DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

RULE-BASED NATURAL LANGUAGE PROCESSING METHODS FOR TURKISH

by

Özlem AKTAŞ

September, 2010
İZMİR

RULE-BASED NATURAL LANGUAGE PROCESSING METHODS FOR TURKISH

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 Doctor of Philosophy in Computer Engineering

by

Özlem AKTAŞ

September, 2010
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"RULE-BASED NATURAL LANGUAGE PROCESSING METHODS FOR TURKISH"** completed by **ÖZLEM AKTAŞ** under supervision of **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 Doctor of Philosophy.

……………………………………
Prof. Dr. Yalçın ÇEBİ

Supervisor

…………………………………… Prof. Dr. Alp KUT	………………………………….. Prof. Dr. Gürer GÜLSEVİN
Thesis Committee Member	Thesis Committee Member
…………………………………… Asst.Prof.Dr. Banu DİRİ	………………………………….. Asst.Prof.Dr. Adil ALPKOÇAK
Examining Committee Member	Examining Committee Member

Prof. Dr. Mustafa SABUNCU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank to my advisor Professor Dr. Yalçın ÇEBİ, thesis tracking committee members Professor Dr. R.Alp KUT and Professor Dr. Gürer GÜLSEVİN, and also my friends and colleagues, Instructor Dr. Kökten Ulaş BİRANT, Research Assistant Emel ALKIM, Research Assistant Çağdaş Can BİRANT, and linguists, Instructor Dr. Özden FİDAN and Research Assistant Dr. Özgün KOŞANER, in Dokuz Eylül University Natural Language Processing Research Group for contribution to this study and sharing their ideas during the development and writing phases of the thesis.

I would also like to thank to Specialist Belgin AKSU and Turkish Linguistic Association (Türk Dil Kurumu, TDK) for their contribution and support to this study.

The infrastructure of this work is supported by Dokuz Eylul University Scientific Research Projects (Bilimsel Araştırma Projeleri, BAP) Coordination Unit, numbered as 2007-KB-FEN-043.

I have special thanks to my parents and my husband Cenk AKTAŞ for their support, patience and making me encouraged during the development and writing phase of the thesis.

Özlem AKTAŞ

RULE-BASED NATURAL LANGUAGE PROCESSING METHODS FOR TURKISH

ABSTRACT

In order to determine morphological properties of a language, a corpus which represents that language should be created. Many large scale corpora generated and have been used for Natural Language Processing (NLP) applications on many languages, such as English, German, Czech, etc, but any large scale Turkish corpora have not be generated yet.

In this study, natural language processing methods for Turkish were developed by using rule-based approach, and also an infrastructure, *Rule-Based Automatical Corpus Generation (RB-CorGen)*, to use the new developed methods was implemented. For testing RB-CorGen on Turkish, the roots, stems and suffixes were obtained from Turkish Linguistic Association (Türk Dil Kurumu, TDK) and Dokuz Eylul University, College of Literature Linguistic Department, the defined tags and grammatical rules were stored in XML formatted file, and documents, include nearly 95 million wordforms, were collected from five Turkish newspapers in electronic environment. The average success rates of Rule-Based Sentence Boundary Detection (RB-SBD) and Rule-Based POS Tagging (RB-POST) methods were determined as 99.66% and 92% respectively. It was seen that the success rate of RB-CorGen increases with the increasing number of rules.

Keywords: Turkish, Corpus, Rule-based, Sentence Boundary Detection, Morphological Analyzer, Part of Speech Tagger.

TÜRKÇE İÇİN KURAL-TABANLI DOĞAL DİL İŞLEME YÖNTEMLERİ

ÖZ

Dillerin biçimbilimsel özelliklerinin belirlenmesi için, dilin özelliklerini temsil edebilecek bir derlem gereklidir. İngilizce, Almanca, Çekçe gibi birçok dil için büyük ölçekli derlemler geliştirilmekte ve Doğal Dil İşleme (DDİ) alanlarında kullanılmaktadır, ancak, büyük ölçekli bir Türkçe derlem henüz geliştirilmemiştir.

Bu çalışmada kural-tabanlı bir yaklaşım kullanılarak Türkçe için Doğal Dil İşleme yöntemleri geliştirilmiş ve yöntemleri gerçekleştirmek için Kural-Tabanlı Otomatik Derlem Oluşturma (en.: Rule-Based Automatically Corpus Generation (RB-CorGen)) adında bir altyapı oluşturulmuştur. RB-CorGen uygulamasını Türkçe üzerinde test etmek amacıyla, elektronik ortamda bulunan gazetelerden yaklaşık 95 milyon kelimelik köşe yazıları derlenmiş, Türkçe kökler, gövdeler ve ekler, Türk Dil Kurumu (TDK) ve Dokuz Eylül Üniversitesi Edebiyat Fakültesi Dilbilim Bölümü'nden temin edilmiş, etiketler ve dilbilgisi kuralları da dilbilimi uzmanları tarafından oluşturularak XML yapısında kaydedilmiştir. Kural-Tabanlı Cümle Sonu Belirleme (RB-SBDT) ve Kural-Tabanlı Kelime Türü Belirleme (RB-POST) yöntemlerinin başarı oranları sırasıyla %99,66 ve %92 olarak belirlenmiştir. Oluşturulan kural sayısı arttıkça başarı oranlarının da arttığı gözlenmiştir.

Anahtar sözcükler: Türkçe, Derlem, Kural-Tabanlı, Cümle Sonu Belirleme, Biçimbilimsel Çözümleyici, Kelime Türü Etiketleyici.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Overview

Proportional to the tendency of continuous improvement of the computer technology during the last few decades, the computer applications and the way of the communication between people and computers are changing fast. The usage of computers has been increased exponentially in many areas in the daily life of people, such as “Communication”, “Data Transferring”, “Natural Language Processing (NLP)”, etc.

“Natural Language Processing (NLP)”, which is one of the application areas of computer technologies, can be defined as the construction of a computing system that processes and understands human natural language. The word “understand” means that the observable behavior of the system must make people assume that it is doing internally the same, or very similar, things that people do when they understand language (Güngördü, 1993). Basically, NLP aims to let computers to understand human’s natural language and even to let them to generate it.

In fact, the studies in NLP are almost old as the development of first computers. Many studies and methods on NLP application areas have been developed, and this field becomes more popular.

Generally, computers are used to process natural language to study in:

- Speech synthesis: The process of converting written text into machine-generated synthetic speech (Sagisaka et al., 1992; Black et al., 1994; Greenwood, 1997; Huang et al, 2001; Sak et al., 2006). A computer system used for converting written text to speech is called a *speech synthesizer*.
- Speech recognition: The process of converting a continuous signal to words (speech-to-text systems) (Zue et al., 2005).

- Automatic summarization: the creation of a shortened version of a text by a computer program (Mani, 2001).
- Natural language generation: The process of generating appropriate responses to any unpredictable inputs by making decisions about the words, word types, word order in the natural language by the system (Hennecke et al., 1997).
- Machine translation (MT): Machine translation (MT) was the first computer-based application related to natural language, which translates one NL into another (Booth, et. al., 1957; Coxhead, 2002).
- Optical character recognition (OCR): The translation of scanned images of handwritten, typewritten or printed text into a form that the computer can manipulate (for example, into ASCII codes) (What is optical character recognition?, (n.d.)).

Natural Language Processing consists of four main analysis levels where each level is strongly related to others: Morphology, Syntax, Semantics and Pragmatics.

Morphology is directly related to word based analysis, which aims to define the structure of words, such as investigation of word types (verb, noun, adjective, etc.), analyzing parts of the words (root, suffix or prefix). The results of morphological analysis are used for further processing in higher level analysis.

Syntactic analysis is generally based on sentences which are more complex components of natural languages than words, and used to determine the structure of sentences and occurrences of words. Syntactic analysis also uses statistics, which can be done in two ways; on letters and words. Letter analysis includes researches such as consonant and vowel letter placements, letter frequencies, relationship between letters such as letter positions according to each other, etc. Word analysis includes researches such as investigation of number of letters in a word, the order of the letters in a word, word frequencies, word orders in a sentence, etc.

Semantic analysis finds out the real structures of sentences and words by using meaning of structures obtained by syntactic analysis and meanings of the words used in the sentence.

Pragmatic analysis lies at the top level of analysis and is a much more complex study than the Morphology, Syntactic and Semantic Analysis. It aims to determine the meaning of discourse involving the contextual information.

In order to carry out NLP studies on any natural language, a representative corpus of that language is needed. There are many definitions about corpus; some of them are listed below:

- “Corpus is a collection of linguistic data, either written texts or a transcription of recorded speech, which can be used as a starting-point of linguistic description or as a means of verifying hypotheses about a language.” (Crystal, 1991).

- “A collection of naturally occurring language text, chosen to characterize a state or variety of a language.” (Sinclair, 1991).

A corpus must be large and representative of the language. A representative corpus has samples of every topic in the language, such as technical words, medicine, spoken language, etc.; large corpus has the large number of data taken from any topic of the language. Both corpora can be used in NLP applications. And also, corpora can be divided into two categories: “Balanced”, and “Unbalanced”. A “Balanced Corpus” is representative. It should include sample of texts from every topic in the language. This corpus should also include these texts in equal weights depending on the quantity of the usage in the language. Large corpus represents “Unbalanced Corpus”, which has large amount of data in one topic or different areas in the language. An unbalanced corpus may be turned into balanced by taking large amount of data from all topics in the language that makes the corpus a “representative” of the language. In fact, it is very difficult to take equal, small pieces of samples from different areas of a natural language into a corpus. Since unbalanced corpus consists of many words from any areas in a language, instead of creating a balanced corpus, an unbalanced corpus may be generated and used for better performance. Whether they are balanced or not, small sized corpora are good enough to carry out letter analysis on it. However, when word analysis is required, a large scale corpus is necessary. Especially to handle some extraordinary words, which are used rarely in the language, an unbalanced corpus is more powerful than the balanced corpus.

1.2 Aim of Thesis

Nowadays, large scale corpus is needed for every language to be able to make analysis on the language and get reliable results about the properties of it. While generating a large scale corpus, it is very important to determine sentences, also stem, root and suffixes of the words in a correct way. Although, large scale corpora have been generated and used for different languages, such as English, German, Czech, etc., large scale corpora for Turkish could not have been developed, yet.

The main goal of this study is to develop an infrastructure with rule-based approach to generate large scale Turkish corpus. This infrastructure can be adapted to any Turkish dialects by the given rules of the Turkish Dialect to be analyzed. During the studies carried out for this thesis, appropriate methods to find the sentences and wordforms in the text; root and suffixes of the words have been developed.

Considering the grammar and rule-based structure of Turkish, the rule-based method has been chosen. Since Turkish is an ‘agglutinative language’ like Finnish, Hungarian, Quechua and Swahili, new words are formed by adding suffixes to the end of roots by using a specific grammatical rule, and there are grammatical rules for suffixes, which of them may follow which other and in what order they will be (Appendix A). The meaning, also type of words are changed or extended by this concatenation. This suffix concatenation can result in relatively long words, which are frequently equivalent to a whole sentence in English.

1.3 Thesis Organization

This thesis is divided into 8 chapters and 6 appendices. The motivation of the thesis and the general description of corpus is given in Chapter 1. Corpora generated for English, Turkish and other languages are told briefly in Chapter 2. Also, Natural Language Processing studies used for both corpora development and linguistic studies, such as sentence boundary detection, stemming, part-of-speech analysis, author detection, etc. are given in Chapter 3.

The infrastructure of Rule-Based Corpus Generation (RBCorGen) software includes database model, structure of used tags, rules and lexicon, is explained briefly in Chapter 4. The algorithms developed for all steps of the RBCorGen are given in details with explanation of implemented classes and methods in Chapter 5.

The results and performance overview of RBCorGen are given in Chapter 6 with the properties of generated data set. The usage of RBCorGen is given briefly in Chapter 7, and finally, the conclusion, in which brief summary and results of this thesis are given in Chapter 8.

CHAPTER TWO
WORKS ON CORPORA DEVELOPMENT FOR SPOKEN LANGUAGES

2.1 Corpus

A corpus can be defined as a special database that includes analysed and tagged texts, and allows specialized processes in Natural Language Processing area such as retrieving the words and suffixes quickly.

By using the corpus, different analyses can be done, such as character recognition operations, cryptanalytical procedures, spell corrections (Church & Gale, 1991), etc. Also, some processes depending on n-gram analysis, such as different word usage statistics, frequencies of letters (Shannon, 1951) and words (Jurafsky & Martin, 2000; Çebi & Dalkılıç, 2004) etc., can be done by using corpus in NLP applications. N-gram analysis is one of the common statistical methods carried out on a corpus. Besides the letter and word frequencies, language model probabilities can be estimated and used in speech recognition systems (Nadas, 1984) by n-gram analysis. It can be used in correcting words by detecting misspelled words and it is useful for OCR (Optical Character Recognition) (Kukich, 1992). And it is commonly used in data compression and encryption. And also, missing words can be estimated for a given text by calculating word n-grams.

2.2 Sample Corpora

There are lots of corpora created for different languages. Some of them are representative, and some are large.

2.2.1 English Corpora

2.2.1.1 Brown Corpus

The Brown Corpus is the first computer-readable general corpus of texts prepared for linguistic research on modern English (Brown Corpus, (n.d.)), which was developed in 1960s, and announced in 1963-1964 at Brown University. In 1964, it included 1 million words with 61,805 different words and in a later edition in 1992; the new Brown corpus included 583 million words with 293,181 different words (Jurafsky & Martin, 2000). The samples in corpus have a wide range of varieties of scripts. Sentences in poems were not included on it because of having special linguistic problems different from scripts. Also drama was excluded, but fiction was included. Making available a carefully chosen and prepared body of material of considerable size in standardized format was aimed while generating Brown Corpus. Samples were chosen for their representative quality. The selection process was done in two phases: an initial subjective classification and decision as to how many samples of each category would be used. The data in the Brown University Library and the Providence Athenaeum were used in most categories. Also, some data were taken from the daily press, for example, the list of American newspapers of which the New York Public Library keeps microfilms (with the addition of the *Providence Journal*), and some periodical materials in the categories *Skills and Hobbies* and *Popular Lore* from the contents of magazine stores in New York City (Table 2.1, Figure 2.1) (Lindebjerg, 1997).

Table 2.1 Text categories in the Brown Corpus (Leech, et al., 2009)

	Genre group	**Category**	**Content of category**	**No. of samples**
I. Informative prose (374)	Press (88)	A	Reportage	44
		B	Editorial	27
		C	Review	17
	General Prose (206)	D	Religion	17
		E	Skills, trades and hobbies	36
		F	Popular lore	48
		G	Belles lettres, biographies, essays	75
		H	Miscellaneous	30
	Learned (80)	J	Science	80
II. Imaginative prose (126)	Fiction (126)	K	General fiction	29
		L	Mystery and detective Fiction	24
		M	Science fiction	6
		N	Adventure and Western	29
		P	Romance and love story	29
		R	Humor	9
TOTAL				500

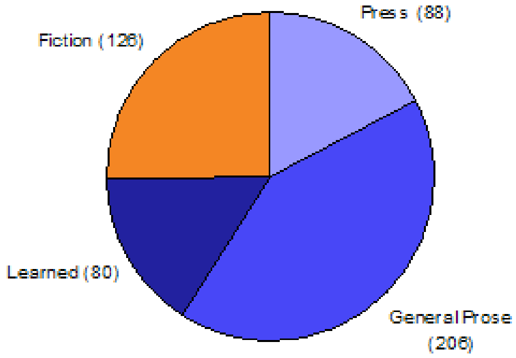


Figure 2.1 Genres represented in the Brown Corpus (CORD The Brown Corpus, (n.d.) a)

Sample tags used in Brown Corpus are given in Table 2.2.

Table 2.2 Sample list of tags in Brown Corpus (CORD The Brown Corpus, (n.d.) b)

Tag	Description	Examples
.	Sentence closer	. ; ? !
(	Left parenthesis	
)	Right parenthesis	
*	Not	
,	Comma	
ABL	Pre-qualifier	Quite, rather
ABN	Pre-quantifier	Half, all
AP	Post-determiner	Many, several, next, a, the, no
CC	Coordinating conjunction	And, or
CD	Cardinal numeral	One, two
DT	Singular determiner	This, that
DTS	Plural determiner	These, those
RP	Adverb/particle	About, off, up

2.2.1.2 British National Corpus (BNC)

The British National Corpus is a very large (over 100 million words) corpus of modern English, both spoken and written. However, non-British English and foreign language words do occur in the corpus (Burnard, 2000). This is a project of Oxford University Press, also including some other members: Longman Group UK Ltd., Chambers, Lancaster University's Unit for Computer Research in the English Language (UCREL), Oxford University Computing Services (OUCS), and the British Library. It was built in four years, and completed in 1994. It was released in February 1995. There are over 6,000,000 sentence units in the whole corpus, which occupies 1.5 gigabytes of disk space 90% of BNC is a written part including extracts from newspapers, journals, academic books, school and university essays, and 10% spoken part includes a large amount of unscripted informal conversation. The text type structure of BNC is given in Table 2.3 (The British National Corpus: facts and figures, (n.d.).

Table 2.3 The text type structure of BNC

BNC	**Text Type**	**(%)**
Written corpus		**90**
	Books	60
	Periodicals (regional and national newspapers, specialist periodicals and journals for all ages and interests)	25
	Other published materials (brochures, advertising leaflets, etc.)	5-10
	Unpublished materials (personal letters and diaries, school and university essays, etc.)	5-10
	Written to be spoken (political speeches, play texts, broadcast scripts, etc.)	< 5
Spoken corpus		**10**
	Transcriptions of natural spontaneous conversations	50
	Transcriptions of recordings made at four specific types of meeting or event: Educational, Business, Institutional, and Leisure.	50

Corpus-oriented Text Encoding Initiative (TEI)-conformant mark-up format known as CDIF (Corpus Document Interchange Format) was used for tagging BNC, but within this format many different formats (e.g. segmentation into words and sentences) were added to make the corpus more readable (Leech et al., 1994).

TEI (Text Encoding Initiative) is an international and interdisciplinary standard, announced in 1987, which helps publishers, scholars, libraries to represent all kinds of linguistic texts for research, by using an encoding scheme. TEI Consortium was set up to maintain and develop this standard in 2000. Until 2002, SGML (Standard Generalized Mark-up Language) was used in TEI standard, which allows us to define elements, specific features of elements, and hierarchical/structural relations between elements, and specifies them in a "Document Type Definition" (DTD) , which makes software to be able to help annotators to make annotation consistently.

Each element in SGML must have a unique name and must be explicitly tagged, such as *<element>* and *</element>* pairs that are called as start and end tags. Elements can have attributes with associated values used in tagging, such as *id,* *name, etc.* (Sperberg-McQueen & Burnard, 1994). In 2002, XML (Extensible Markup Language) has been used as TEI standard to make the annotations more efficient and readable. XML is more descriptive, that means it can define structure of texts rather than defining what can be done with the text, and independent from Application Development Environment and any platforms (Encoding the British National Corpus, (n.d.)).

The basic document structure of BNC is given in the Figure 2.2.

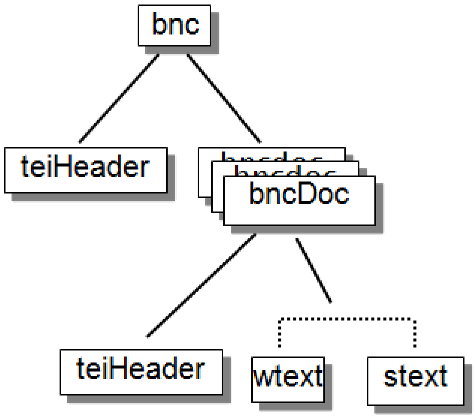


Figure 2.2 Basic document structure of BNC

"wtext" and "stext" contains "written" and "spoken" parts of corpus, and parsed by using XML structure (Figure 2.3). There are 6,026,284 tagged sentences and 98,363,784 tagged words in the BNC.

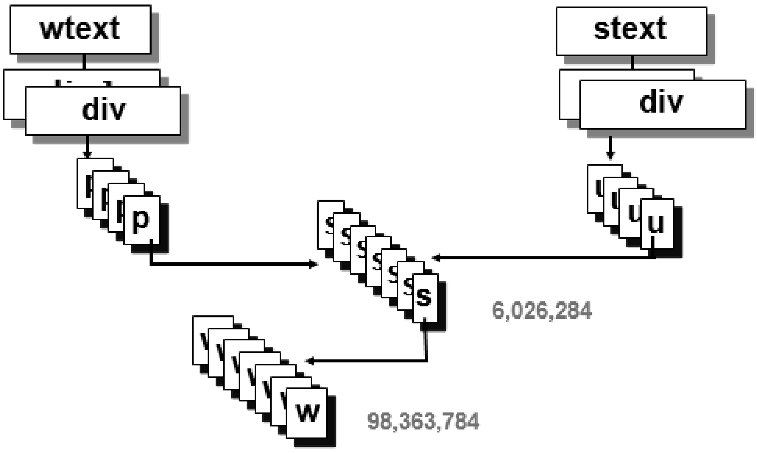


Figure 2.3 XML structure used in BNC

Written texts are organized hierarchically into various kinds of *division* such as;

```
<div level=”1”>
 <div level=”2”>... </div>
       <div level=”2”>...</div>
</div>
```

where divisions can be chapter, section, story, subsection, column, front, part, recipe, leaflet, etc. All spoken texts are divided into “conversations”.

In XML structure of BNC, paragraphs of written part are tagged as in Table 2.4.

Table 2.4 Paragraph tags used in BNC for written part

Tag	**Meaning**
<p>	Paragraph
<head>	headings or captions
<list>	lists
<quote>	quotes
<lg>	verse lines
<hi>	typographic highlighting
<corr>	corrected passages
<gap>	deliberate omissions
<pb/>	page breaks

Spoken texts are also organized hierarchically, by using the tags given in Table 2.5.

Table 2.5 XML tags used in BNC for spoken part

Tag	**Meaning**
<u who=”XXX”>	A stretch of speech initiated by speaker identified as XXX
<align with=”XXX”/>	a synchronization point
<shift>	changes in voice quality (e.g. whispering, laughing, etc.)
<vocal>	non-verbal but vocalised sounds (e.g. coughs, humming noises etc.)
<event>	non-verbal and non-vocal events (e.g. passing lorries, animal noises, and other matters considered worthy of note.)
<pause>	significant pauses (silence)
<unclear>	unclear passages (passages that are inaudible or incomprehensible)

Also, detailed information on speakers is given in the text header of spoken part.

An unannotated example of a raw BNC text is given in Figure 2.4.

```
<bncDoc id=BDFX8 n=093802>
<header type=text creator='natcorp' status=new update=1994-07-13>
  <fileDesc>
    <titStmt>
      <title>
        General Practitioners Surgery -- an electronic transcription
      </title>
      <respStmt>
        <resp> Data capture and transcription </resp>
        <name> Longman ELT </name>
      </respStmt>
    </titStmt>
    <ednStmt n=1> Automatically-generated header </ednStmt>
    <extent kb=7 words=128> </extent>
    <pubStmt>
      <respStmt>
        <resp> Archive site </resp>
        <name> Oxford University Computing Services </name>
      </respStmt>
      <address>
        13 Banbury Road, Oxford OX2 6NN U.K.
        ...
        Internet mail: natcorp@ox.ac.uk
      </address>
      <idno type=bnc n=093802> 093802 </idno>
      <avail region=world status=unknown>
        Exact conditions of use not currently known to
        the archiving agency.
        ...
        Distribution of any part of the corpus must
        include a copy of the corpus header.
      </avail>
      <date value=1994-07-13> 1994-07-13 </date>
    </pubStmt>
    <srcDesc>
      <recStmt>
        <rec type=DAT>
        </rec>
      </recStmt>
    </srcDesc>
  </fileDesc>
  <profDesc>
    <creation date='?'> Origination/creation date not known </creation>
    <partics>
      <person age=X educ=0 flang=EN-GBR id=PS22T n=W0001 sex=m soc=AB>
        ...
      </person>
      <person id=FX8PS000 n=W0000> ... </person>
      <person id=FX8PS001 n=W0002> ... </person>
    </partics>
...
...
</bncDoc>
```

Figure 2.4 An unannotated example of a raw BNC text

2.2.1.3 The Bank of English

The Bank of English is a collected from samples in modern English language, which is held on computer for using in linguistics (Järvinen, 1994).

The Bank of English was started to be collected in 1980 by COBUILD, which was based within the School of English at Birmingham University, and launched in 1991 by COBUILD and The University of Birmingham. The aim was making the scale of the corpus to 200 million words and 103 million words were collected and tagged until 1993. It had 450 million words in January 2002, 525 million words as of 2005and it continues to grow. It has spoken and written part as in BNC. The written part contains books, newspapers, magazines, letters, etc. and the spoken part includes speech from BBC World Service radio broadcasts, and the American National Public Radio, meetings, conversations, etc. The data are either collected from electronic environment or from scanning some books. Whole corpus is divided into 11 subcorpora or text-type categories. Abbreviations used for subcorpora are given in Table 2.6 (The Bank of English User Guide, (n.d.)).

Table 2.6 Abbreviation list of subcorpora in the Bank of English Corpus

Abbreviation	**Full Title**
oznews	Australian news
ukephem	UK ephemera
ukmags	UK magazines
Ukspok	UK spoken
usephem	US ephemera
bbc	BBC World Service
npr	National Public Radio
ukbooks	UK books
usbooks	US books
times	Times newspaper
today	Today newspaper

2.2.1.4 English Gigaword

It is an English corpus having 1,756,504,000 words and 4,111,240 documents. It is a product of Linguistic Data Consortium. It includes data from Agence France Press English Service, Associated Press Worldstream English Service, The New York times Newwire Service and Xinhua News Agency English Service (Parker et al., 2009).

Sample text from English Gigaword corpus is given in Figure 2.5.

```
<DOC id="LTW_ENG_20081201.0001" type="story" >
<HEADLINE>
Road Map in Iraq: When Mr. Obama Takes Office, a Sovereign Iraqi
Government and a U.S. Withdrawal Timetable Will Be in Place
</HEADLINE>
<TEXT>
<P>
The following editorial appeared in Sunday's Washington Post:
</P>
<P>
Barack Obama recently reiterated his campaign promise to order up a plan
for the withdrawal of U.S. forces from Iraq. But the Iraqi parliament
has beaten him to it. Its ratification Thursday …………. toward that goal.
</P>
</TEXT>
</DOC>
```

Figure 2.5 Sample tagged text from English Gigaword

2.2.1.5 American National Corpus

The American National Corpus (ANC) is aimed to contain a core corpus of at least 100 million words, including both written and spoken (transcripts) data. The genres in the ANC are expanded from BNC to include new types of language data that have become available in recent years, such as web blogs and web pages, chats, email, and music lyrics. In Spring 2010, the ANC produced its second release of over 22 million words of American English, where it was 11 million in the first release in 2003 (Ide & Suderman, 2003).

2.2.2 Turkish Corpora

Some Turkish corpora are listed below:

- Koltuksuz Corpus
- Yıldız Technical University (YTU) Corpus
- Dalkilic Corpus
- METU Turkish Corpus
- TurCo Turkish Corpus

There are also other corpora for Turkish (Güngör, 1995).

2.2.2.1 Koltuksuz Corpus

Koltuksuz Corpus can be called as the first corpus generated for Turkish language, used for letter statistics and to find out some of the characteristics of Turkish Language. It has 6,095,457 characters and formed of 24 novels and stories of 22 different authors (Koltuksuz, 1995).

2.2.2.2 Yıldız Technical University (YTU) Corpus

YTU Corpus has 4,263,847 characters from 14 different documents: 3 Novels, 1 PhD Thesis, 1 Transcription, 9 Articles and created for compression based morphology study by Diri (2000).

2.2.2.3 Dalkilic Corpus

There are two different corpora prepared by Dalkilic (2001) and Dalkilic and Dalkilic (2001). They are;

- *Dalkilic Corpus*: It has 1,473,738 characters from the newspaper "Hurriyet" web archive (01/01/1998 – 06/01/1998 mainpage and 01/01/1998 – 06/30/1998 authors) and generated for letter statistics and defining the characteristics of Turkish language (Dalkılıç, 2001).

- *Dalkilic Corpus*: It is the combination of some the previous Turkish corpora (Koltuksuz, YTÜ and Dalkilic corpora) with a size of 11,749,977 characters (Dalkılıç & Dalkılıç, 2001).

2.2.2.4 METU Turkish Corpus

It is a collection of over one million words of post-1990 written Turkish samples (METU Turkish Corpus Project, (n.d.); Say, Zeyrek, et al., 2002; Say, Özge, et al., 2002).

The document types in METU Corpus are listed in Table 2.7.

Table 2.7 Document types in METU Corpus

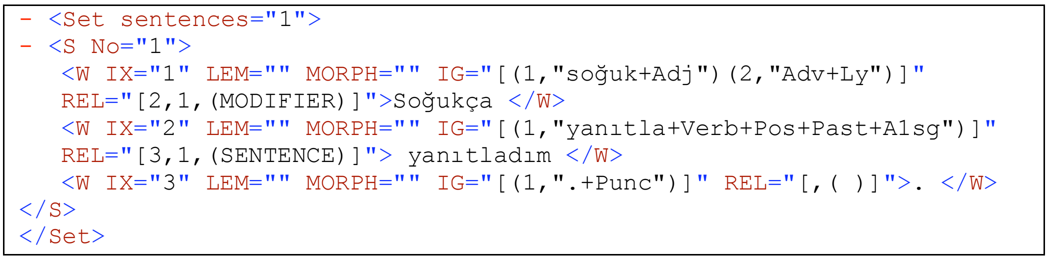
Genre	Percentage of entire corpus (%)
Novel	24
Story	21
Article	16
Essay	14
Research	12
Travel Writing	4
Conversation	2
Others (Biography, Auto-biography, Reference, Diary, etc.)	7

For tagging process of paragraphs, quotas, lists, and other elements' citation information XCES, one of the application of TEI, is used. Some tags used in corpus are given in the following table.

Table 2.8 Tags in METU Corpus

Tag Name	Meaning
<text>	Tags texts
<body>	Tags the unit of texts
<opener>	Tags the data in the introduction part of texts, such as Date, Keywords, etc.
<head>	Indicates the header of the structures like text, poem, etc.
<p>	Paragraph
<q>	Quotas
<poem>	Poems
<table>	Table
<list>	List
<abbr>	Abbreviation
<date>	Date
<hi>	Highlighted words and phrases like bold, underlined, etc.

Sample tagged text in METU corpus is given in the following figure.



```
- <Set sentences="1">
- <S No="1">
   <W IX="1" LEM="" MORPH="" IG="[(1,"soğuk+Adj")(2,"Adv+Ly")]"
   REL="[2,1,(MODIFIER)]">Soğukça </W>
   <W IX="2" LEM="" MORPH="" IG="[(1,"yanıtla+Verb+Pos+Past+A1sg")]"
   REL="[3,1,(SENTENCE)]"> yanıtladım </W>
   <W IX="3" LEM="" MORPH="" IG="[(1,".+Punc")]" REL="[,( )]">. </W>
</S>
</Set>
```

Figure 2.6 Sample tagged text in METU Corpus

2.2.2.5 TurCo Turkish Corpus

TurCo is known as first corpus created for word statistics, which has a capacity of 362.449MB, and 50,111,828 words (Dalkilic & Cebi, 2002).

