith(oğlu)</constraint>
    <constraint>!T:inList(rule16ExceptList.txt)</constraint>
    <constraint>!S:inList(rule16ExceptList.txt)</constraint>
    <constraint>!T:derivedFrom(Verb,exceptList.txt)</constraint>
</NounPhraseRule>

<!--  connects nominative proper noun to a noun word which has an proper noun
sense which is eliminated in disambiguation process -->
<NounPhraseRule>
    <ID>17</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+A3sg,+Pnon,+Nom</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>Prop</linkType>
    <priority>70</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>!S:endsWith(oğlu)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
</NounPhraseRule>

<!--  Connects noun which can be a proper noun to a proper noun  -->
<NounPhraseRule>
    <ID>18</ID>
    <sourceType>POS</sourceType>
    <source>Noun</source>
    <targetType>POS</targetType>
    <target>Prop</target>
    <linkType>Prop</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isPossible(Prop)</constraint>
    <constraint>!S:endsWith(oğlu)</constraint>
    <constraint>![[!S:inList(person.first_name.txt)]^
    [T:inList(person.first_name.txt)]]</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
```

```
</NounPhraseRule>

<!--  Connects noun which is also a possible proper noun to a noun which is also
a possible proper noun  -->
<NounPhraseRule>
    <ID>19</ID>
    <sourceType>POS</sourceType>
    <source>Noun</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>Prop</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isPossible(Prop)</constraint>
    <constraint>!S:endsWith(oğlu)</constraint>
    <constraint>S:contains(Nom)</constraint>  <!--can be deleted -->
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
</NounPhraseRule>

<!--  kendi kalemi - his own pen -->
<NounPhraseRule>
    <ID>20</ID>
    <sourceType>SSF</sourceType>
    <source>kendi,+Nom</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>Bsiz</linkType>
    <priority>100</priority>
    <constraint>!T:contains(Pnon)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
</NounPhraseRule>

<!--  bütün bunlar - all of these  -->
<NounPhraseRule>
    <ID>21</ID>
    <sourceType>LIST</sourceType>
    <source>PronModifierList.txt</source>
    <targetType>LIST</targetType>
    <target>ModifyablePronList.txt</target>
    <linkType>A</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  konuyla ilgili - about the subject -->
<NounPhraseRule>
    <ID>22</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropInsList.txt</source>
    <targetType>LIST</targetType>
    <target>InsSifatList.txt</target>
    <linkType>About</linkType>
    <priority>80</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
</NounPhraseRule>

<!--  konuya ilişkin - related to the subject -->
<NounPhraseRule>
    <ID>23</ID>
    <sourceType>LIST</sourceType>
```

```
    <source>NounPropPronDatList.txt</source>
    <targetType>LIST</targetType>
    <target>DatSifatList.txt</target>
    <linkType>About</linkType>
    <priority>80</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
</NounPhraseRule>

<!--  konudan bağımsız - free from the subject -->
<NounPhraseRule>
    <ID>24</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropPronAblList.txt</source>
    <targetType>LIST</targetType>
    <target>AblSifatList.txt</target>
    <linkType>From</linkType>
    <priority>80</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
</NounPhraseRule>

<!--  saat 5'te - at 5 o'clock -->
<NounPhraseRule>
    <ID>25</ID>
    <sourceType>SF</sourceType>
    <source>saat</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>Time</linkType>
    <priority>120</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:derivedFrom(Num)</constraint>
  </NounPhraseRule>

<!--  saat 3 - 3 o'clock -->
<NounPhraseRule>
    <ID>26</ID>
    <sourceType>SF</sourceType>
    <source>saat</source>
    <targetType>POS</targetType>
    <target>Num</target>
    <linkType>Time</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  24 ekim perşembe günü, 24 October Thursday - Connects October to Thursday
-->
<NounPhraseRule>
    <ID>27</ID>
    <sourceType>LIST</sourceType>
    <source>AyList.txt</source>
    <targetType>LIST</targetType>
    <target>GunList.txt</target>
    <linkType>Takisiz</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  24 ekim 2000 = 24 october 2000 - Connects October to 2000 -->
<NounPhraseRule>
```

