DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DETERMINATION OF TURKISH WORD TYPES

by
Ümit HALLAÇ

September, 2007
İZMİR

DETERMINATION OF TURKISH WORD TYPES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science in
Computer Engineering, Computer Engineering Program

by

Ümit HALLAÇ

September, 2007
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DETERMINATION OF TURKISH WORD TYPES"** completed by **ÜMİT HALLAÇ** under supervision of **ASSOC.PROF.DR. YALÇIN ÇEBİ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Assoc.Prof.Dr. Yalçın ÇEBİ

Supervisor

.....

(Jury Member)

.....

(Jury Member)

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Assoc. Prof. Dr. Yalçın ÇEBİ for his assistance and encouragement throughout the progress of this study.

I also would like to thank to Asst. Prof. Dr. Gökhan DALKILIÇ for leading me to this research.

Finally, I would like to express my deepest gratitude to my parents for their endless support.

Ümit HALLAÇ

DETERMINATION OF TURKISH WORD TYPES

ABSTRACT

Natural Language Processing (NLP) is a field of research that studies the problems of automated generation and understanding of natural languages. Part-of-speech (POS) tagging is a sub-problem in NLP which is interested in tagging the words in a document with the appropriate parts-of-speech. POS tagging has many uses in fields such as: full text searching, information retrieval, speech synthesis and pronunciation and high level text analysis.

This study introduces TurPOS, a new rule-based part-of-speech tagger system that was developed for Turkish. The system aims to assign the appropriate word classes for each word in a given Turkish document.

TurPOS uses a text corpora produced by a morphological analyzer as the input document. The system also uses a rule file that contains the list of grammatical Turkish rules. The structure of the rule file is quite simple and flexible. This makes it possible that the system can be used for tagging other languages, simply by modifying the rule file according to the grammar of the language.

Keywords: Natural Language Processing, part-of-speech tagging, rule-based Turkish POS tagger.

TÜRKÇE KELİMELERİN TÜRLERİNİN BELİRLENMESİ

ÖZ

Doğal Dil İşleme (DDİ), doğal dillerin otomatik olarak anlaşılması ve üretilmesi problemlerini inceleyen bir araştırma alanıdır. Sözcük Türleri Belirleme (STS) ise, DDİ'nin bir alt problemidir ve bir metin içindeki kelimelere en uygun kelime türlerinin atanmasıyla ilgilidir. STS uygulamaları pratikte birçok alanda kullanılmaktadır. Bu alanlardan bazıları: tam metin arama, bilgi edinme, konuşma sentezi ve telaffuz ile yüksek seviye metin çözümlemedir.

Bu çalışma, Türkçe için geliştirilen kural tabanlı bir Sözcük Türleri Belirleyicisi olan TurPOS'u tanıtmaktadır. Sistem, verilen bir Türkçe dokümandaki her sözcüğe, en uygun sözcük sınıfını atamayı amaçlamaktadır.

TurPOS, girdi dokümanı olarak biçimbirimsel çözümleme uygulaması tarafından üretilen bir derlem kullanmaktadır. Sistem ayrıca, Türkçe gramer kurallarını içeren bir de kural dosyası kullanmaktadır. Bu kural dosyasının yapısı oldukça sade ve esnektir. Bu sayede, yalnızca kural dosyasını farklı dil gramerlerine göre yeniden uyarlayarak, sistemin başka dillerde de kullanılması mümkün olacaktır.

Anahtar sözcükler: Doğal Dil İşleme, sözcük türleri saptama, kural tabanlı Türkçe sözcük türleri belirleyicisi.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Natural Language Processing

The computer applications and the way of the communication between people and computers are changing fast, proportional to the tendency of continuous improvement of the computer technology during the last few decades.

Although the idea of developing intelligent computers have raised among computer scientists in 1950s, the studies were quite limited depending on the restricted resources and knowledge. Today's computers are much faster and strong in computations compared to their ancestors. As a consequence, today's computer scientists have the opportunity to make computers behave in a more intelligent and human-like way.

One of the main tasks to create human-like computers is creating computers that are capable of analyzing and interpreting people's language. This task is called as Natural Language Processing (generally referred as NLP) and basicly aims to make computers understand human's natural language and even to make them generate it.

The studies in NLP are almost old as the development of first computers (Booth, Brandwood & Cleave, 1958). Up to today, many studies and methods regarding NLP have been developed and published. As a matter of natural language complexity, the progress in this field is much slower, compared to oher fields of computing.

Natural Language Processing consists of four main analysis levels where each level is strongly related to others. Dividing the complex NLP problem into levels help researchers to focus on each level separately, produce output for each and combine results to produce the overall results. Figure 1.1 shows these basic analysis levels.

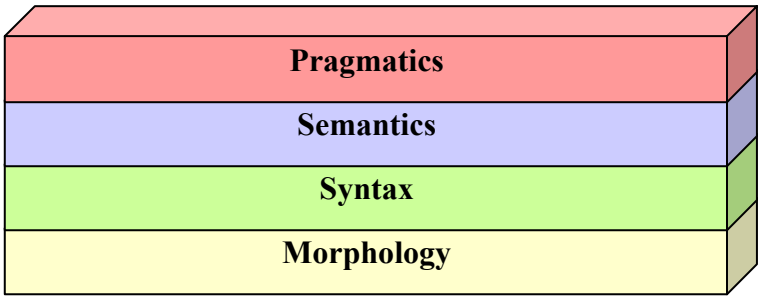

Figure 1.1 Basic analysis levels of NLP

Morphology lays at the bottom level and is directly related to word based analysis. Morphology aims to define the structure and formation of words. The results of morphological analysis are used for further processing in higher level analysis.

Syntax is a higher level of analysis than morphology. Syntactic analysis is based on sentences which are more complex components of natural languages than words. Similarly this analysis is used to determine the structure of sentences and grouping of words.

Semantic analysis is the study of meaning that is obtained by syntactic structure of the sentence and meanings of the words of this sentence.

Pragmatic analysis lays at the top and is a much more complex study than the ones in lower levels. It aims to determine the meaning of discourse involving the world knowledge and the contextual information.

1.2 Part-of-Speech Tagging

Each analysis level in Figure 1.1 consists of many sub-problems. One of the main sub-problems of NLP in Syntax Level is Part-of-Speech (POS) Tagging.

Words of language are grouped by linguists into classes according to their similar syntactic behaviour. Those word classes are called parts-of-speech (POS) and the most important three of them are: noun, verb and adjective (Manning & Schutze, 1999).

Formally, “Part-of-speech tagging is the process of assigning a part-of-speech label to each of a sequence of words.” (Jurafsky & Martin, 2000, p. 314).

Figure 1.2 describes the POS tagging process. A group of words that compose a sentence are analyzed to find out the appropriate word classes for each word in the context.

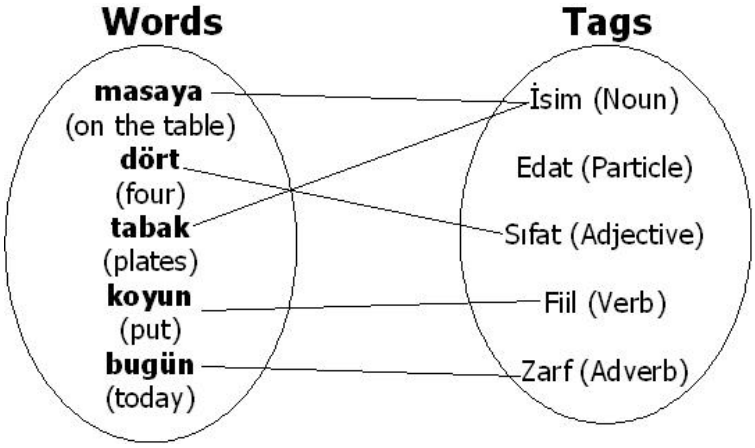

Figure 1.2 Part-of-Speech tagging process aims to allocate appropriate word classes to words of sentences.

Investigating the available research on POS tagging, several approaches can be encountered. Guilder (1995) makes a distinction among POS taggers according to taggers’ automation degree in training and tagging process. Two approaches subject to this distinction are:

- Supervised Tagging
- Unsupervised Tagging

The most important difference between these two approaches is that supervised taggers generally require a pre-tagged corpora to be used in the tagging process, whereas unsupervised taggers do not.

Another distinction in POS tagging can be done according to characteristics of POS taggers. There are three basic approaches encountered in this distinction:

1. Rule-based Tagging
2. Stochastic Tagging
3. Combination (hybrid) Tagging

Rule-based approaches generally use a lexicon and a list of grammatical rules of natural language. This method basicly applies the rules to a word group including words with several possible word classes (e.g. both adjective and noun) for word class disambiguation.

Stochastic tagging approach aims to resolve the ambiguities of word classes by computing probabilities and frequencies. Some stochastic tagging models include Hidden Markov Models (HMM) to tag words of documents.

Combination approach which may also be called as hybrid approach, as its name suggests, combines the advantages of both approaches to improve the overall performance of the tagging system.

There are many POS tagging systems available today and some of them report great accuracy (Cutting, Kupiec, Pedersen & Sibun, 1992; Brill, 1995; Voutilainen, 1995; Garside, 1997). Those systems are mostly developed for English text and not suitable for porting into other languages. Therefore, researchers have to rebuild their tagging systems for different natural languages.

POS tagging has many practical uses in full text searching, information retrieval, speech synthesis and pronunciation and high level text analysis.

1.3 Main Characteristics of Turkish

Turkish is an agglutinative language which belongs to the Ural-Altaic family of languages of Asia. Modern Turkish has many grammatical and lexical similarities with other Turkic languages such as Turkmen, Gagauz, Azerbaijani (Azeri), Kazakh, Kyrgyz, Tatar, etc.

The main characteristics of the language is given as the following:

- Words are formed adding suffixes to the end of root words,
- It has a complex but regular grammar,

- Nouns have neither genders nor definite articles,
- The language is phonetic (each letter has a specific sound),
- Turkish has a vowel harmony,
- Sentences are generally in "Sentence – Object – Verb" order,
- The Turkish alphabet has 29 letters. It does not contain the letters Qq, Xx and Ww but has the following ones: Şş, Çç, Üü, Öö, Ğğ, Iı and İi.

There are 3 basic word types in Turkish when words are categorized according to their meanings or tasks. These word classes are as the following:

1. Nouns
2. Verbs
3. Prepositions ("postposition" fits much better for this word class in Turkish)

Each word in Turkish can be grouped under one of those three word classes. Nouns and verbs have lexical meanings on their own, whereas prepositions have grammatical meanings.