TurCo consists of text data taken from 11 different websites, and novels and stories in Turkish that belong to more than 100 authors, which parts were collected from websites (98.11%) and novels and stories (1.89%).

In order to make TurCo larger, to include more words, it is generated as unbalanced corpus. The document types in the corpus have different sizes as given in Table 2.9.

Table 2.9 NOW (Number of Words), files' size and distribution % in TurCo

Site #	Web Sites	NOW	Corpora Files' Sizes[1] (MB)	Percentage of entire corpus (%)
1	www.tbmm.gov.tr	23,396,817	170.747	46.69
2	www.stargazete.com.tr	9,746,093	69.103	19.45
3	www.hurriyet.com.tr	9,415,716	69.140	18.79
4	Turkish novels and stories	4,668,306	33.571	1.89
5	www.die.gov.tr	948,116	6.387	9.32
6	www.arabul.com	753,571	4.994	1.50
7	www.pcmagazine.com.tr	527,757	3.722	1.05
8	www.bilimteknoloji.com.tr	203,620	1.450	0.41
9	www.abgs.gov.tr	160,562	1.249	0.32
10	www.lazland.com	135,519	0.954	0.27
11	www.yeniasir.com.tr	96,857	0.707	0.19
12	www.pankitap.com	58,894	0.425	0.12
	TOTAL	50,111,828	362.449	100.00

[1] Includes only Turkish alphabet and space character

In TurCo, Number of Words (NOW), number of different words (NODW) and Different Word Usage Ratio (DWUR) are calculated and given in Table 2.10. NODW in all sites are 1,235,056, but some words are repeated in different sites. These words are picked up from TurCo and calculated again. The result of this, NODW in TurCo is 686,804. According to this result, DWUR in TurCo is 1.37%.

Table 2.10 NOW, NODW and DWUR in TurCo

Site #	NOW	NOW Ratio (%)	NODW	NODW Ratio (%)	DWUR (%)
1	23.396.817	46,69	342.544	27,74	1,46
2	9.746.093	19,45	255.024	20,65	2,62
3	9.415.716	18,79	99.432	8,05	1,06
4	4.668.306	9,32	309.030	25,02	6,62
5	948.116	1,89	20.760	1,68	2,19
6	753.571	1,50	42.208	3,42	5,60
7	527.757	1,05	46.743	3,78	8,86
8	203.620	0,41	29.228	2,37	14,35
9	160.562	0,32	13.103	1,06	8,16
10	135.519	0,27	37.057	3,00	27,34
11	96.857	0,19	25.294	2,05	26,11
12	58.894	0,12	14.633	1,18	24,85
Total	50.111.828	100,00	1.235.056	100,00	2,74
TurCo	50.111.828		686.804		1,37

2.2.3 Corpora of Other Languages

2.2.3.1 The Czech National Corpus (CNC)

The Czech National Corpus (CNC) is a non-commercial, academic project, which contains written Czech (Kucera, 2002).

The idea of CNC was first mentioned in 1990, and the work is started in 1994 when Faculty of Arts at Charles University, Prague, founded the Czech National Corpus Institute. It was signed by 8 signatories, representatives of the some institutions such as, Faculty of Mathematics and Physics, Charles University, Masaryk University, Palack University, Institute of Czech Language, Academy of Sciences, etc.

It has synchronous and diachronic parts. Some parts of the synchronous are: Database and dictionaries (Electronic databases and dictionaries), SYN2000 (Balanced representative of contemporary written Czech and contains about 100 million words), ORAL (Spoken Czech) (Czech National Corpus, (n.d.)).

2.2.3.2 Croatian National Corpus

It has 30 million words and 101.3 million tokens as of March 03th, 2010 and is still growing. It includes contemporary Croatian covering different media, genres, styles, fields and topics (Croatian National Corpus: Home Page, (n.d.)). The document types used in the corpus is given in Table 2.11.

Table 2.11 Document types in Croatian National Corpus

	Genre	**Percentage of entire corpus (%)**
Informative Texts		**74**
Newspapers (37%)		
	Daily	22
	Weekly	9
	Bi-weekly	6
Magazines, journals (16%)		
	weekly	9
	monthly	4
	bi-, tri-monthly	3
Books, brochures, correspondence... (21%)		
	publicistics	4
	popular texts	3.5
	correspondence, ephemera	0.5
	arts and sciences	13
Imaginative texts (fiction): prose		**23**
	novels	13
	stories	5
	essays	4
	diaries, (auto)biographies...	1
Mixed texts		**3**

2.2.3.3 PAROLE

PAROLE has collection of modern Dutch texts, which are younger than 1980. The data included in PAROLE is given in Table 2.12 (PAROLE CORPUS-Information, (n.d.)), which has over 20,000,000 words.

Table 2.12 Document types in Dutch PAROLE.

Distribution according to publication medium			**Number of words**	**Percentage of entire corpus (%)**
Books			3,247,136	15.98 %
Newspapers	articles		12,970,841	63.85 %
	quotations		217,500	1.07 %
Periodicals	Local papers	quotations	52,235	0.26 %
	Periodicals	articles	1,201,721	5.92 %
		quotations	176,962	0.87%
Miscellaneous	Pamphlets	quotations	163,022	0.80 %
	8 o'clock news		1,280,986	6.31 %
	Jeugdjournaal (News for young people)		1,005,079	4.95 %
Total			**20,315,482**	100 %

Some of tags used in the corpus are given in Table 2.13.

Table 2.13 Tags for POS Tagging

Abbreviation	**Meaning**
ADJ	Adjective
ADP	Adposition
ADV	Adverb
ART	Article
CON	Conjunction
DET	Determiner
INT	Interjection
NOU	Noun
NUM	Numeral
PRN	Pronoun
RES	Residual
UNIQUE	Unique Membership Class
VRB	Verb

PAROLE was improved to be multilingual, which contains the languages Belgian French, Catalan, Danish, Dutch, English, French, Finnish, German, Greek, Irish, Italian, Norwegian, Portuguese and Swedish. It has 20,000 entries per language.

2.2.3.4 French Corpus

The French Corpus that includes the tagging of the anaphors was created by the CRISTAL-GRESEC (Stendhal-Grenoble 3 University, France) team and XRCE (Xerox Research Centre Europe, France) in the framework of the call launched by the DGLF-LF (national institution for the French language and the languages spoken in France). This corpus has over 1 million annotated words from scientific and human science articles, books (some stored in CD-ROM), newspapers (especially Le Monde newspaper), periodicals (HERMES and CNRS-Infos), etc. (Modern French Corpus, (n.d.)). The data in the corpus are:

- Two books, edited by the CNRS, which have 77.591 and 124.990 words.
- 204 articles, extracted from CNRS Info, a magazine which contains short popular scientific articles from the CNRS laboratories (201.280 words).

- 14 articles dealing with Hermès Human Sciences (111.886 words).
- 136 articles, extracted from "Le Monde", dealing with economics (roughly 180 760words).
- 13 booklets of the Official Journal of the European Communities (roughly 337.000 words).

The annotation scheme was defined in XML format and the annotation process was done manually by two qualified linguists.

2.2.3.5 COSMAS (Corpus Search Management Analysis System)

It is a German corpus having more than 3,750,000,000 running words, into which new words are added each day, and world's largest collection of German texts. It was created in 1964 and still growing. The texts are younger than 1950, and covers all time to the present. Only 1.1 billion words are available to public because of copyright restrictions. It is a product of "Institut für Deutsche Sprache, Mannheim" (COSMAS, German Corpus, (n.d.)). *Deutsches Referenzkorpus (DeReKo)* is the official name of the full corpus archive since 2004.

CHAPTER THREE
COMMONLY USED METHODS FOR NATURAL LANGUAGE PROCESSING APPLICATIONS

In order to generate a corpus, some main processes must be done, such as;

- Sentence boundary detection,
- Stemming and root finding,
- Part-of-Speech examination.

3.1 Sentence Boundary Detection

The first process of generating a corpus, which is a representative of the language, is determination of sentences, which is very complicated and hard to solve, but an important part of the corpus generation.

Different approaches have been tried to find out sentence boundaries in some languages. The most known approaches are "Rule Based" and "Machine Learning". Manually collected rules, which are usually encoded in terms of regular expression grammars, and supplementary lists of abbreviations, common words, proper names, and appropriate feature sets of syntactic information, are used in rule-based approach such as in the study of Aberdeen et al. (1995), in which sentence-splitting module that contains nearly 100 regular-expression rules. Developing a good rule base system is an ambiguous task itself and hard to design. So, different approaches are developed to solve the sentence boundary disambiguation by using "Machine Learning", such as Maximum Entropy approach of Reynar & Ratnaparki (1997), the Decision Tree Classifier approach of Riley (1989), and Neural Network approach of Palmer & Hearst (2000). Also, there are hybrid systems such as the Mikheev's work (1997), which integrates part-of-speech tagging task based on Hidden Markov model of the language and the Maximum Entropy into sentence boundary detection.

For English, a module named `Sentence` in `Lingua` library, which is used for splitting text into sentences, was developed in Perl and distributed freely in 2001 (Yona, 2001). This module contains the function `get_sentences` that splits text into its sentences by using regular expression and a list of abbreviations.

```
get_sentences( $text )
add_acronyms( @acronyms )
get_acronyms( )
set_acronyms( @my_acronyms )
get_EOS( )
set_EOS( $new_EOS_string )
set_locale( $new_locale )
```

Figure 3.1 Functions in `Sentence` module

The Bondec system (Wang & Huang, 2003) is a sentence boundary detection system for English, which has three independent applications (Rule-based, HMM, and Maximum Entropy). Three files were created, `train.dat`, `test.dat`, and `heldout.dat`, from Palmer's raw data files, the Wall Street Journal (WSJ) Corpus (Palmer & Hearst, 1997). The `train.dat` file is used for training purpose in HMM and ME. There are 21,026 sentences in this training set, 95.25% (20,028) of them are delimited by a period; 3.47% (727) of them ends with a quotation mark and 0.69% (146) of them ends with a question mark. The `heldout.dat` file has 9721 sentences, which was used for cross-validation and performance tuning; while the test set, which has 9758 sentences, was only available for final performance measurements. Maximum Entropy Model is the main method of this system, which achieved an error rate less than 2% on part of the WSJ Corpus. The performances of these three applications are given in Table 3.1.

Table 3.1 Performance comparison of three methods

Method	Precision	Recall	F1	Error Rate
RuleBased	99.56%	76.95%	86.81%	16.25%
HMM	91.43%	94.46%	92.92%	10.00%
MaxEnt	99.16%	97.62%	98.38%	1.99%

An ontology based approach on sentence boundary detection for Turkish was developed by Temizsoy and Çiçekli in 1998. In the same year, a new method, in which simple Turkish sentences were generated, was developed by Çiçekli and Korkmaz (1998). They used a functional linguistic theory called Systemic-Functional Grammar (SFG) to represent the linguistic resources, and FUF (Functional Unification Formalism) text generation system as a software tool to carry out them.

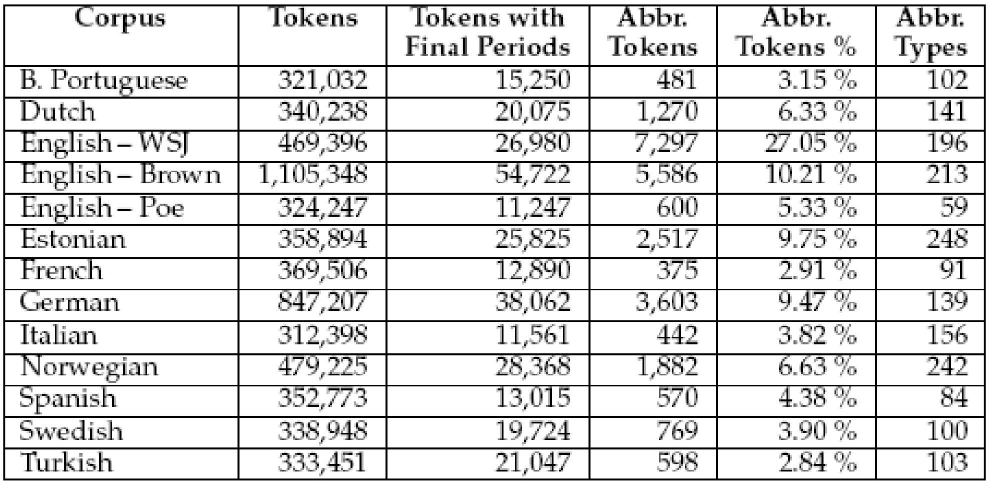
Other well-known study on sentence boundary detection for Turkish is developed by Dinçer and Karaoğlan (2004), in which a rule-based approach was used. This study was tested on a collection of Turkish news texts having 168,375 tokens, including punctuations, and 12,026 sentences, which are morphologically analyzed and disambiguated by Hakkani-Tür et. al. (2002) and success rate was measured as 96.02%. The rules were generated as all combinations around a dot with a triple. For example, [`w * W`] denotes the situation where a letter sequence `w` which starts with a lower-case character, is followed by a dot (represented by asterisk "*") which is then followed by a letter sequence `W` which starts with an uppercase character. The symbols and their meanings are listed in Table 3.2.

Table 3.2 Notation

Symbol	**Meaning**
w	All letter sequences starting with a lowercase character.
W	All letter sequences start with an uppercase character
#	All number sequences. (Real, integer cardinal or ordinal, date, time, telephones, etc.)
T	Apostrophe (‘)
TT	Quote character (“)
K	Dash (-)
V	Comma (,)
(	Open parentheses
)	Close parentheses
:	Colon
;	Semi colon
P	All punctuation including not listed ones such as %, &, $, etc.
EOS	End of Sentence
~EOS	Not End of Sentence
∞	All kind of tokens (w, W, #, T, TT, K, V, “(“, “)”, P)

The well-known highest success rate for Turkish sentence boundary method was denoted by Kiss & Strunk (2006) for multilingual sentence boundary detection including Turkish, which was measured as 98.74% mean value of eleven languages' test results, English, Brazilian Portuguese, Dutch, Estonian, French, German, Italian, Norwegian, Spanish, Swedish, and Turkish (Table 3.3). For Turkish, it has the success rate of 98.69%. It was implemented by using the log-likelihood ratio algorithm by Dunning (1993) and tested on the part of METU Turkish Corpus (Say et.al, 2002), which only included Turkish newspaper, Milliyet.

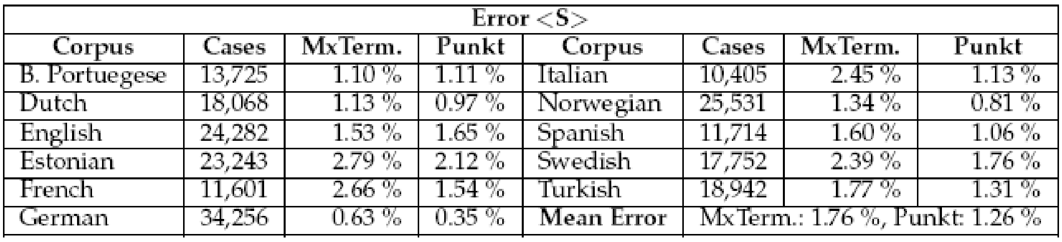
Table 3.3 Statistical properties of the test corpora



Corpus	Tokens	Tokens with Final Periods	Abbr. Tokens	Abbr. Tokens %	Abbr. Types
B. Portuguese	321,032	15,250	481	3.15 %	102
Dutch	340,238	20,075	1,270	6.33 %	141
English – WSJ	469,396	26,980	7,297	27.05 %	196
English – Brown	1,105,348	54,722	5,586	10.21 %	213
English – Poe	324,247	11,247	600	5.33 %	59
Estonian	358,894	25,825	2,517	9.75 %	248
French	369,506	12,890	375	2.91 %	91
German	847,207	38,062	3,603	9.47 %	139
Italian	312,398	11,561	442	3.82 %	156
Norwegian	479,225	28,368	1,882	6.63 %	242
Spanish	352,773	13,015	570	4.38 %	84
Swedish	338,948	19,724	769	3.90 %	100
Turkish	333,451	21,047	598	2.84 %	103

Error rates in this study are given in Table 3.4.

Table 3.4 Error rates



Error <S>							
Corpus	Cases	MxTerm.	Punkt	Corpus	Cases	MxTerm.	Punkt
B. Portuegese	13,725	1.10 %	1.11 %	Italian	10,405	2.45 %	1.13 %
Dutch	18,068	1.13 %	0.97 %	Norwegian	25,531	1.34 %	0.81 %
English	24,282	1.53 %	1.65 %	Spanish	11,714	1.60 %	1.06 %
Estonian	23,243	2.79 %	2.12 %	Swedish	17,752	2.39 %	1.76 %
French	11,601	2.66 %	1.54 %	Turkish	18,942	1.77 %	1.31 %
German	34,256	0.63 %	0.35 %	Mean Error	MxTerm.: 1.76 %, Punkt: 1.26 %		

3.2 Stemmers

The process of reducing derived/inflected words to their stem or root is called “Stemming”. There are many algorithms generated for stemming in many languages, such as Porter Stemming Algorithm for English (Porter, 1980), Stemming Engine for Polish (Weiss, 2005), Swedish, German, Spanish, Greek Stemming Algorithms, etc.

Some of the algorithms that determine root or stem of words in Turkish such as Identified Maximum Match (IMM) Algorithm (Köksal, 1975), AF Algorithm (Solak & Can, 1994), Longest-Match (L-M) Algorithm (Alpkoçak et al., 1995), Root Finding Method without Dictionary (Cebiroğlu & Adalı, 2002), FindStem Algorithm (Sever & Bitirim, 2003), Extended Finite State Approach (Oflazer, 2003), etc. are investigated and summarized.

Identified Maximum Match (IMM) Algorithm is developed by Köksal in 1975. It is left-to-right parsing algorithm, which tries to find the maximum length substring that is matched with in a root lexicon. If a match is found, the remaining part of the word is considered as the suffixes, this part is searched in a suffix morpheme forms dictionary and morphemes are identified one by one until there is no element.

In 1993, Solak and Oflazer developed an algorithm which used a dictionary that has 23,000 words based on the Turkish Writing Guide as the source (Solak and Oflazer, 1993). The words are listed in a sorted order in an ordered sequential array to be able to make fast search. Each entry of the dictionary contains a root in Turkish and a series of flags showing certain properties of that word. If the bit corresponding to a certain flag is set for an entry, it means that the word has the property represented by that flag. 64 different flags are reserved for each entry, but only 41 flags have been used. Some of the flags are given in the Table 3.5.

Table 3.5 Example of flags

Flag	Property of the word for which this flag is set	Examples
CL_NONE	Belongs to none of the two main root classes	RAĞMEN, VE
CL_ISIM	Is a nominal root	BEYAZ, OKUL
CL_FIIL	Is a verbal root	SEV, GEZ
IS_OA	Is a proper noun	AYŞE, TÜRK
IS_OC	Is a proper noun which has a homonym that is not a proper noun	MISIR, SEVGİ
IS_SAYI	Is a numeral	BİR, KIRK
IS_KI	Is a nominal root which can directly take the relative suffix –Kİ	BERİ, ÖBÜR
IS_SD	Is a nominal root ending with a consonant which is softened when a suffix beginning with a vowel is attached.	AMAÇ,PARMAK, PSİKOLOG
IS_SDD	Is a nominal root ending with a consonant which has homonym whose final consonant is softened when a suffix beginning with a vowel is attached.	ADET, KALP

The root of the word is searched in the dictionary using a maximal match algorithm. In this algorithm, first the whole word is searched in the dictionary, if it is found then it is assumed that the word has no suffixes and it does not need to be parsed. If not, then right-to-left parsing is done. A letter from the right is removed and the left letters are searched as a word if it exists in the dictionary. This step is repeated until the root is found. If no root is found after the letter at the beginning or the word is removed, the word's structure is accepted as incorrect. In order to obtain reliable results from this parser, all of the rules and their exceptions must be implemented. But they could not obtain all rules and exceptions in Turkish language.

AF algorithm works with a lexicon that includes actively used stems for Turkish in which each record is explained with 64 tags (Solak & Can, 1994). The examined word is looked up in the lexicon iteratively by pruning a letter from right at each step. If the character array matches with any of the root words in the lexicon, then the morphological analysis for that word is finished. The process is repeated until a single letter is left from the word. The AF algorithm is summarized as:

```
1. Remove suffixes that are added with punctuation marks from the word.
2. Search the word in dictionary.
3. If a matched root found, add the word into root words list.
4. If the word remained as a single letter, the root words list is empty then go to step 6, if root words list has at least one element then go to step 7.
```

```
5. Remove the last letter from the word and go to step 2.
6. Add the examined word into unfounded record and exit.
7. Get the root word from the root words list.
8. Apply morphological analysis to the root word.
9. If the result of morphological analysis is positive then add the root word to the stems list.
10. If there is any element(s) in root words list then go to step 7.
11. Choose the all stems in the stems list as a word stem.
```

Although, this algorithm finds all possible stems of the word, it is far away to find "correct" stem.

Longest-Match (L-M) Algorithm is based on the word search logic over a lexicon that covers Turkish word stems and their possible variances (Kut et al., 1995). Here is the algorithm:

```
1. Remove suffixes that are added with punctuation marks from the word.
2. Search the word in the dictionary.
3. If a root is matched, go to step 5.
4. If the word remained as a single letter, go to step 6. Otherwise, remove the last letter from the word and go to step 2.
5. Choose the found root as a stem and go to step 7.
6. Add the examined word into unfounded records.
7. Exit.
```

This algorithm finds the first stem matched with character array that is gained by removing the last letter iteratively. This algorithm is far away to find "correct" root or stem, because first matched substring of word may not be the correct stem.

In 2002, a new method is developed in which roots can be found without dictionary by Cebiroğlu and Adalı. It is claimed and proved that by analyzing a word, its root and suffixes can be formulated. The suffixes, which can be attached to a root, are divided into groups and finite state machines are formed by formulating the order of suffixes for each of these groups. A main machine is formed by combining these machines specific to the groups. In the morphological analysis, the root is obtained by extracting the suffixes from the end to the beginning of word. The abbreviations that are used in suffixes are:

U:	ı,i,u,ü	C:	c,ç
A:	a,e	I:	ı,i
D:	d,t	():	the letters not obligatory

where "-cU" can be -cı, -ci, -cu, -cü.

In this method, it was assumed that the morphological rules can be determined with finite state machines. Rules may be interpreted from right to left and from last to beginning to reach to the root of the word. Different modules are developed for all sets dependent to each other. Table 3.6 shows the affix-verbs in Turkish that is determined as a set of the affix-verbs.

Table 3.6 The affix-verbs in Turkish

1	–(y)Um	6	–m	11	–cAsInA
2	–sUn	7	–n	12	–(y)DU
3	–(y)Uz	8	–k	13	–(y)sA
4	–sUnUz	9	–nUz	14	–(y)mUş
5	–lAr	10	–DUr	15	–(y)ken

The finite state machine of the implementation of the data in Table 3.6 is given in Figure 3.2.

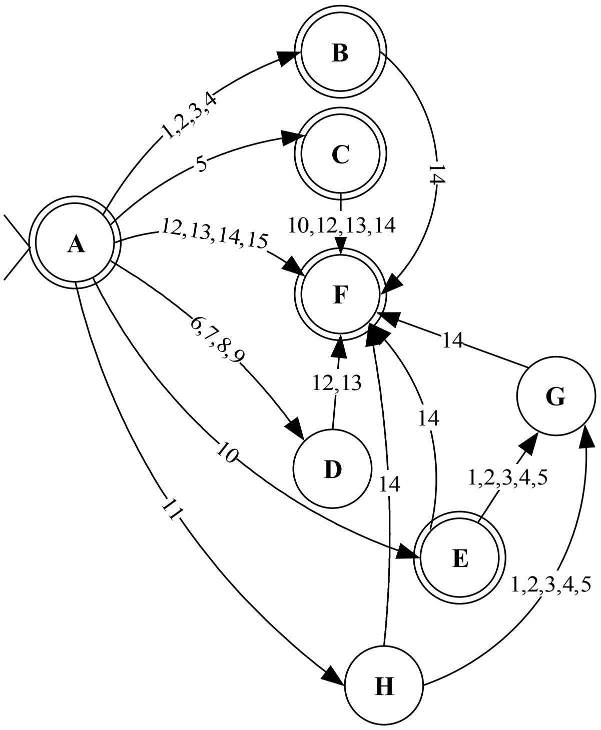


Figure 3.2 Finite state machine of Table 3.6.

For example, the word "çalışkan-mış-sınız" is examined by this finite state machine as;

- `- sUnUz` affix moves from A to B state,
- `- (y)mUş` affix moves from B to F state
- If the last affix `-n` is tried to move anywhere from F state, it is not possible to move, so the process is stopped.

Since F state is final; the root is accepted as "çalışkan" in the example given above. But the correct root is "çalış-", so this algorithm gave wrong result.

For all sets like the affixes that are used for nouns and verbs new finite state machines are implemented. They are all combined in one finite state machine at the end and the roots are found. The main finite state machine is given in Figure 3.3.

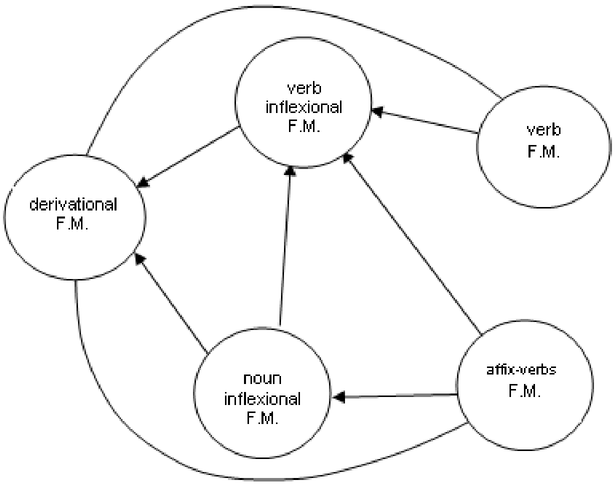


Figure 3.3 The main finite state machine

In 2003, a method by extended finite state approach is developed by Oflazer. In this approach, a Turkish word is represented as a sequence of Inflectional Groups (IGs), separated by ^DBs denoting derivation boundaries, in the following general form:

`root + Infl1^DB+Infl2^DB+…… ^DB+Infl`$_n$

where `Infl`$_i$ denotes relevant inflectional features including the part-of-speech for the root, or any of the derived forms. For example, the derived determiner "sağlamlaştırdığımızdaki" (*en: (the thing existing) at the time we caused (something) to become strong*) would be represented as:

```
sağlam+Adj ^DB+Verb+Become ^DB+Verb+Caus+Pos
      ^DB+Adj +PastPart+P1sg^DB
      +Noun+Zero+A3sg+Pnon+Loc^DB+Det
```

This word has 6 IGs:

1. `sağlam+Adj`
2. `+Verb+Become`
3. `+Verb+Caus+Pos`
4. `+Adj+PastPart+P1sg`
5. `+Noun+Zero+A3sg +Pnon+Loc`
6. `+Det`

A sentence then would be represented as a sequence of the IGs. When a word is considered as a sequence of IGs, syntactic relation links only emanate from the last IG of a (dependent) word, and land on one of the IG's of the (head) word on the right (with minor exceptions) (Figure 3.4).

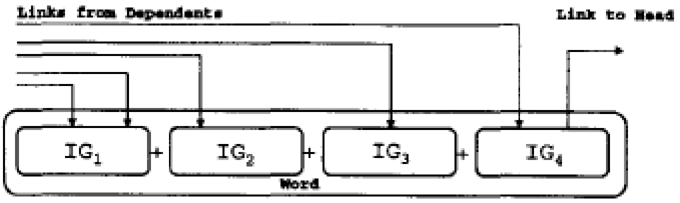
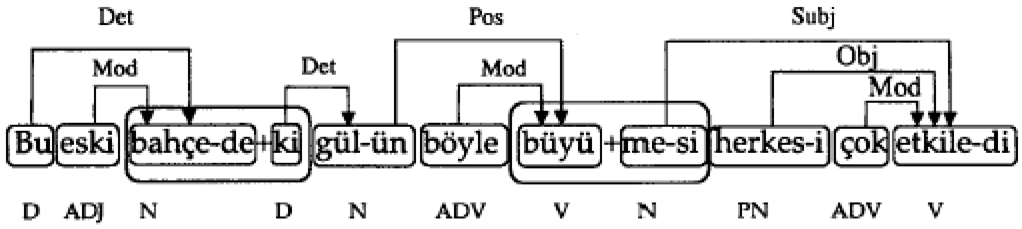


Figure 3.4 Links and inflectional groups

A dependency tree for a sentence laid on top of the words segmented along IG boundaries is given in Figure 3.5.



Last line shows the final POS for each word.

Figure 3.5 Dependency links in an example Turkish sentence

The approach relies on augmenting the input with *channels* that reside above the IG sequence and *laying* links representing dependency relations in these channels. The parser, which was implemented for this approach, has a number of iteration. A new empty channel is on top of the input in each iteration, and any possible links are established by using these channels, until no new links can be added. The symbol “0” indicates that the channel segment is not used while “1” indicates that the channel is used by a link that starts at some IG on the left and ends at some IG on the right, that is, the link is just crossing over the IG. If a link starts from an IG (ends on an IG), then a start (stop) symbol denoting the syntactic relation is used on the right (left) side of the IG. The syntactic relations (along with symbols used) that are encoded in the parser are the following:

> 4 S (Subject), 0 (Object), M (Modifier, adv/adj), P (Possessor), C (Classifier),
> D (Determiner), T (Dative Adjunct), L (Locative Adjunct), A: (Ablative Adjunct),
> I (Instrumental Adjunct).

In 2003, Sever & Bitirim developed a new method called FindStem. This method contains a pre-processing step that converts all letters of the word into their cases and singles out the letters after the punctuation mark in the word. It has three components;"Find the Root", "Morphological Analysis" and "Choose the Stem".

In "Find the Root" component, all possible roots of the examined word are found by starting with the first character of the examined word and searching the lexicon for this item. Then the next character is appended to the item and searched in the lexicon again. This operation continues until the item becomes equal to the examined word or until the system understands that there are no more relevant roots for the examined word in the lexicon. Then, found roots and production rules are used to derive the examining word. In lexicon, the class of all words and possible syntactic changes during combining a root with suffix is coded for the Morphological Analysis component.

A morphological analyzer is used in "Morphological Analysis" component. All possible stems can be found by using this component.

In the last component, "Choose the Stem", the stem is chosen by a selection between derivations in the derivations list.