```
    <ID>28</ID>
    <sourceType>LIST</sourceType>
    <source>AyList.txt</source>
    <targetType>POSFI</targetType>
    <target>Num+Card</target>
    <linkType>Time</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- dün gece - the night of yesterday   -->
<NounPhraseRule>
    <ID>29</ID>
    <sourceType>LIST</sourceType>
    <source>ZamanModifierList.txt</source>
    <targetType>LIST</targetType>
    <target>ZamanList.txt</target>
    <linkType>Time</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  Rektör Atalar - Rector Atalar   -->
  <NounPhraseRule>
    <ID>30</ID>
    <sourceType>LIST</sourceType>
    <source>meslek.txt</source>
    <targetType>POS</targetType>
    <target>Prop</target>
    <linkType>Title</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- yaklaşık 200 - nearly 200   -->
<NounPhraseRule>
    <ID>31</ID>
    <sourceType>LIST</sourceType>
    <source>preNumberModifierList.txt</source>
    <targetType>POS</targetType>
    <target>Num</target>
    <linkType>NN</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  altın saat - golden watch  -->
<NounPhraseRule>
    <ID>32</ID>
    <sourceType>LIST</sourceType>
    <source>takisizList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropPnonList.txt</target>
    <linkType>Takisiz</linkType>
    <priority>100</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>

</NounPhraseRule>

<!-- 30 civarında - around 30 -->
<NounPhraseRule>
    <ID>33</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>postNumberModifierList.txt</target>
```

```
    <linkType>postNumber</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- 30'dan fazla - more than 30 -->
<NounPhraseRule>
    <ID>34</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+Abl</source>
    <targetType>LIST</targetType>
    <target>postNumberNounAblModifierList.txt</target>
    <linkType>postNumber</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:derivedFrom(Num)</constraint>
</NounPhraseRule>

<!-- 30'a yakın - near to 30 -->
<NounPhraseRule>
    <ID>35</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+Dat</source>
    <targetType>LIST</targetType>
    <target>postNumberNounDatModifierList.txt</target>
    <linkType>postNumber</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:derivedFrom(Num)</constraint>
</NounPhraseRule>

<!-- 30'a yakın adam - nearly 30 men - Connects yakın to adam -->
<NounPhraseRule>
    <ID>36</ID>
    <sourceType>ANY</sourceType>
    <source>ANY</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>A</linkType>
    <priority>100</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>S:hasEndLink(postNumber)</constraint>
</NounPhraseRule>

<!-- hint asıllı - Indian Origined -->
<NounPhraseRule>
    <ID>37</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+Nom</source>
    <targetType>LIST</targetType>
    <target>postNomPropModifierList.txt</target>
    <linkType>withProp</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- çok da güzel araba - very good car - Connects da to güzel -->
<NounPhraseRule>
    <ID>38</ID>
    <sourceType>LIST</sourceType>
    <source>dedaList.txt</source>
    <targetType>POS</targetType>
    <target>Adj</target>
    <linkType>adverbConj</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
```

```
    <constraint>S-1:contains(AdjMdfy)</constraint>
</NounPhraseRule>

<!-- Connects " to any word -->
<NounPhraseRule>
    <ID>39</ID>
    <sourceType>LIST</sourceType>
    <source>tirnakList.txt</source>
    <targetType>ANY</targetType>
    <target>ANY</target>
    <linkType>fromTirnak</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T->E:hasBeginLink(toTirnak)</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T->E:hasBeginLink(toTirnak)</constraint>
</NounPhraseRule>

<!-- Connects any word to " -->
<NounPhraseRule>
    <ID>40</ID>
    <sourceType>ANY</sourceType>
    <source>ANY</source>
    <targetType>LIST</targetType>
    <target>tirnakList.txt</target>
    <linkType>toTirnak</linkType>
    <priority>125</priority>
    <constraint>![[S:contains(Verb)]^[!S-1:inList(tirnakList.txt)]]</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!T:hasBeginLink(fromTirnak)</constraint>
</NounPhraseRule>