1.4 Aim of the Thesis

This thesis is a part of a study which fundamentally aims to develop original methods to make computers capable of analyzing and understanding the Turkish language according to the characteristics of this language.

This thesis also aims to provide an intellectual contribution on NLP studies for Turkish and other Turkic languages, since the variety and the quantity of these studies are quite limited.

1.5 Organization of the Thesis

The overall outline of this thesis can be summarized as follows:

Chapter Two includes fundamentals of NLP and an overview of part-of-speech (POS). This chapter describes how POS tagging takes place in analyzing natural languages and indicates the main problems of this analysis process. Previous works on POS tagging also takes place in Chapter Two.

In Chapter Three, a rule-based POS tagger system that was developed mainly for Turkish (TurPOS) is introduced to the reader. The design, architecture and the overall performance of the TurPOS system is analyzed deeply in this chapter.

Chapter Four includes the conclusion part which summarizes the thesis and proposes future work regarding the studies in the context of this thesis.

There are 4 appendices included within the thesis. Appendix A contains a sample rule file used in TurPOS and Appendix B contains a sample text corpora file. Appendix C contains a sample document tagged by TurPOS. And finally, Appendix D contains the TurPOS user interface with some views captured from the system with brief descriptions for each of the screens.

CHAPTER TWO
PREVIOUS WORKS

Research on part-of-speech tagging can be said to begin with the development of the Brown Corpus in 1960s, since first POS tagging studies were based on it. The reason behind creating a large corpus of English was to use this data in computer analysis. The Brown Corpus consisted of complete sentences gathered from various resources including about 1,000,000 English words.

One of the first studies in POS tagging was a deterministic rule-based tagger which focused on tagging the words of the Brown Corpus (Greene & Rubin, 1971). The tagger (called TAGGIT) achieved an accuracy of 77%. The results of this study showed the complexity of the problem.

Brill's simple rule-based part-of-speech tagger which makes modifications on itself to improve performance, achieved an accuracy of 96% (Brill, 1992). The accuracy of this tagger was improved to 97.5% with some advances (Brill, 1994). Another well known research on rule-based POS tagging is the ENGTWOL tagger (Voutilainen, 1995) that uses the Constraint Grammar approach of Karlsson, Voutilainen, Heikkilä & Anttila (1995).

There are various POS tagging approaches that rely on stochastic methods (DeRose, 1988; Church, 1988; Charniak, 1993). Modern stochastic taggers are mostly based on Hidden Markov Model (HMM) to choose the appropriate tag for a word. The Xerox POS tagger is also based on a HMM with a result of 96% accuracy (Cutting, Kupiec, Pedersen & Sibun, 1992).

The most famous combination tagging system is Brill's transformation-based tagger (Brill, 1995). The tagger determines ambiguous word classes using rules like other rule-based taggers. On the other hand, like stochastic taggers, it includes a machine learning mechanism that provides rules to be induced from the text.

CLAWS is also a combination tagger which is based on a HMM with a rule-based component to handle idioms (Garside, 1987, 1997). This tagger reports the accuracy of 97%. Another combination tagger developed by Tapanainen and Voultilainen, uses

ENGCG and the Xerox Tagger to independently tag the same document and combine the results (1994).

Studies on Turkish POS tagging are quite limited. A tagging tool for Turkish, that is implemented on the PC-KIMMO environment (Antworth, 1990), was published by Oflazer and Kuruoz (1994).

CHAPTER THREE
TURPOS

3.1 Overview of TurPOS Tagger

TurPOS is a rule-based part-of-speech tagging system developed mainly for Turkish language. The system aims to define and tag the classes of words in Turkish texts, like any other POS taggers.

Developing a rule-based tagger for a natural language mainly requires three inputs: a complete lexicon which includes the list of whole words of the language, a list of grammatical rules for that language and the text to be tagged. The format of lexicon and rule list may differ according to aspects of the application.

As a rule-based tagger, TurPOS requires a list of grammatical rules for Turkish as an input. The analysis on word groups in sentences is achieved by processing these rules (TDK Güncel Türkçe Sözlük, (n.d.)).

Another input to the TurPOS system is the result document produced from the morphological analysis process. This result file includes the words of the document that will be tagged, with its possible stems, suffixes and word classes. Including all the possible word classes that a word in the document may have, avoids the requirement for searching a complete lexicon in the system for every word. Figure 3.1 shows the inputs and outputs of TurPOS.

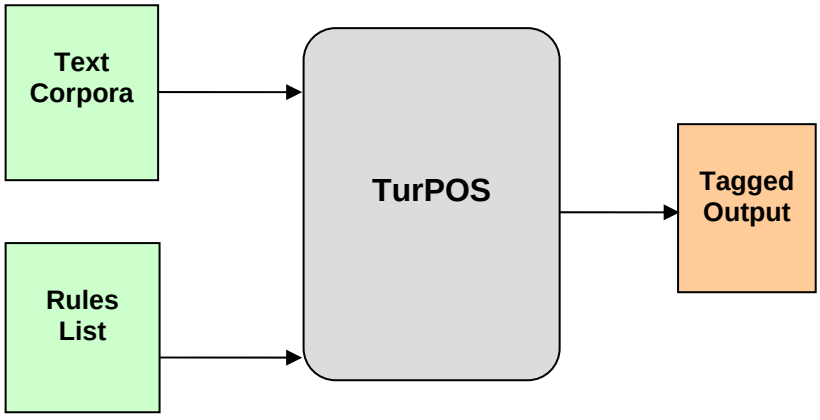

Figure 3.1 Inputs and outputs of the TurPOS system.

All input and output file formats in TurPOS are based on W3C XML. The format and the structure of these files are explained in further details, later in this chapter.

TurPOS is part of a large NLP project which also includes Sentence Boundary Detection (Aktaş, 2006) and Morphological Analyzer applications (Çebi & Varlıklar, 2005). The overall design of the TurPOS system depends on interactions with these applications on a large scale. Beyond the constraints based on interacting with other applications, the design aspects of TurPOS are based on goals of providing fast and accurate tagging and portability among Turkic languages such as Turkmen, Gagauz, Azerbaijani (Azeri), Kazakh, Kyrgyz, Tatar, etc.

The TurPOS tagger was developed as a Windows application in Microsoft .NET Visual Studio 2005 (.NET Framework 2.0) environment using the C# programming language. The main reason behind this decision was for compatibility between sentence boundary detection and morphologic analyzer applications that supplement TurPOS.

3.2 TurPOS Modules

TurPOS consists of four modules:

- Rule Parser Module
- Stem Reader Module
- Tagger Module
- User Interface Module

The Rule Parser Module is responsible for loading the rule list into the system. Module works only once on program start up.

The Stem Module is used for reading the text corpora from an input file and load it into the system as a user-defined data structure. The input file of this module is provided by an external system which analyzes documents morphologically, stems words and returns all possible stems, suffixes and word classes to TurPOS.

The main module of the system which tags input documents is the Tagger Module. The module processes rules list on morphologically analyzed text corpora and produces the output. The tagged text output is written into an XML file for storage. Figure 3.2 shows the general structure of the TurPOS tagging system with its modules.

The User Interface Module, as its name suggests, is the module that deals with the interactions of TurPOS with users. Beyond including the full functionality of the system, the module also includes some functions for easier maintenance on the rule file such as adding new rules to the rule file.

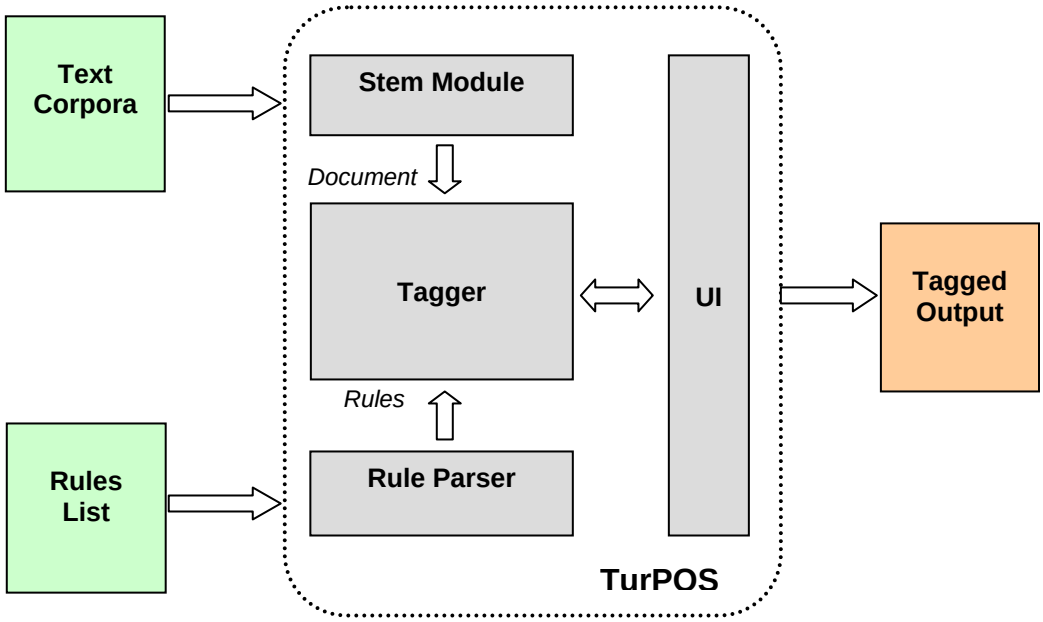

Figure 3.2 General structure of TurPOS. The system consists of three modules.

3.2.1 Rule Parser Module

Syntax is the grammatical arrangement of words in sentences. Any natural language has predefined rules to describe the absolute and relative positions of word classes in sentences. These rules are called grammatical or syntactical rules or sometimes context frame rules. Almost all POS taggers use those grammatical rules of a natural language as the basic rules to analyze and tag the text of this language.

TurPOS uses a rule file that is completely composed from the grammatical rules of Turkish. The rule file includes individual rules defining the order of word classes that are eligible or not eligible to be used in a sentence. It also includes rules defining the positions of punctuation marks against word classes within a sentence.

The format of the rule list in a POS tagger changes according to the aims and aspects of the tagger. Some systems use rule compilers (Antworth, 1990), whereas some systems use standard or customized database management systems. Some systems, including the TurPOS, use text based rule files.

Developers need to concern about few issues when deciding the format of rule list in a tagger. First of all, the rule list should be stored in a format that is efficiently parseable by the system. Secondly, the list should be in a form that users can easily read, understand and modify. That way, the performance of the system can be improved by adding new rules into or removing incorrect rules from the rule list. This also makes the system flexible and scalable.