This algorithm finds all possible stems of the word by eliminating the stems that are not in the derivation list. The algorithm is:

```
1. Remove suffixes that are added with punctuation marks from the word.
2. Find all possible roots of the word in a lexicon and add them into root words list.
3. If root words list is empty, add the word into unfounded records and exit.
4. Get the root word from root words list.
5. Apply morphological analysis to the root word.
6. After morphological analysis, add the formed derivations into derivations list.
7. If there is any element(s) in root words list then go to step 4.
8. Choose the stem by a selection between derivations in the derivations list.
```

3.3 Part of Speech (POS) Tagging

In a sentence, words are grouped into classes according to their similar syntactic behavior by linguist. Those word classes are called Parts-of-Speech (POS) of which well-known three are: noun, verb and adjective (Manning & Schutze, 1999).

Part-of-speech tagging is defined as “a process in which a part-of-speech label is assigned to each of words in sequence” (Jurafsky & Martin, 2000, p. 314). The POS tagging process is simply given in Figure 3.6.

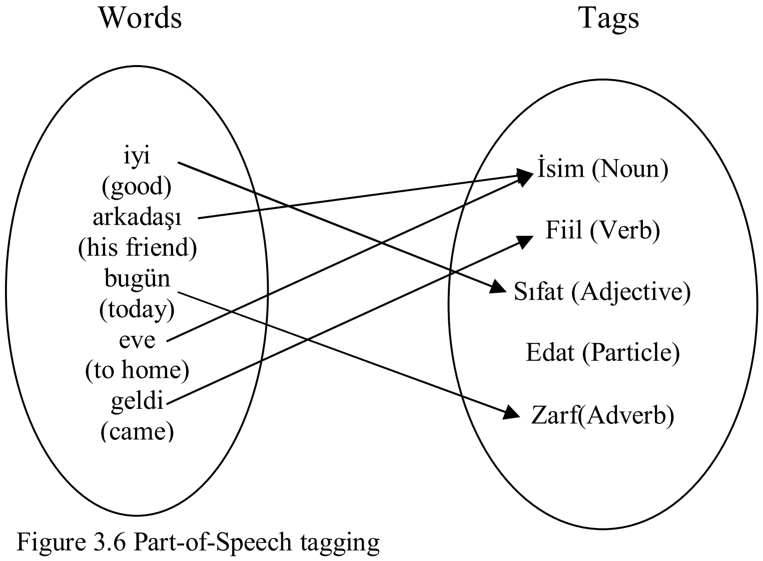


Figure 3.6 Part-of-Speech tagging

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

There are many aspects about classifying POS tagging processes, such as Guilder announced in 1995, in which a distinction among POS taggers were made according to taggers' automation degree in training and tagging process (Guilder, 1995). Two approaches in this classification are:

1. Supervised Tagging
2. Unsupervised Tagging

In supervised methods, users check out the results and accept one result as true and generally require pre-tagged corpora to be used in the tagging process. In unsupervised methods, the results are checked out automatically by computers and the appropriate solution is chosen as true, unsupervised taggers do not require pre-tagged corpora.

Another classification in POS tagging has been done according to characteristics of POS taggers. There are three basic approaches in this classification:

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

Rule-based approaches generally use a lexicon and a list of hand-written grammatical rules of natural language. This method basically applies the rules to a word group including words with several possible word classes (e.g. both adjective and noun) for word class disambiguation (e.g. Greene & Rubin, 1971; Brill, 1992; Oflazer & Kuruoz, 1994; Voutilainen, 1995a).

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 (e.g. DeRose, 1988; Church, 1988; Cutting et al., 1992; Charniak, 1993).

Combination (Hybrid) tagging approach combines the advantages of both approaches to improve the overall performance of the tagging system (e.g. Cutting et al., 1992; Tapanainen & Voultilainen, 1994; Brill, 1995; Garside, 1987, 1997; Altinyurt et. al, 2006).

Research on part-of-speech tagging may have begun with the development of the Brown Corpus in 1960s, because first POS tagging studies were based on it. By creating a large corpus of English, the researchers aimed to make some analysis on the language in electronical environment. The Brown Corpus includes 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 in the Brown Corpus (Greene & Rubin, 1971). The tagger (TAGGIT) achieved an accuracy of 77%.

There are various POS tagging approaches that rely on stochastic methods, such as DeRose (1988), Church (1988), Charniak (1993), etc. 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 et al., 1992).

Brill's simple rule-based part-of-speech tagger achieved an accuracy of 96% in 1992 (Brill, 1992). The accuracy of this tagger was improved to 97.5% with some changes in 1994 by the author himself (Brill, 1994). Another well known research on rule-based POS tagging is the ENGTWOL tagger (Voutilainen, 1995b) that uses the Constraint Grammar approach of Karlsson et al. (1995).

The well-known hybrid tagging system is Brill's transformation-based tagger (Brill, 1995). This tagger determines ambiguous word classes using rules like other rule-based taggers. Also, it includes a machine learning mechanism like stochastic taggers, which provides rules to be constructed from the text.

CLAWS is also a hybrid 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 hybrid tagger developed by Tapanainen & Voultilainen (1994), which uses ENGCG: Constraint Grammar Parser of English (Karlsson et al., 1995) and the Xerox Tagger (Cutting et al., 1992) to tag the same document and combine the results independently.

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

In 2006, a composite approach for part of speech tagging in Turkish, which combines rule-based and statistical approaches with use of some characteristics of the language in terms of heuristics, such as frequencies and n-gram (unigram, bigram, and trigram) probabilities, was announced by Altinyurt et. al. In this work, it was shown that using hybrid approach increases the accuracy between 12% and 17% as to using only morphological analyzer.

Besides, TurPOS, a new rule-based part-of-speech tagger system that was developed for Turkish by Hallaç in 2007. TurPOS uses a text corpora produced by a morphological analyzer as the input document and a rule file that contains the list of grammatical Turkish rules. This makes the system usable for tagging other languages, by simply modifying the rule file according to the grammar of the language.

3.4 Other Works

There are many works in natural language processing area, such as author detection systems, translation systems between languages, spell checkers and correctors, etc.

In 2003, an automatic author detection system was developed by Diri & Amasyalı for Turkish. In this system, 18 authors were used for training by figuring out 22 style markers for each author. The success rate was detected as 84% in average, which was the highest rate until 2008. In 2008, a new study was released, which is called "*Determination of Author Characteristics*", developed by Gündü (Gündü, 2008). In this work, three different training sets and two test sets were generated from two Turkish newspapers with different specifications. In this system, 10 authors were used for training by figuring out 17 different style markers. While examining the text, all authors were scored by looking at the similarity between their texts and the unknown text. The success rate was detected as 86% and it was increased to 92% by adding a different parameter, called as Average Number of Wordforms in a Sentence (ANWS) parameter. Also, this work showed that using author scoring process with similarity coefficients and n-grams increases accuracy as 10%.

In 2004, researchers from Boğaziçi and Sabancı Universities built an open-source software platform called "*A natural language processing infrastructure for Turkish",* which served as a common infrastructure that can be used in the development of new applications involving the processing of Turkish (Say et. al.,2004). This platform has some variant features such as a lexicon, a morphological analyzer/generator, and a Definite Clause Grammar (DCG) parser/generator that translates Turkish sentences to predicate logic formulas, and a knowledge base framework. One of the developed applications by this study is a natural language interface for generating SQL queries and JAVA code.

For detecting misspelled words in Turkish texts, a study was released by using syllable n-gram frequencies in 2007 (Asliyan et. al.). In this work, three databases of syllable monogram, bigram and trigram frequencies are constructed using the syllables that are derived from five different Turkish corpora. Then, the system takes words in Turkish text as an input and gives the result for each word as "Misspelled Word" or "Correctly Spelled Word" by computing the probability distribution of words. If the probability distribution of a word is zero, it is decided that this word is misspelled. This system reached 97% success rate to detect misspelled words.

Also, many works on translation systems between Turkic languages were done, such as "*A Prototype Machine Translation System between Turkmen and Turkish*" (Tantuğ et. al., 2006), "*Machine Translation between Turkic Languages*" (Tantuğ et. al., 2007), "*Türk Dilleri Arası Çeviri Altyapısı (eng: An Infrastructure for Translation between Turkic Languages*)" (Alkım et. al., 2009).

CHAPTER FOUR
INFRASTRUCTURE AND DATABASE MODEL FOR RB-CorGen

4.1 Used Technologies

The project was developed as Windows application in Microsoft .NET Visual Studio 2005 (.NET Framework 2.0) environment by using the C# programming language, and MS SQL Server was used to store data such as lexicon, tags, etc. Additionally, Crystal Reports application is used in *Getting and Storing Data* step of the project. Some reports are designed by using this application, which are used for evaluation for downloaded documents.

The rule list should be stored in a format that can be parsed efficiently by the system. And also, the list should be in a form that users can read, understand and modify easily. Consequently, 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 more flexible and scalable. Considering this issues, rule files based on the XML (Extensible Markup Language) standards was designed in this project (Ray, 2003). The advantages for using an XML based rule file can be listed as the following:

- Increased human readability provides easy modification,
- A simple text editor is sufficient for rule list modifications instead of programs/tools,
- Parsing rules from text based file is faster than parsing from complex tables in a database,
- 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.

4.2 Used Tags

The tags for suffixes in Turkish Language have been created by linguists according to the structure of language. Full list of the tags, which are used in Stem / Root Parsing, are given in Appendix C3 and some samples are given in Table 4.1.

Table 4.1 Sample tags used in *Stem / Root Parsing* step

Suffix	**Tag**	**Meaning**	**Expression**				
-(y)Im	DuEKGr2T1	2. Grup 1. Tekil Kişi Eki *2.Group 1. Person Inflectional Suffix*	Du Dilbilgisi Ulamı *(Grammatical Category)*	E Eylem *(Verb)*	K Kişi *(Person)*	Gr2 2.Grup *(2. Group)*	T1 1.Tekil *(1st Singular)*
Ø Il/(I)n (I)ş (A, I)r /(A, I)t / (A,I)rt / Dır	DuEC (DuECEt DuECEdil DuECDonus DuECIstes DuECEttir)	Çatı Ekleri Grubu *(Voice Morphemes Group)* (Etken (*Active)* Edilgen (*Passive*) Dönüşlü *(Reflexive)* İşteş *(Reciprocal)* Ettirgen *(Causative)*)	Du Dilbilgisi Ulamı *(Grammatical Category)*	E Eylem *(Verb)*	C Çatı Eki *(Voice Morpheme)*		
-CA -lIk …	TBAA (TBAA-ca TBAA-lik...)	Addan Ad Yapan Yapım Eki Grubu *(Nominal Derivational Morphemes Group)*	TB Türetim Biçimbirim *(Derivational Morphemes*)	A Ad *(Noun)*	A Ad *(Noun)*		
-A/E -(A)k	TBAE (TBAE-a TBAE-k...)	Addan Eylem Yapan Yapım Eki *(Verbal Derivational Morphemes Group)*	TB Türetim Biçimbirim *(Derivational Morphemes*)	A Ad *(Noun)*	E Eylem *(Verb)*		

Abbreviations are used, which were chosen according to the semantic or structural meaning that the suffix adds to the word, for tags. As an example; “DuEKGr2T1” abbreviation indicates one of the group 2 people suffixes, which shows the first person singular suffix that can be added only after the suffixes of continuous, present, future and necessity tenses grammatically. The structure of the abbreviation is given as:

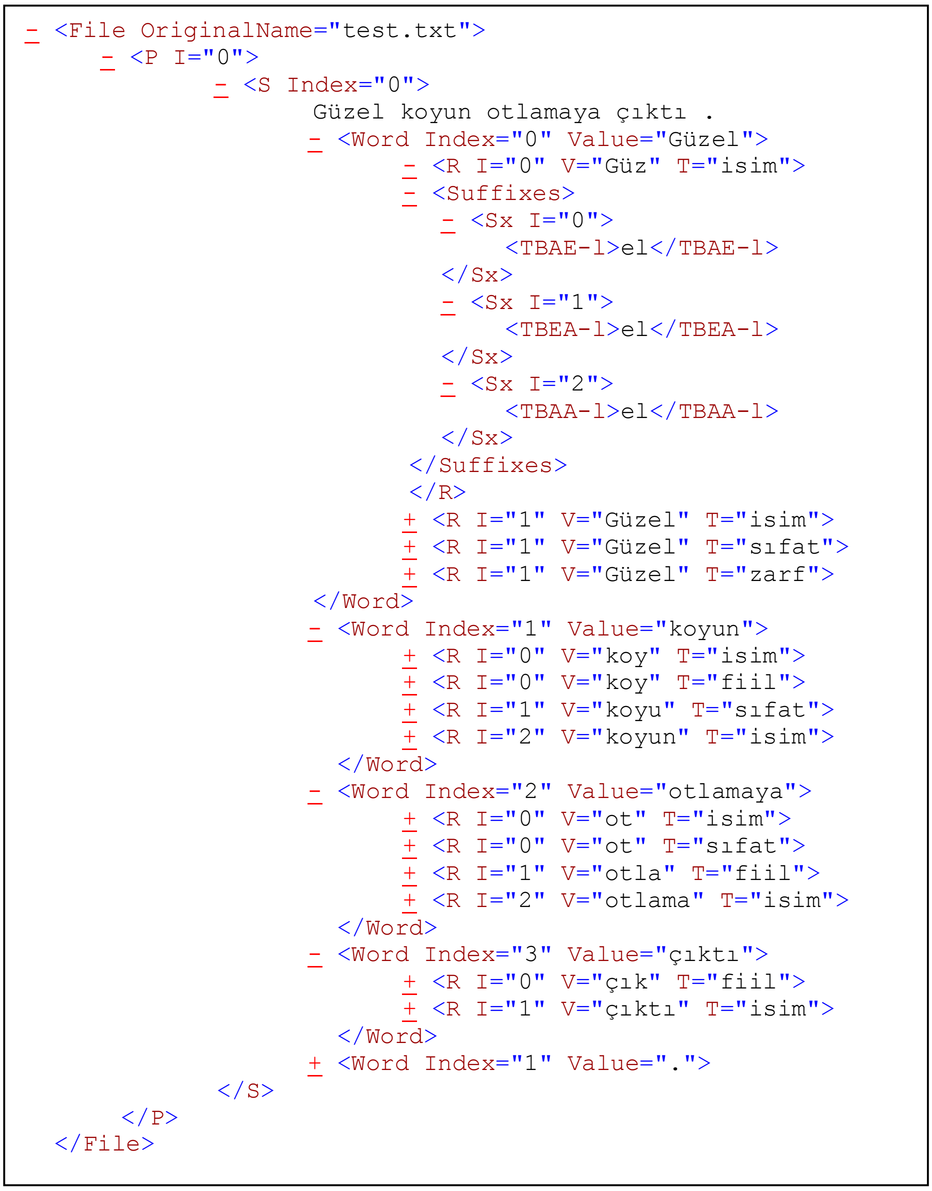
```
 Du                 -   E      - K          - Gr2        - T1
 Dilbilgisi Ulamı  - Eylem    - Kişi       - 2.Grup     -  1.Tekil
(Gramatical Suffix - Verb     - Person     - 2. Group   -  1.Singular)
```

All tags for the suffixes of the 2. Group People are given in Table 4.2.

Table 4.2 Tags for the suffixes of the group 2 people

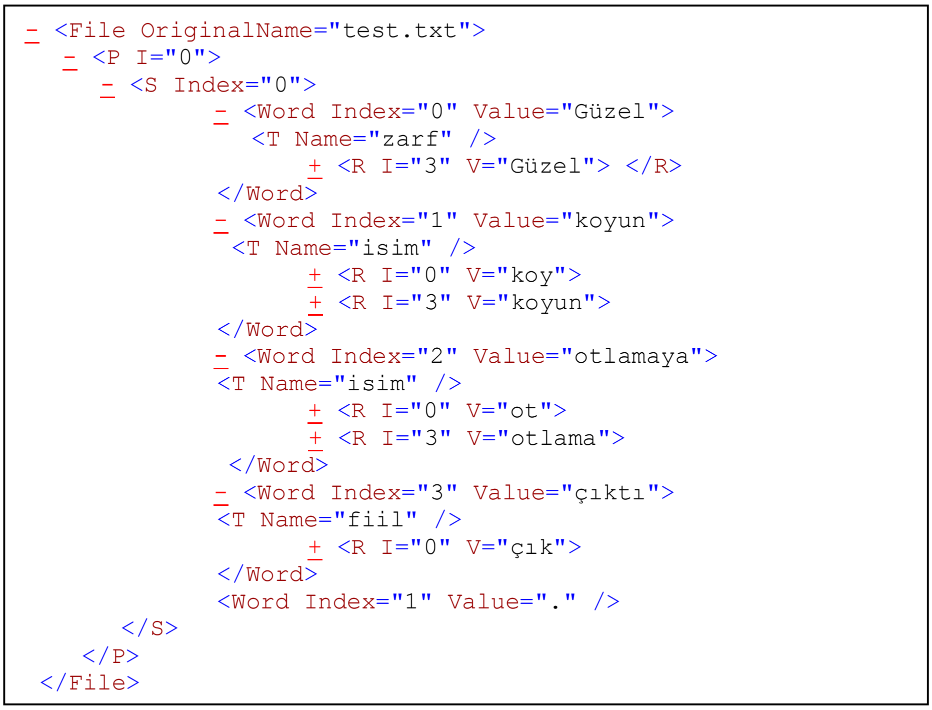
Suffix	**Tag**	**Expression**
{-(y)Im}	DuEKGr2T1	First Person Singular
{-sIn}	DuEKGr2T2	Second Person Singular
{-Ø}	DuEKGr2T3	Third Person Singular
{-(y)Iz}	DuEKGr2C1	First Person Plural
{-sIn-Iz}	DuEKGr2C2	Second Person Plural
{-lAr}	DuEKGr2C3	Third Person Plural

Sample parsed documents are given in Figure 4.1 and Figure 4.2, which are outputs of *Parsing Stems/Roots and Suffixes* and *POS Tagging* steps, and meanings of used tags are given in Table 4.3.



```
- <File OriginalName="test.txt">
     - <P I="0">
             - <S Index="0">
                    Güzel koyun otlamaya çıktı .
                  - <Word Index="0" Value="Güzel">
                        - <R I="0" V="Güz" T="isim">
                        - <Suffixes>
                           - <Sx I="0">
                                <TBAE-l>el</TBAE-l>
                             </Sx>
                           - <Sx I="1">
                                <TBEA-l>el</TBEA-l>
                             </Sx>
                           - <Sx I="2">
                                <TBAA-l>el</TBAA-l>
                             </Sx>
                           </Suffixes>
                           </R>
                        + <R I="1" V="Güzel" T="isim">
                        + <R I="1" V="Güzel" T="sıfat">
                        + <R I="1" V="Güzel" T="zarf">
                    </Word>
                  - <Word Index="1" Value="koyun">
                        + <R I="0" V="koy" T="isim">
                        + <R I="0" V="koy" T="fiil">
                        + <R I="1" V="koyu" T="sıfat">
                        + <R I="2" V="koyun" T="isim">
                      </Word>
                  - <Word Index="2" Value="otlamaya">
                        + <R I="0" V="ot" T="isim">
                        + <R I="0" V="ot" T="sıfat">
                        + <R I="1" V="otla" T="fiil">
                        + <R I="2" V="otlama" T="isim">
                      </Word>
                  - <Word Index="3" Value="çıktı">
                        + <R I="0" V="çık" T="fiil">
                        + <R I="1" V="çıktı" T="isim">
                      </Word>
                  + <Word Index="1" Value=".">
               </S>
        </P>
  </File>
```

Figure 4.1 Sample parsed document of *Parsing Stems/Roots and Suffixes* step including only one sentence



```
- <File OriginalName="test.txt">
  - <P I="0">
    - <S Index="0">
            - <Word Index="0" Value="Güzel">
               <T Name="zarf" />
                   + <R I="3" V="Güzel"> </R>
              </Word>
            - <Word Index="1" Value="koyun">
              <T Name="isim" />
                   + <R I="0" V="koy">
                   + <R I="3" V="koyun">
              </Word>
            - <Word Index="2" Value="otlamaya">
              <T Name="isim" />
                   + <R I="0" V="ot">
                   + <R I="3" V="otlama">
               </Word>
            - <Word Index="3" Value="çıktı">
              <T Name="fiil" />
                   + <R I="0" V="çık">
              </Word>
              <Word Index="1" Value="." />
      </S>
    </P>
  </File>
```

Figure 4.2 Sample parsed document of *POS Tagging* step including only one sentence

Table 4.3 Meanings of tags in XML file

Tag	Item (XML Value)	Meaning
File		File Section
	OriginalName	File name
P		Paragraph Section
	I	Index (Paragraph Number)
S		Sentence Section
	Index	Index (Sentence Number)
Word		Word Section
	Index	Index (Word Number)
	Value	Word Value
	T Name	Root Type
	T	Stem/Root Type
	R	Stem/Root
	I	Index (Root Number)
	V	Stem/Root Value
Suffixes		Suffixes Section
	Sx	Suffix Value
	I	Index (Suffix Number)

4.3 Rule Lists

In this study, the rules were collected by the linguists and stored in XML (Extensible Markup Language) formatted lists, which are used in sentence boundary detection, morphological analysis and POS tagging processes.

25 rules for *Sentence Boundary Detection* process were generated by the linguists to be used in this process. Full list of the rules, which are used in *Sentence Boundary Detection* step, are given in Appendix B1and some samples are also given in Figure 4.3.

```
<rules>
      <rule EOS="False">L.L</rule>
      <rule EOS="True">L.U</rule>
      <rule EOS="True">L.#</rule>
      ...
      <rule EOS="False">#.-</rule>
      <rule EOS="False">#.#</rule>
      <rule EOS="False">#.U</rule>
</rules>
```

Figure 4.3 Sample rules in rule list for *Sentence Boundary Detection* process

In this rule file, XML format is created in triple group (e.g. “L.L”). The meanings of characters used in this process are given in Chapter 4.

For concordance of suffixes, 16 rules were generated by the linguists and stored in a text file to be used in *Morphological Analysis* process to determine the stem / root and suffixes. The tags of suffixes are used to indicate successive rules. Full list of the rules, which are used in *Morphological Analysis* step, are given in Appendix B2 and some samples are also given in Figure 4.4.

```
TB,E,A,S
E,DuEC,DuEKipYet,DuEOlz,YS,DuEZ\DuEG\DuEKip\,YS,K,DuEK,
E,DuEC,DuEKipYet,DuEOlz,YS,Ytu,YS,K,DuEK,
A,YS,DuASay,YS,DuAUy,YS,DuADur,
E,DuEC,DuEOlz,Ytu,YS,DuASay,YS,DuAUy,YS,DuADur,
```

Figure 4.4 Sample rules in rule list for *Parsing Stems/Roots and Suffixes* process

As an example; the rule “E,DuEC,DuEKipYet,DuEOlz,YS,Ytu,YS,K,DuEK,” indicates that; a verb can get a suffix from DuEC (Eylem Çatı Ekleri *(Voice Morphemes)*) group, and after that it can get suffixes from DuEKipYet (Yeterlilik Eki *(abilitative suffix)*), DuEOlz (Eylem Olumsuzluk Eki *(Negation Suffix)*), YS (Yardımcı Ses *(Buffer Sound)*), K (Koşaç – Ek fiil *(Verb Compunds)*), DuEK- (Eylem Kişi Çekim Ekleri *(Personal Suffixes)*) suffix groups respectively. The meanings of tags used in this process are given in Chapter 4.

For *Rule-Based POS Tagging* process, 85 rules are generated to determine concordance of suffixes by the linguists and stored in an XML file. Full list of the rules, which are used in *Rule-Based POS Tagging* step, are given in Appendix B3 and some samples are also given in Figure 4.5.