<!--  "Dur" ihtarı - "Stop" warning  -->
<NounPhraseRule>
    <ID>41</ID>
    <sourceType>LIST</sourceType>
    <source>tirnakList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>QoutationBsiz</linkType>
    <priority>80</priority>
    <constraint>!T:hasEndLink(toTirnak)</constraint>
    <constraint>!T:contains(+Pnon)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!T:inList(BirlesikIsimList.txt)</constraint>
</NounPhraseRule>

<!-- sözkonusu adam - aforementioned man -->
<NounPhraseRule>
    <ID>42</ID>
    <sourceType>LIST</sourceType>
    <source>IsimSifatList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>NA</linkType>
    <priority>100</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
</NounPhraseRule>

<!--  ve adam - and man -->
```

```
<NounPhraseRule>
    <ID>43</ID>
    <sourceType>LIST</sourceType>
    <source>veVeyaVsList.txt</source>
    <targetType>ANY</targetType>
    <target>ANY</target>
    <linkType>connector</linkType>
    <priority>100</priority>
    <constraint>[[T:isNPBeginning()]|[T:inList(NounPropPronList.txt)]]
    </constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!T:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>T:isEndToken()</constraint>
    <constraint>S~T:connectorAgreement()</constraint>
</NounPhraseRule>

<!-- çocuk ve - kid and  -->
<NounPhraseRule>
    <ID>44</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropPronList.txt</source>
    <targetType>LIST</targetType>
    <target>veVeyaVsList.txt</target>
    <linkType>connector</linkType>
    <priority>105</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:hasBeginLink(connector)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>S~T:hasSameInflection()</constraint>
</NounPhraseRule>

<!-- Connects comma to any word  -->
<NounPhraseRule>
    <ID>45</ID>
    <sourceType>SF</sourceType>
    <source>,</source>
    <targetType>ANY</targetType>
    <target>ANY</target>
    <linkType>CommaConnector</linkType>
    <priority>105</priority>
    <constraint>[[T:isNPBeginning()]|[T:inList(NounPropPronList.txt)]|
    [S:isPossible(Noun)]]</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isConnectedToLink(connector)</constraint>
</NounPhraseRule>

<!--  Connects any noun to a comma  -->
<NounPhraseRule>
    <ID>46</ID>
    <sourceType>ANY</sourceType>
    <source>ANY</source>
    <targetType>SF</targetType>
    <target>,</target>
    <linkType>CommaConnector</linkType>
    <priority>105</priority>
    <constraint>[[S:inList(NounPropPronList.txt)]|[S:isPossible(Noun)]]
    </constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:hasBeginLink(CommaConnector)</constraint>
    <constraint>S~T:hasSameInflectionAtEnd()</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
</NounPhraseRule>

<!--  Connects dash to any word -->
<NounPhraseRule>
    <ID>47</ID>
    <sourceType>SF</sourceType>
    <source>-</source>
    <targetType>ANY</targetType>
    <target>ANY</target>
    <linkType>TireLink</linkType>
```

```
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!T:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>!T:derivedFrom(Conj)</constraint>
</NounPhraseRule>

<!-- Connects any word to a dash  -->
<NounPhraseRule>
    <ID>48</ID>
    <sourceType>ANY</sourceType>
    <source>ANY</source>
    <targetType>SF</targetType>
    <target>-</target>
    <linkType>TireLink</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!S:derivedFrom(Conj)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
</NounPhraseRule>

<!-- insanlık dışı davranış - inhuman behaviour  -->
<NounPhraseRule>
    <ID>49</ID>
    <sourceType>LIST</sourceType>
    <source>SifatIsimTamlamari.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>ANP</linkType>
    <priority>105</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
</NounPhraseRule>

<!-- 3 metre 30 santim - 3 meters and 30 cm - Connects meters to cm  -->
<NounPhraseRule>
    <ID>50</ID>
    <sourceType>LIST</sourceType>
    <source>uzunlukList.txt</source>
    <targetType>LIST</targetType>
    <target>uzunlukList.txt</target>
    <linkType>uzunluk</linkType>
    <priority>150</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S->T:containsPOS(Punc)</constraint>
</NounPhraseRule>