Considering the issues mentioned above, in the design period of TurPOS, using a rule file based on the World Wide Web Consortium (W3C) XML standard was decided. The advantages for using an XML based rule file can be listed as the following:

- Increased human readability provides easy modification,
- No programs/tools but a simple text editor is sufficient for rule list modifications,
- Parsing rules from text based file is faster than parsing from complex DBMS tables,
- Removing/changing only one rule is possible since rules are independent from each other,
- Rule list has no limit on number of rules it contains

Table 3.1 shows the parts-of-speech used in TurPOS. The list also includes the punctuation marks that can take place in sentences. This is a design decision to improve the accuracy of tagging results in the system considering the locations of punctuation marks in sentences.

Table 3.1 The word classes used in the TurPOS system and corresponding descriptions in English.

PARTS of SPEECH	DESCRIPTION
sıfat	adjective
isim	noun
zamir	pronoun
fiil	verb
zarf	adverb
edat	preposition
bağlaç	conjunction
sayı	number
nokta	dot
virgül	comma
ikinokta	colon
noktalıvirgül	semicolon
soru	question
ünlem	exclamation

Table 3.2 shows sample rules and their corresponding descriptions from the TurPOS rule file.

Table 3.2 Sample rules and corresponding rule descriptions from the rule file.

RULE SAMPLE	RULE DESCRIPTION
<Rule RuleId="2" RuleType="sözdizim" RuleState="true"> <Item ItemType="sıfat" /> <Item ItemType="sıfat" /> <Item ItemType="isim" /> </Rule>	An adjective group must be followed by a noun which is described or determined by these adjectives.
<Rule RuleId="3" RuleType="sözdizim" RuleState="false"> <Item ItemType="sıfat" /> <Item ItemType="virgül" /> <Item ItemType="isim" /> </Rule>	There can not be a comma between an adjective and a noun which is determined by this adjective.

As seen in Table 3.2, the rule format of the rule list is quite simple and flexible. Only required attributes take place in a rule. As a decision of design, all attributes and elements in the rule file are written in English. But the values of these attributes are written in Turkish (and possibly will be written in other Turkic languages in the future).

A rule in the rule list is basicly an element with three attributes and undefined number of item elements within. The *RuleId* attribute of a rule is a unique rule number, used to determine a rule in the file. *RuleType* is a string attribute which defines the type of the rule. Basic rule type for TurPOS is grammatical rules and the *RuleType* attribute for a grammatical rule is expressed as “sözdizim” (which means “syntax” in English) in the rule file. Finally the third attribute, the *RuleState* attribute defines whether syntax of given word classes that rule contains are applicable or not. In other words, when value of the *RuleState* attribute is “false” in a rule, then there is no possibility for words of a sentence to be ordered as in given sequence, in this natural language.

As mentioned above, each rule element has sub elements of items, which correspond to words or punctuation items in sentences. Item element has only one attribute: *ItemType*. The *ItemType* attribute defines the word class of an item. If the item is a punctuation mark, the *ItemType* attribute includes its type such as “virgül” (comma).

An extended sample rule file used in TurPOS is included in Appendix A.

The Rule Parser module is an extended XML parser including few entities to maintain the rule list. The module works in the following way: The whole rule file is read once on startup of the TurPOS system. The parser then parses rules individually and loads them onto the memory for use at the rest of the program runtime. Since the rule list is not considered to be changed during tagging, the rule file processing is done once at startup to get rid of its overhead for the system.

3.2.2 Stem Reader Module

The Stem Reader Module is the part that is responsible for loading the morphologically analyzed documents into the system. The module parses the XML file containing the

document to be tagged and stores it into its entities for use in the tagging process. The Stem Reader entities and their hierarchy are given in Figure 3.3.

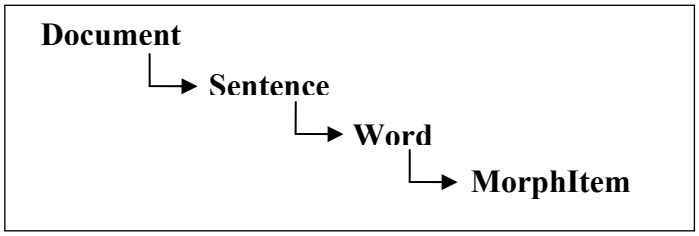

Figure 3.3 The Stem Reader Module entities and their hierarchy

Unlike most POS taggers, TurPOS doesn't include a built-in lexicon. Instead, the classes of words in a document are read from the text corpora which contains the morphologically analyzed document. This design difference in TurPOS is due to the fact that it is a part of a complete NLP project with a morphological analyzer and a sentence boundary detection system.

Parsing the possible word classes of words from the document itself has an advantage against a built-in lexicon structure. This eliminates the extra overhead for searching the lexicon for each word and parsing word classes of words, because the text corpora which contain the document to be tagged is parsed in any circumstances in a POS tagger. Parsing and storing word classes of words at that time do not bring out any extra overheads.

Similar to the Rule Parser, Stem Reader Module also parses the complete text corpora file once, just before starting the tagging process. The complete document is loaded into the system for use in the Tagger Module.

As mentioned above, the text corpora is produced as an output file from the morphological analyzer program that was also developed by our workgroup at the university. The text corpora file is based on W3C XML standard. Contents of this file are basicly the required inputs for the TurPOS system and other details to be returned in the overall results. An example document including a single sentence from the text corpora is given in Figure 3.4.

```
<?xml version="1.0" encoding="utf-8" ?>
<Document DocumentId = "0">
    <Sentence SentenceId = "0">
        <Word WordId = "0" Value = "Koyun">
            <Morph MorphId = "0" Stem="koyun" Type ="isim" Suffix =
                "ISIM_YALIN_BOS"></Morph>
            <Morph MorphId = "1" Stem="koy" Type ="isim" Suffix ="ISIM_YALIN_BOS +
                ISIM_TAMLAMA_IN"></Morph>
            <Morph MorphId = "2" Stem="koy" Type ="isim" Suffix ="ISIM_YALIN_BOS +
                ISIM_SAHIPLIK_SEN_IN"></Morph>
            <Morph MorphId = "3" Stem="koy" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
                FIIL_EMIR_SIZ_IN"></Morph>
        </Word>
        <Word WordId = "1" Value = "masaya">
            <Morph MorphId = "0" Stem="masa" Type ="isim" Suffix ="ISIM_YALIN_BOS +
                ISIM_YONELME_E"></Morph>
        </Word>
        <Word WordId = "2" Value = "çıktı">
            <Morph MorphId = "0" Stem="çık" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
                FIIL_GECMISZAMAN_DI"></Morph>
            <Morph MorphId = "1" Stem="çıktı" Type ="isim" Suffix ="ISIM_YALIN_BOS"></Morph>
        </Word>
        <Word WordId = "3" Value = ".">
            <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
        </Word>
    </Sentence>
</Document>
```

Figure 3.4 An example document including a single sentence from the text corpora

As seen in Figure 3.4, the structure of the text corpora file is quite simple. The hierararchy of the file is similar to the hierarchy of a natural language document where a document is composed of individual sentences and each sentence is composed of words.

The text corpora file include the *Morph* elements for each word as an additional structure. These elements are included to list all possible stems and suffixes produced from the morphological analysis stage for each word.

For a better understanding on the use of *Morph* elements in the file, we may analyze the first word (defined with WordId = "0") in Figure 3.4. The word "Koyun" has 4 possible "stem + suffix" sequences according to the results of the morphological analysis. The possible structures for this word are given below with brief explanations:

- **Koyun:** The word itself is the stem and means "sheep" in English. It is used as a noun in singular form without any suffixes.

- **Koy-un :** The word consists of a noun stem and a possession suffix for singular third person. The word has the meaning similar to the phrase "the cove's" as in the sentence "The cove's view was wonderful".

- **Koy-un :** The word consists of a noun stem (the same stem as in the previous structure) and a possession suffix for singular second person. Its meaning can be defined as the phrase "your cove" as in the sentence "Your cove has a wonderful view".

- **Koy-un :** The word consists of a verb stem and a verb suffix defining a meaning of order for singular second person. It has the meaning similar to "put" as in the sentence "Put your gun down!".

Starting from the outmost element in the file, the *Document* element defines a natural language document and includes only one attribute, *DocumentId,* to identify the document uniquely. The second level element is the *Sentence* element. This element includes the *SentenceId* attribute to identify the sentence uniquely in the document.

The *Word* element is the third level element in the text corpora. It stores words of sentences in a document and uniquely identified by its *WordId* attribute. The second attribute for this element is the *Value* element which represents the word as in the form it was used in the document. As mentioned above, *Word* elements include *Morph* elements which include 4 attributes. The first one is the *MorhphId* attribute to identify the element inside its parent word. *Stem* is the second attribute in the element and stores the stem of the word produced by the morphological analyzer. Third attribute, *Type*, stores the word class

for this word. The last attribute is the *Suffix* attribute which contains the information on suffixes for that word produced by the morphological analyzer.

An important point about the text corpora file is the decision of adding punctuation marks into the document structure as parts-of-speech. It is obvious that relative or absolute positions of punctuation marks in sentences have considerable contributions on accuracy of system results. Using punctuation marks as word classes has the advantage of mentioned contributions without any modification requirement in format of the file.

A sample text corpora file used in TurPOS is given in Appendix B.

3.2.3 Tagger Module

The main part in the TurPOS tagger is the Tagger Module. As the module name suggests, all real work is done in this module. The Tagger Module is responsible for picking required inputs and process them to produce the most appropriate tagging results.

The basic algorithm of the tagging process can be expressed as:

- Getting the rule list from the Rule Parser
- Getting the document from the Stem Reader
- Tagging the document
- Returning the tagged output in a format of preference

Part of the Tagger Module code that implements the above algorithm is given in Figure 3.5.

```
Rules rules;                        //Stores the rule list
Document doc = new Document();      //Stores the document


//Try parsing the rule file
try
{
   RuleXmlParser ruleParser = new RuleXmlParser(RulesPathTextBox.Text);
   rules = ruleParser.ParseRules(RuleType.sözdizim);
}
catch (TurPOSException e1)
{
   MessageBox.Show(this, e1.Message, EXCEPTION_CAPTION);
   return;
}


//Try parsing the document file
try
{
   StemXmlReader stemReader = new StemXmlReader(StemsPathTextBox.Text);
   doc = stemReader.ReadDocument();
}
catch (Exception e2)
{
   MessageBox.Show(this, e2.Message, EXCEPTION_CAPTION);
   return;
}


//Do tagging
MainTagger tagger = new MainTagger(doc, rules);
tagger.Tag();


//Get tagged output
XmlDocument xmlDoc = tagger.Analyze();
```

Figure 3.5 Part of the Tagger Module code which implements the tagging process.