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  .............
  <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="v" />
    <Item ItemType="isim" />
  </Rule>
  ................
</Document>
```

Figure 4.5 Sample rules in rule list for *POS Tagging* process

A rule in the rule list is basically 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 POS Tagging Module is *word order* rules, which is called as “sözdizim” in Turkish. The third attribute, *RuleState,* defines whether syntax of word types' sequences that the rule contains is applicable or not. 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 Turkish. Each rule element has sub-elements (items), which correspond to words or punctuation items in sentences. *Item* element has only one attribute called *ItemType*, which defines the word type of an item. If the item is a punctuation mark, the *ItemType* attribute includes its type as punctuation given in Table 4.5, which shows the types used in POS Tagging Module, including the punctuation marks that can take place in sentences.

Table 4.5 The word types used in the POS Tagging System and corresponding descriptions in English

Parts of Speech	**Description**
sıfat	adjective
isim	noun
zamir	pronoun
fiil	verb
zarf	adverb
edat	preposition
sayı	number
nokta	dot
virgül	comma
ikinokta	colon
noktalıvirgül	semicolon

4.4 Database Model

In order to find stem/root possibilities, a wordform is analyzed by starting with first character from left and comparing with the words taken from TDK and stored in the database. Also the suffixes with used tags were stored in database. The lexicon and suffixes were stored in 7 different tables:

- Kökler
- KöklerSanal
- Gövde
- Kelimeler
- Grup
- Ek
- Ekler

Database diagram of the project is given in Figure 4.6.

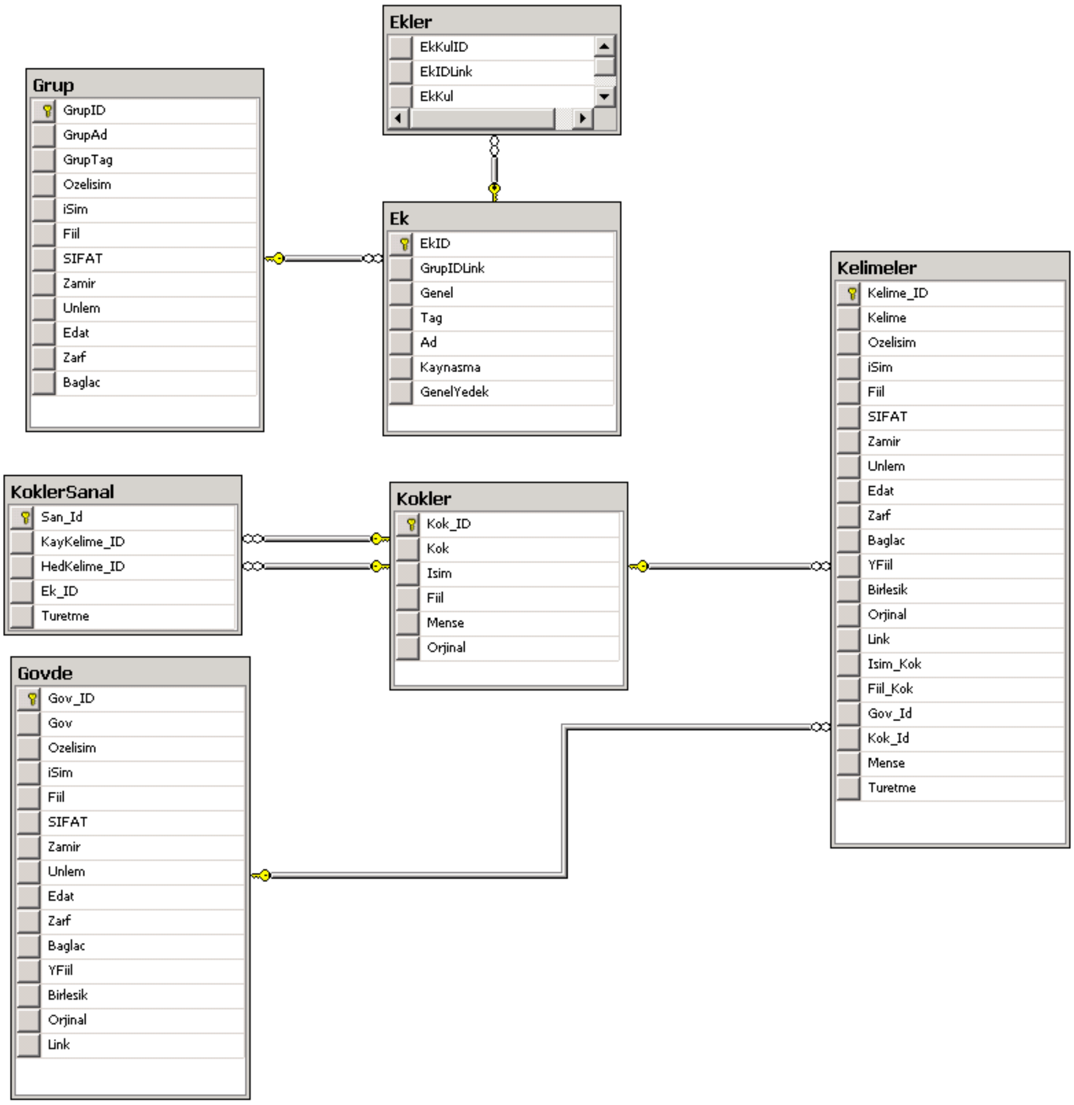


Figure 4.6 Database diagram of the Automatic Corpus Generation project

4.4.1 The Table "Kokler"

The lexicon of Turkish, which was obtained from TDK, was stored in this table. Some of the roots in this table are given in Table 4.6 and in Appendix C.2.1.

Table 4.6 Sample rootsin the table "Kokler"

Kok_ID	Kok	Isim	Fiil	Mense	Orjinal
16869	ab	True	False	Far.	True
16870	aba	True	False	Ar.	True
16871	abadî	True	False	Far.	True
16872	abajur	True	False	Fr.	True
16873	abaküs	True	False	Fr.	True
16874	aban	False	True		True
16875	abandone	True	False	Fr.	True
16876	abanî	True	False	Far.	True
16877	abanoz	True	False	Far.	True
33457	abı	True	False	Ar.	False

In this table, there are 6 fields:

- Kok_ID: Unique index number of root.
- Kok: The root value.
- Isim: A Boolean data, which indicates whether the root is a Noun or not.
- Fiil: Boolean data, which indicates whether the root is a Verb or not.
- Mense: Data, which indicates the origin of root.
- Orjinal: Boolean data, which indicates whether the root is original or not, which means whether the root is changed according to the vowel changing rules or not.

4.4.2 The Table "KoklerSanal"

The derived roots in the lexicon by using vowel changing rules were stored in this table. Some of the roots are given Table 4.7.

Table 4.7 Sample data in the "Kokler" table

San_Id	KayKelime_ID	HedKelime_ID	Turetme
157	16870	33457	aba->abı Darlama_Önceki_Düz
158	16875	33458	abandone->abandonü Darlama_Önceki_Yuvarlak
159	16878	33459	abart->abard Yumuşama
160	16880	33460	abat->abad Yumuşama
161	16888	33461	abide->abidi Darlama_Önceki_Düz
162	16890	33462	abiye->abiyi Darlama_Önceki_Düz
163	16891	33463	abla->ablı Darlama_Önceki_Düz
164	16892	33464	ablak->ablağ Yumuşama
165	16894	33465	ablatya->ablatyı Darlama_Önceki_Düz
166	16896	33466	abluka->abluku Darlama_Önceki_Yuvarlak
167	16897	33467	abone->abonü Darlama_Önceki_Yuvarlak
168	16899	33468	aborda->abordu Darlama_Önceki_Yuvarlak

In this table, there are 4 fields:

- San_ID: Unique index number of root.
- KayKelime_ID: Index of root, from which the new root is derived. It is used as foreign key of the field “Kok_ID” in “Kokler” table.
- HedKelime_ID: Index of new root, which is derived. It is used as foreign key of the field “Kok_ID” in “Kokler” table.
- Turetme: The data about which kind of derivation rule is applied to this root.

If a root is derived, the data in “Original” field of “Kokler” table is recorded as “false”. The original value of any root can be found by controlling the unique index number, “Kok_ID”, in “HedKelime_ID” field of the “KoklerSanal” table. For example, the root *abı*, which has unique index as 33457 in table “Kokler” (Table 6), is derived from the root *aba*, which has unique index as 16870, by using vowel changing rule “aba->abı Darlama_Önceki_Düz” and stored in the table “KoklerSanal” with unique index 157.

4.4.3 The Table “Govde”

The stems in the lexicon in Turkish were stored in this table. Some of the stems are given in Table 4.8 and in Appendix C.2.2.

Table 4.8 Sample data from the “Govde” table in the database

Gov_ID	*Gov*	*Ozel isim*	*isim*	*Fiil*	*Sifat*	*Zamir*	*Unlem*	*Edat*	*Zarf*	*Baglac*	*YFiil*	*Birlesik*	*Orjinal*
1	a	False	False	False	False	False	True	False	False	False	False	False	False
2	ab	False	True	False	False	False	False	False	False	False	False	False	False
3	aba	True	True	False	True	False	False	False	False	False	False	False	False
4	aba güreşi	False	True	False	False	False	False	False	False	False	False	True	False
5	abac	False	True	False	False	False	False	False	False	False	False	False	False
6	abacılık	False	True	False	False	False	False	False	False	False	False	False	False
7	abadi	False	True	False	False	False	False	False	False	False	False	False	False
8	abajur	False	True	False	False	False	False	False	False	False	False	False	False
9	abajurcu	False	True	False	False	False	False	False	False	False	False	False	False
10	abajurculuk	False	True	False	False	False	False	False	False	False	False	False	False

In this table, there are 14 fields:

- Gov_ID: Unique index of stem.
- Gov: The value of stems.
- Ozelisim: Boolean data, which indicates whether the stem is a special name or not.
- Isim: Boolean data, which indicates whether the stem is a name or not.
- Fiil: Boolean data, which indicates whether the stem is a verb or not.
- Sifat: Boolean data, which indicates whether the stem is an adjective or not.
- Zamir: Boolean data, which indicates whether the stem is a pronoun or not.
- Unlem: Boolean data, which indicates whether the stem is an exclamation or not.
- Edat: Boolean data, which indicates whether the stem is a preposition or not.
- Zarf: Boolean data, which indicates whether the stem is an adverb or not.
- Baglac: Boolean data, which indicates whether the stem is a conjunction or not.
- YFiil: Boolean data, which indicates whether the stem is an auxiliary verb or not.
- Birlesik: Boolean data, which indicates whether the stem is a compound name or not.
- Orjinal: Boolean data, which indicates whether the root is original or not that means whether the root is changed according to the vowel changing rules.

4.4.4 The Table "Kelimeler"

All words in the lexicon, stems and roots, in Turkish were stored in this table. Some of the words are given in Table 4.9.

Table 4.9 Sample data from the table “Kelimeler”

ID	**Kelime**	**Ozel**	**Isim**	**Fiil**	**Sıfat**	**Zamir**	**Unlem**	**Edat**	**Zarf**	**Baglac**
1	a	False	False	False	False	False	True	False	False	False
2	ab	False	True	False	False	False	False	False	False	False
3	aba	True	True	False	True	False	False	False	False	False
4	aba güreşi	False	True	False	False	False	False	False	False	False
5	abacı	False	True	False	False	False	False	False	False	False
6	abacılık	False	True	False	False	False	False	False	False	False
7	abadi	False	True	False	False	False	False	False	False	False
8	abajur	False	True	False	False	False	False	False	False	False
9	abajurcu	False	True	False	False	False	False	False	False	False
10	abajurculuk	False	True	False	False	False	False	False	False	False
11	abajurlu	False	False	False	True	False	False	False	False	False
12	abajursuz	False	False	False	True	False	False	False	False	False
13	abaküs	False	True	False	False	False	False	False	False	False
14	abal	False	False	False	True	False	False	False	False	False
15	aban	False	False	True	False	False	False	False	False	False

There are 21 fields in this table, first 14 of which are same as in the table “Govde”. The different fields from “Govde” table are:

- Link: Boolean data, which indicates whether the word is derived from another word by using vowel changing rules or not.
- Isim_Kok: Boolean data that indicates whether the type of the root, from which this stem is derived, is a name or not.
- Fiil_Kok: Boolean data that indicates whether the type of the root, from which this stem is derived, is a verb or not.
- Gov_Id: The unique index number that indicates the index of the stem, from which this word is derived.
- Kok_Id: Unique index number that indicates the index of the root, from which this stem is derived.
- Mense: Data, which indicates the origin of the word.
- Turetme: Data, which shows the vowel changing rule such as `fokurdama-> fokurdamı` Darlaşma_Önceki_Düz, `forvet->forved` Yumuşama etc.

4.4.5 The Table “Grup”

The meanings of tags, which are used to define the rules in the process of parsing stem/root, are stored in this table. This table has 12 fields, which are named as the “word types” in a language, are used to indicate the word type that the suffix group can be added to.

The group names in table “Grup” are given in Table 4.10.

Table 4.10 Sample Data from table “Grup”

Grup ID	*GrupAd*	*GrupTag*	*OzelIsim*	*Isim*	*Fiil*	*Sıfat*	*Zamir*	*Unlem*	*Edat*	*Zarf*	*Baglac*
1	Sayı	DuASay	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	Uyum	DuAUy	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	Durum	DuADur	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	Cinsiyet	DuACins	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	Zaman	DuEZ	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	Görünüş	DuEG	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	Kiplik	DuEKip	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8	Çatı	DuEC	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	Olumsuzluk	DuEOlz	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	Kişi 1. Grup	DuEKGr1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
11	Kişi 2. Grup	DuEKGr2	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
12	Kişi 3. Grup	DuEKGr3	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	Kişi 4. Grup	DuEKGr4	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	Koşaç	K	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	Adlaştırma Yantümcesi	YtuAdl	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

4.4.6 The Table “Ek”

The tags that are used in the process of parsing stem/root are stored in a table called “Ek”. The groups and general notations of the suffixes are stored in this table, some of which are given in Table 4.11.

Table 4.11 Sample data from the table "Ek"

Ek ID	GrupID Link	Genel	Tag	Ad	Kaynasma	GenelYedek
60	14	{-DI}	KDi	Geçmiş Zaman Koşacı	y	{-(y)DI}
61	14	{-mI}	KMis	Tantsallık Koşacı	y	{-(y)mI}
62	14	{-sA}	KSa	Koşul Koşacı	y	{-(y)sA
63	15	{-DIk}	YtuAdlDik	Adlaştırma Belirticisi	NULL	{-DIk}
64	15	{-AcAk}	YtuAdlAcak	Adlaştırma Belirticisi	y	{-(y)AcAk}
65	15	{-mA}	YtuAdlMa	Adlaştırma Belirticisi	NULL	{-mA}
66	15	{-mAk}	YtuAdlMak	Adlaştırma Belirticisi	NULL	{-mAk}
67	16	{-DIk}	YtuOrDik	Sıfat Yantümcesi Belirticisi (Ortaç)	NULL	{-DIk}
68	16	{-An}	YtuOrAn	Sıfat Yantümcesi Belirticisi (Ortaç)	y	{-(y)An}
69	16	{-AcAk}	YtuOrAcak	Sıfat Yantümcesi Belirticisi (Ortaç)	y	{-(y)AcAk}
70	16	{-I}	YtuOrIs	Sıfat Yantümcesi Belirticisi (Ortaç)	y	{-(y)I}

There are 7 fields in this table:

- Ek_ID: Unique index of suffix.
- GrupIDLink: The unique index number that indicates the index of the group, to which this suffix belongs.
- Genel: The general notation of the suffix.
- Tag: The tag for suffix that is used in the XML output of *Stem/Root Parsing* process.
- Ad: Expression of tag.
- Kaynasma: The buffer letter if it is needed while the tag is added to any word.
- GenelYedek: The general notation for suffix with possible buffer letter that is used in the XML output of *Stem/Root Parsing* process.

4.4.7 The Table “Ekler”

All suffixes, with their modified versions according to morphophonemic processes vowel and consonant harmonies, are stored this table. Some of the suffixes are given in Table 4.12.

Table 4.12 Sample data from the table “Ekler”

EkKulID	**EkIDLink**	**EkKul**
816	177	üt
817	177	d
818	177	id
819	177	ud
820	177	üd
821	177	t
822	177	d
823	178	tay
824	178	tey
825	179	tay
826	179	tey
827	180	t
828	180	ti
829	180	tu
830	180	tü
831	181	av
832	181	ev
833	181	v

There are 3 fields in this table:

- EkKulID: Unique index of suffix.
- EkIDLink: The index number that indicates the index of the suffix in “Ek” table, from which this suffix derived by using vowel changing rules.
- EkKul: The suffix as it can be used in the language.

CHAPTER FIVE
ALGORITHMS AND SOFTWARE STRUCTURE OF RB-CorGen

Turkish is an ‘agglutinative language’ like Finnish, Hungarian, Quechua and Swahili, where it is classified where new words are formed by adding suffixes to the end of roots (Appendix A). In Turkish, there are grammatical rules for suffixes that which of them may follow which other and in what order they will be. By this concatenation the meaning of words are changed or extended. This suffix concatenation can result in relatively long words, which are frequently equivalent to a whole sentence in English (e.g. Osmanlılaştıramadıklarımızdansınız (*eng: You were of those whom we might consider not converting into an Ottoman.*)). Besides the rules for suffixes, Turkish has many grammatical rules, such as word types ordering, structure of compounds, etc. The corpus development processes are defined as follows:

1. Getting and Storing Data
2. Sentence Boundary Detection
3. Morphological Analysis
    a. Finding stem and inflectional suffixes
    b. Finding root and derivational suffixes
4. Part-of-Speech Tagging

The main block diagram of corpus generation processes is given in Figure 5.1.

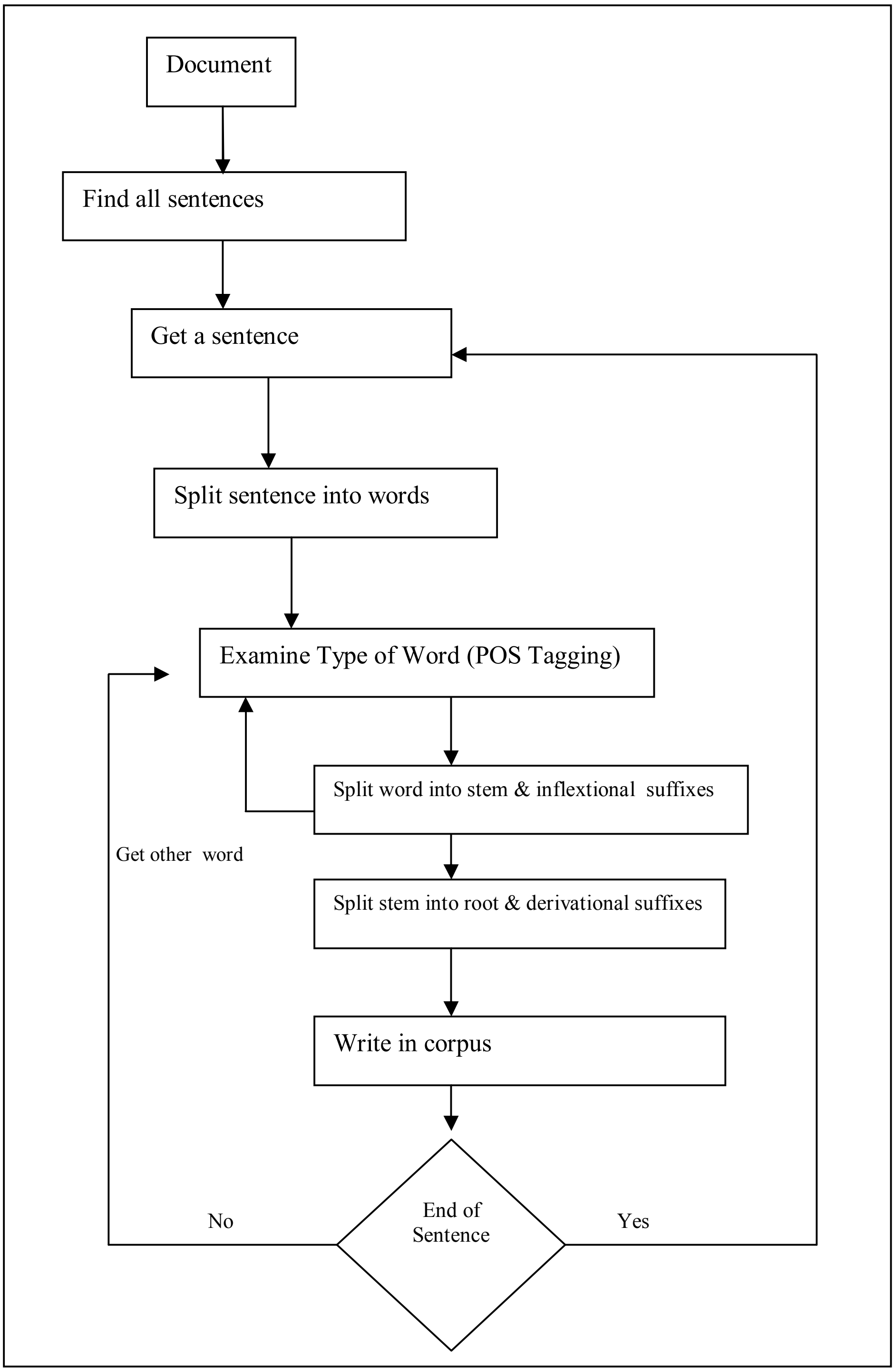

Document
Find all sentences
Get a sentence
Split sentence into words
Examine Type of Word (POS Tagging)
Split word into stem & inflextional suffixes
Get other word
Split stem into root & derivational suffixes
Write in corpus
No
End of Sentence
Yes

Figure 5.1 Block diagram of processes in generating corpus

5.1 Getting and Storing Data

At first, the electronic data should be taken from web or any scanned documents and stored in a database to be able to use efficiently in the project as data set. For this purpose, the project “Döküman İndirici *(en: Document Downloader)*” that was developed by Kızılay (2009) is used.

In this project, the electronic data is taken from web by URL links of the newspapers and stored in database to be able to use efficiently in the project. For classifying texts of corpus, a database model, which supports 6 different document types such as newspaper, report, magazine, book, parliamentary report and official gazette, was designed. Metadata of documents such as URL of document, header of document, size of document etc is stored in this database model. For collecting electronic data, a module that downloads articles from 5 different newspapers “Milliyet”, “Hürriyet”, “Radikal”, “Vatan”, ”Akşam” was implemented. Downloaded articles are stored in a storage media and also metadata of these documents are stored in database.

Besides, some different reports about the downloaded data can be generated by using this application.

The main interface of the *Document Downloader* application is given in Figure 5.2.

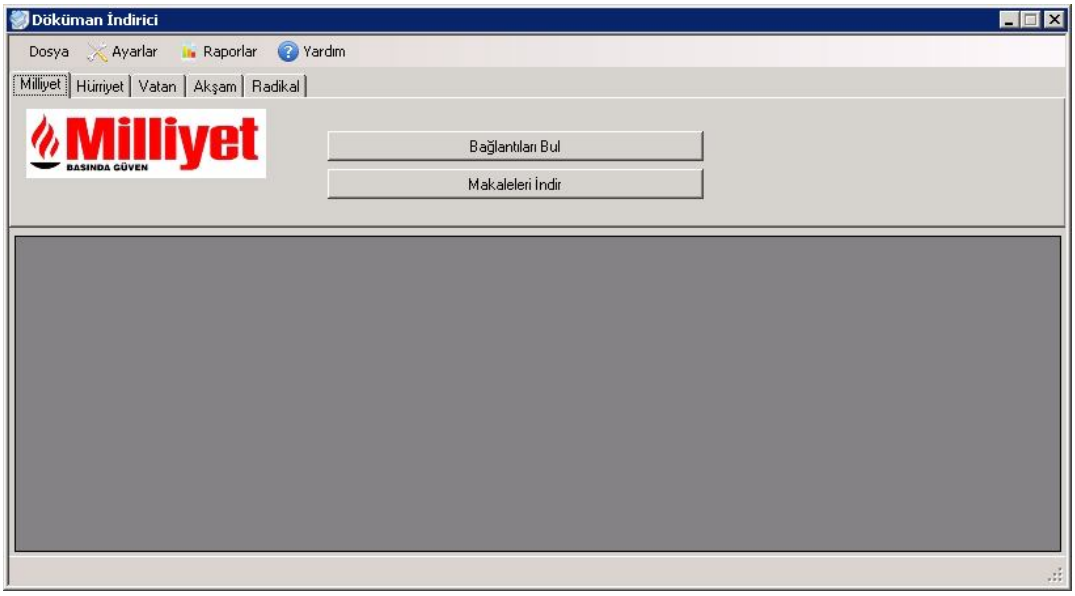


Figure 5.2 Main interface of the *Document Downloader* application

5.2 Rule-Based Sentence Boundary Detection

Turkish sentences generally end with known punctuations such as “.”, “…”, “!”, “?”. A punctuation mark which is commonly used as an end of sentence determination symbol may also be used in an abbreviation, as a decimal point in a number, in an e-mail addresses etc. This situation is called “ambiguity”. The sentence boundary determination process becomes harder with the increasing amount of ambiguities.

In Turkish there are some ambiguities in finding sentence boundaries like in any other languages. For example;

- Uluslar, bu ekonomik buhran sonucunda 2. Dünya Savaşı’nı yaşamıştır.
  Nations lived the 2.World War as a reason of this economic crisis.
- Bu sezon kaybedilen maç sayısı 2. Dünya Kupası’na katılma şansı azalıyor.
  The lost game number in this season is 2. The World Cup attendance chance decreases.

In the first sentence, the "." character is used for enumerate, but in the second sentence it indicates end of sentence. And after ".", both of them have the same word that begins with uppercase. So, this causes an ambiguity for the process of finding end of sentence. In this study, in order to solve such kinds of ambiguities, different solutions were developed.

In order to find end of sentences, a rule list is created and stored in XML (Extensible Markup Language) format (Ray, 2003). Full list of the rules are given in Appendix B1 and some samples are given in Figure 5.3.

```
<rules>
      <rule EOS="False">L.L</rule>
      <rule EOS="True">L.U</rule>
      <rule EOS="True">L.#</rule>
      ...
      <rule EOS="False">#.-</rule>
      <rule EOS="False">#.#</rule>
      <rule EOS="False">#.U</rule>
</rules>
```

Figure 5.3 Sample rules in rule list for sentence boundary detection

XML format is created in triple group (e.g. "L.L"). The first character indicates the first character of the word before punctuation mark, second character is the punctuation mark itself, and the third character indicates the first character of the word after punctuation mark (Figure 5.4).

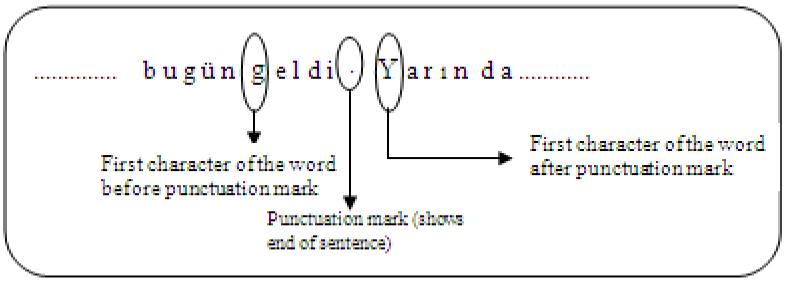

.... bugün geldi . Yarın da ....
First character of the word before punctuation mark
Punctuation mark (shows end of sentence)
First character of the word after punctuation mark

Figure 5.4 The characters used in the rules

The meanings of characters, which are used in sentence boundary rules, are given in Table 5.1.

Table 5.1 Meanings of the characters in the sentence boundary rule list

Character	Meaning	Character	Meaning
.	EOS punctuations (. … ! ?)	(	(
L	Lowercase	)	)
U	Uppercase	/	/
#	Number	‘	‘
?	Any character	“	“
-	-		
,	,		

First process to find sentences is paragraph determination. If a text stream has “enter (‘\n’)” character, this character is assumed as the end of paragraph and all text from the beginning to this character are taken as a “paragraph”. After a paragraph is determined, characters are taken one by one and checked if it is one of the punctuation marks used for sentence boundary rule list (“.”, “…”, “!”, “?”). In an ordinary situation, only the end of sentence punctuation marks might be good enough to determine the sentence boundaries. However, besides the complicated structure of the Turkish, there may be many ambiguities caused by the punctuation marks such as using them for abbreviations, e-mail and web addresses, etc.

In order to solve ambiguities caused from abbreviations, an additional rule list, in which abbreviations are given, is used. The abbreviation list was taken from Turkish Linguistic Association (TDK) and accepted as is. The list was also defined in XML format. Full list of the abbreviations are given in Appendix C1 and some samples are given in Figure 5.5.

```
<abbrevations>
<abbr>              A            </abbr>
<abbr>              AA           </abbr>
<abbr>              AAFSE        </abbr>
<abbr>              AAM          </abbr>
<abbr>              AB           </abbr>
<abbr>              ABD          </abbr>
<abbr>              ABS          </abbr>
<abbr>              ADSL         </abbr>
<abbr>              AET          </abbr>
<abbr>              ……           </abbr>
<abbr>              HAVAŞ        </abbr>
<abbr>              HDD          </abbr>
<abbr>              hek          </abbr>
<abbr>              ……           </abbr>
<abbr>              zf           </abbr>
<abbr>              zm           </abbr>
<abbr>              ZMO          </abbr>
<abbr>              zool         </abbr>
<abbr>              ……           </abbr>
</abbrevations>
```

Figure 5.5 Sample abbrevations in the abbreviation list

Also, the roman numbers were added into the abbreviation list (Figure 5.6).