<!-- Connects special words to modifies texts  -->
<NounPhraseRule>
    <ID>51</ID>
    <sourceType>LIST</sourceType>
    <source>DogalDiller.txt</source>
    <targetType>LIST</targetType>
    <target>YaziCinsIsimleri.txt</target>
    <linkType>A</linkType>
    <priority>80</priority>
</NounPhraseRule>

<!-- Çarşamba günü saat 02:00'de - On wendesday at 2 o'clock - Connects günü to
02:00'de  -->
<NounPhraseRule>
    <ID>52</ID>
    <sourceType>SSF</sourceType>
    <source>gün,+P3</source>
    <targetType>POS</targetType>
    <target>Noun</target>
```

```
    <linkType>Time</linkType>
    <priority>100</priority>
    <constraint>S+1:contains(saat)</constraint>
    <constraint>T:derivedFrom(Num)</constraint>
</NounPhraseRule>

<!--   3. Takım - 3. Team -->
<NounPhraseRule>
  <ID>53</ID>
  <sourceType>POSFI</sourceType>
  <source>Num+Ord</source>
  <targetType>LIST</targetType>
  <target>NounPropList.txt</target>
  <linkType>Dn</linkType>
  <priority>150</priority>
  <constraint>S->T:numberOfWords(0)</constraint>
  <constraint>!T:derivedFrom(Verb)</constraint>
  <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
  <constraint>!T:inList(IsimSifatList.txt)</constraint>
</NounPhraseRule>

<!-- bir yarış arabası - a race car - Connects bir to arabası -->
<NounPhraseRule>
    <ID>54</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Dn</linkType>
    <priority>120</priority>
    <constraint>T:hasEndLink(Bsiz)</constraint>
    <constraint>!S->T:containsLink(About)</constraint>
    <constraint>!S->T:containsMorpheme(+Gen)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!S->T:containsMorpheme(Adj+Rel)</constraint>
</NounPhraseRule>

<!--  24 ekim - 24 October  -->
<NounPhraseRule>
    <ID>55</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>AyList.txt</target>
    <linkType>Time</linkType>
    <priority>200</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--  Adj + nouns actıng as adjectives -->
<NounPhraseRule>
    <ID>56</ID>
    <sourceType>POS</sourceType>
    <source>Adj</source>
    <targetType>LIST</targetType>
    <target>IsimSifatList.txt</target>
    <linkType>A</linkType>
    <priority>80</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
</NounPhraseRule>
```

```
<!-- amerikan savaş uçağı - american war plane -->
<NounPhraseRule>
    <ID>57</ID>
    <sourceType>LIST</sourceType>
    <source>IsimSifatList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>A</linkType>
    <priority>120</priority>
    <constraint>T:hasEndLink(Bsiz)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsMorpheme(+Gen)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
</NounPhraseRule>

<!--   yeni yetkili - The new person in charge -->
<NounPhraseRule>
    <ID>58</ID>
    <sourceType>POS</sourceType>
    <source>Adj</source>
    <targetType>LIST</targetType>
    <target>IsimOlmayanHeadNounList.txt</target>
    <linkType>A</linkType>
    <priority>100</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!T:inList(IsimSifatList.txt)</constraint>
</NounPhraseRule>

<!--   bir yetkili - a person in charge     -->
<NounPhraseRule>
    <ID>59</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>IsimOlmayanHeadNounList.txt</target>
    <linkType>A</linkType>
    <priority>100</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!T:inList(IsimSifatList.txt)</constraint>
</NounPhraseRule>

<!--   Fatih Bey - Mr. Fatih  -->
<NounPhraseRule>
    <ID>60</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+Nom</source>
    <targetType>LIST</targetType>
    <target>PropNamePostUnvanList.txt</target>
    <linkType>Proper</linkType>
    <priority>120</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
</NounPhraseRule>

<!-- Avukat Adem - Advocate Adem  -->
```

```
<NounPhraseRule>
    <ID>61</ID>
    <sourceType>LIST</sourceType>
    <source>meslek.txt</source>
    <targetType>POSPI</targetType>
    <target>Prop,+Nom</target>
    <linkType>Title</linkType>
    <priority>105</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- daha hassas - more sensitive  -->
<NounPhraseRule>
    <ID>62</ID>
    <sourceType>POSFI</sourceType>
    <source>Adverb+AdjMdfy</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>A</linkType>
    <priority>80</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Adj)</constraint>
    <constraint>!T:isLastWord()</constraint>
    <constraint>!T:derivedFrom(Verb)</constraint>
</NounPhraseRule>