The Tagger Module decides the word class of a word using an accumulator model. The basic logic behind this model is to accumulate the value of a possible word class for a word for each use. The final decision is given according to the word class with the highest accumulator value.

The tagger processes each rule in the rule list against each sentence in the document. The sequence of word classes given in a rule, are compared to the sequences of sentence chunks with the number of words that is same as the number of items in this rule. For each word with a syntax that matches with a rule, an accumulator is incremented by one for the

word class defined in this rule. The accumulation process is explained step by step according to the given sample sentence below:

Boş teneke çok ses çıkartır. (Empty can rattles the most.)

Assuming we have a rule with a syntax "adjective + noun", the tagger processes the sentence in groups of two words. This is because the rule contains 2 items. The word groups and the order of this process is as the following:

1. boş teneke
2. teneke çok
3. çok ses
4. ses çıkartır
5. çıkartır .

The tagger investigates the possible word classes of words given above for each of the steps. In step 1, the word group "boş teneke" ("empty can" in English) matches with the grammatical sequence "adjective + noun". So the adjective word class accumulator value for "boş" and the noun word class accumulator value for "teneke" is incremented by one. The other word groups are also processed with this rule. All rules in the rule list are applied to whole sentences in the text corpora. After this process, the word classes with the highest accumulator values are picked as the result tags for each word.

Another important point with these steps is that the step 5 includes a word and a punctuation mark. As mentioned above, TurPOS takes punctuation marks into account as a syntax item to make more appropriate decisions.

A sample tagged output file of TurPOS is given in Appendix C.

3.2.4 User Interface Module

The User Interface Module is the TurPOS component which provides a Graphical User Interface with the functionalities of the system. The module includes the following options:

- Adding a new rule to the rule file,
- Loading different rule and text corpora files from the file system,
- Tagging the text corpora using the rule file,
- Displaying and saving the tagged output file.

Some views captured from the TurPOS User Interface is included in the Appendix D.

3.3 Performance Overview

This section includes the test results for each TurPOS modules separately and the overall performance of the system. The performance tests were established on a personal computer with AMD Athlon-XP 2600+ CPU and 704MB memory.

Measuring the time that is required to load rule files with different sizes is a reasonable way to test the performance of the Rule Parser Module. Since this is a once-in-runtime process, the only overhead of this module to the system would be as much as this measured time.

The Rule Parser tests were established using 3 different rule files including 10, 100 and 200 rules respectively. The average parsing times for those rule files are displayed in Table 3.3.

Table 3.3 TurPOS rule parser performance test results

Number of Rules	Average Parsing Time
10 rules	0.3271 milliseconds
100 rules	1.1916 milliseconds
200 rules	2.0696 milliseconds

The results show that the time required for loading rules is not directly proportional to the number of rules. The increment in the loading time decreases against the number of rules. Results show that loading 200 rules takes 2.0696 milliseconds, which is an acceptable time.

The Stem Reader Module is also a kind of special parser module like the Rule Parser. Stem Reader parses the complete text corpora once before the tagging process starts. To calculate the overhead of this module to the overall system performance, tests that are based on measurement of parsing documents with different sizes were established.

Table 3.4 shows the test results for the Stem Reader. According to the limited amount of analyzed text, tests were established on 3 different documents with 24, 105, 220 sentences each. The average parsing time for a document with 220 sentences is about 22 milliseconds.

Table 3.4 TurPOS stem reader performance test results

Total Number of Sentences	**Total Number of Words**	**Total Number of Morph Items**	**Average Parsing Time**
24	130	237	2,7038 milliseconds
105	561	1025	10,7154 milliseconds
220	1189	2172	22,9329 milliseconds

Table 3.5 shows the average tagging time and the average time for writing tagged output into files for various sized documents. The same documents were used for tests in Table 3.4 and Table 3.5 for a better investigation on performance results.

Table 3.5 TurPOS tagger performance test results

Total Number of Sentences	**Total Number of Words**	**Avg. Tagging Time**	**Avg. Time for Storing Results**
24	130	51,0734 milliseconds	4,5064 milliseconds
105	561	324,2662 milliseconds	6,6295 milliseconds
220	1189	802,2535 milliseconds	10,5151 milliseconds

3.4 Tagging Results

Some inflected verbs and some nouns are homonym in Turkish. This generally causes ambiguities in defining word classes of these words. TurPOS disambiguates word classes

in such situations, when the text corpora includes required information about these words. Two examples that are correctly tagged by TurPOS are as the following:

"Şair hissettiklerini yazar." (The poet writes his feelings) (1)

"Yazar olmak sorumluluk gerektirir." (Being a writer requires responsibility.) (2)

The word "yazar" in sentence (1) is tagged as a "verb", whereas the word in sentence (2) is tagged as a "noun" by the system.

In Turkish, some words like "bu, şu, o" ("this" and "that" may cover these words in English) are categorized as adjectives when a noun follows them. These words are categorized as pronouns when no nouns follow them. This word class ambiguity can only be resolved using the syntax. TurPOS resolves this ambiguity. Two examples are given below:

"O okulu sevdi." (He loved that school.) (3)

"O, okulu sevdi." (He loved the school.) (4)

In sentence (3), the word "o" is an adjective and it modifies the word "okul". Whereas in sentence (4), the comma is included between these two words. No commas may exist between an adjective and a noun which is modified by this adjective according to the Turkish grammar. So in sentence (4), the word "o" is a pronoun.

It also should be noted that genders of sentence subjects are not defined in Turkish. English descriptions of sentences (3) and (4) include "he" as the subject, although neither sentences have clue on gender of the subject. This also creates an ambiguity as far as the meaning of the given sentences are concerned.

24 randomly chosen Turkish sentences were taken from search engines to calculate the overall tagging results. Each sentence has at least one word with one or more possible word classes. The text corpora was produced by analyzing those sentences morphologically. Table 3.6 shows the accuracy achieved by TurPOS on this text corpora.

Table 3.6 TurPOS tagging results based on total words

Number of Total Words	**Number of Correctly Tagged Words**	**Number of Incorrectly Tagged Words**	**Accuracy**
130	120	10	92,3 %

The tagging results given in Table 3.6 are very encouraging, with an accuracy of 92,3%. Considering that tests were established using a limited rule list, the accuracy of the system can be incremented using a larger scale rule list, supported by linguists.

CHAPTER FOUR
CONCLUSIONS

4.1 Conclusions

Part-of-speech (POS) tagging is a sub-problem of Natural Language Processing. It is an important level of analysis in the research of understanding human language using computer programs. POS tagging has many uses in fields such as: full text searching, information retrieval, speech synthesis, pronunciation and the high level text analysis.

TurPOS is a rule-based POS tagger, with no built-in lexicon structure. Instead, it uses a text corpora which include possible word classes, possible morphemes and suffixes for each word that takes place in the document to be tagged. This feature eliminates the extra space requirement for the lexicon and the time requirement for looking up a word in it.

All input and output files used in TurPOS are based on the W3C XML standard. Beyond the mentioned advantages of format of the rule file in 3.2.1, this design decision also brings out the ease of communication with other systems (like the morphological analyzer) that supplement TurPOS tagger.

Several different approaches have been developed by computer scientists to achieve results with higher accuracy. Obviously, language characteristics play an important role in choosing the approach to be followed for tagging a natural language. A rule-based approach was used in this study.

The complexity of POS tagging problem has been proven by great amount of research on tagging of natural languages (Greene & Rubin, 1971; Jurafsky & Martin, 2000). Considering this fact, the tagging results of TurPOS are very encouraging in terms of accuracy and performance.

For improvements to the current system, recommendations and future works are given in Section 4.2.

4.2 Recommendations and Future Works

As mentioned in 3.2.2, TurPOS uses a text corpora produced by a morphological analyzer application. Therefore, the overall tagging results depend on the results of the morphological analyzer. The drawbacks of this dependency can be eliminated by improving the accuracy of the morphological analyzer that is used.

The success of the results of a rule-based parser is strongly tied to the amount and the quality of its rule list. To achieve higher accuracy on tagging results, the list of rules should be maintained continously, observing the effect of these modifications on results. Because of the lack of linguistic support on development process, TurPOS, currently has a limited number of rules in its rule list. For further accuracy on tagging results, the TurPOS rule list should be maintained and expanded adding more rules into it.

Current TurPOS contains only grammatical rules, whereas some taggers also contain rules that use morphological information to resolve ambiguities (Guilder, 1995). Such rules might also be included into TurPOS rule list for a better performance. As mentioned in section 3.2.2, the text corpora which contain the document to be tagged by TurPOS is produced by a morphological analyzer. Therefore, all words in this document contain stem and suffix details, which makes it possible to use rules using morphological information about the words to be tagged.

One of the main goals on developing TurPOS was to make the system portable enough to be used as a POS tagger for other natural languages. The first study to achieve this goal should be on other Turkic languages including Turkmen, Gagauz, Azerbaijani (Azeri), Kazakh, Kyrgyz, Tatar, Uighur, Uzbek and Gagauz. These languages share many grammatical features with Turkish and it would be much easier to adapt the system to work succesfully for them, as well.

One level further from this study, there should be a semantic analysis system to find out appropriate meanings of words of a document according to their context. TurPOS aims to disambiguate the word classes of words, but still some ambiguities may exist for some words. As given in 3.2.2, the word “koyun” may have 4 possible meanings. 3 of them are of noun types, whereas the other one is of verb type. Assuming that this part-of-speech

ambiguity has been resolved by TurPOS and the word has been tagged as a “noun”, there exists still 3 possibilities. This meaning ambiguity should also be resolved by another analysis system to decide the word’s appropriate meaning. A successful semantic analysis system built on the results of this study can make the use of natural languages in daily life much more possible.

REFERENCES

Aktaş, Ö. (2006). *Türkçe için Verimli bir Cümle Sonu Belirleme Yöntemi*. Akademik Bilişim 2006 / Bilgi Teknolojileri Kongresi IV, Pamukkale Üniversitesi, 9-11 Şubat.

Antworth, E. L. (1990). *PC-KIMMO: A Two-level Processor for Morphological Analysis*. Summer Institute of Linguistics, Dallas, TX.

Booth, A. D., L Brandwood & J. P. Cleave. 1958. *Mechanical resolution of linguistic problems.* London: Butterworths Scientific Publications.