```
<abbrevations>
<abbr>          ……            </abbr>
<abbr>          I             </abbr>
<abbr>          V             </abbr>
<abbr>          IX            </abbr>
<abbr>          X             </abbr>
<abbr>          XV            </abbr>
<abbr>          ……            </abbr>
<abbr>          XXX           </abbr>
<abbr>          ……            </abbr>
</abbrevations>
```

Figure 5.6 Sample roman numbers in the abbrevation list

Abbreviations and rule lists were written in two files by using XML standard and separated from the program source codes to allow users making changes in these files easily and independently from the program source. Therefore, the adaptation of any Turkish dialects will be easy.

The “ ” (space) character is searched after the “.” (dot) character in order to define e-mail and web addresses, such as www.deu.edu.tr. If the character after “.” (dot) character is not a “ ” (space) character and in lowercase, it is assumed as e-mail or web address.

In conversation texts, conversations are indicated by special character, “-” (hypen), after the character “:” (colon), which is also used for bulleting. These characters cause ambiguity, because of being used for bulleting. The sentences come after the “:” character assumed as if it belongs to one sentence; all lines were read and combined together as one sentence. The flow diagram of the algorithm is given in Figure 5.7.

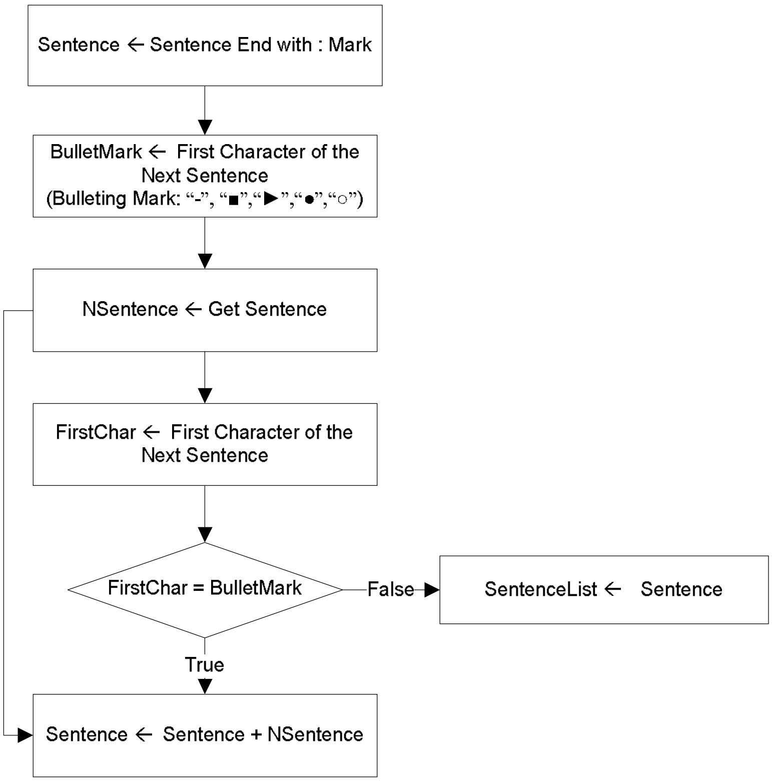

Sentence ← Sentence End with : Mark
BulletMark ← First Character of the Next Sentence (Bulleting Mark: “-”, “■”,“►”,“•”,“○”)
NSentence ← Get Sentence
FirstChar ← First Character of the Next Sentence
FirstChar = BulletMark
False
SentenceList ← Sentence
True
Sentence ← Sentence + NSentence

Figure 5.7 Flow diagram of the Bulleting Algorithm

Since these blocks of sentences cause ambiguity, and they cannot be separated from conversation texts, they were asked to the user to determine the type of them in the program. Then, the user defines the type of the sentence block as “conversation sentences”, tagged as DLG (Dialog), or “bulleting text”, tagged as BL (Bulleted List). This made the program more reliable.

The main flow diagram of the sentence boundary detection algorithm is given in Figure 5.8.

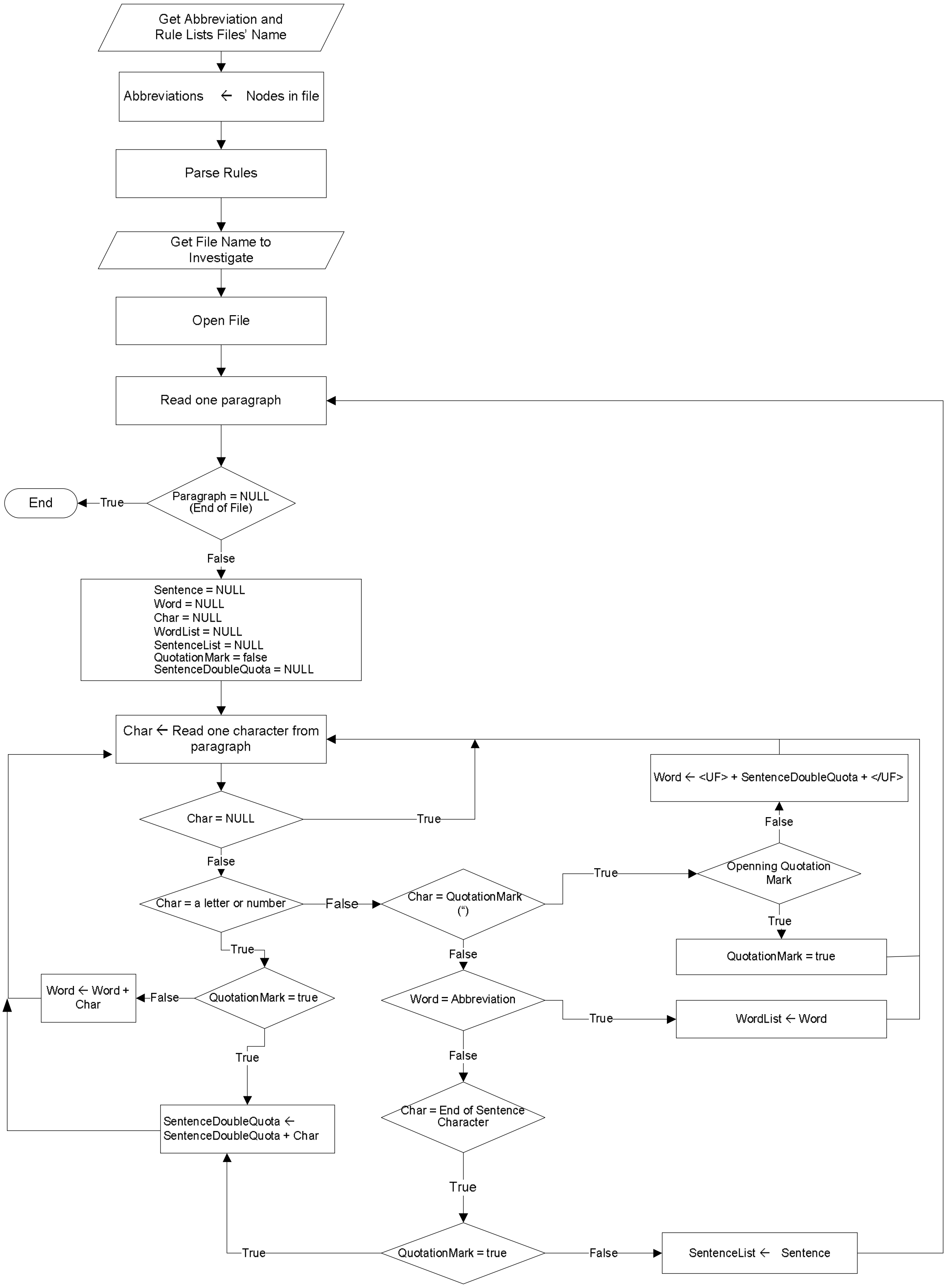


Figure 5.8 Flow diagram of sentence boundary detection algorithm

5.3 Morphological Analysis

A Turkish word is analyzed according to its root and the suffixes, which are added after the root. For word analysis, at first the sentences and then words should be determined. The output of the *Rule-Based Sentence Boundary Detection (RB-SBD) module* is used as input in the *Rule-Based Morphological Analysis (RB-MA) (or Word Detector (RB-WD)) Algorithm*. The main steps of the algorithm are as follows:

1. Take words from the text that will be analyzed.
2. Find all meaningful stem possibilities by using "Enhanced Search Method - From Left to Right (ESM)" (Çebi vd. 2006), define *Stem Possibilities Space*.
3. Find the inflectional suffix possibilities according to each root possibility, define *Inflectional Suffix Possibilities Space*.
4. Find all meaningful root possibilities of all stems by using ESM, define *Root Possibilities Space*.
5. Find the derivational suffix possibilities according to each root possibility of a stem, define *Derivational Suffix Possibilities Space*.
6. Eliminate wrong combinations in *Stem / Root and Suffix Possibilities Spaces*.
7. Save all the combinations in a text file to make analyzing a word easier in the future.
8. If the words, which will be analyzed, are finished go to step 9, else go to step 1.
9. Save all the combinations and present the result to the user in an XML file.

The root and stem lists are taken from Turkish Linguistic Association (Türk Dil Kurumu, TDK). These lists are stored in database with their modified versions according to the morphophonemic processes that cause root deformations, which rules are given in Appendix A.4. Sample modified roots and stems are given in Appendix C.2.3.

Flow diagram of *Stem / Root Parsing Algorithm* is given in Figure 5.9.

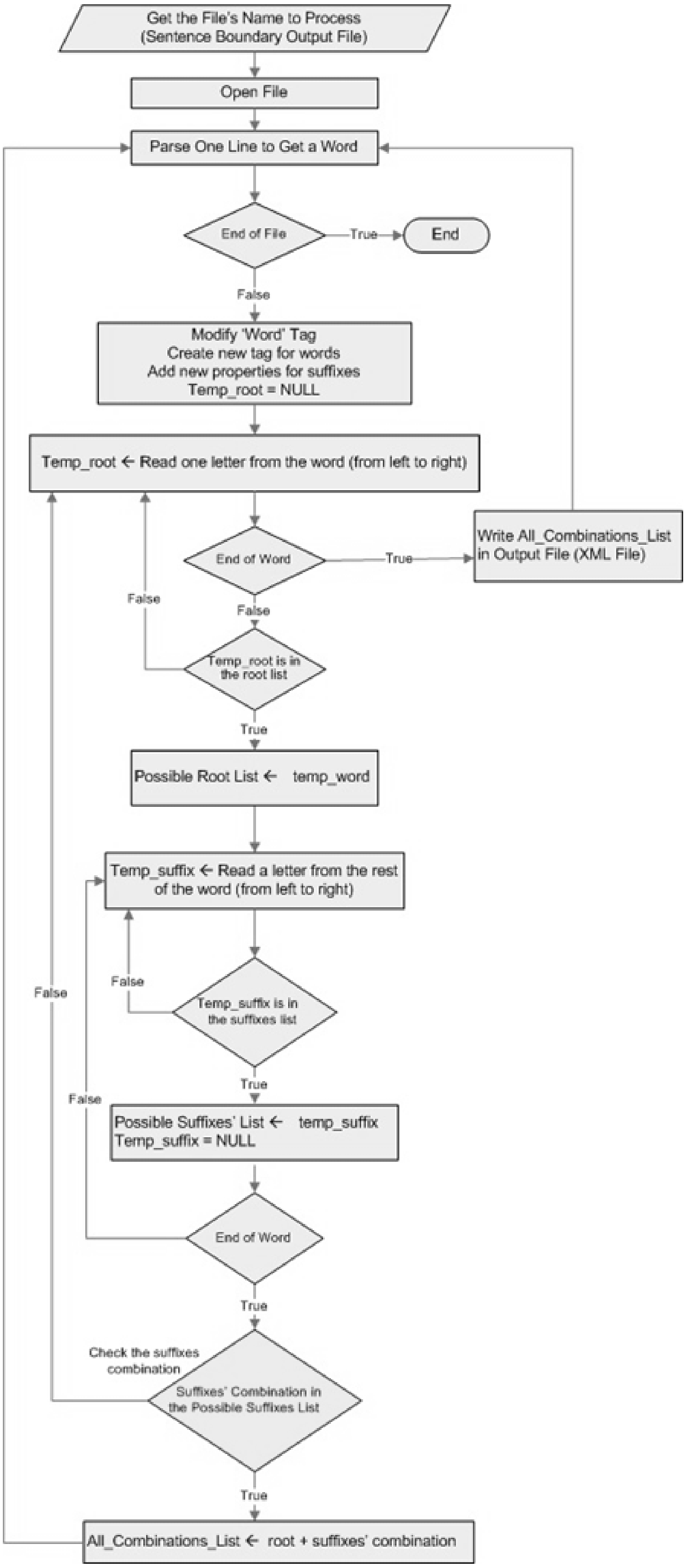

Get the File's Name to Process
(Sentence Boundary Output File)
Open File
Parse One Line to Get a Word
End of File
True
End
False
Modify 'Word' Tag
Create new tag for words
Add new properties for suffixes
Temp_root = NULL
Temp_root ← Read one letter from the word (from left to right)
End of Word
True
Write All_Combinations_List in Output File (XML File)
False
False
Temp_root is in the root list
True
Possible Root List ← temp_word
Temp_suffix ← Read a letter from the rest of the word (from left to right)
False
Temp_suffix is in the suffixes list
False
True
False
Possible Suffixes' List ← temp_suffix
Temp_suffix = NULL
End of Word
True
Check the suffixes combination
Suffixes' Combination in the Possible Suffixes List
True
All_Combinations_List ← root + suffixes' combination

Figure 5.9 Flow diagram of Rule-Based Morphological Analysis Algorithm

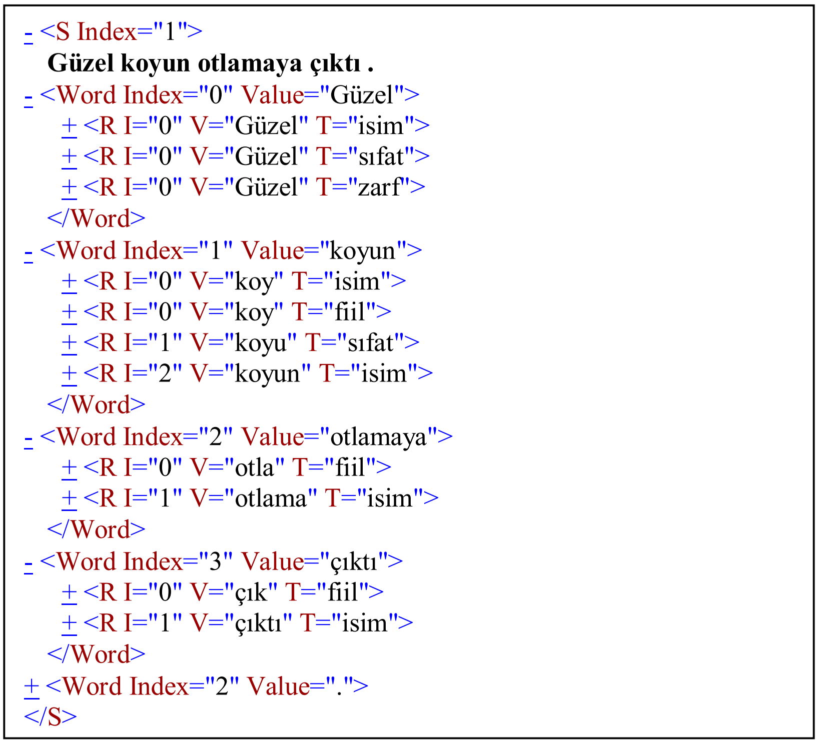
In this algorithm, two steps, *Parsing Stems and Inflectional Suffixes* and *Parsing Roots and Derivational Suffixes*, are applied consecutively. In the first step, the examined wordform is searched in the stems lexicon iteratively by pruning a letter from left to right at each step. If splited character array matches with any of the stems in the lexicon, the rest of the wordform is also checked by pruning a letter from left to right iteratively if it is a suffix or not in inflectional suffixes list, which is collected by linguists in Dokuz Eylul University, College of Social Science and Literature Linguistic Department. If both the stem and suffixes parts of the wordform are found in the lists, they are tagged and stored as possible stem and suffixes of the examined wordform. The search process is repeated until a single letter is left from the wordform.

For example, three different stems and suffix combination possibilities were found in finding stem process of the wordform "koyun" (Table 5.2).

Table 5.2 Example application of Enhanced Search Method- From Left to Right

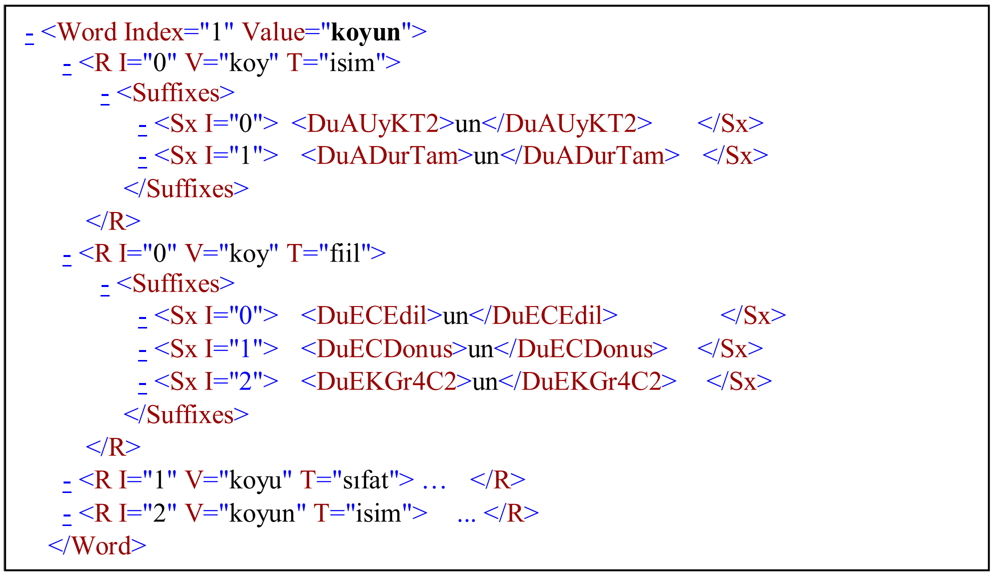
	Possible Stem	**Possible Buffer Letter**	**Possible Suffixes**	**Result**
Step 1	K		o y u n	False
	K	o	y u n	False
Step 2	Ko		y u n	False
	Ko	y	u n	False
Step 3	Koy		u n	True
	Koy	u	n	False
Step 4	Koyu		n	True
	Koyu	n		False
Step 5	Koyun			True

In the first step, *k* is taken as stem, *o* is taken as possible buffer letter, *yun* and *oyun* is searched as character by character in the suffixes list. Since there are no accepted rules for these combinations, they are eliminated and search process continues with next step. This process continues until there is no character assumed as suffix, as seen in Step5. Then, the solution space for the wordform is created. For example, possible stems of wordforms in the sentence “Güzel koyun otlamaya çıktı.” are given in Figure 5.10, and some possible suffixes of the wordform “koyun” are given in Figure 5.11.



```
- <S Index="1">
   Güzel koyun otlamaya çıktı .
- <Word Index="0" Value="Güzel">
    + <R I="0" V="Güzel" T="isim">
    + <R I="0" V="Güzel" T="sıfat">
    + <R I="0" V="Güzel" T="zarf">
   </Word>
- <Word Index="1" Value="koyun">
    + <R I="0" V="koy" T="isim">
    + <R I="0" V="koy" T="fiil">
    + <R I="1" V="koyu" T="sıfat">
    + <R I="2" V="koyun" T="isim">
   </Word>
- <Word Index="2" Value="otlamaya">
    + <R I="0" V="otla" T="fiil">
    + <R I="1" V="otlama" T="isim">
   </Word>
- <Word Index="3" Value="çıktı">
    + <R I="0" V="çık" T="fiil">
    + <R I="1" V="çıktı" T="isim">
   </Word>
+ <Word Index="2" Value=".">
</S>
```

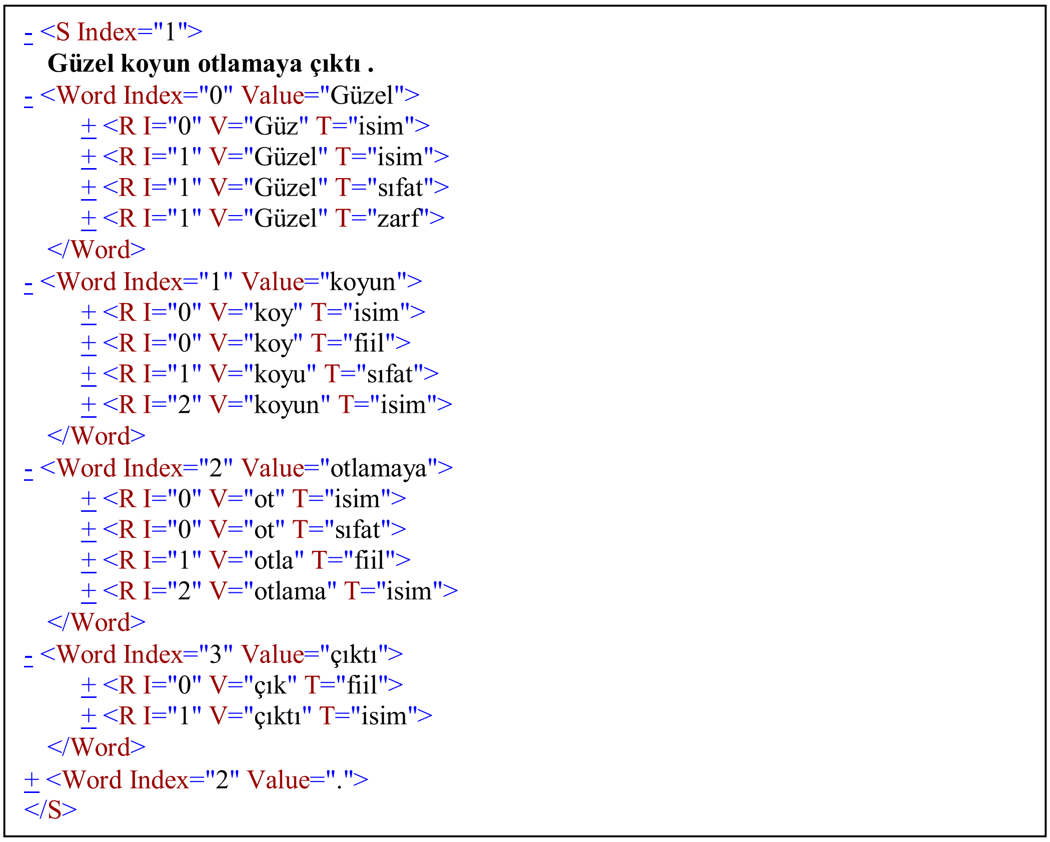
Figure 5.10 Possible stems of wordforms in the sentence “Güzel koyun otlamaya çıktı.”



```
- <Word Index="1" Value="koyun">
   - <R I="0" V="koy" T="isim">
      - <Suffixes>
         - <Sx I="0"> <DuAUyKT2>un</DuAUyKT2>     </Sx>
         - <Sx I="1">  <DuADurTam>un</DuADurTam>   </Sx>
        </Suffixes>
     </R>
   - <R I="0" V="koy" T="fiil">
      - <Suffixes>
         - <Sx I="0">  <DuECEdil>un</DuECEdil>            </Sx>
         - <Sx I="1">  <DuECDonus>un</DuECDonus>   </Sx>
         - <Sx I="2">  <DuEKGr4C2>un</DuEKGr4C2>    </Sx>
        </Suffixes>
     </R>
   - <R I="1" V="koyu" T="sıfat"> …   </R>
   - <R I="2" V="koyun" T="isim">    ... </R>
  </Word>
```

Figure 5.11 Some possible suffixes of the wordform “koyun”

In the *Parsing Roots and Derivational Suffixes* step, the possible stems are parsed by using same algorithm as parsing stems and possible combinations of derivational suffixes are found. Finally, all possible roots/stems and suffixes are stored in an XML structured file. For example, possible roots of the stem "*güzel*" are found as "*güz*" and "*güzel*"; possible roots of the stem "*koyun*" are found as "*koy*", "*koyu*" and "*koyun*"; possible roots of the stem "*otlamaya*" are found as "*ot*", "*otla*" and "*otlama*"; possible roots of the stem "*çıktı*" are found as "*çık*" and "*çıktı*" as in different types (Figure 5.12).