<!--  bir kitap - a book - Connects noun to countable nouns  -->
<NounPhraseRule>
    <ID>63</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>countableNounList.txt</target>
    <linkType>Dn</linkType>
    <priority>200</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--   bir adam -  a man      -->
<NounPhraseRule>
    <ID>64</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Dn</linkType>
    <priority>90</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
    <constraint>T:turnsInto(Adj)</constraint>
</NounPhraseRule>

<!--   Salı günü - Tuesday    -->
<NounPhraseRule>
    <ID>65</ID>
    <sourceType>LIST</sourceType>
    <source>GunList.txt</source>
    <targetType>SSF</targetType>
    <target>gün,Noun</target>
    <linkType>gun</linkType>
    <priority>90</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- Direktör Ali - Director Ali - The proper noun is a person name  -->
  <NounPhraseRule>
    <ID>66</ID>
    <sourceType>LIST</sourceType>
```

```
    <source>meslek.txt</source>
    <targetType>POS</targetType>
    <target>Prop</target>
    <linkType>Title</linkType>
    <priority>70</priority>
    <constraint>[S->T:numberOfWords(1)]^
    [S+1:inList(personTitle.txt)]</constraint>
</NounPhraseRule>

<!-- Connects nom. Prop. noun to nom. noun which is possible a proper noun-->
<NounPhraseRule>
    <ID>67</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+A3sg,+Pnon,+Nom</source>
    <targetType>POSPI</targetType>
    <target>Noun,A3sg,+Pnon,+Nom</target>
    <linkType>Proper</linkType>
    <priority>75</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
</NounPhraseRule>

<!--  İlyas Çiçekli  -->
<NounPhraseRule>
    <ID>68</ID>
    <sourceType>POSPI</sourceType>
    <source>Prop,+Nom</source>
    <targetType>POS</targetType>
    <target>Adj</target>
    <linkType>Prop</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ilce.txt)</constraint>
</NounPhraseRule>

<!--  Connect nominative noun which is possibly a proer noun to a proper noun-->
<NounPhraseRule>
    <ID>69</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+Nom</source>
    <targetType>POS</targetType>
    <target>Prop</target>
    <linkType>Proper</linkType>
    <priority>105</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isPossible(Prop)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
</NounPhraseRule>

<!-- Connects special words to modifies liquids -->
<NounPhraseRule>
    <ID>70</ID>
    <sourceType>LIST</sourceType>
    <source>SiviModifierList.txt</source>
    <targetType>LIST</targetType>
    <target>siviList.txt</target>
    <linkType>A</linkType>
    <priority>80</priority>
</NounPhraseRule>

<!-- Connects adjectives which are possibly noun to p3 nounss  -->
<NounPhraseRule>
    <ID>71</ID>
    <sourceType>LIST</sourceType>
```

```
    <source>AdvAdjList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Belirtisiz</linkType>
    <priority>40</priority>
    <constraint>!T:contains(Pnon)</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isPossible(Noun)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!T:hasEndLink(Bsiz)</constraint>
    <constraint>!T:hasEndLink(Belirtisiz)</constraint>
    <constraint>!T:hasEndLink(Bli)</constraint>
    <constraint>!T:inList(BirlesikIsimList.txt)</constraint>
</NounPhraseRule>

<!-- liderlerinden Aslı - Aslı, one of the leaders of..-Aslı is found as noun-->
<NounPhraseRule>
    <ID>72</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+A3pl,+Abl</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>ofProp</linkType>
    <priority>80</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
</NounPhraseRule>

<!-- Erzurum 3. ceza mahkemesi -  Erzurum 3. Fine court - Connects Erzurum to
court -->
<NounPhraseRule>
    <ID>73</ID>
    <sourceType>LIST</sourceType>
    <source>TurkiyeSehir.txt</source>
    <targetType>LIST</targetType>
    <target>yereAitIsimler.txt</target>
    <linkType>Bsiz</linkType>
    <priority>80</priority>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
    <constraint>!T:contains(Pnon)</constraint>
    <constraint>S:contains(Nom)</constraint>
</NounPhraseRule>