Brill E. (1992). *A Simple Rule-Based Part of Speech Tagger*. In "Proceedings of the 3rd Conference on Applied Natural Language Processing", Trento, Italy, pp. 152-155.

Brill E. (1994). *Some Advances in Transformation Based Part of Speech Tagging*. In "Proceedings of the Twelfth International Conference on Artificial Intelligence" (AAAI-94), Seattle, WA.

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.

Charniak, E. (1993). *Statistical Language Learning*. New Haven, CT: The MIT Press.

Church, K. W. (1988). *A stochastic parts program and noun phrase parser for unrestricted text*. In Second Conference on Applied Natural Language Processing, pp. 136-143. ACL.

Cutting, D., Kupiec, J., Pedersen, J. O. & Sibun, P. (1992). *A practical part-of-speech tagger*. In Third Conference on Applied Natural Language Processing, pp. 133-140. ACL.

Çebi, Y. & Varlıklar, Ö. (2005). *Türkçe Derlem Oluşturmada Karşılaşılan Sorunlar ve Çözüm Önerileri.* Bilgisayar Destekli Dil Bilimi Çalıştayı Bildirileri, 14 Mayıs 2005. Ankara: TDK Yayınları.

DeRose, S. J. (1988). *Grammatical category disambiguation by statistical optimization*. Computational Linguistics, 14, 31-39.

Garside R., Leech G. & Sampson G. (1987). *The Computational Analysis of English: a corpus-based approach*. Longman Group UK Limited.

Garside, R., Leech, G. & McEnery, A. (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. Addison Wesley Longman Inc., New York.

Greene B. B. & Rubin G. M. (1971). *Automatic Grammatical Tagging of English*. Department of Linguistics, Brown University, Providence, R.I.

Guilder, L. V. (1995). *Automated Part of Speech Tagging: A Brief Overview*. Retrieved January 25, 2007, from http://www.georgetown.edu/faculty/ballc/ling361/tagging_overview.html.

Jurafsky, D., Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey: Prentice-Hall.

Karlsson, F., Voutilainen, A., Heikkilä, J., & Anttila, A. (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Berlin: Mouton de Gruyter.

Manning, C. D. & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: The MIT Press.

Oflazer, K. & Kuruoz, I. (1994). *Tagging and morphological disambiguation of Turkish text*. In Fourth Conference on Applied Natural Language Processing, pp. 144-149.

Tapanainen, P. & Voutilainen, A. (1994). *Tagging accurately - Don't guess if you know*. In "Proceedings of the Fourth ACL Conference on Applied Natural Language Processing", Stuttgart.

TDK Güncel Türkçe Sözlük (n.d.). Retrieved April 19, 2007, from http://www.tdk.org.tr.

Voutilainen, A. (1995). *Morphological disambiguation*. In Karlsson, F., Voutilainen, A., Heikkilä, J., and Anttila, A. (Eds.), *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*, pp. 165-284. Berlin: Mouton de Gruyter.

APPENDIX A
SAMPLE RULE FILE

This chapter includes the contents of a sample rule file prepared for use in TurPOS to inform the reader about the implementation details.

Rules.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Rule RuleId="1" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
  </Rule>
  <Rule RuleId="2" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
  </Rule>
  <Rule RuleId="3" RuleType="sözdizim" RuleState="true">
    <Item ItemType="zarf" />
  </Rule>
  <Rule RuleId="4" RuleType="sözdizim" RuleState="true">
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="5" RuleType="sözdizim" RuleState="true">
    <Item ItemType="virgül" />
  </Rule>
  <Rule RuleId="6" RuleType="sözdizim" RuleState="true">
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="7" RuleType="sözdizim" RuleState="true">
    <Item ItemType="zamir" />
  </Rule>
  <Rule RuleId="8" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
    <Item ItemType="sıfat" />
    <Item ItemType="isim" />
  </Rule>
  <Rule RuleId="9" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
    <Item ItemType="isim" />
  </Rule>
  <Rule RuleId="10" RuleType="sözdizim" RuleState="false">
    <Item ItemType="sıfat" />
    <Item ItemType="virgül" />
    <Item ItemType="isim" />
  </Rule>
  <Rule RuleId="11" RuleType="sözdizim" RuleState="true">
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="12" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="13" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="14" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
  </Rule>
```

Rules.xml (cont'd)

```
<Rule RuleId="15" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
</Rule>
<Rule RuleId="16" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="isim" />
</Rule>
<Rule RuleId="17" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="edat" />
</Rule>
<Rule RuleId="18" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="19" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="edat" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="20" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="21" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="22" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="23" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zamir" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="24" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zamir" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="25" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="sıfat" />
  <Item ItemType="isim" />
</Rule>
<Rule RuleId="26" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="sıfat" />
</Rule>
<Rule RuleId="27" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="zarf" />
</Rule>
<Rule RuleId="28" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
</Rule>
```

Rules.xml (cont'd)

```
<Rule RuleId="29" RuleType="sözdizim" RuleState="false">
  <Item ItemType="sıfat" />
  <Item ItemType="virgül" />
  <Item ItemType="sıfat" />
</Rule>
<Rule RuleId="30" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zarf" />
  <Item ItemType="sıfat" />
  <Item ItemType="isim" />
</Rule>
<Rule RuleId="31" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
  <Item ItemType="nokta" />
</Rule>
<Rule RuleId="32" RuleType="sözdizim" RuleState="true">
  <Item ItemType="sıfat" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
  <Item ItemType="nokta" />
</Rule>
<Rule RuleId="33" RuleType="sözdizim" RuleState="false">
  <Item ItemType="sıfat" />
  <Item ItemType="ikinokta" />
  <Item ItemType="isim" />
</Rule>
<Rule RuleId="34" RuleType="sözdizim" RuleState="false">
  <Item ItemType="sıfat" />
  <Item ItemType="noktalıvirgül" />
  <Item ItemType="isim" />
</Rule>
<Rule RuleId="35" RuleType="sözdizim" RuleState="false">
  <Item ItemType="sıfat" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="36" RuleType="sözdizim" RuleState="true">
  <Item ItemType="fiil" />
  <Item ItemType="isim" />
  <Item ItemType="sıfat" />
  <Item ItemType="isim" />
  <Item ItemType="nokta" />
</Rule>
<Rule RuleId="37" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
  <Item ItemType="nokta" />
</Rule>
<Rule RuleId="38" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="39" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
  <Item ItemType="nokta" />
</Rule>
<Rule RuleId="40" RuleType="sözdizim" RuleState="true">
  <Item ItemType="edat" />
</Rule>
<Rule RuleId="41" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
```

Rules.xml (cont'd)

```
    <Item ItemType="isim" />
    <Item ItemType="edat" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="42" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="edat" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="43" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="44" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="45" RuleType="sözdizim" RuleState="false">
    <Item ItemType="zarf" />
    <Item ItemType="virgül" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="46" RuleType="sözdizim" RuleState="false">
    <Item ItemType="edat" />
    <Item ItemType="virgül" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="47" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="48" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="49" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="50" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="edat" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="51" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
```

Rules.xml (cont'd)

```
    <Item ItemType="isim" />
    <Item ItemType="edat" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="52" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="53" RuleType="" RuleState="true">
    <Item ItemType="zamir" />
    <Item ItemType="virgül" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="54" RuleType="" RuleState="true">
    <Item ItemType="zamir" />
    <Item ItemType="virgül" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="55" RuleType="" RuleState="true">
    <Item ItemType="fiil" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="56" RuleType="" RuleState="true">
    <Item ItemType="zamir" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
  <Rule RuleId="57" RuleType="" RuleState="true">
    <Item ItemType="zamir" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="58" RuleType="" RuleState="true">
    <Item ItemType="zamir" />
    <Item ItemType="edat" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="59" RuleType="" RuleState="true">
    <Item ItemType="zamir" />
    <Item ItemType="edat" />
    <Item ItemType="fiil" />
    <Item ItemType="nokta" />
  </Rule>
 </Document>
```

APPENDIX B
SAMPLE TEXT CORPORA

This chapter includes the contents of a sample text corpora file produced by a morphological analyzer.

Stems1.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Document DocumentId = "0">

  <Sentence SentenceId = "0">
    <Word WordId = "0" Value = "Büyük">
      <Morph MorphId = "0" Stem="büyük" Type ="sıfat" Suffix
="ISIM_YALIN_BOS"></Morph>
    </Word>
    <Word WordId = "1" Value = "kapı">
      <Morph MorphId = "0" Stem="kapı" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
      <Morph MorphId = "1" Stem="kap" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_TAMLAMA_I"></Morph>
      <Morph MorphId = "2" Stem="kap" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_O_I"></Morph>
      <Morph MorphId = "3" Stem="kap" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_BELIRTME_I"></Morph>
    </Word>
    <Word WordId = "2" Value = "kapalıydı">
      <Morph MorphId = "0" Stem="kapalı" Type ="fiil" Suffix ="ISIM_YALIN_BOS +
IMEK_HIKAYE_DI"></Morph>
    </Word>
    <Word WordId = "3" Value = ".">
      <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
    </Word>
  </Sentence>

  <Sentence SentenceId = "1">
    <Word WordId = "0" Value = "Koyun">
      <Morph MorphId = "0" Stem="koyun" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
      <Morph MorphId = "1" Stem="koy" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_TAMLAMA_IN"></Morph>
      <Morph MorphId = "2" Stem="koy" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_SEN_IN"></Morph>
      <Morph MorphId = "3" Stem="koy" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_EMIR_SIZ_IN"></Morph>
```

Stems1.xml (cont'd)

```
    </Word>
    <Word WordId = "1" Value = "masaya">
      <Morph MorphId = "0" Stem="masa" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_YONELME_E"></Morph>
    </Word>
    <Word WordId = "2" Value = "çıktı">
      <Morph MorphId = "0" Stem="çık" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_GECMISZAMAN_DI"></Morph>
      <Morph MorphId = "1" Stem="çıktı" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
    </Word>
    <Word WordId = "3" Value = ".">
      <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
    </Word>
  </Sentence>

  <Sentence SentenceId = "2">
    <Word WordId = "0" Value = "Koyun">
      <Morph MorphId = "0" Stem="koyun" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
      <Morph MorphId = "1" Stem="koy" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_TAMLAMA_IN"></Morph>
      <Morph MorphId = "2" Stem="koy" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_SEN_IN"></Morph>
      <Morph MorphId = "3" Stem="koy" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_EMIR_SIZ_IN"></Morph>
    </Word>
    <Word WordId = "1" Value = "masaya">
      <Morph MorphId = "0" Stem="masa" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_YONELME_E"></Morph>
    </Word>
    <Word WordId = "2" Value = "bu">
      <Morph MorphId = "0" Stem="bu" Type ="sıfat" Suffix
="ISIM_YALIN_BOS"></Morph>
      <Morph MorphId = "1" Stem="bu" Type ="zamir" Suffix
="ZAMIR_YALIN_BOS"></Morph>
    </Word>
    <Word WordId = "3" Value = "tabağı">
      <Morph MorphId = "0" Stem="tabak" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_O_I"></Morph>
      <Morph MorphId = "1" Stem="tabak" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_BELIRTME_I"></Morph>
      <Morph MorphId = "2" Stem="tabak" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_TAMLAMA_I"></Morph>
```