```
- <S Index="1">
  Güzel koyun otlamaya çıktı .
- <Word Index="0" Value="Güzel">
      + <R I="0" V="Güz" T="isim">
      + <R I="1" V="Güzel" T="isim">
      + <R I="1" V="Güzel" T="sıfat">
      + <R I="1" V="Güzel" T="zarf">
  </Word>
- <Word Index="1" Value="koyun">
      + <R I="0" V="koy" T="isim">
      + <R I="0" V="koy" T="fiil">
      + <R I="1" V="koyu" T="sıfat">
      + <R I="2" V="koyun" T="isim">
  </Word>
- <Word Index="2" Value="otlamaya">
      + <R I="0" V="ot" T="isim">
      + <R I="0" V="ot" T="sıfat">
      + <R I="1" V="otla" T="fiil">
      + <R I="2" V="otlama" T="isim">
  </Word>
- <Word Index="3" Value="çıktı">
      + <R I="0" V="çık" T="fiil">
      + <R I="1" V="çıktı" T="isim">
  </Word>
+ <Word Index="2" Value=".">
</S>
```

Figure 5.12 Sample results of Parsing Roots and Derivational Suffixes step

Detailed information about tags and usage of the rule list are given in Chapter 4.

5.4 Rule-Based Part of Speech (POS) Tagging

In this study, the *Rule-Based POS Tagging* (RB-POST) *Method* is used to tag the word classes in Turkish texts, which is improved version of TurPOS that was developed as unsupervised method by Hallaç in 2007. The *Rule-Based POS Tagging*

Method tries to solve the ambiguities caused in word type determination and also stem/root finding processes.

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,
- Text to be tagged.

In this project, a complete lexicon was taken from TDK to be used in this project, a list of grammatical rules was collected by linguists in Dokuz Eylul University, College of Social Science and Literature Linguistic Department and input document was taken as output of the morphological analysis process. The output file of morphological analysis process includes the words of the document that will be tagged, with its possible stems, suffixes and word types. All input and output file formats in POS tagging are based on XML structure.

Rule-Based POS Tagging method consists of three modules: Main Tagger Module and two auxiliary modules called as Rule Parser and Stem Reader. Basic flow diagram of Rule-Based POS Tagging is given in Figure 5.13.

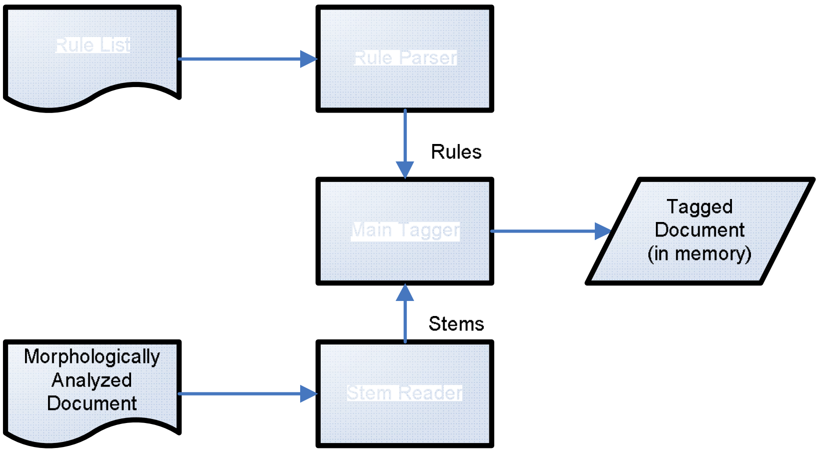

Rule List
Rule Parser
Rules
Main Tagger
Tagged Document (in memory)
Stems
Morphologically Analyzed Document
Stem Reader

Figure 5.13 General Flow Diagram of Rule-Based POS Tagging Algorithm.

5.4.1 Rule Parser Module

Rule Parser Module works only once at program start up and reads the list of rules into the system. In order to analyze and tag the text, most of the POS taggers use some basic rules, which are defined to describe the absolute and relative positions of word types in sentences in a natural language. These predefined rules are called “grammatical” or “syntactical” rules, or sometimes “context frame” rules.

The rule file used in *Rule-Based POS Tagging* Module includes individual rules defining the order of word types that are eligible or not eligible to be used in a sentence. It also includes rules which show the positions of punctuation marks as to word types in a sentence. XML structure is used to store the rules, samples of which are given in Table 5.3.

Table 5.3 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 cannot be a comma between an adjective and a noun which is determined by this adjective.

The currently used rules for POS Tagging Module are given in the Appendix B3.

The *Rule Parser Module* works in the following way: The whole rule file is read once on startup of the system. Then, the parser parses rules individually and loads them into 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 decrease system overhead.

5.4.2 Stem Reader Module

Stem Reader Module loads the morphologically analyzed documents into the system. The input file of this module is the output file of morphological analysis module, which includes possible stems with type of them and possible combinations of suffixes. Then, it stores parsed data into its entities to use in the POS tagging process.

Unlike most POS taggers, Rule-Based POS Tagger doesn't include a built-in lexicon. Instead, the types of words in a document are read from the text, which contains the morphologically analyzed document, and stored into the memory. Parsing the possible types 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 found words.

Similar to the Rule Parser, Stem Reader Module also parses the output file of morphological analysis module once and stores into the memory, just before starting the tagging process to be used in the Tagger Module.

5.4.3 Tagger Module

The main part of the POS Tagging is the *Tagger Module*, which processes rule list on morphologically analyzed XML document and produces the output. The tagged text output is also designed in XML format.

The Tagger Module decides the word type 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 type with the highest accumulator value.

The tagger processes each rule in the rule list on each sentence in the document, whether the word types are sequenced true or not. For each matched type in a rule, an accumulator is incremented by one for the word class defined in this rule. The accumulation process is explained step by step by giving an example sentence:

Boş teneke çok ses çıkartır. (*en: Empty tin noises the most.*)

Assume that there is a rule such as “`adjective + noun`”. The tagger processes the sentence in groups of two words because the rule contains 2 items. The word groups and the order of control processing are:

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

Then, the tagger investigates the possible word types of words for each of the word groups. In step 1, the types of words in the word group “`boş teneke`” matches with the grammatical sequence “`adjective + noun`”. So the adjective word type accumulator value for “`boş`” and the noun word type accumulator value for “`teneke`” are incremented by one. The other word groups are also processed with this rule. Steps of processing word groups are given in Table 5.4.

Table 5.4 Control processes of word groups in sample sentence for first rule

Controlled Rule: Adjective + Noun -> True			
Step #	**Word Group**	**Word Types**	**Accumulators**
1	boş teneke	boş: Adjective	Acc_Adjective_boş +=1
		teneke: Noun	Acc_Noun_teneke +=1
2	teneke çok	teneke: Noun	Acc_Adjective_teneke
		çok: Adjective and Adverb	Acc_Noun_çok
3	çok ses	çok: Adjective and Adverb	Acc_Adjective_çok+=1
		ses: Noun	Acc_Noun_ses +=1
4	ses çıkartır	ses: Noun	Acc_Adjective_ses
		çıkartır: Verb	Acc_Noun_çıkar
5	çıkartır	çıkartır: Verb	Acc_Adjective_çıkar

This operation continues to process all rules on all sentences of the text corpora. After the process is finished, then the word types with the highest accumulator values are picked as the result tags for each word. In this example, the values of all accumulators for all words are compared, then the types of “boş” and “çok” are accepted as “adjective”, the types of “teneke” and “ses” are accepted as “noun”, the type of “çıkar” is accepted as “verb” since Acc_Adjective_boş, Acc_Adjective_çok, Acc_Noun_teneke, Acc_Noun_ses and Acc_Verb_çıkar have the highest values among the accumulators of these words.

The flow diagram of this algorithm is given in Figure 5.14.

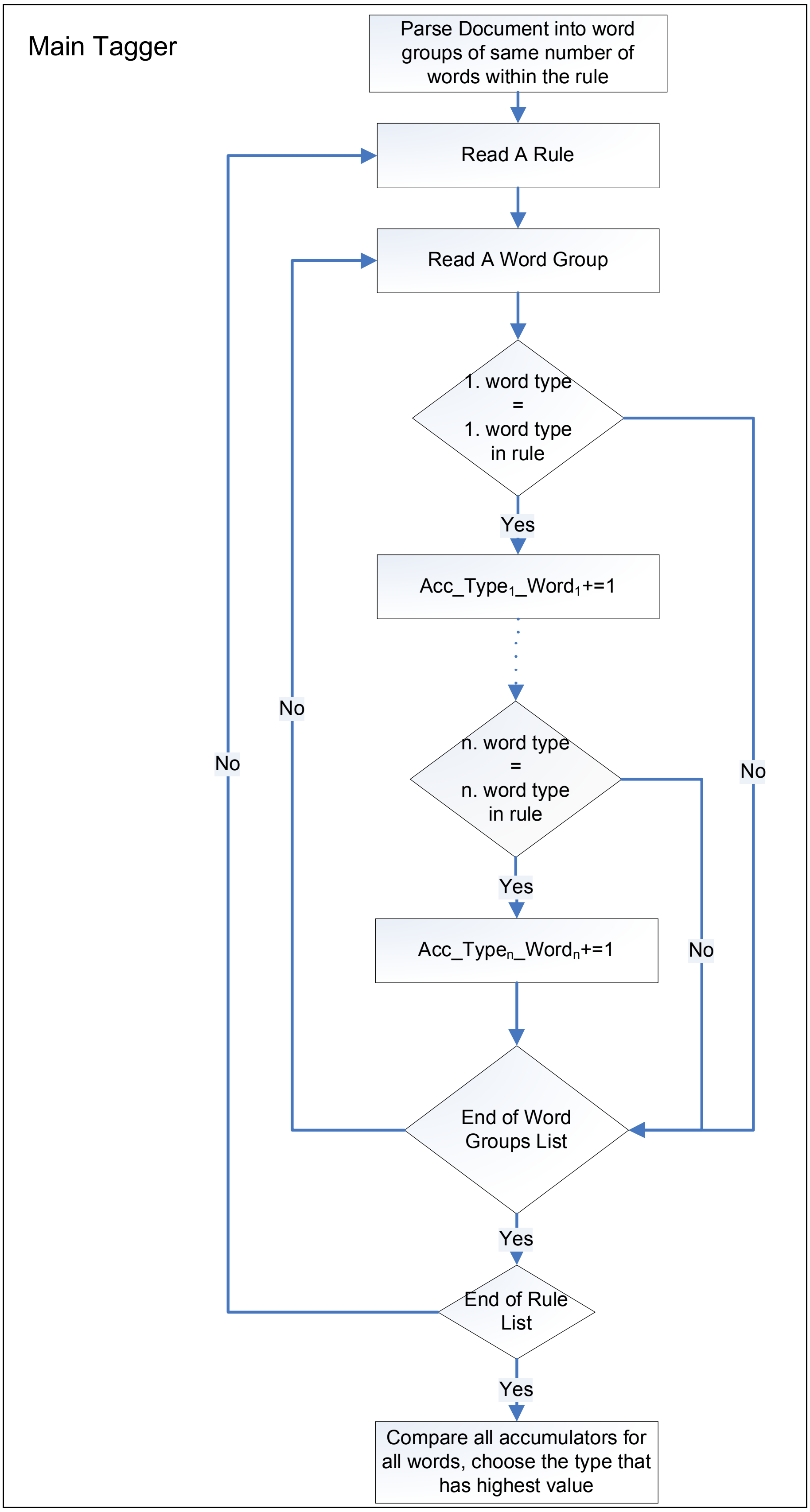

Main Tagger
Parse Document into word groups of same number of words within the rule
Read A Rule
Read A Word Group
1. word type = 1. word type in rule
Yes
Acc_Type1_Word1+=1
No
No
n. word type = n. word type in rule
No
Yes
Acc_Typen_Wordn+=1
No
End of Word Groups List
Yes
End of Rule List
Yes
Compare all accumulators for all words, choose the type that has highest value

Figure 5.14 Flow diagram of Main Tagger in Rule Based POS Tagging Algorithm

5.5 Software Structure

The main structure of the Rule-Based Corpus Generation (RBCorGen) consists of four modules (Figure 5.15);

- CorpusGeneration as Main Module,
- SentenceBoundary Library for Sentence Boundary Detection,
- WordDetector Library for Root/Suffix Separation,
- POSTagging Library for Word Class Definition.

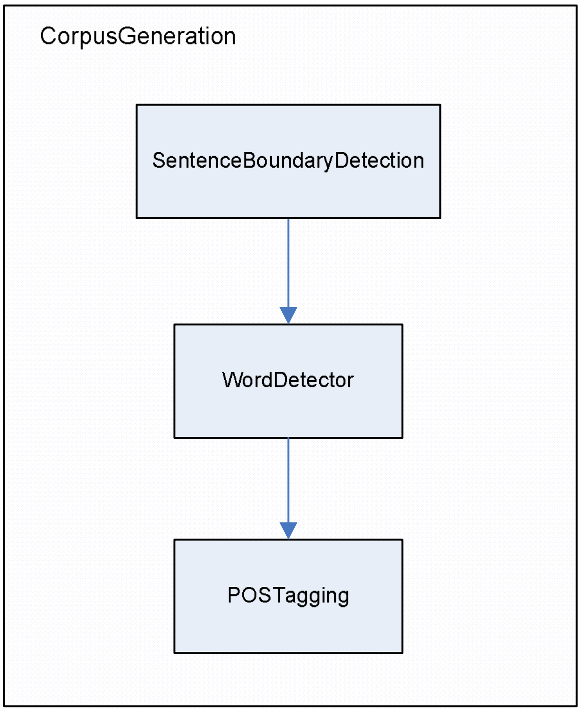


Figure 5.15 Developed modules in the software

The class diagram of RB-CorGen, was implemented in C# programming language, is given in Figure 5.16.

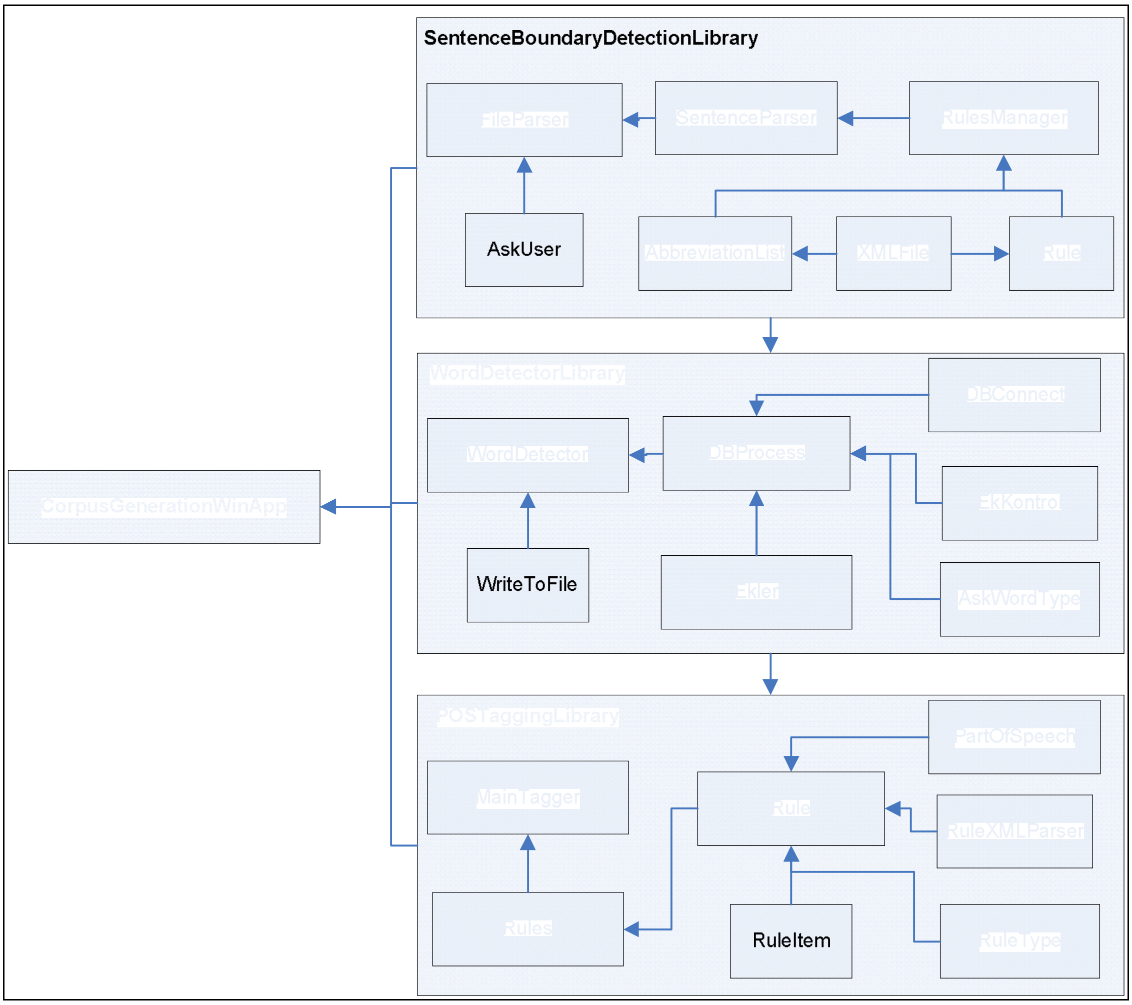
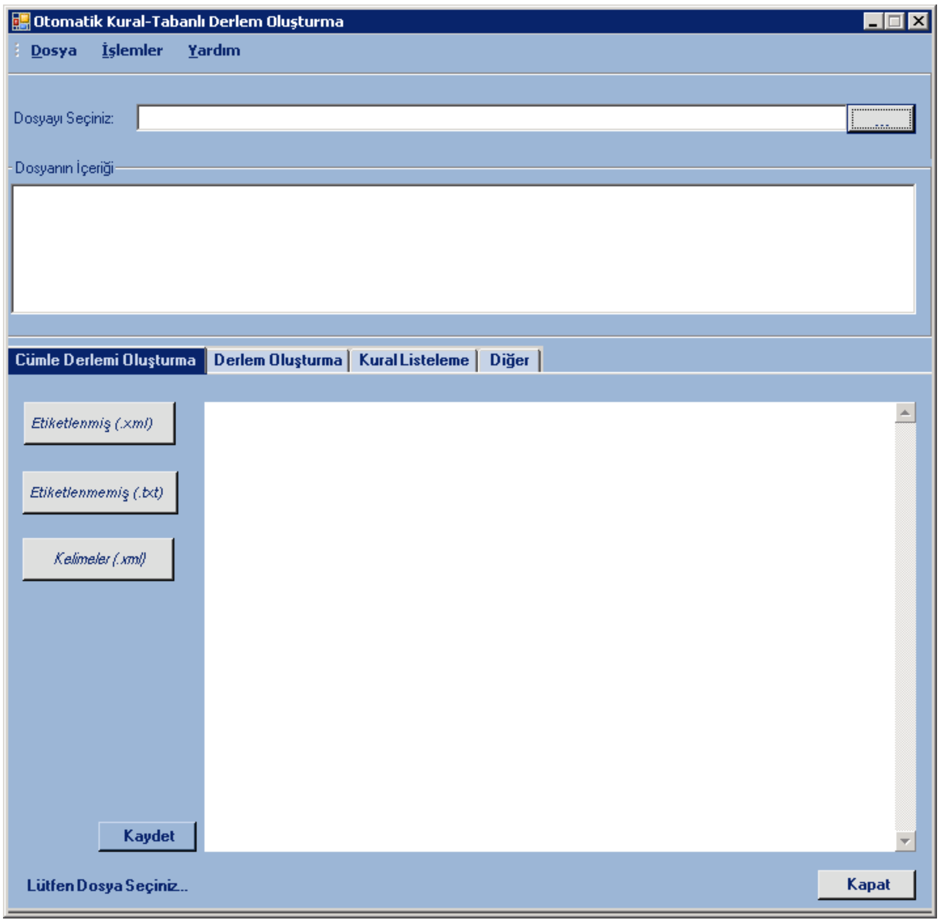


Figure 5.16 The main classes and modules of RBCorGen

CorpusGenerationWinApp is the main module of the RBCorGen, in which main form is generated (Figure 5.17).



Lütfen Dosya Seçiniz..

Figure 5.17 Main form of RB-CorGen

CorpusGenerationWinApp module includes 5 submodules;

- SentenceBoundaryLibrary
- SSentenceBoundaryLibrary
- STSentenceBoundaryLibrary
- POSTaggingLibrary
- WordDetectorLibrary

Rule-Based Sentence Boundary Detection for Turkish (RB-SBD) was implemented in *SentenceBoundaryLibrary* module to generate sentence corpus and parse the sentences into wordforms to be used in Rule-Based Morphological Analysis for Turkish (RB-MA) module as input. This module has 7 classes, in which 12 methods were implemented (Figure 5.18).

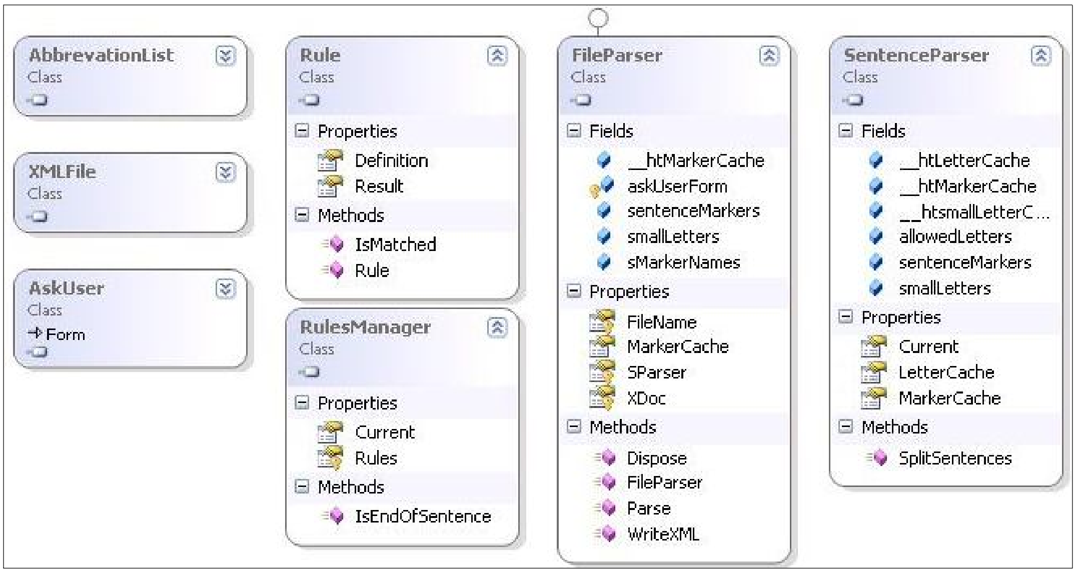


Figure 5.18 Methods in SentenceBoundaryLibrary

Explanations of classes and main methods in *SentenceBoundaryLibrary* module are given in Table 5.5.

Table 5.5 Explanation of Methods in SentenceBoundaryLibrary

Class Name	**Explanation**	**Method Name**	**Explanation**
AbbreviationList		AbbreviationList	Parses and loads the abbreviation list by using XMLFile.
XMLFile		XMLFile	Parses and loads an XML file using DOM.
Rule	Represents a rule that is already defined in an XML file, and managed by a *RulesManager.*	Rule	Stores a rule
		IsMatched	Controls the Rule if it is matched with the sent data.
RulesManager	Applies predefined rules in XML File. If any rule could be applied returns rule result otherwise return true to indicate End Of Sentence.	IsEndOfSentence	Controls all rules in XML file and defines whether the data end of sentence or not.
FileParser	Loads the input text document and parses it by using *SentenceParser*.	FileParser	Create the member variables and defines the hashed letters.
		Parse	Loads the input text document, reads lines one by one and controls whether they are sentences or not.
		WriteXML	Writes the results in XML format.

Table 5.5 Explanation of Methods in SentenceBoundaryLibrary (Cont’d)

SentenceParser	Splits a candidate line, which has been splited with linebreak and sent by *FileParser*.	SplitSentences	Splits the sent text into sentences by using RulesManager and returns list of sentences containing a list of words in order.
AskUser	Used for ambiguities in the decisions of the text whether it is bulleted list or conversation.	AskUser	Generates a form to ask the user whether the text is bulleted list or conversation text.

Rule-Based Morphological Analysis for Turkish (RB-MA) was implemented in *WordDetectorLibrary* module to parse wordforms into stems, roots and suffixes. This module takes the output of RBSBDT as input and has 7 classes, in which 25 methods were implemented (Figure 5.19).

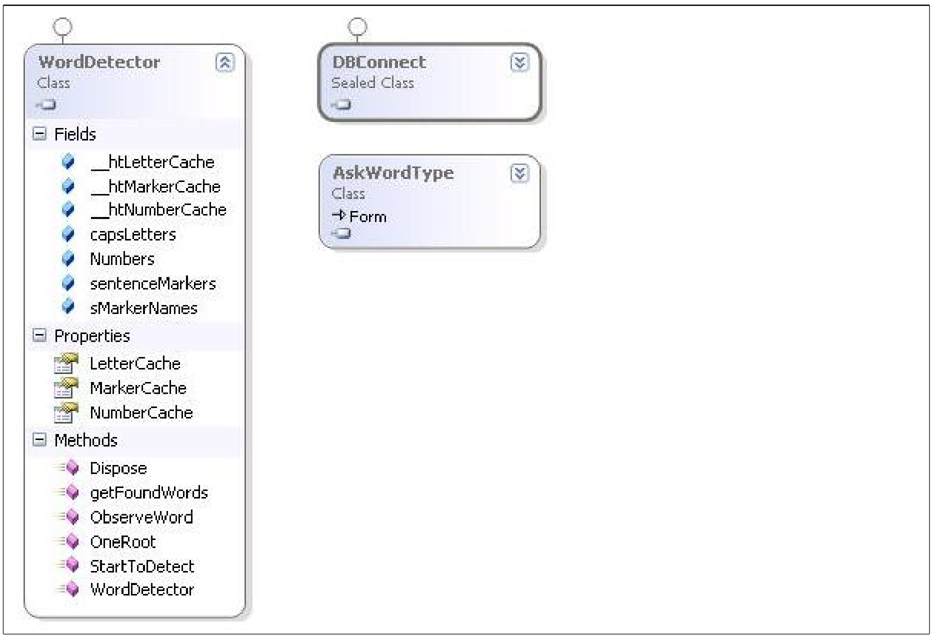


Figure 5.19 Methods in WordDetectorLibrary

Explanations of classes and main methods in *WordDetectorLibrary* module are given in Table 5.6.

Table 5.6 Explanation of Methods in WordDetectorLibrary

Class Name	**Explanation**	**Method Name**	**Explanation**
DBConnect		DBConnect	Used to make connection to database
DBProcesses	Used for all process those use the database.	RootController	Checks the root database whether it includes the searched root or not.
		SuffixSeperator	If the searched characters are found in root database, the rest of the wordform is checked whether it is generated by suffixes or not.
		CheckSuffix	Checks the suffixes database whether it includes the searched suffix part or not.
		CheckWordTypes	Returns all possible types of the searched word.
		suffixTypeControl	Checks whether the suffixes are suitable to the word type or not.
		suffixWithoutTags	Returns all possible suffixes combinations without tags.
Ekler		getEkXMLtag	Returns tags of the searched suffix.
EkKontrol		controlEkDizisi	Controls the order of the suffixes whether true or not.
WordDetector		StartToDetect	Used to read the words from the input file and make root-suffix separation.
		ObserveWord	Makes root-suffix separation, called by *StartToDetect* method.
		getFoundWords	Reads the file "foundBefore.txt" and returns the read data.
WriteToFile		WriteToFile	Writes the results in XML format.
AskWordType	Used for ambiguities in the decisions of the word type whether it is special name or not.	AskWordType	Generates a form to ask the user whether the word type whether it is special name or not.

Rule-Based Part of Speech Tagging for Turkish (RBPOST) was implemented in *POSTagging* module. This module has 15 classes, in which 35 methods were implemented (Figure 5.20).

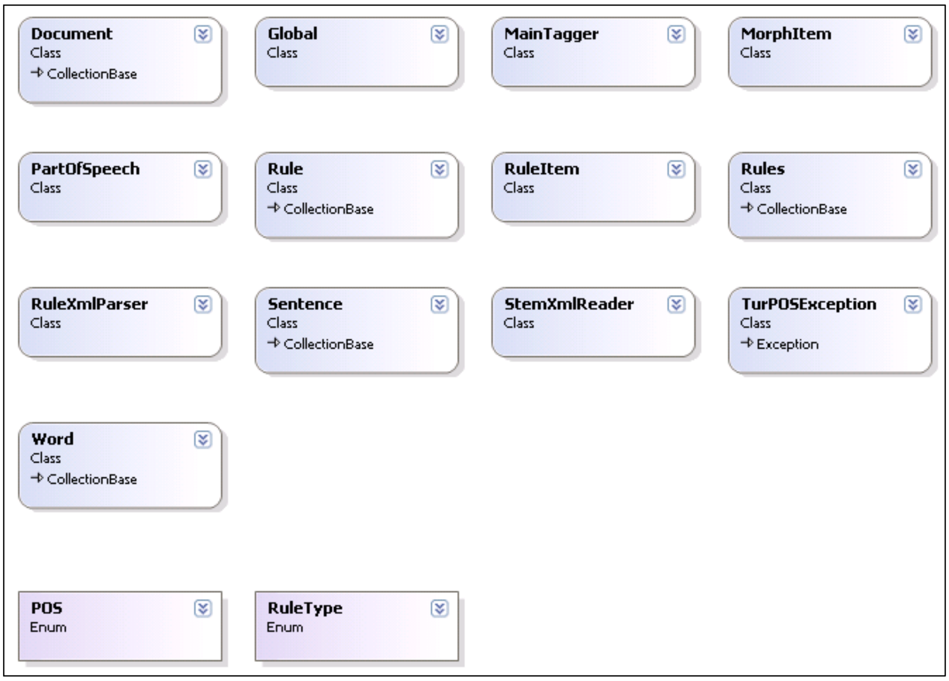


Figure 5.20 Methods in POSTaggingLibrary

Explanations of classes and main methods in *POSTagging* module are given in Table 5.7.

Table 5.7 Explanation of Methods in POSTaggingLibrary

Class Name	Explanation	Method Name	Explanation
PartOfSpeech	Includes POS item and stores unique values of POS items.	PartOfSpeech	
Rule		Add	Adds a new RuleItem into the Rule.
		Remove	Removes a RuleItem from the Rule.
		ToXMLString	Creates an xml string from the current Rule entity.
RuleItem			Gets the word type from the rule.
Rules	Collection class for *Rules*, the parsed rule list including unique rules.		
RuleType	Stores the type of rules, such as word order (sözdizim).	RuleType	
RuleXmlParser	Reads the rules file node by node, parses the rules and stores into a *Rules* object.	ParseRules	

Table 5.7 Explanation of Methods in TurPOSLibrary (Cont'd)

MainTagger	Determines word types in sentences of documents.	Tag	Accumulates possible types for each word of document.
		Analyze	Analyzes the document, defines types of each word and writes the output into an XML file.
		Accumulate	Increases *accumulator* if rule state is true, and decreases it if rule state is false.
Document	Stores the data that will be analyzed.	Sentence	Returns the Sentence at the given location.
		Add	Adds a Sentence into the document.
		Remove	Removes the Sentence at the given location.
StemXmlReader	Reads the stems file and stores its contents into a *Document* object.	ReadDocument	Reads document and returns document instance that contains the stems file.
Sentence		Add	Adds a Word into the Sentence.
		Remove	Removes the Word at the given location.
Word		AnalyzeAccumulator	Analyzes the accumulator and returns name of the probable word type.
		IncrementAccumulator	Increments accumulator for the WordType.
		AddWordType	Adds a new word type into possible word types of the searched Word.

CHAPTER SIX
CASE STUDY

In order to test the Rule-Based Corpus Generation (RBCorGen), four main modules were carried out:

1. Dataset Generation,
2. Rule-Based Sentence Boundary Detection (RB-SBD),
3. Rule-Based Morphological Analyser (RB-MA),
4. Rule-Based Part-of-Speech Tagging (RB-ROST).

6.1 Dataset Generation

In order to generate dataset used in this case study, an application was developed for collecting documents such as newspaper, report, magazine, book, parliamentary report and official gazette from electronic environment. By using this module, articles from 5 different Turkish newspapers “Milliyet”, “Hürriyet”, “Radikal”, “Vatan”, ”Akşam” were downloaded and stored on disk, also metadata of these documents, such as URL of document, header of document, size of document, etc., were stored in a database (Appedix D). A total of 195.256 articles which contain 93.228.892 words were downloaded and generated collection size was 1.05 GB. The list of the documents downloaded for the dataset is given in Table 6.1 and details of the documents are given in Table 6.2.

Table 6.1 Documents in Dataset according to the newspaper names

Newspaper Name	**Number of Articles**	**Number of Words**
Milliyet	43.465	19.536.744
Hürriyet	65.599	31.585.895
Radikal	35.159	17.605.413
Vatan	41.250	18.505.364
Akşam	9.783	5.995.476
TOTAL	195.256	93.228.892

Table 6.2 Details of documents in Dataset

Year	**Newspaper Name**	**Number of Articles**	**Article Size (bytes)**	**Number of Words**
2009	Milliyet	7.987	28.042.637	3.692.542
2009	Akşam	3.129	13.412.243	1.872.178
2009	Hürriyet	5.709	19.572.040	2.604.667
2009	Radikal	4.354	18.318.948	2.382.129
2009	Vatan	3.752	13.036.081	1.735.620
2008	Hürriyet	9.556	35.162.196	4.699.045
2008	Radikal	5.305	21.780.759	2.806.544
2008	Vatan	6.718	23.018.968	3.090.063
2008	Akşam	2.421	10.743.743	1.512.270
2008	Milliyet	8.714	31.141.424	4.082.979
2007	Milliyet	4.685	16.630.367	2.089.244
2007	Akşam	2.075	9.279.697	1.304.930
2007	Hürriyet	9.043	33.933.670	4.513.839
2007	Radikal	4.401	17.406.528	2.228.127
2007	Vatan	6.639	22.820.262	3.055.629
2006	Akşam	1.621	7.033.136	995.826
2006	Radikal	4.255	16.613.193	2.124.421
2006	Vatan	6.026	20.416.090	2.705.875
2006	Hürriyet	7.682	30.025.234	3.994.601
2006	Milliyet	4.977	17.334.200	2.174.640
2005	Milliyet	4.486	16.077.155	2.018.390
2005	Radikal	3.925	15.032.947	1.925.788
2005	Akşam	534	2.184.361	308.496
2005	Hürriyet	6.652	24.352.596	3.240.341
2005	Vatan	5.677	19.762.531	2.611.727
2004	Milliyet	4.556	15.780.034	2.006.730
2004	Radikal	3.742	14.206.582	1.818.379
2004	Hürriyet	5.901	21.680.718	2.882.916
2004	Vatan	6.029	19.823.904	2.626.374
2003	Hürriyet	4.973	17.719.157	2.347.657
2003	Milliyet	3.471	11.788.371	1.498.700
2003	Radikal	3.655	13.629.813	1.733.173
2003	Vatan	5.801	18.141.953	2.418.832
2002	Hürriyet	3.926	13.952.947	1.847.157
2002	Vatan	608	1.960.937	261.244
2002	Milliyet	3.851	13.191.525	1.691.409
2002	Radikal	3.407	12.747.119	1.617.097
2001	Hürriyet	2.739	9.742.045	1.317.156
2001	Milliyet	1.456	4.768.595	611.997
2001	Radikal	2.115	7.703.079	969.755
2000	Hürriyet	2.707	9.889.002	1.329.063
1999	Hürriyet	3.025	9.804.854	1.304.374
1998	Hürriyet	2.618	7.879.868	1.043.888
1997	Hürriyet	1.068	3.571.005	461.191
TOTAL		195.256	711.112.514	93.557.003

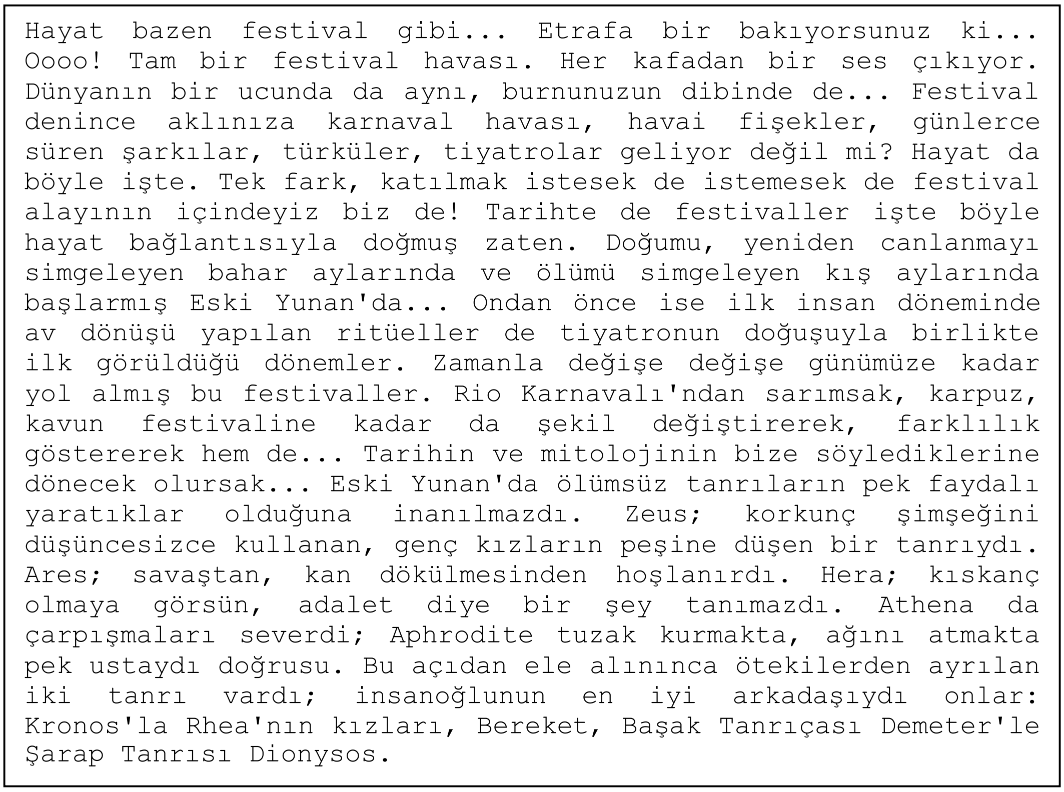
6.2 Rule-Based Sentence Boundary Detection (RB-SBD)

By applying RB-SBD method onto the sample data, the results may be written into two different formats as;

1- Parsed sentences are written into a text (.txt) file,
2- Parsed and tagged sentences are written into an XML file,

Besides, after sentence determination process, all wprdforms are also tagged and written into an XML file.

A part of the sample document to be parsed is given in the following figure, and full document is given in Appendix C.4.1.1.



Hayat bazen festival gibi... Etrafa bir bakıyorsunuz ki... Oooo! Tam bir festival havası. Her kafadan bir ses çıkıyor. Dünyanın bir ucunda da aynı, burnunuzun dibinde de... Festival denince aklınıza karnaval havası, havai fişekler, günlerce süren şarkılar, türküler, tiyatrolar geliyor değil mi? Hayat da böyle işte. Tek fark, katılmak istesek de istemesek de festival alayının içindeyiz biz de! Tarihte de festivaller işte böyle hayat bağlantısıyla doğmuş zaten. Doğumu, yeniden canlanmayı simgeleyen bahar aylarında ve ölümü simgeleyen kış aylarında başlarmış Eski Yunan'da... Ondan önce ise ilk insan döneminde av dönüşü yapılan ritüeller de tiyatronun doğuşuyla birlikte ilk görüldüğü dönemler. Zamanla değişe değişe günümüze kadar yol almış bu festivaller. Rio Karnavalı'ndan sarımsak, karpuz, kavun festivaline kadar da şekil değiştirerek, farklılık göstererek hem de... Tarihin ve mitolojinin bize söylediklerine dönecek olursak... Eski Yunan'da ölümsüz tanrıların pek faydalı yaratıklar olduğuna inanılmazdı. Zeus; korkunç şimşeğini düşüncesizce kullanan, genç kızların peşine düşen bir tanrıydı. Ares; savaştan, kan dökülmesinden hoşlanırdı. Hera; kıskanç olmaya görsün, adalet diye bir şey tanımazdı. Athena da çarpışmaları severdi; Aphrodite tuzak kurmakta, ağını atmakta pek ustaydı doğrusu. Bu açıdan ele alınınca ötekilerden ayrılan iki tanrı vardı; insanoğlunun en iyi arkadaşıydı onlar: Kronos'la Rhea'nın kızları, Bereket, Başak Tanrıçası Demeter'le Şarap Tanrısı Dionysos.

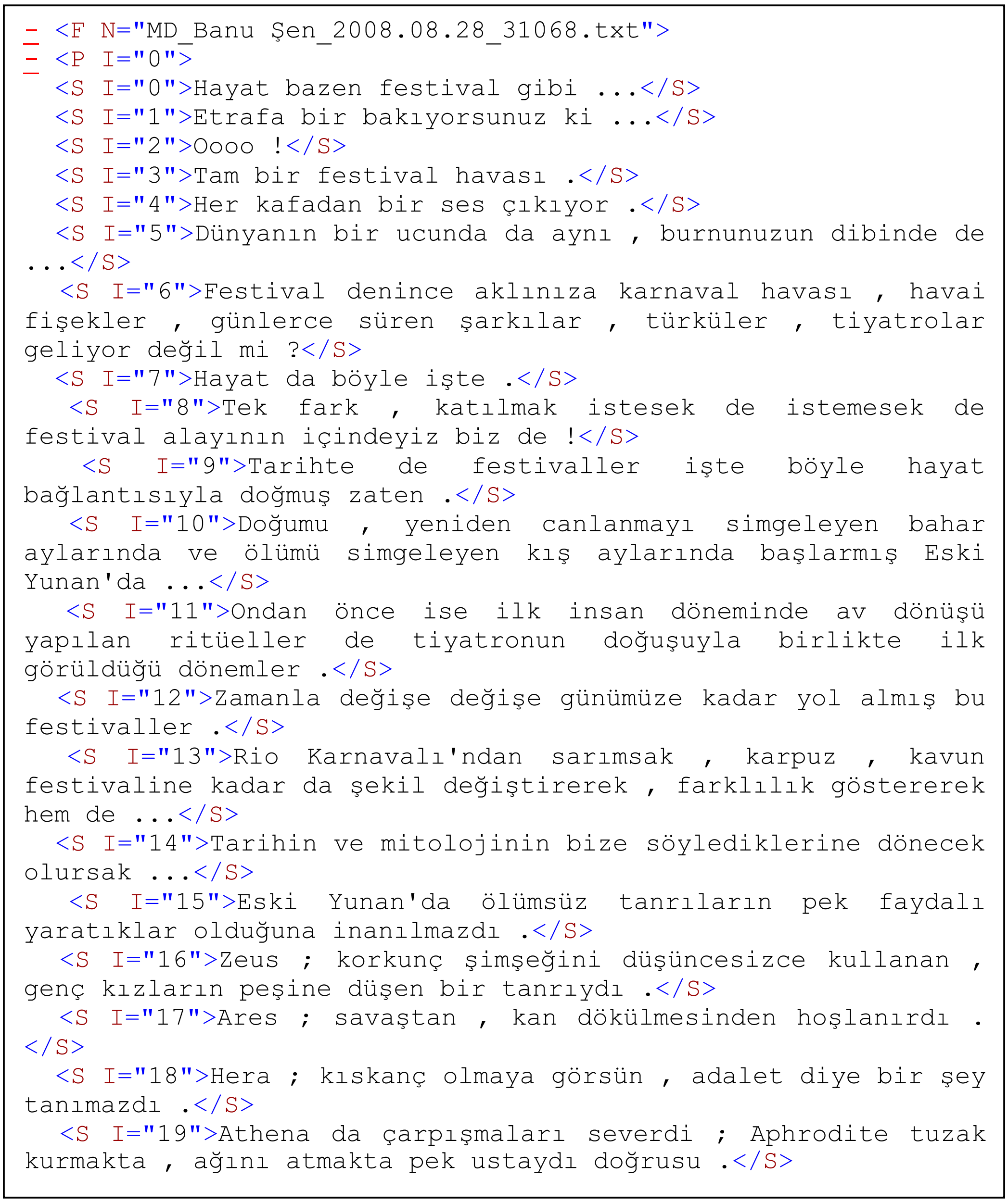
Figure 6.1 Sample parsed document

Parsed sentences of document part in Figure 6.1 are given as text form in Figure 6.2, and full sentence list is given in Appendix C.4.1.2.