<!--  General Muşaviri - General Consultant - Used for wrong disambiguation -->
<NounPhraseRule>
    <ID>74</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropNomList.txt</source>
    <targetType>POSPI</targetType>
    <target>Prop,+Nom</target>
    <linkType>Bsiz</linkType>
    <priority>80</priority>
    <constraint>T:isPossible(+P3)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!T:hasEndLink(Belirtisiz)</constraint>
    <constraint>!T:hasEndLink(Bsiz)</constraint>
    <constraint>[![S+1->T-1:containsPOS(Noun)]|
    [S+1->T-1:containsPOS(Prop)]]^[!S->T:numberOfWords(0)]</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
</NounPhraseRule>

<!--  Mucahid Kutlu - connects person names to surnames     -->
<NounPhraseRule>
    <ID>75</ID>
    <sourceType>POS</sourceType>
```

```
    <source>Noun</source>
    <targetType>POS</targetType>
    <target>Prop</target>
    <linkType>Proper</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:inList(person.first_name.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
</NounPhraseRule>

<!-- BJK Ankara Şubesi - BJK Ankara Branch - Connects institution names to city
names -->
<NounPhraseRule>
    <ID>76</ID>
    <sourceType>LIST</sourceType>
    <source>kurumList.txt</source>
    <targetType>LIST</targetType>
    <target>TurkiyeSehir.txt</target>
    <linkType>Prop</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isPossible(Prop)</constraint>
</NounPhraseRule>

<!--  Connects person names to nouns which are possibly proper noun    -->
  <NounPhraseRule>
    <ID>77</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropList.txt</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>Proper</linkType>
    <priority>105</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isPossible(Prop)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>S:inList(person.first_name.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
</NounPhraseRule>

<!--  Connects person names founds as noun to adjectives which is possibly a
proper noun   -->
<NounPhraseRule>
    <ID>78</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropList.txt</source>
    <targetType>POS</targetType>
    <target>Adj</target>
    <linkType>Proper</linkType>
    <priority>105</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S:isFirstCharacrerCapital()</constraint>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>S:inList(person.first_name.txt)</constraint>
    <constraint>!T:inList(TurkiyeSehir.txt)</constraint>
    <constraint>!T:inList(ulke.txt)</constraint>
</NounPhraseRule>

<!--400 ve ustunde eleman - 400 and more employee - connects 400 to ve -->
<NounPhraseRule>
    <ID>79</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>SF</targetType>
    <target>ve</target>
    <linkType>postNumber</linkType>
    <priority>111</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T+1:inList(postNumberModifierList.txt)</constraint>
```

```
</NounPhraseRule>

<!--400 ve ustunde eleman - 400 and more employee - connects ve to üstünde -->
<NounPhraseRule>
    <ID>80</ID>
    <sourceType>SF</sourceType>
    <source>ve</source>
    <targetType>LIST</targetType>
    <target>postNumberModifierList.txt</target>
    <linkType>postNumber</linkType>
    <priority>110</priority>
    <constraint>S-1:contains(+Num+)</constraint>
</NounPhraseRule>

<!--30 ile 40 - 30 and 40 - Conncets 30 to ile -->
<NounPhraseRule>
    <ID>81</ID>
    <sourceType>POS</sourceType>
    <source>Num</source>
    <targetType>SF</targetType>
    <target>ile</target>
    <linkType>numberConnector</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T+1:contains(+Num+)</constraint>
</NounPhraseRule>

<!--30 ile 40  - connects ile to 40 -->
<NounPhraseRule>
    <ID>82</ID>
    <sourceType>SF</sourceType>
    <source>ile</source>
    <targetType>POS</targetType>
    <target>Num</target>
    <linkType>numberConnector</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>S-1:contains(+Num+)</constraint>
</NounPhraseRule>