Stems1.xml (cont'd)

```
    </Word>
    <Word WordId = "4" Value = ".">
      <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
    </Word>
  </Sentence>

  <Sentence SentenceId = "3">
    <Word WordId = "0" Value = "O">
      <Morph MorphId = "0" Stem="o" Type ="sıfat" Suffix
="ISIM_YALIN_BOS"></Morph>
      <Morph MorphId = "1" Stem="o" Type ="zamir" Suffix
="ZAMIR_YALIN_BOS"></Morph>
    </Word>
    <Word WordId = "1" Value = ",">
      <Morph MorphId = "0" Stem="," Type ="virgül" Suffix =""></Morph>
    </Word>
    <Word WordId = "2" Value = "artık">
      <Morph MorphId = "0" Stem="artık" Type ="zarf" Suffix
="ZARF_YALIN_BOS"></Morph>
      <Morph MorphId = "1" Stem="artık" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
    </Word>
    <Word WordId = "3" Value = "büyüdü">
      <Morph MorphId = "0" Stem="büyü" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_GECMISZAMAN_DI"></Morph>
    </Word>
    <Word WordId = "3" Value = ".">
      <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
    </Word>
  </Sentence>

  <Sentence SentenceId = "4">
   <Word WordId = "0" Value = "Kaç">
     <Morph MorphId = "0" Stem="kaç" Type ="fiil" Suffix
="FIIL_YALIN_BOS"></Morph>
     <Morph MorphId = "1" Stem="kaç" Type ="sıfat" Suffix
="ISIM_YALIN_BOS"></Morph>
    </Word>
   <Word WordId = "1" Value = "kişi">
     <Morph MorphId = "0" Stem="kişi" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
    </Word>
   <Word WordId = "2" Value = "kaldık">
```

Stems1.xml (cont'd)

```
     <Morph MorphId = "0" Stem="kal" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_BELIRTME_DIK"></Morph>
     <Morph MorphId = "1" Stem="kal" Type ="isim" Suffix ="ISIM_YALIN_BOS +
IMEK_HIKAYE_DI + IMEK_BIZ_K"></Morph>
    </Word>

   <Word WordId = "3" Value = "?">
     <Morph MorphId = "0" Stem="?" Type ="soru" Suffix =""></Morph>
   </Word>
  </Sentence>

  <Sentence SentenceId = "5">
   <Word WordId = "0" Value = "Tavşan">
     <Morph MorphId = "0" Stem="tavşan" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
   </Word>
   <Word WordId = "1" Value = "kaç">
     <Morph MorphId = "0" Stem="kaç" Type ="fiil" Suffix
="FIIL_YALIN_BOS"></Morph>
     <Morph MorphId = "1" Stem="kaç" Type ="sıfat" Suffix
="ISIM_YALIN_BOS"></Morph>
   </Word>
   <Word WordId = "2" Value = ",">
     <Morph MorphId = "0" Stem="," Type ="virgül" Suffix =""></Morph>
   </Word>
   <Word WordId = "3" Value = "tazı">
     <Morph MorphId = "0" Stem="tazı" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
   </Word>
   <Word WordId = "4" Value = "tut">
     <Morph MorphId = "0" Stem="tut" Type ="fiil" Suffix
="FIIL_YALIN_BOS"></Morph>
    </Word>
   <Word WordId = "5" Value = ".">
     <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
   </Word>
  </Sentence>

  <Sentence SentenceId = "6">
   <Word WordId = "0" Value = "Sayın">
     <Morph MorphId = "0" Stem="sayın" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
     <Morph MorphId = "1" Stem="sayı" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_SEN_IN"></Morph>
```

Stems1.xml (cont'd)

```
     <Morph MorphId = "2" Stem="say" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_EMIR_SIZ_IN"></Morph>
     <Morph MorphId = "3" Stem="say" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_SEN_IN"></Morph>
     <Morph MorphId = "4" Stem="say" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_TAMLAMA_IN"></Morph>
   </Word>
   <Word WordId = "1" Value = "Kutlu">
     <Morph MorphId = "0" Stem="kutlu" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
     <Morph MorphId = "1" Stem="kut" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_BULUNMA_LI"></Morph>
   </Word>
   <Word WordId = "2" Value = "istifasını">
     <Morph MorphId = "0" Stem="istifa" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_O_I + ISIM_BELIRTME_I"></Morph>
   </Word>

   <Word WordId = "3" Value = "açıkladı">
     <Morph MorphId = "0" Stem="açıkla" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_GECMISZAMAN_DI"></Morph>
   </Word>
   <Word WordId = "4" Value = ".">
     <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
   </Word>
  </Sentence>

  <Sentence SentenceId = "7">
   <Word WordId = "0" Value = "Büyüklerinizi">
     <Morph MorphId = "0" Stem="büyük" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_COGUL_LER + ISIM_SAHIPLIK_SIZ_INIZ + ISIM_BELIRTME_I"></Morph>
   </Word>
   <Word WordId = "1" Value = "sayın">
     <Morph MorphId = "0" Stem="sayın" Type ="isim" Suffix
="ISIM_YALIN_BOS"></Morph>
     <Morph MorphId = "1" Stem="sayı" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_SEN_IN"></Morph>
     <Morph MorphId = "2" Stem="say" Type ="fiil" Suffix ="FIIL_YALIN_BOS +
FIIL_EMIR_SIZ_IN"></Morph>
     <Morph MorphId = "3" Stem="say" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_SAHIPLIK_SEN_IN"></Morph>
     <Morph MorphId = "4" Stem="say" Type ="isim" Suffix ="ISIM_YALIN_BOS +
ISIM_TAMLAMA_IN"></Morph>
   </Word>
```

Stems1.xml (cont'd)

```
   <Word WordId = "2" Value = ".">
     <Morph MorphId = "0" Stem="." Type ="nokta" Suffix =""></Morph>
   </Word>
  </Sentence>
</Document>
```

APPENDIX C
SAMPLE TAGGED DOCUMENT

This chapter includes the contents of a sample tagged output document produced by TurPOS.

TaggedOutput1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Document DocumentId="0">
   <Sentence SentenceId="0">
      <Word WordId="0" Value="Büyük">
         <Type Name="sıfat" />
      </Word>
      <Word WordId="1" Value="kapı">
         <Type Name="isim" />
      </Word>
      <Word WordId="2" Value="kapalıydı">
         <Type Name="fiil" />
      </Word>
      <Word WordId="3" Value=".">
         <Type Name="nokta" />
      </Word>
   </Sentence>

   <Sentence SentenceId="1">
      <Word WordId="0" Value="Koyun">
         <Type Name="isim" />
      </Word>
      <Word WordId="1" Value="masaya">
         <Type Name="isim" />
      </Word>
      <Word WordId="2" Value="çıktı">
         <Type Name="fiil" />
      </Word>
      <Word WordId="3" Value=".">
         <Type Name="nokta" />
      </Word>
   </Sentence>

   <Sentence SentenceId="2">
      <Word WordId="0" Value="Tüm">
         <Type Name="sıfat" />
      </Word>
```

TaggedOutput1.xml (cont'd)

```
      <Word WordId="1" Value="yeni">
         <Type Name="isim" />
      </Word>
      <Word WordId="2" Value="kitapları">
         <Type Name="isim" />
      </Word>
      <Word WordId="3" Value="okurlar">
         <Type Name="fiil" />
      </Word>
      <Word WordId="4" Value=".">
         <Type Name="nokta" />
      </Word>
   </Sentence>

   <Sentence SentenceId="3">
      <Word WordId="0" Value="Konuklar">
         <Type Name="fiil" />
         <Type Name="isim" />
      </Word>
      <Word WordId="1" Value="bu">
         <Type Name="sıfat" />
      </Word>
      <Word WordId="2" Value="akşam">
         <Type Name="isim" />
      </Word>
      <Word WordId="3" Value="yemeğe">
         <Type Name="isim" />
      </Word>
      <Word WordId="4" Value="gelecek">
         <Type Name="fiil" />
      </Word>
      <Word WordId="5" Value=".">
         <Type Name="nokta" />
      </Word>
   </Sentence>

   <Sentence SentenceId="4">
      <Word WordId="0" Value="Gelecek">
         <Type Name="isim" />
      </Word>
      <Word WordId="1" Value="yılın">
         <Type Name="isim" />
      </Word>
      <Word WordId="2" Value="müfredatı">
```

TaggedOutput1.xml (cont'd)

```
      <Type Name="isim" />
    </Word>
    <Word WordId="3" Value="hazırlandı">
      <Type Name="fiil" />
    </Word>
    <Word WordId="4" Value=".">
      <Type Name="nokta" />
    </Word>
  </Sentence>

  <Sentence SentenceId="5">
    <Word WordId="0" Value="Yazar">
      <Type Name="isim" />
    </Word>
    <Word WordId="1" Value="olmak">
      <Type Name="fiil" />
    </Word>
    <Word WordId="2" Value="sorumluluk">
      <Type Name="isim" />
    </Word>
    <Word WordId="3" Value="gerektirir">
      <Type Name="fiil" />
    </Word>
    <Word WordId="4" Value=".">
      <Type Name="nokta" />
    </Word>
  </Sentence>

  <Sentence SentenceId="6">
    <Word WordId="0" Value="Şair">
      <Type Name="isim" />
    </Word>
    <Word WordId="1" Value="hissettiklerini">
      <Type Name="fiil" />
    </Word>
    <Word WordId="2" Value="yazar">
      <Type Name="fiil" />
    </Word>
    <Word WordId="3" Value=".">
      <Type Name="nokta" />
    </Word>
  </Sentence>

  <Sentence SentenceId="7">
    <Word WordId="0" Value="O">
```

TaggedOutput1.xml (cont'd)