```
1_____ Hayat bazen festival gibi....
2_____ Etrafa bir bakıyorsunuz ki....
3_____ Oooo!
4_____ Tam bir festival havası.
5_____ Her kafadan bir ses çıkıyor.
6_____ Dünyanın bir ucunda da aynı, burnunuzun dibinde de....
7_____ Festival denince aklınıza karnaval havası, havai
fişekler, günlerce süren şarkılar, türküler, tiyatrolar
geliyor değil mi?
8_____ Hayat da böyle işte.
9_____ Tek fark, katılmak istesek de istemesek de festival
alayının içindeyiz biz de!
10_____ Tarihte de festivaller işte böyle hayat bağlantısıyla
doğmuş zaten.
11_____ Doğumu, yeniden canlanmayı simgeleyen bahar aylarında
ve ölümü simgeleyen kış aylarında başlarmış Eski Yunan'da....
12_____ Ondan önce ise ilk insan döneminde av dönüşü yapılan
ritüeller de tiyatronun doğuşuyla birlikte ilk görüldüğü
dönemler.
13_____ Zamanla değişe değişe günümüze kadar yol almış bu
festivaller.
14_____ Rio Karnavalı'ndan sarımsak, karpuz, kavun
festivaline kadar da şekil değiştirerek, farklılık göstererek
hem de....
15_____ Tarihin ve mitolojinin bize söylediklerine dönecek
olursak....
16_____ Eski Yunan'da ölümsüz tanrıların pek faydalı
yaratıklar olduğuna inanılmazdı.
17_____ Zeus; korkunç şimşeğini düşüncesizce kullanan, genç
kızların peşine düşen bir tanrıydı.
18_____ Ares; savaştan, kan dökülmesinden hoşlanırdı.
19_____ Hera; kıskanç olmaya görsün, adalet diye bir şey
tanımazdı.
20_____ Athena da çarpışmaları severdi; Aphrodite tuzak
kurmakta, ağını atmakta pek ustaydı doğrusu.
```

Figure 6.2 Parsed sentences in the file of Figure 6.1

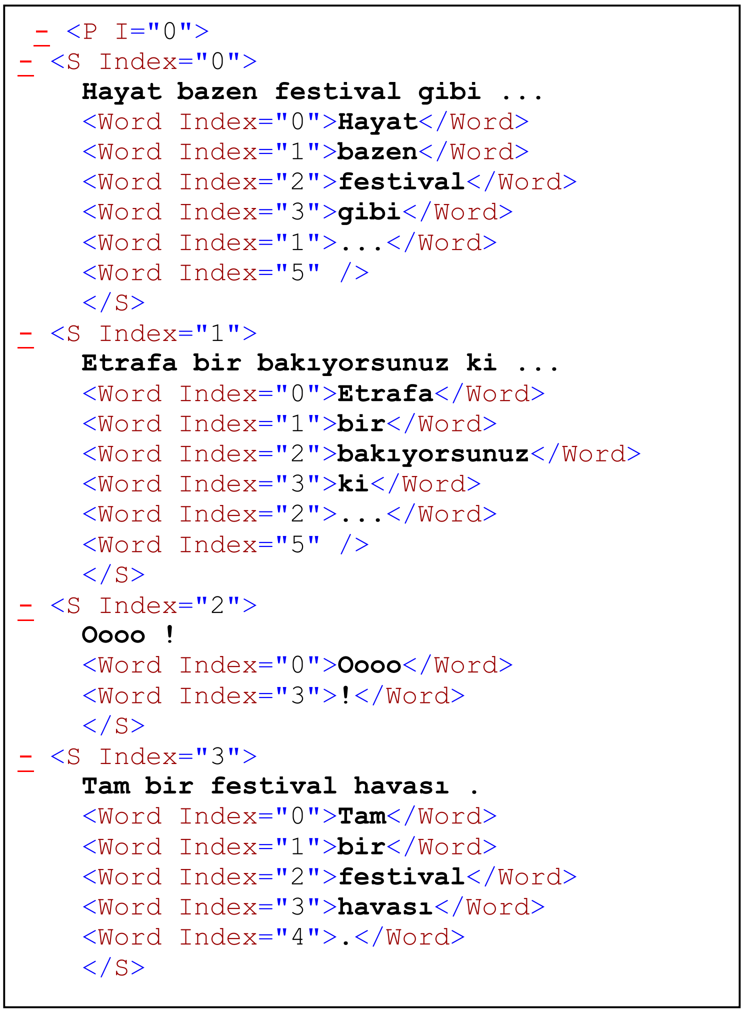
Parsed sentences of document part in Figure 6.1 are given as tagged in XML format in Figure 6.3, and full sentence list is given in Appendix C.4.1.2.



```
- <F N="MD_Banu Şen_2008.08.28_31068.txt">
- <P I="0">
  <S I="0">Hayat bazen festival gibi ...</S>
  <S I="1">Etrafa bir bakıyorsunuz ki ...</S>
  <S I="2">Oooo !</S>
  <S I="3">Tam bir festival havası .</S>
  <S I="4">Her kafadan bir ses çıkıyor .</S>
  <S I="5">Dünyanın bir ucunda da aynı , burnunuzun dibinde de
...</S>
  <S I="6">Festival denince aklınıza karnaval havası , havai
fişekler , günlerce süren şarkılar , türküler , tiyatrolar
geliyor değil mi ?</S>
  <S I="7">Hayat da böyle işte .</S>
  <S I="8">Tek fark , katılmak istesek de istemesek de
festival alayının içindeyiz biz de !</S>
  <S I="9">Tarihte de festivaller işte böyle hayat
bağlantısıyla doğmuş zaten .</S>
  <S I="10">Doğumu , yeniden canlanmayı simgeleyen bahar
aylarında ve ölümü simgeleyen kış aylarında başlarmış Eski
Yunan'da ...</S>
  <S I="11">Ondan önce ise ilk insan döneminde av dönüşü
yapılan ritüeller de tiyatronun doğuşuyla birlikte ilk
görüldüğü dönemler .</S>
  <S I="12">Zamanla değişe değişe günümüze kadar yol almış bu
festivaller .</S>
  <S I="13">Rio Karnavalı'ndan sarımsak , karpuz , kavun
festivaline kadar da şekil değiştirerek , farklılık göstererek
hem de ...</S>
  <S I="14">Tarihin ve mitolojinin bize söylediklerine dönecek
olursak ...</S>
  <S I="15">Eski Yunan'da ölümsüz tanrıların pek faydalı
yaratıklar olduğuna inanılmazdı .</S>
  <S I="16">Zeus ; korkunç şimşeğini düşüncesizce kullanan ,
genç kızların peşine düşen bir tanrıydı .</S>
  <S I="17">Ares ; savaştan , kan dökülmesinden hoşlanırdı .
</S>
  <S I="18">Hera ; kıskanç olmaya görsün , adalet diye bir şey
tanımazdı .</S>
  <S I="19">Athena da çarpışmaları severdi ; Aphrodite tuzak
kurmakta , ağını atmakta pek ustaydı doğrusu .</S>
```

Figure 6.3 Parsed and tagged sentences in the file of Figure 6.1

The tagged wordforms of the document part given in Figure 6.1 are given in Figure 6.4, and full sentence list is given in Appendix C.4.1.3.



```
 - <P I="0">
- <S Index="0">
    Hayat bazen festival gibi ...
    <Word Index="0">Hayat</Word>
    <Word Index="1">bazen</Word>
    <Word Index="2">festival</Word>
    <Word Index="3">gibi</Word>
    <Word Index="1">...</Word>
    <Word Index="5" />
    </S>
- <S Index="1">
    Etrafa bir bakıyorsunuz ki ...
    <Word Index="0">Etrafa</Word>
    <Word Index="1">bir</Word>
    <Word Index="2">bakıyorsunuz</Word>
    <Word Index="3">ki</Word>
    <Word Index="2">...</Word>
    <Word Index="5" />
    </S>
- <S Index="2">
    Oooo !
    <Word Index="0">Oooo</Word>
    <Word Index="3">!</Word>
    </S>
- <S Index="3">
    Tam bir festival havası .
    <Word Index="0">Tam</Word>
    <Word Index="1">bir</Word>
    <Word Index="2">festival</Word>
    <Word Index="3">havası</Word>
    <Word Index="4">.</Word>
    </S>
```

Figure 6.4 Parsed sentences with wordforms

6.3 Rule-Based Morphological Analyser (RB-MA)

The Word Detector Module, which is used for stem/root separating, was implemented in four different procedures to make the application flexible for users:

1- Finding all possible roots/stems and all suffixes of wordforms (The suffixes are tagged as XML structure) (FAPRS),

2- Finding all possible stems and inflectional suffixes of wordforms (The suffixes are tagged as XML structure) (FAPSIS),

3- Finding all possible roots/stems and suffixes of wordforms (The suffixes are not tagged) (FAPRS-not tagged),

4- Eliminating possible roots/stems by using suffixes types (FAPRS -

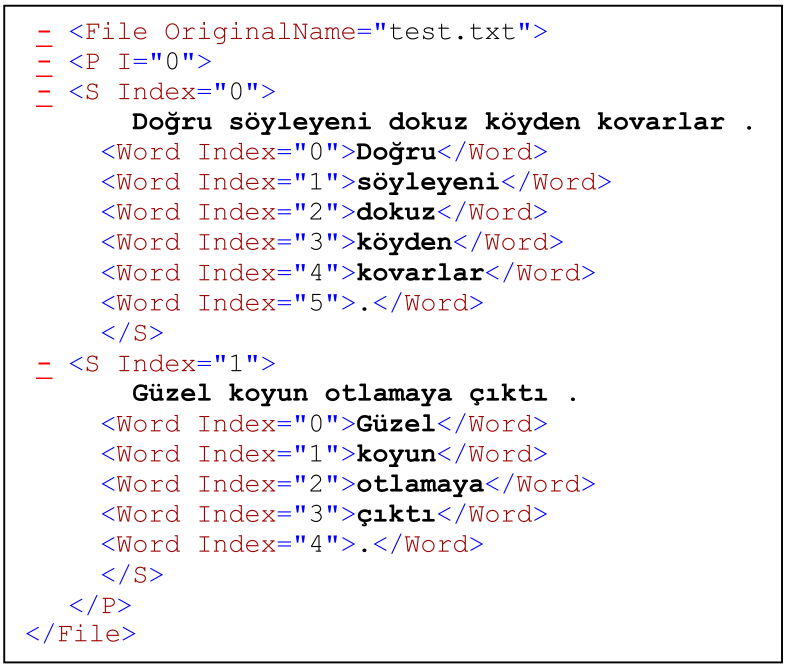
eliminated).

As an example, these modules were carried out for the following sentences;

Doğru söyleyeni dokuz köyden kovarlar. (1)
(They fire the person, who tells the truth.)

Güzel koyun otlamaya çıktı. (2)
(The beautiful sheep has gone for grazing.)

In the word analysis, at first, the input document was parsed into its sentences with wordforms by using the *sentence boundary detection – with wordforms* module (Figure 6.5).

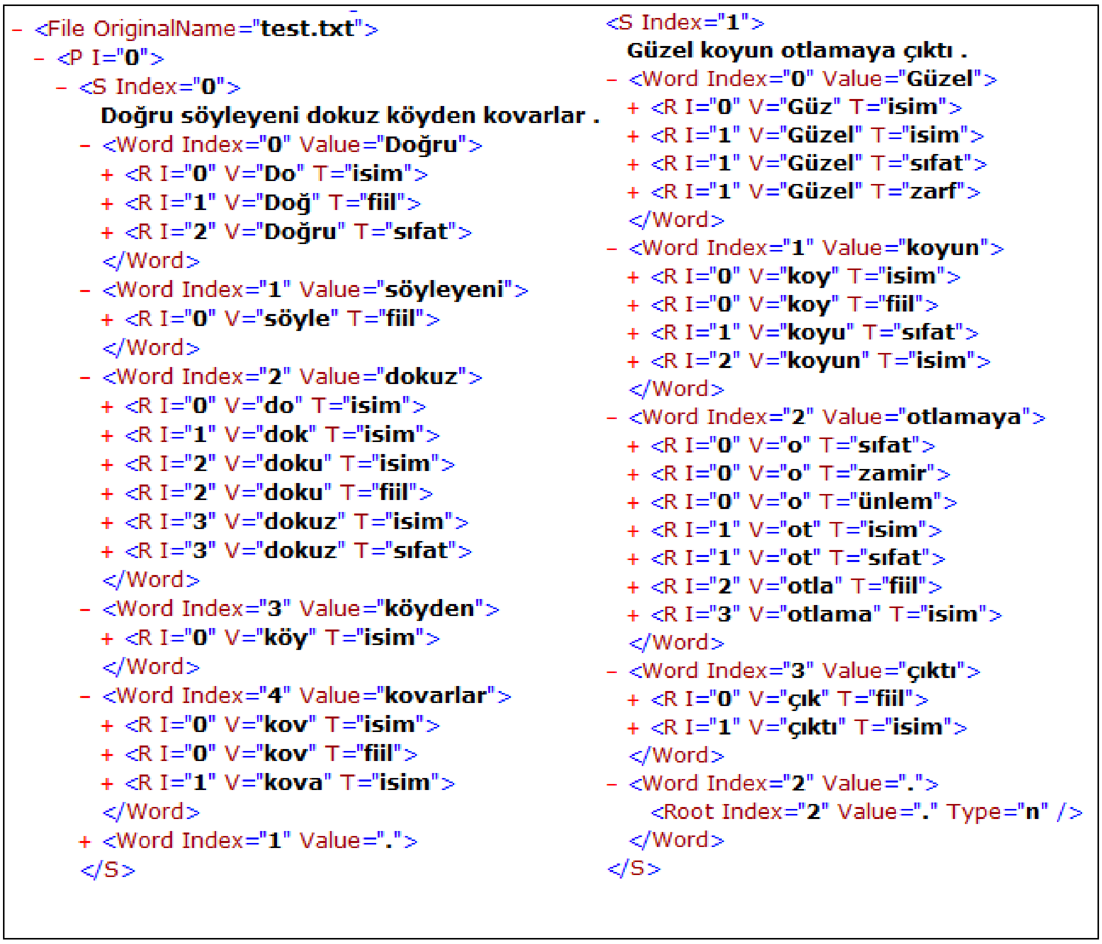


```
- <File OriginalName="test.txt">
- <P I="0">
- <S Index="0">
      Doğru söyleyeni dokuz köyden kovarlar .
    <Word Index="0">Doğru</Word>
    <Word Index="1">söyleyeni</Word>
    <Word Index="2">dokuz</Word>
    <Word Index="3">köyden</Word>
    <Word Index="4">kovarlar</Word>
    <Word Index="5">.</Word>
    </S>
- <S Index="1">
      Güzel koyun otlamaya çıktı .
    <Word Index="0">Güzel</Word>
    <Word Index="1">koyun</Word>
    <Word Index="2">otlamaya</Word>
    <Word Index="3">çıktı</Word>
    <Word Index="4">.</Word>
    </S>
  </P>
</File>
```

Figure 6.5 The wordforms of sample sentences

The parsed document was taken as input of RB-WD. All possible roots, stems, derivational suffixes and inflectional suffixes of the wordforms were given as an output of the “Finding all possible roots/stems and all suffixes (FAPRS)” procedure, and all suffixes are tagged in an XML formatted file with the possible roots/stems.

Outputs of the sample sentences (1) and (2) for this procedure is given in Figure 6.6, and detailed output document was given in Appendix C.4.2.1.



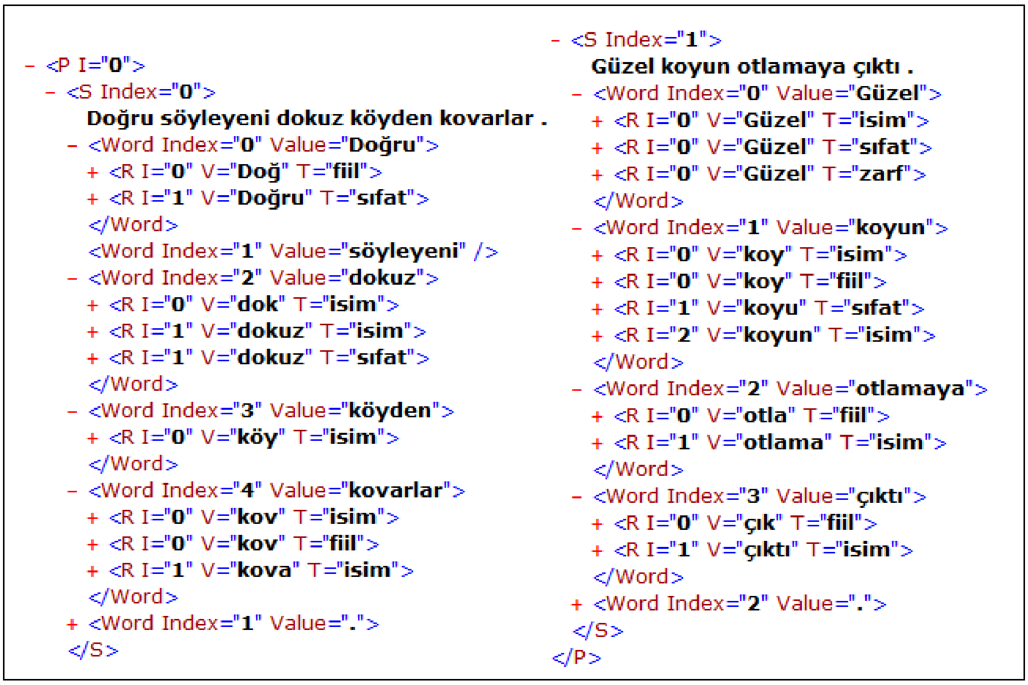
```
- <File OriginalName="test.txt">
  - <P I="0">
    - <S Index="0">
        Doğru söyleyeni dokuz köyden kovarlar .
      - <Word Index="0" Value="Doğru">
        + <R I="0" V="Do" T="isim">
        + <R I="1" V="Doğ" T="fiil">
        + <R I="2" V="Doğru" T="sıfat">
        </Word>
      - <Word Index="1" Value="söyleyeni">
        + <R I="0" V="söyle" T="fiil">
        </Word>
      - <Word Index="2" Value="dokuz">
        + <R I="0" V="do" T="isim">
        + <R I="1" V="dok" T="isim">
        + <R I="2" V="doku" T="isim">
        + <R I="2" V="doku" T="fiil">
        + <R I="3" V="dokuz" T="isim">
        + <R I="3" V="dokuz" T="sıfat">
        </Word>
      - <Word Index="3" Value="köyden">
        + <R I="0" V="köy" T="isim">
        </Word>
      - <Word Index="4" Value="kovarlar">
        + <R I="0" V="kov" T="isim">
        + <R I="0" V="kov" T="fiil">
        + <R I="1" V="kova" T="isim">
        </Word>
      + <Word Index="1" Value=".">
      </S>
```

```
<S Index="1">
  Güzel koyun otlamaya çıktı .
- <Word Index="0" Value="Güzel">
  + <R I="0" V="Güz" T="isim">
  + <R I="1" V="Güzel" T="isim">
  + <R I="1" V="Güzel" T="sıfat">
  + <R I="1" V="Güzel" T="zarf">
  </Word>
- <Word Index="1" Value="koyun">
  + <R I="0" V="koy" T="isim">
  + <R I="0" V="koy" T="fiil">
  + <R I="1" V="koyu" T="sıfat">
  + <R I="2" V="koyun" T="isim">
  </Word>
- <Word Index="2" Value="otlamaya">
  + <R I="0" V="o" T="sıfat">
  + <R I="0" V="o" T="zamir">
  + <R I="0" V="o" T="ünlem">
  + <R I="1" V="ot" T="isim">
  + <R I="1" V="ot" T="sıfat">
  + <R I="2" V="otla" T="fiil">
  + <R I="3" V="otlama" T="isim">
  </Word>
- <Word Index="3" Value="çıktı">
  + <R I="0" V="çık" T="fiil">
  + <R I="1" V="çıktı" T="isim">
  </Word>
- <Word Index="2" Value=".">
    <Root Index="2" Value="." Type="n" />
  </Word>
</S>
```

Figure 6.6 Output file of"Finding all possible roots/stems and all suffixes (FAPRS)" procedure

The number of wordforms (NOW) in sentence 1 and sentence 2 were 5 and 4, total number of possible roots and stems (NOR) were 14 and 17, and number of suffixes (NOS) were 57 and 75 respectively. Detailed analysis of each wordform is given in Table 6.3.

In the "Finding all possible stems and inflectional suffixes (FAPSIS)" procedure, all possible stems and inflectional suffixes of the wordforms were given as an output, and all results are tagged in an XML formatted file. Outputs of sample sentences (1) and (2) for this choice is given in Figure 6.7, and detailed result document for sentence 2 was given in Appendix C.4.2.2.

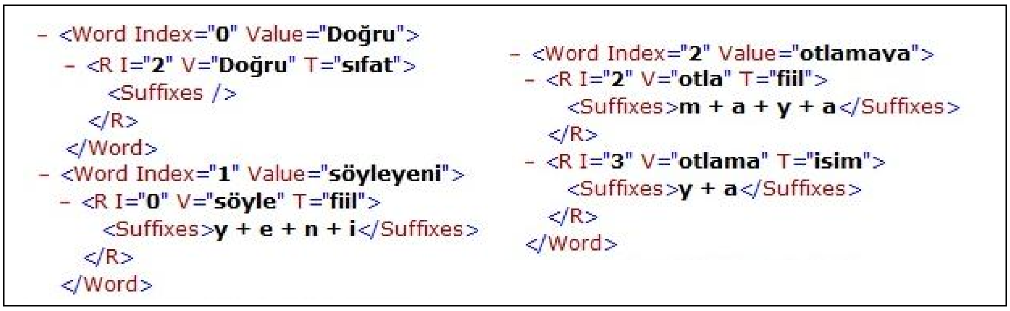


```
- <P I="0">
  - <S Index="0">
      Doğru söyleyeni dokuz köyden kovarlar .
    - <Word Index="0" Value="Doğru">
      + <R I="0" V="Doğ" T="fiil">
      + <R I="1" V="Doğru" T="sıfat">
      </Word>
      <Word Index="1" Value="söyleyeni" />
    - <Word Index="2" Value="dokuz">
      + <R I="0" V="dok" T="isim">
      + <R I="1" V="dokuz" T="isim">
      + <R I="1" V="dokuz" T="sıfat">
      </Word>
    - <Word Index="3" Value="köyden">
      + <R I="0" V="köy" T="isim">
      </Word>
    - <Word Index="4" Value="kovarlar">
      + <R I="0" V="kov" T="isim">
      + <R I="0" V="kov" T="fiil">
      + <R I="1" V="kova" T="isim">
      </Word>
    + <Word Index="1" Value=".">
    </S>
  - <S Index="1">
      Güzel koyun otlamaya çıktı .
    - <Word Index="0" Value="Güzel">
      + <R I="0" V="Güzel" T="isim">
      + <R I="0" V="Güzel" T="sıfat">
      + <R I="0" V="Güzel" T="zarf">
      </Word>
    - <Word Index="1" Value="koyun">
      + <R I="0" V="koy" T="isim">
      + <R I="0" V="koy" T="fiil">
      + <R I="1" V="koyu" T="sıfat">
      + <R I="2" V="koyun" T="isim">
      </Word>
    - <Word Index="2" Value="otlamaya">
      + <R I="0" V="otla" T="fiil">
      + <R I="1" V="otlama" T="isim">
      </Word>
    - <Word Index="3" Value="çıktı">
      + <R I="0" V="çık" T="fiil">
      + <R I="1" V="çıktı" T="isim">
      </Word>
    + <Word Index="2" Value=".">
    </S>
  </P>
```

Figure 6.7 Output file of "Finding all possible stems and inflectional suffixes (FAPSIS)" procedure

As given in Table 6.3, total number of possible roots and stems (NOR) value was decreased to 9 and 11, number of suffixes (NOS) value was decreased to 20 in Sentence 2 by using the FAPSIS module.

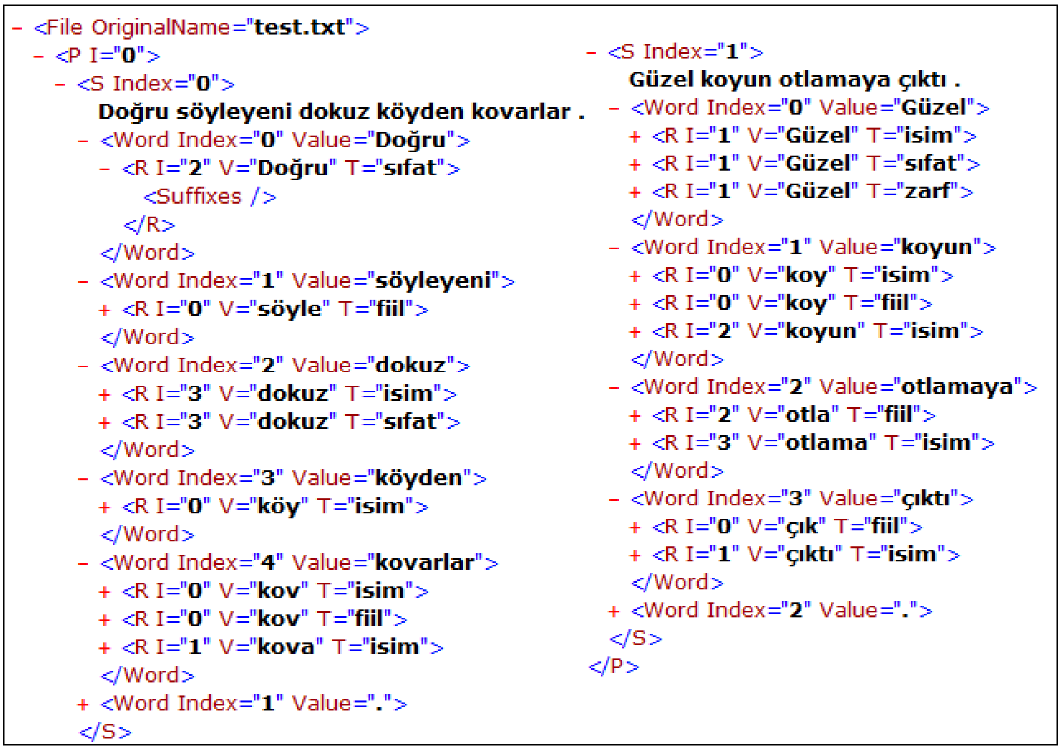
"Finding all possible roots/stems and suffixes of wordforms (The suffixes are not tagged) (FAPRS-not tagged)" procedure works like FAPRS, and all possible roots, stems and suffixes of the wordforms were given as an output in an XML formatted file. The possible combinations of suffixes were not tagged separately, but given in a combined form. Sample part of outputs for the wordforms "*doğru, söyleyeni, otlamaya*" in the sentences (1) and (2) are given in Figure 6.8, and detailed output of this procedure is given in Appendix C.4.2.3.



```
- <Word Index="0" Value="Doğru">
  - <R I="2" V="Doğru" T="sıfat">
      <Suffixes />
    </R>
  </Word>
- <Word Index="1" Value="söyleyeni">
  - <R I="0" V="söyle" T="fiil">
      <Suffixes>y + e + n + i</Suffixes>
    </R>
  </Word>
- <Word Index="2" Value="otlamaya">
  - <R I="2" V="otla" T="fiil">
      <Suffixes>m + a + y + a</Suffixes>
    </R>
  - <R I="3" V="otlama" T="isim">
      <Suffixes>y + a</Suffixes>
    </R>
  </Word>
```

Figure 6.8 Sample output file of "Finding all possible roots/stems and suffixes of wordforms (The suffixes are not tagged) (FAPRS-not tagged)"

All possible stems, roots and suffixes of the wordforms were found and the suffixes types, which were not suitable for combining with the stem/root, were eliminated, and tagged as an output in an XML formatted file by using "Eliminating possible roots/stems by using suffixes types (FAPRS - eliminated)" procedure for disambiguation in root/stem possibilities. Sample results of the sentences (1) and (2) for this choice is given in Figure 6.9, and detailed output of these sentences is given in Appendix C.4.2.4.



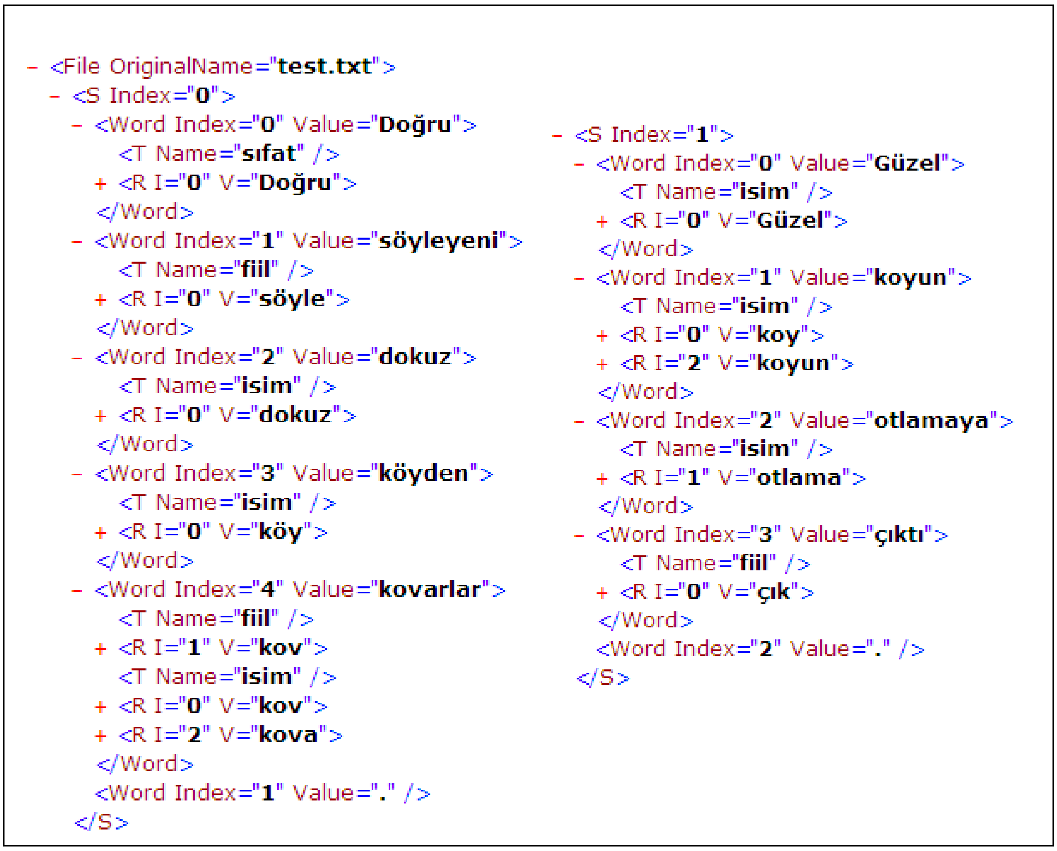
```
- <File OriginalName="test.txt">
  - <P I="0">
    - <S Index="0">
        Doğru söyleyeni dokuz köyden kovarlar .
      - <Word Index="0" Value="Doğru">
        - <R I="2" V="Doğru" T="sıfat">
            <Suffixes />
          </R>
        </Word>
      - <Word Index="1" Value="söyleyeni">
        + <R I="0" V="söyle" T="fiil">
        </Word>
      - <Word Index="2" Value="dokuz">
        + <R I="3" V="dokuz" T="isim">
        + <R I="3" V="dokuz" T="sıfat">
        </Word>
      - <Word Index="3" Value="köyden">
        + <R I="0" V="köy" T="isim">
        </Word>
      - <Word Index="4" Value="kovarlar">
        + <R I="0" V="kov" T="isim">
        + <R I="0" V="kov" T="fiil">
        + <R I="1" V="kova" T="isim">
        </Word>
      + <Word Index="1" Value=".">
      </S>
    - <S Index="1">
        Güzel koyun otlamaya çıktı .
      - <Word Index="0" Value="Güzel">
        + <R I="1" V="Güzel" T="isim">
        + <R I="1" V="Güzel" T="sıfat">
        + <R I="1" V="Güzel" T="zarf">
        </Word>
      - <Word Index="1" Value="koyun">
        + <R I="0" V="koy" T="isim">
        + <R I="0" V="koy" T="fiil">
        + <R I="2" V="koyun" T="isim">
        </Word>
      - <Word Index="2" Value="otlamaya">
        + <R I="2" V="otla" T="fiil">
        + <R I="3" V="otlama" T="isim">
        </Word>
      - <Word Index="3" Value="çıktı">
        + <R I="0" V="çık" T="fiil">
        + <R I="1" V="çıktı" T="isim">
        </Word>
      + <Word Index="2" Value=".">
      </S>
    </P>
```

Figure 6.9 Sample output file of "Eliminating possible roots/stems by using suffixes types (FAPRS - eliminated)" module

As given in Table 6.3, total number of possible roots and stems (NOR) value was decreased to 8 and 11, number of suffixes (NOS) value was decreased to 7 and 12 in Sentence 1 and 2 respectively according to the FAPRS module by using the this module.

6.4 Rule-Based POS Tagging (RB-POST)

RB-POS takes the output of RB-WD as input and eliminates the root/stem possibilities, which do not match the word ordering rules, and gives an XML file as output. Sample tagged document of the input file (Figure 6.9), is given in Figure 6.10, and detailed output is given in Appendix C.4.3.