<!-- subat tarihli anlaşma - agreement with February date - - Connects Şubat to
tarih    -->
<NounPhraseRule>
    <ID>83</ID>
    <sourceType>LIST</sourceType>
    <source>AyList.txt</source>
    <targetType>SSF</targetType>
    <target>tarih</target>
    <linkType>takisiz</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!--   "gerçek" adam - “real” man    -->
<NounPhraseRule>
    <ID>84</ID>
    <sourceType>LIST</sourceType>
    <source>tirnakList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>QoutationAdj</linkType>
    <priority>100</priority>
    <constraint>S-1:contains(Adj)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!S:hasEndLink(Proper)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
    <constraint>[![T:isPossible(Verb)]^[S:isPossible(Adverb)]]</constraint>
```

```
    <constraint>!S:inList(Rule1ExceptList.txt)</constraint>
</NounPhraseRule>

<!--  adamlardan 2'si - two of the men  -->
<NounPhraseRule>
    <ID>85</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropPronAblList.txt</source>
    <targetType>POSPI</targetType>
    <target>Noun,+P3</target>
    <linkType>Bsiz</linkType>
    <priority>100</priority>
    <constraint>T:derivedFrom(Num)</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!S:hasEndLink(Proper)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
    <constraint>[![T:isPossible(Verb)]^[S:isPossible(Adverb)]]</constraint>
</NounPhraseRule>

<!--  yüzde 5 - 5 percent -->
<NounPhraseRule>
    <ID>86</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+Loc</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>A</linkType>
    <priority>100</priority>
    <constraint>T:derivedFrom(Num)</constraint>
    <constraint>S:derivedFrom(Num)</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- Prof. Atalar   -->
<NounPhraseRule>
    <ID>87</ID>
    <sourceType>LIST</sourceType>
    <source>kisaltmalar.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Title</linkType>
    <priority>70</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- benim arabam - my car -->
<NounPhraseRule>
    <ID>88</ID>
    <sourceType>POSPI</sourceType>
    <source>Pron,+Gen</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Bli</linkType>
    <priority>90</priority>
    <constraint>!T:contains(Pnon)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>S~T:agree(Person)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T->E:hasP3SGNounDerivedFromVerb()</constraint>
    <constraint>!T:hasEndLink(Bli)</constraint>
    <constraint>!T:inList(SifatIsimTamlamari.txt)</constraint>
</NounPhraseRule>
```

```
<!-- Avulat Ali -Advocate Ali   -->
<NounPhraseRule>
    <ID>89</ID>
    <sourceType>LIST</sourceType>
    <source>meslek.txt</source>
    <targetType>POS</targetType>
    <target>Noun</target>
    <linkType>Title</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isPossible(Prop)</constraint>
</NounPhraseRule>

<!-- en az - the least -->
<NounPhraseRule>
    <ID>90</ID>
    <sourceType>POSFI</sourceType>
    <source>Adverb+AdjMdfy</source>
    <targetType>POSFI</targetType>
    <target>Adverb+AdjMdfy</target>
    <linkType>Eap</linkType>
    <priority>110</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- en az 10 - at least 10 - Connects az to 10 -->
<NounPhraseRule>
    <ID>91</ID>
    <sourceType>POSFI</sourceType>
    <source>Adverb+AdjMdfy</source>
    <targetType>POS</targetType>
    <target>Num</target>
    <linkType>Eap</linkType>
    <priority>110</priority>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsPOS(Punc)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S+1->T-1:numberOfWord(",1,EQ)</constraint>
    <constraint>!S+1->T-1:numberOfWord(',1,EQ)</constraint>
    <constraint>!S->T:containsUnconnectedToken()</constraint>
</NounPhraseRule>

<!--  konuya ilişkin - related to the topic  -->
<NounPhraseRule>
    <ID>92</ID>
    <sourceType>LIST</sourceType>
    <source>DatSifatList.txt</source>
    <targetType>ANY</targetType>
    <target>ANY</target>
    <linkType>About</linkType>
    <priority>80</priority>
    <constraint>S:hasEndLink(About)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasBeginLink(Takisiz)</constraint>
    <constraint>!S:hasEndLink(Proper)</constraint>
    <constraint>!T:hasBeginLink(NA)</constraint>
    <constraint>[![T:isPossible(Verb)]^[S:isPossible(Adverb)]]</constraint>
    <constraint>!S:inList(Rule1ExceptList.txt)</constraint>
</NounPhraseRule>