```
        <Type Name="sıfat" />
      </Word>
      <Word WordId="1" Value="okulu">
        <Type Name="isim" />
      </Word>
      <Word WordId="2" Value="sevdi">
        <Type Name="fiil" />
      </Word>
      <Word WordId="3" Value=".">
        <Type Name="nokta" />
      </Word>
    </Sentence>
```

APPENDIX D
TURPOS USER INTERFACE

This chapter includes some screenshots captured during the TurPOS run-time. Each figure contains a brief explanation about the screen.

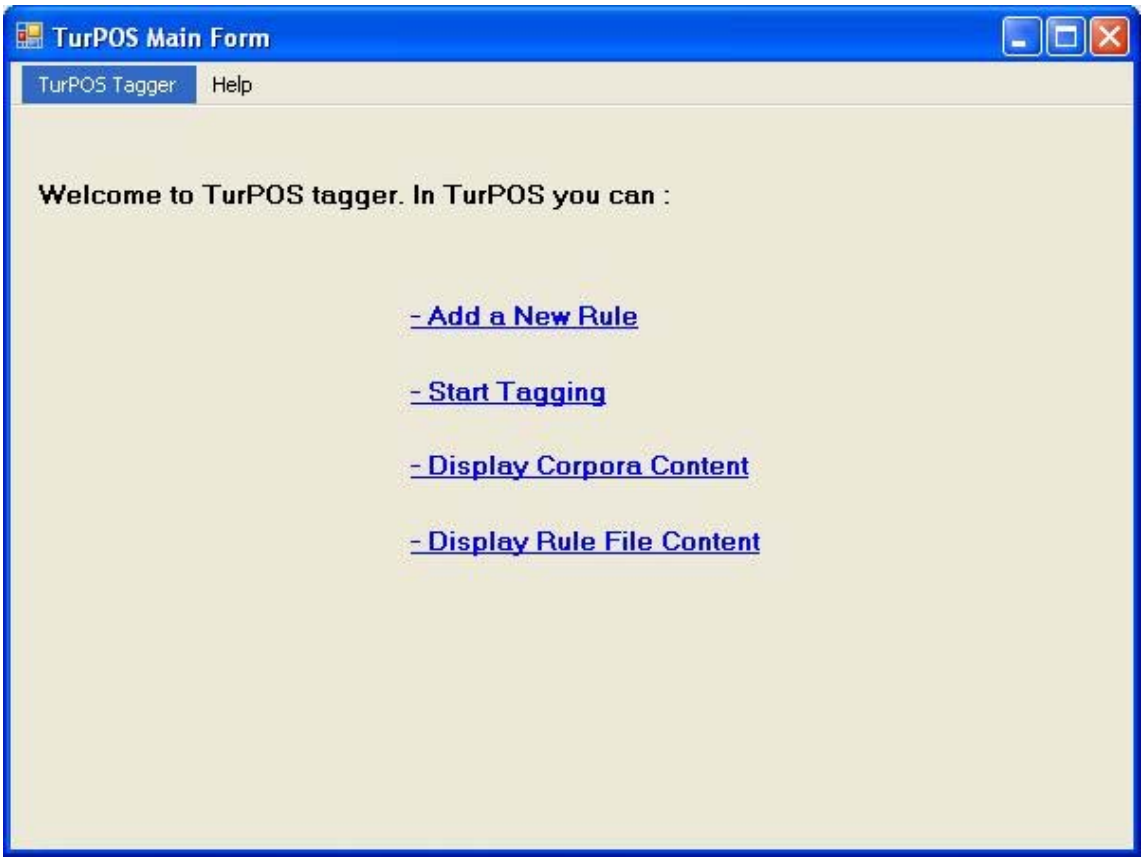

Figure 1 TurPOS main screen: Entry point for TurPOS features.

Figure 1 shows the TurPOS “Main Form” screen. This page currently includes 4 options as listed in the figure. This is the netry point for the application.

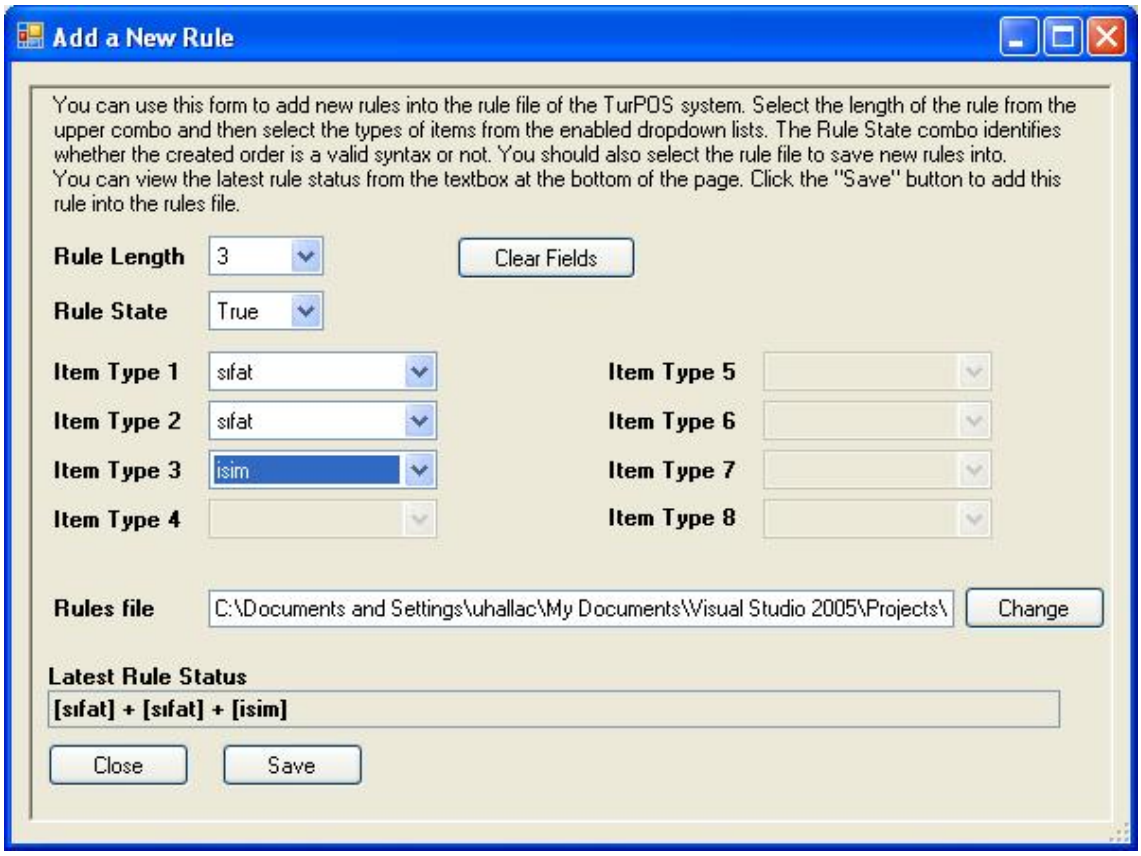

Figure 2 Add a new rule screen: Adding a new rule to TurPOS rule file.

Figure 2 shows the "Add a New Rule" screen of TurPOS. Using this page, one can add a new rule into TurPOS rule file. The length of the rule defines the number of items in this new rule. As seen in the picture, a length of 3 enables 3 item type comboboxes in the page.

As mentioned in section 3.2.1, the rule state defines whether the word class sequence given in this rule is legitimate for the grammar of the natural language, or not. According to the rule in Figure 2, the word classes can be ordered as "sıfat + sıfat + isim" ("adjective + adjective + noun").

The page allows a maximum length of 8 items. So, there are 8 item type comboboxes that can be filled in by the user. These comboboxes include the word classes defined in Table 3.1.