<!- connects any word to comma  -->
<NounPhraseRule>
    <ID>93</ID>
    <sourceType>ANY</sourceType>
    <source>ANY</source>
```

```
    <targetType>SF</targetType>
    <target>,</target>
    <linkType>CommaConnector</linkType>
    <priority>105</priority>
    <constraint>S:isPossible(Prop)</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:hasBeginLink(CommaConnector)</constraint>
</NounPhraseRule>

<!-- connects comma to any word  -->
<NounPhraseRule>
    <ID>94</ID>
    <sourceType>SF</sourceType>
    <source>,</source>
    <targetType>ANY</targetType>
    <target>ANY</target>
    <linkType>CommaConnector</linkType>
    <priority>105</priority>
    <constraint>T:isPossible(Prop)</constraint>
    <constraint>S->T:numberOfWords(0)</constraint>
    <constraint>T:isConnectedToLink(connector)</constraint>
</NounPhraseRule>

<!-- onun arabası - his car-->
<NounPhraseRule>
    <ID>95</ID>
    <sourceType>POSPI</sourceType>
    <source>Pron,+Gen</source>
    <targetType>LIST</targetType>
    <target>NounPropList.txt</target>
    <linkType>Bli</linkType>
    <priority>90</priority>
    <constraint>T:contains(+P3)</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>S~T:agree(Person)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T->E:hasP3SGNounDerivedFromVerb()</constraint>
    <constraint>!T:inList(SifatIsimTamlamari.txt)</constraint>
</NounPhraseRule>

<!--in case wrong disambiguation of P3sg+Nom ( Pnon+Acc ), connects nominative
noun to Pnon+Acc noun -->
<NounPhraseRule>
    <ID>96</ID>
    <sourceType>LIST</sourceType>
    <source>NounPropList.txt</source>
    <targetType>LIST</targetType>
    <target>NounPropPnonAccList.txt</target>
    <linkType>Bsiz</linkType>
    <priority>80</priority>
    <constraint>T:isPossible(P3sg)</constraint>
    <constraint>S:contains(Nom)</constraint>
    <constraint>S~T:agree(Plural)</constraint>
    <constraint>!T:hasEndLink(YearN)</constraint>
    <constraint>!T:hasEndLink(postNumber)</constraint>
    <constraint>!T:hasEndLink(Belirtisiz)</constraint>
    <constraint>[![S+1->T-1:containsPOS(Noun)]|
    [S+1->T-1:containsPOS(Prop)]]^[!S->T:numberOfWords(0)]</constraint>
    <constraint>!S->T:containsPOS(Verb,exceptList.txt)</constraint>
    <constraint>!S:derivedFrom(Verb,exceptList.txt)</constraint>
    <constraint>[![S->T:numberOfWords(1)]^[S->T:containsPOS(Punc)]]</constraint>
    <constraint>!S->T:containsList(Rule10BlockerList.txt)</constraint>
    <constraint>!S->T:containsList(NPBlockerList.txt)</constraint>
    <constraint>!T:inList(BirlesikIsimList.txt)</constraint>
</NounPhraseRule>
```

```
<!-- X benzeri Y -  Y like X -  Connects X to benzeri  -->
<NounPhraseRule>
    <ID>97</ID>
    <sourceType>POSPI</sourceType>
    <source>Noun,+Nom</source>
    <targetType>LIST</targetType>
    <target>postNomNounModifierList.txt</target>
    <linkType>postNomNounModifier</linkType>
    <priority>100</priority>
    <constraint>S->T:numberOfWords(0)</constraint>
</NounPhraseRule>

<!-- Connects special words to modifies clothes-->
<NounPhraseRule>
    <ID>98</ID>
    <sourceType>LIST</sourceType>
    <source>GiyecekModifierList.txt</source>
    <targetType>LIST</targetType>
    <target>GiyecekList.txt</target>
    <linkType>A</linkType>
    <priority>80</priority>
</NounPhraseRule>
```