The "Rules File" textbox contains the path of the rule file that this new rule will be appended into. The "Latest Rule Status" textbox displays the final order of word classes defined by this rule.

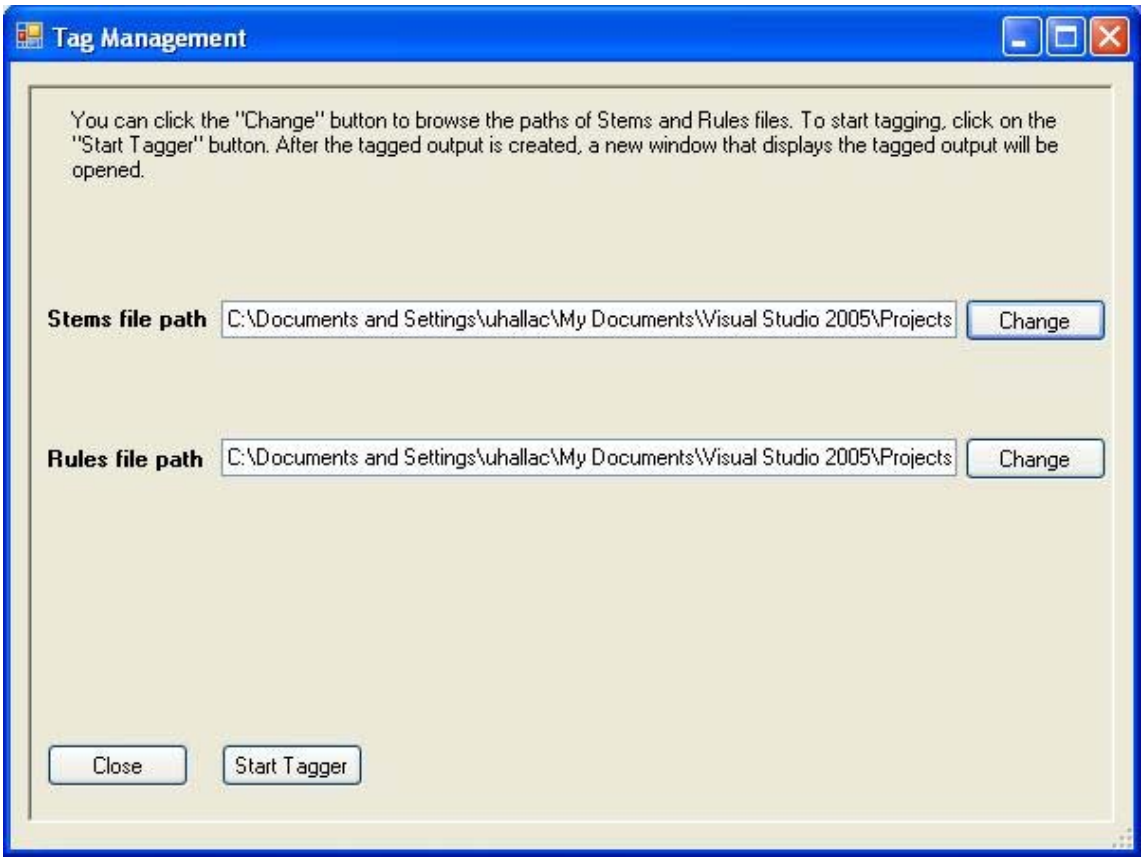

Figure 3 Tag management screen: Choosing the paths for text corpora and rule list file before tagging.

Figure 3 shows the TurPOS "Tag Management" screen. This page lets user to choose the rule file and the text corpora file to be used, just before starting the tagging process.

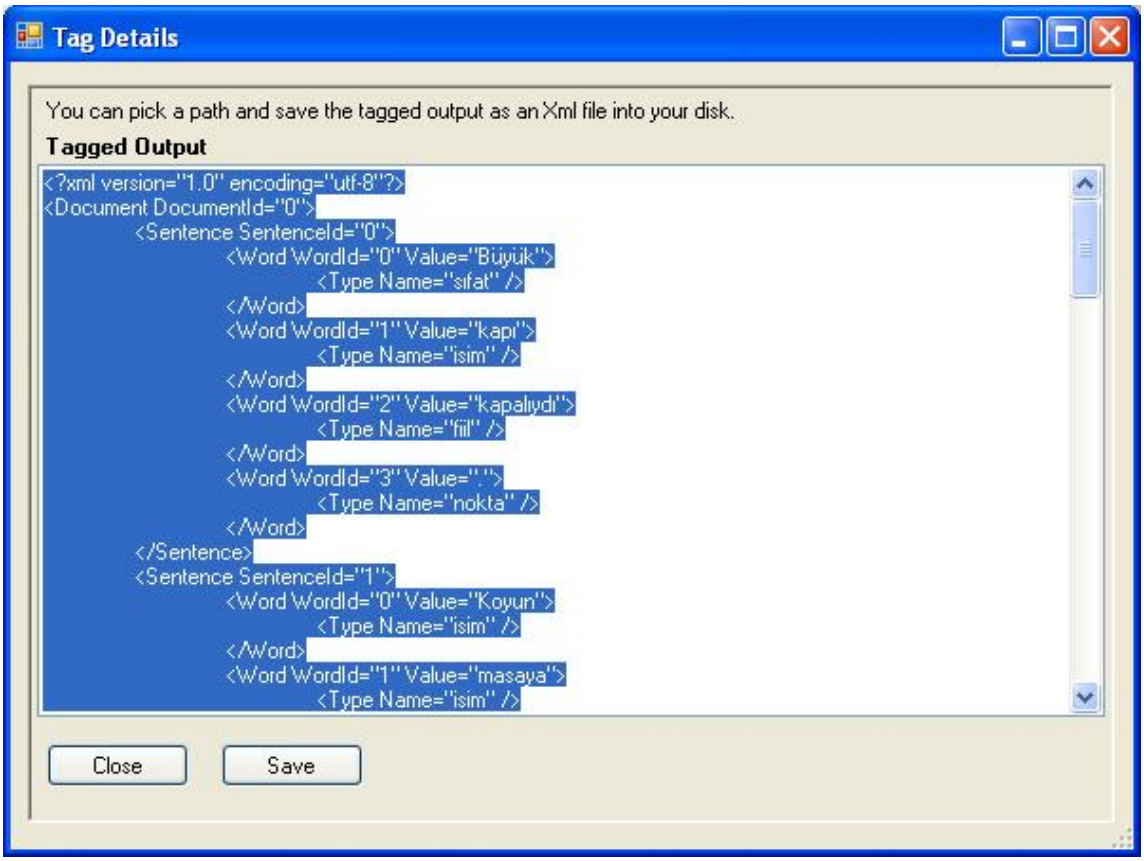

Figure 4 Tag details screen: Displaying the tagged output of the document.

Clicking the “Start Tagger” on the “Tag Management” page in Figure 3 starts the tagging process with the given input files and then the tagged output is produced. The tagged document output is displayed in the “Tag Details” page, which is shown in Figure 4. In this page, user has the opportunity to save the tagged output into an XML file.

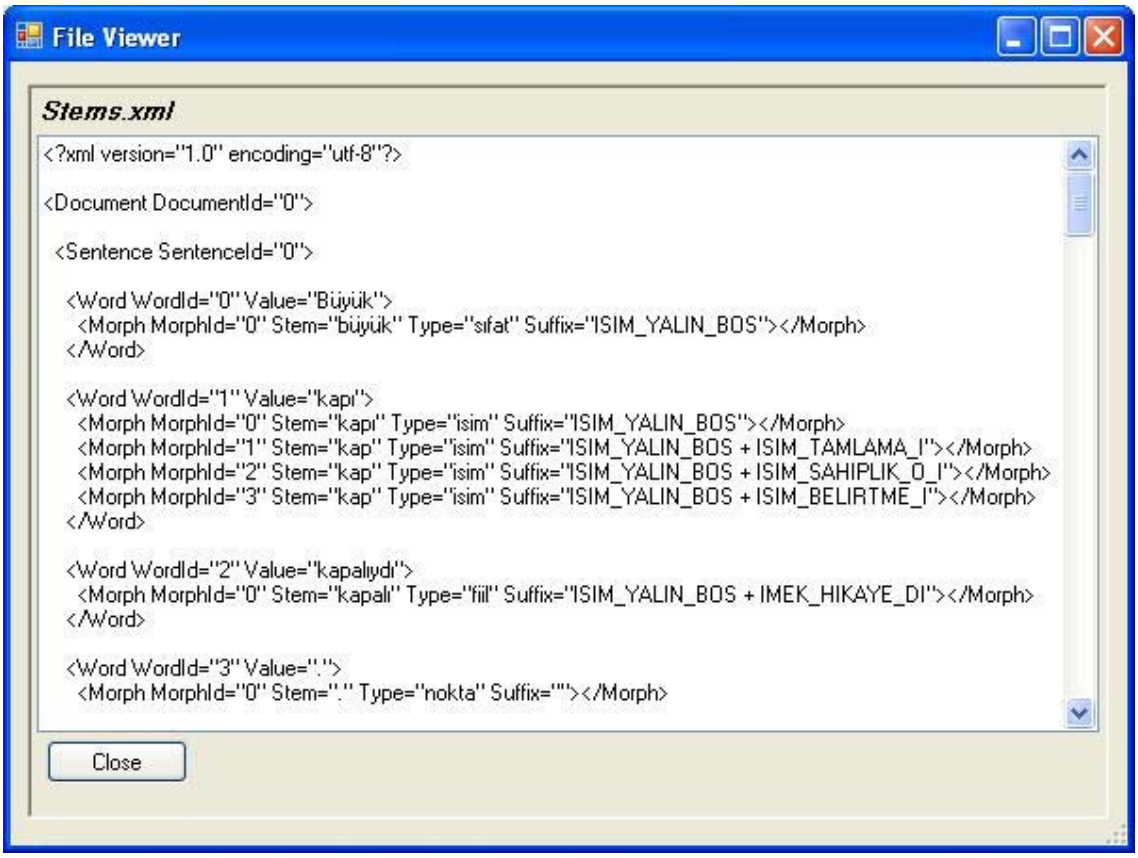

Figure 5 File viewer screen: Displaying the corpora content.

Figure 5 shows the “File Viewer” screen which displays the contents of the text corpora that will be tagged.

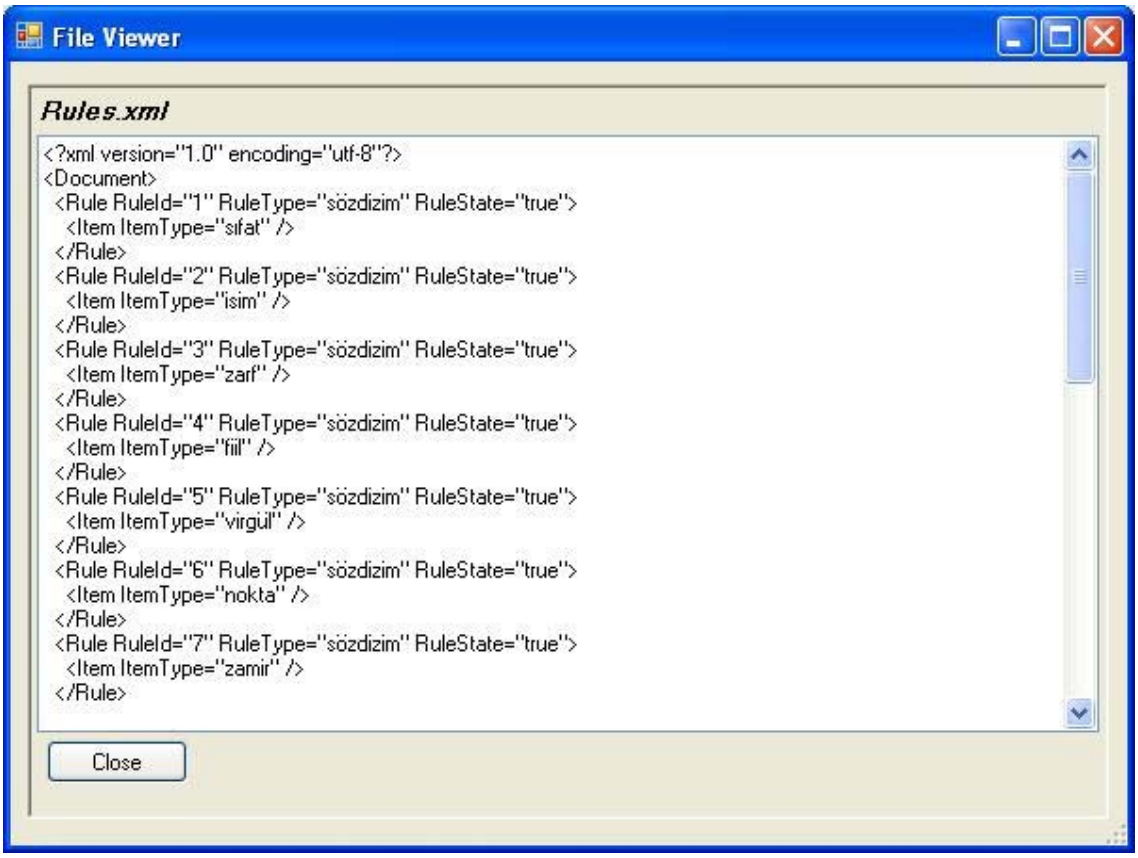

Figure 6 File viewer screen: Displaying the rule file.

Figure 6 also shows the “File Viewer” screen which displays the content of the rule file that will be used in the tagging process.