```
- <File OriginalName="test.txt">
  - <S Index="0">
    - <Word Index="0" Value="Doğru">
        <T Name="sıfat" />
      + <R I="0" V="Doğru">
      </Word>
    - <Word Index="1" Value="söyleyeni">
        <T Name="fiil" />
      + <R I="0" V="söyle">
      </Word>
    - <Word Index="2" Value="dokuz">
        <T Name="isim" />
      + <R I="0" V="dokuz">
      </Word>
    - <Word Index="3" Value="köyden">
        <T Name="isim" />
      + <R I="0" V="köy">
      </Word>
    - <Word Index="4" Value="kovarlar">
        <T Name="fiil" />
      + <R I="1" V="kov">
        <T Name="isim" />
      + <R I="0" V="kov">
      + <R I="2" V="kova">
      </Word>
      <Word Index="1" Value="." />
    </S>
```

```
- <S Index="1">
  - <Word Index="0" Value="Güzel">
      <T Name="isim" />
    + <R I="0" V="Güzel">
    </Word>
  - <Word Index="1" Value="koyun">
      <T Name="isim" />
    + <R I="0" V="koy">
    + <R I="2" V="koyun">
    </Word>
  - <Word Index="2" Value="otlamaya">
      <T Name="isim" />
    + <R I="1" V="otlama">
    </Word>
  - <Word Index="3" Value="çıktı">
      <T Name="fiil" />
    + <R I="0" V="çık">
    </Word>
    <Word Index="2" Value="." />
  </S>
```

Figure 6.10 Output file of “POS tagging” module

The total number of possible roots and stems (NOR) was decreased to 7 and 6, and number of suffixes (NOS) was decreased to 7 and 11 in sentences (1) and (2) respectively according to the *FAPRS - eliminated* module by using the *POS tagging* module (Table 6.3).

Table 6.3 Number of Possible Stems / Roots (NOR) and Number of Suffixes (NOS) values in the analysis

			Number of Suffixes			
Wordform	**Possible Root / Stem**	**Possible Type of Root / Stem**	**FAPRS**	**FAPSIS**	**FAPRS - Eliminated**	**POS Tagging**
Doğru	Do	isim	5	Eliminated	Eliminated	Eliminated
	Doğ	fiil	10	2	Eliminated	Eliminated
	doğru	sıfat	0	0	0	0
söyleyeni	söyle	fiil	10	Eliminated	2	2
dokuz	do	isim	3	Eliminated	Eliminated	Eliminated
	dok	isim	6	5	Eliminated	Eliminated
	doku	isim	1	Eliminated	Eliminated	Eliminated
	doku	fiil	1	Eliminated	Eliminated	Eliminated
	dokuz	isim	0	0	0	0
	dokuz	sıfat	0	0	0	Eliminated
köyden	köy	isim	2	1	2	2
kovarlar	kov	isim	7	2	1	1
	kov	fiil	7	2	1	1
	kova	isim	5	2	1	1
Güzel	güz	isim	3	Eliminated	3	3
	güzel	isim	0	0	0	0
	güzel	sıfat	0	0	0	Eliminated
	güzel	zarf	0	0	0	Eliminated
koyun	koy	isim	7	5	2	2
	koy	fiil	7	5	2	2
	koyu	sıfat	7	5	Eliminated	Eliminated
	koyun	isim	0	0	0	0
otlamaya	o	sıfat	5	Eliminated	Eliminated	Eliminated
	o	zamir	5	Eliminated	Eliminated	Eliminated
	o	ünlem	5	Eliminated	Eliminated	Eliminated
	ot	isim	8	Eliminated	Eliminated	Eliminated
	ot	sıfat	8	Eliminated	Eliminated	Eliminated
	otla	fiil	9	1	1	Eliminated
	otlama	isim	5	1	1	1
çıktı	çık	fiil	6	3	3	3
	çıktı	isim	0	0	0	Eliminated
TOTAL			**132**	**34**	**19**	**18**

The total number of possible roots and stems (NOR) in FAPRS module and number of suffixes (NOS) were 31 and 132 respectively, as given in Table 6.3. After the parsing processes, NOR and NOS value were decreased to 14 and 18. The results were quite successive. Some sample sentences analyzed in RB-CorGen is given in Appendix C.4.4 and C.4.5.

6.5 Performance Overview

The software was tested on a computer, which had an Intel Core 2 6600 2.40 GHz processor and 4 GB RAM. The software used on this computer was Windows Server 2003 with SP2, .NET 2005 and SQL Server 2005.

Data sets or test sets used in the success rate determination of any related methods carried out by different researchers (Dinçer & Karaoğlan, 2004; Kiss & Strunk, 2006) could not be obtained to be able to check the success rates of new developed methods against with related works. Therefore, new data set and test set were collected by using Document Downloader program to carry out the tests on the new developed methods, RB-SBD, RB-WD and RB-POST, and determine success rates.

6.5.1. Rule-Based Sentence Boundary Detection (RB-SBD)Module

The algorithm complexity of the Rule-Based Sentence Boundary Detection (RB-SBD) Module is O(n^2) in worst case, because for all characters in a sentence all rules are compared whether they are compatible or not.

The accuracy of the module was calculated by comparing the number of the sentences found by RB-SBD module, with the number of sentences in the original texts, which were counted by linguists. The formulas for error rate and accuracy of the test are:

$$\mathrm{e} = \frac{\mathrm{F}}{\mathrm{N}}\,,\ \mathrm{A} = 1 - \frac{\mathrm{F}}{\mathrm{N}}\,, \qquad (1)$$

where e: error rate, F: Number of sentences predicted False, A: Accuracy, N: Total Number of Sentences.

A test set was generated from the data set to test RB-SBD Method. There were 10 different columnists and 20 columns of each from the first newspaper (Milliyet) and 10 different columnists and 20 columns of each from the second newspaper (Yeni Asır) in the Test Set (TS). The texts were used as is, there were not any corrections on them. The number of columns and sentences in the test sets are given in Table 6.4.

Table 6.4 Numbers of columns and sentences in the test set

Newspaper 1 (Milliyet)			**Newspaper II (Yeni Asır)**		
Name of Columnist	**Number of Columns**	**Number of Sentences**	**Name of Columnist**	**Number of Columns**	**Number of Sentences**
C1	20	798	C1	20	582
C2	20	1.746	C2	20	1.458
C3	20	406	C3	20	546
C4	20	834	C4	20	1.126
C5	20	862	C5	20	1.316
C6	20	697	C6	20	797
C7	20	546	C7	20	972
C8	20	1.252	C8	20	795
C9	20	661	C9	20	634
C10	20	532	C10	20	852
Total	200	8.334	Total	200	9.078
Total Number of Sentences = 17.412					

The results were given in Table 6.5.

Table 6.5 Accuracy of Sentence Boundary Detection Module

Columns	**# of Sentences**	**# of Sentences Predicted True**	**# of Sentences Predicted False**	**e (%)**	**AR (%)**	**Accuracy – Except Misspellings (%)**
NP1	8.334	8.306	28	0.34	99.66	99.80
NP2	9.078	9.042	36	0.4	99.60	99.76
TOTAL	**17.412**	**17.348**	**64**	**0.37**	**99.63**	**99.78**

The method was tested on 17.412 sentences (Table 6.5); 17.348 sentences were resolved correctly, 64 sentences were resolved inaccurately, and the average success rate was calculated as 99.63 with the original texts, and 99.78 after the misspellings were ignored.

Also, 10-Fold Cross Validation technique was used for testing this module in details. At first, the True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) values were determined and counted by the linguists in each dataset of 10 datasets including 20.351 sentences totally, which were generated from randomly chosen texts in DataSet downloaded and generated by the Document Downloader program, to use in the calculations of SENS (Sensitivity), SPEC (Specificity), PREC(Precision), ACC (Accuracy), and AUC (Area Under the Curve) values (Formulas 2, 3, 4,5,6).

SENS (Sensitivity) = TP / (TP + FN) (2)

SPEC (Specificity or True Negative Rate) = TN / N = TN / (FP + TN) = 1 – (FP/ (FP+TN)) (3)

PREC (Precision) = TP / (TP+FP) (4)

ACC (Accuracy) = (TP + TN) / (TP + FP + TN + FN) (5)

AUC (Area under the Curve) = $$\frac{SENS*(\frac{\mathrm{FP}}{\mathrm{FP+TN}}))}{2} + SPEC * SENS + \frac{SPEC*(1-(\frac{TP}{(TP+FN)}))}{2} \quad (6)$$

Table 6.6 The SENS, SPEC, ACC and AUC values

Dataset	TP	FP	FN	TN	TOTAL	SENS	SPEC	PREC	ACC	AUC
1	2.070	3	4	1	2.078	0,9981	0,7500	0,9986	99,66%	0,8740
2	2.250	4	2	1	2.257	0,9991	0,8000	0,9982	99,73%	0,8996
3	1.973	14	0	3	1.990	1	0,8235	0,9930	99,30%	0,9118
4	1.989	12	0	1	2.002	1	0,9231	0,9940	99,40%	0,9615
5	2.001	8	0	2	2.011	1	0,8000	0,9960	99,60%	0,9000
6	1.985	6	0	1	1.992	1	0,8571	0,9970	99,70%	0,9286
7	2.014	5	0	0	2.019	1	1,0000	0,9975	99,75%	1,0000
8	1.994	5	0	1	2.000	1	0,8333	0,9975	99,75%	0,9167
9	1.990	2	0	0	1.992	1	1,0000	0,9990	99,90%	1,0000
10	2.007	2	1	0	2.010	0,9995	1,0000	0,9990	99,85%	0,9998
Average					2.035,1	0,9997	0,8787	0,997	99.66%	94.84%

As given in Table 6.6., the results of the test was encouraging with the average values of ACC and AUC values, which were determined as 99.66% and 94.84%, and are the highest values for sentence boundary detection process in Turkish.

Some paragraphs from texts and correctly resolved sentences by the program are given in Table 6.7.

Table 6.7 Sample paragraphs and correctly resolved sentences

Original Text	**Parsed Sentences**

Biliyor musunuz, geçenlerde 'Çırağan Palace Hotel Kempinski'nin Tuğra Restaurant'ı 'Dünyanın en iyi 10 mutfağı' arasına girdi.
 	
Do you know that Çırağan Palace Hotel Kempinski's Tuğra Restaurant had a degree in the 'The Best 10 Kitchens of the World' recently.
	`<P I="0">`
	`<S Index="0">Biliyor musunuz, geçenlerde 'Çırağan Palace Hotel Kempinski'nin Tuğra Restaurant'ı 'Dünyanın en iyi 10 mutfağı' arasına girdi.`
	`</S>`

Düşünün 7 milyar insanın yaşadığı koca dünya, binlerce otel, lokanta ve...ilk on arasında bizim Tuğra Restaurant... Üstelik dünyanın en saygın uzmanlarından oluşan jüri tarafından seçildi.	`<P I="2">`
	`<S Index="0">Düşünün 7 milyar insanın yaşadığı koca dünya, binlerce otel, lokanta ve...ilk on arasında bizim Tuğra Restaurant.... </S>`
	`<S Index="1">Üstelik dünyanın en saygın uzmanlarından oluşan jüri tarafından seçildi. </S>`

O yemekler, o müzik ve Boğaz... Kendinizi kesinlikle zaman tüneline sokar, en azından 150 yıl öncesine gidersiniz. Kendinizi 'sultan' sanabilirsiniz.
 	
Think that a huge world, in which 7 billions people live, thousands of hotels, restaurants, and...our Tuğra Restaurant had a degree in first 10... Besides, it had been chosen by the jury, which was comprised from the most respected specialists.
	`<P I="4">`
	`<S Index="0">O yemekler, o müzik ve Boğaz.... </S>`
	`<S Index="1"> Kendinizi kesinlikle zaman tüneline sokar, en azından 150 yıl öncesine gidersiniz. </S>`
	`<S Index="2"> Kendinizi 'sultan' sanabilirsiniz. </S>`

Bu da, Kont von M...'yi, bahçesini, en güzel çeşitlemeler içinde birbiriyle kesişerek şirin vadiler oluşturan tepelerden birinde kurmaya yöneltmiş. *This situation directed Kont von M… to set up his garden on one of the hills that create sweet valleys by intersecting each others with the best variations.*	`<Sentence Index="0"> Bu da, Kont von M...'yi bahçesini, en güzel çeşitlemeler içinde birbiriyle kesişerek şirin vadiler oluşturan tepelerden birinde kurmaya yöneltmiş.</Sentence>` Normally, "…" punctuation is used at the end of sentence, but it was used in the sentence in place of the name of a person in this sentence, this can generate an ambiguity.

Inaccuracies in the results were generally caused by misspellings in the texts (Table 6.8). But sometimes, they are caused by the rules. For example, in the first sentence in Table 6.8, the “…” mark is used inaccurately. According to the sentence boundary rules, a capital letter must be placed after this mark and new sentence must begin. But in this sentence this character was used to shorten the cited text.

Table 6.8 Sample paragraphs and inaccurately resolved sentences

Original Sentence	**Parsed Sentences**

Devamı şöyle: Millî Eğitim Bakanı’nın imzasıyla tüm okullara gönderilen genelgede... deniliyordu.
It continues such that: It is said ... in the notice that was signed by the Head of the Department of Education and sent to all schools.
	`<Sentence Index="1">Devamı şöyle: Millî Eğitim Bakanının imzasıyla tüm okullara gönderilen genelgede.</Sentence>`
	`<Sentence Index="2"> deniliyordu. </Sentence>`

Ama, düz yolda gitmeyi bilmeden, bir elinizde telefon, ağzınızda sigara... bu bir.
But, there is a telephone in one of your hands; a cigarette in your mouth without knowing to go on the straight road… this is first.
	`<Sentence Index="4">Ama, düz yolda gitmeyi bilmeden, bir elinizde telefon, ağzınızda sigara.</Sentence>`
	`<Sentence Index="5"> bu bir. </Sentence>`

Telekom Genel Müdürü Mehmet Ekinalan her fırsatta Telekom'un 'muhteşem!' faaliyetlerini öve öve bitiremiyor.
Mehmet Ekinalan, who is the Manager of the Telecommunication Department, praises the ‘magnificent!’ activities of the department all the time.
	`<Sentence Index="0">Telekom Genel Müdürü Mehmet Ekinalan her fırsatta Telekom'un 'muhteşem!</Sentence>`
	`<Sentence Index="1">' faaliyetlerini öve öve bitiremiyor. </Sentence>`

Tetikçileri var, devlet içinde devlet olmuşlar, devlet adına çalışıyorlar, devlet adamlarıyla ahbap çavuşlar.. şu, bu!
They have triggermen, create a state in the state, work for the government, good friends with government… this, that!
	`<Sentence Index="0">Tetikçileri var, devlet içinde devlet olmuşlar, devlet adına çalışıyorlar, devlet adamlarıyla ahbap çavuşlar.</Sentence>`
	`<Sentence Index="1"> şu, bu!</Sentence>`

- Bugün saat kaçta gideceksin?
- 2. Sen?
- 5.
- At what time will you go today?
- 2. You?
- 5.
	`<Paragraph Index="0">`
	`<Sentence Index="0"> Bugün saat kaçta gideceksin? </Sentence> </Paragraph>`
	`<Paragraph Index="1">`
	`<Sentence Index="0">`**`2. Sen?`**
	`</Sentence>`
	`</Paragraph>`
	`<Paragraph Index="2">`
	`<Sentence Index="0">` **`5.`** `</Sentence>`
	`</Paragraph>`
	There are two sentences: “2.” and “Sen? (You?)”, in the second line. The algorithm assumed the “.” (dot) mark used after the number “2” as enumeration not for bulleting, by using the rule defined in the list, and took all line as one sentence.

The total process time of this analysis was 179 milliseconds (0.179 seconds), details of which are given in Table 6.9.

Table 6.9 Process time of sentence boundary detection module

Newspaper 1 (Milliyet)			**Newspaper II (Yeni Asır)**		
Name of Columnist	**Number of Sentences**	**Process Time (milliseconds)**	**Name of Columnist**	**Number of Sentences**	**Process Time (milliseconds)**
C1	798	11	C1	582	6
C2	1.746	26	C2	1.458	10
C3	406	0	C3	546	4
C4	834	13	C4	1.126	8
C5	862	14	C5	1.316	9
C6	697	8	C6	797	7
C7	546	5	C7	972	8
C8	1.252	18	C8	795	7
C9	661	8	C9	634	6
C10	532	4	C10	852	7
Total	**8.334**	**107**		**9.078**	**72**

6.5.2. Rule-Based Morphological Analyser (RB-MA)Module

The algorithm complexity of the Rule-Based Morphological Analyser (RB-MA) Module is O(n!) in worst case. In this module, finding all possible roots, stems and suffixes process took very long time at the fist stages of this study. In order to increase overall system performance, some improvements and modifications were realized.

Because of the difficulty of the manually counting all possible roots, stems and suffixes' combinations by linguists, instead of using all test set used in performance analysis of RB-SBD Method, only small part of it was used in the RB-MA Method.

In this module, while controlling whether the part of wordform is valid root/stem/suffix or not, algorithm tried to connect database and controlled it in the related table. Each connection took 18 ms in average. Algorithm connected over 80.000 times to analyze the sentence;

```
Dümenin terbiye edemediğini kayalar terbiye eder.   (3)
(The rocks chasten it, which the rudder cannot chasten.)
```

Therefore, analysis of this sentence takes 1440 seconds, which means 24 minutes. Considering that a column in a newspaper has approximately 300 sentences, analysis of each column takes 120 hours, which means 5 days. These values cannot be acceptable for the usability of the system.

In order to solve this problem, lists of root, stem and suffixes were kept in the memory when the program started. It connects database only one time at the beginning and all root, stem and suffix information is saved into three lists. Then, character combinations are checked by using these lists. The process time decreased from 24 minutes to 24 seconds for the sample sentence (3) by using this approach, which causes a system speed increase by 60 times.

Although this new process time was better than the previous one, one column could be analyzed in 2 hours by using this value of time. This performance was also not acceptable and made the system unusable. Therefore, new improvement was done. The suffix combination control process of the algorithm was changed. In order to control each suffix whether it is suitable for using with previous found suffixes, control process was done after the determination of each suffix combination, so the efficiency of the algorithm was increased by 25% and the sentence (3) was analyzed in 6 seconds.

This value was achieved by using the given rule file, which includes 15 rules. By increasing number of rules, the suffix possibilities and the time of checking processes are expected to be decreased.

A refinement was made by saving the found roots, stems and suffixes of the analyzed word to a file, in order to avoid re-analysis of the same word. This process was increased the system performance.

After the refinements and improvements, a text, which contains 308 sentences and nearly 3000 words, was analyzed in 3 seconds by *RB-WD* module.

6.5.3. Rule-Based POS Tagging (RB-POST) Module

The algorithm complexity of the Rule-Based Sentence POS Tagging (RB-POST) Module is O(n^2) in worst case.

Since RB-POST is a once-in-runtime process, the only overhead of this module to the system would be as much as the 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 6.10.

Table 6.10 POS Tagging rule parser performance test results

Number of Rules	Average Parsing Time (millisec.)
10	0.3271
100	1.1916
200	2.0696

The results show that the time required for loading rules is not directly proportional to the number of rules, and 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 input documents 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.

According to the limited amount of analyzed text, tests were carried out on 3 different documents with 6, 89, and 226 sentences each. The average parsing time for a document with 226 sentences was about 18 milliseconds (Table 6.11).

Table 6.11 POS Tagging stem reader performance test results

Total Number of Sentences	**Total Number of Words**	**Total Number of Morph Items**	**Average Parsing Time (millisec.)**
6	30	51	0,6342
89	418	737	6,7597
226	1072	1888	18,1594

140 Turkish sentences were randomly taken from the data set to calculate the overall tagging results. Each sentence has at least one word with more than one possible word classes. The accuracy achieved by using this algorithm on this text corpus is given in Table 6.12.

Table 6.12 POS tagging results based on total words

Number of Total Words	**Number of Correctly Tagged Words**	**Number of Incorrectly Tagged Words**	**Accuracy**
780	718	62	92 %

The success rate of *RB-POST* were determined as 92%. Since this test was carried out by using a limited rule list, the accuracy of the program can be incremented using a larger scale rule list, supported by linguists.

CHAPTER SEVEN
USAGE AND USER INTERFACES OF RBCorGen

"Rule-Based Automatic Corpus Generation (RB-CorGen)" application consists of two big sub-applications:

1. Document Downloader
2. Automatic Rule-Based Corpus Generation

In the "Document Downloader (*Döküman İndirici*)", the electronic data is taken from web by URL links of the newspapers and stored in database to be able to use efficiently in RB-CorGen.

In "Automatic Corpus Generation (*Otomatik Derlem Oluşturma*) (RB-CorGen)" application, there is a user-friendly interface to generate sentence corpus, make morphological analysis of the words in the sentences and apply part-of-speech tagging process on the analysis.

7.1 Document Downloader

User interface of "Document Downloader (*Döküman İndirici*)" project was implemented in Turkish (Figure 7.1.).

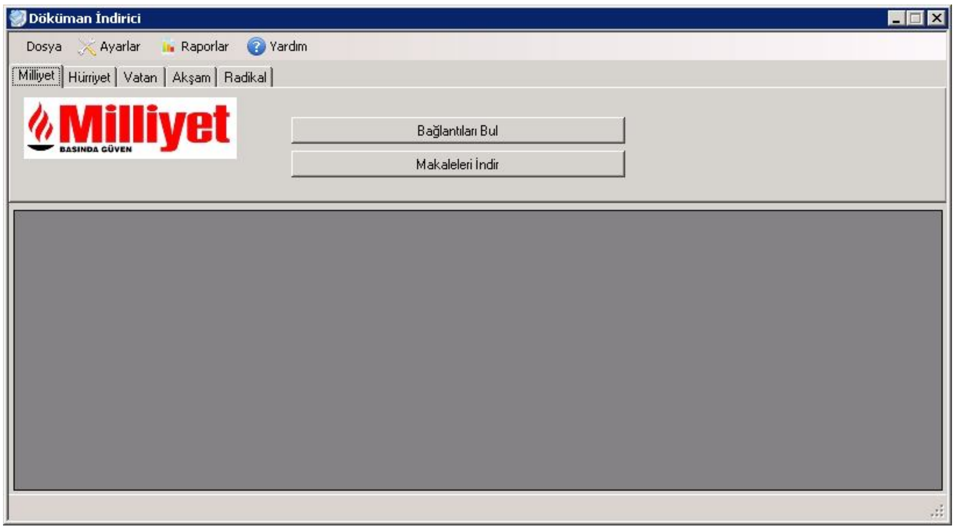
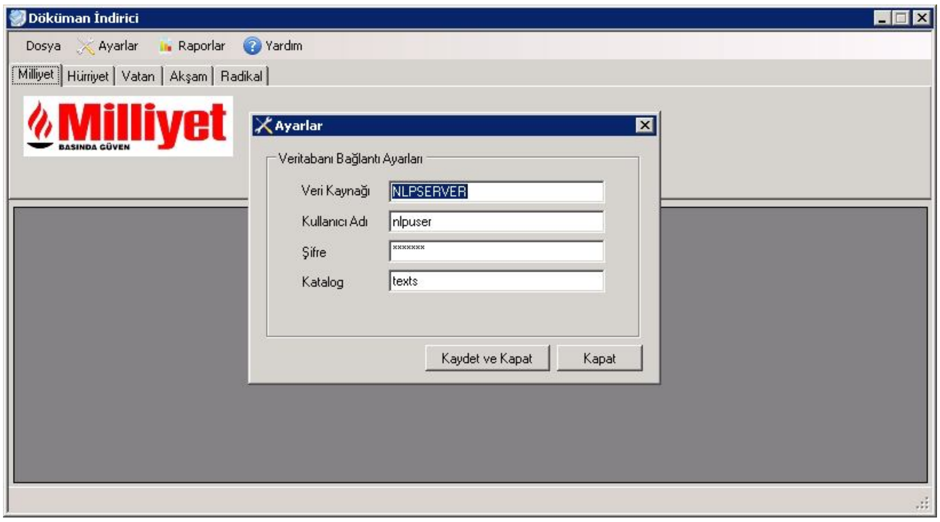


Figure 7.1 Main screen of the *Document Downloader* application

Application has two main parts; user menu that is at the top of the window and tab blocks of the resources that are on the centre of it. On the menu bar section, user can change database connection settings, get reports, and open help. For changing database connection settings, user clicks “Settings (*Ayarlar*)” menu button (Figure 7.2).



Veri Kaynağı: NLPSERVER
Kullanıcı Adı: nlpuser
Şifre: *******
Katalog: texts

Figure 7.2 Database connection settings window

In this settings window user can change data source, username, password, and catalog values. On help menu, user can open “How I do (*Nasıl Yaparım*)” user manual and “About (*Hakkında*)” information of the application (Figure 7.3).

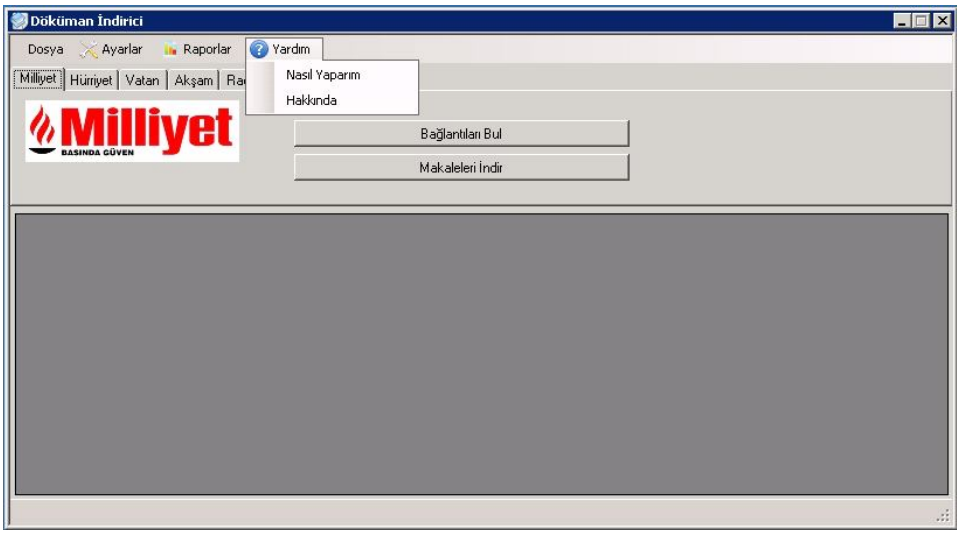
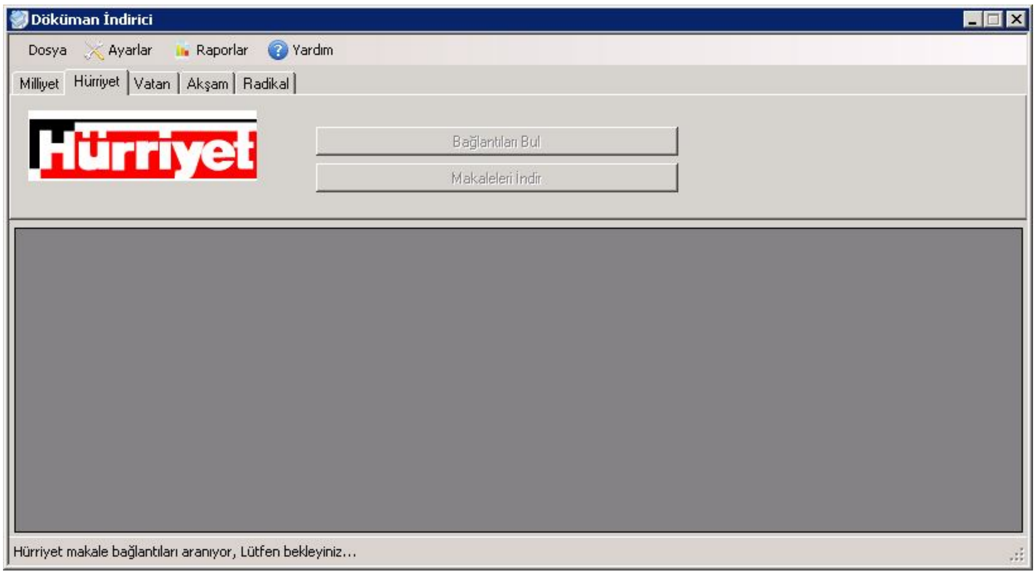


Figure 7.3 Help menu of application

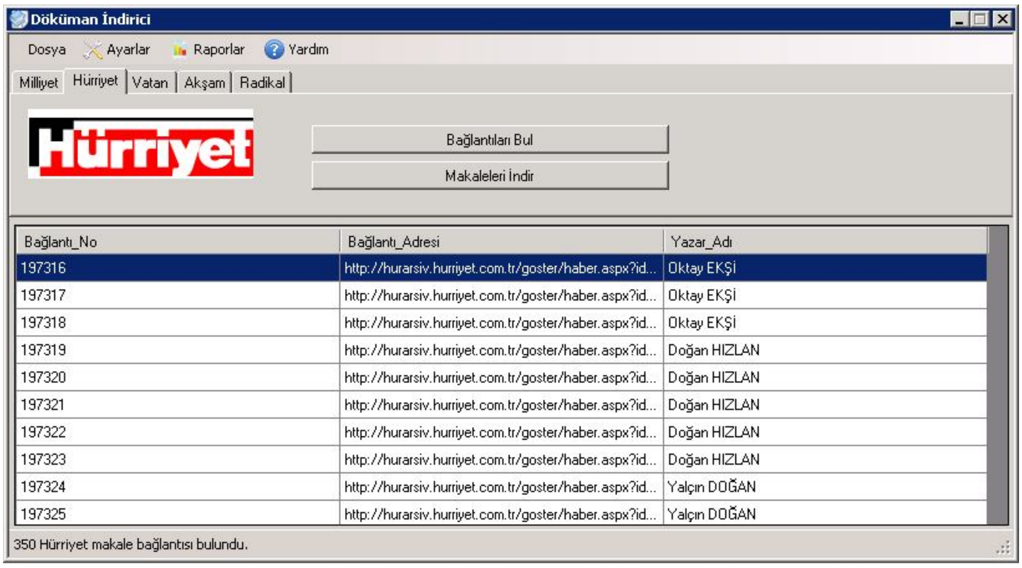
General usage of the application is quite simple. User selects the tab section which columns of the newspaper the user wants to download, and then the user clicks on the "Find Links (*Bağlantıları Bul*)" button and all buttons become disabled until operation is completed (Figure 7.4). Finding links operation has a long run time, so user should be patient on this operation.



Hürriyet makale bağlantıları aranıyor, Lütfen bekleyiniz...

Figure 7.4 Disabled user interface

When the link finding operation ends, the links are listed on a grid table. Also the article number is given on the status bar of the application (Figure 7.5).

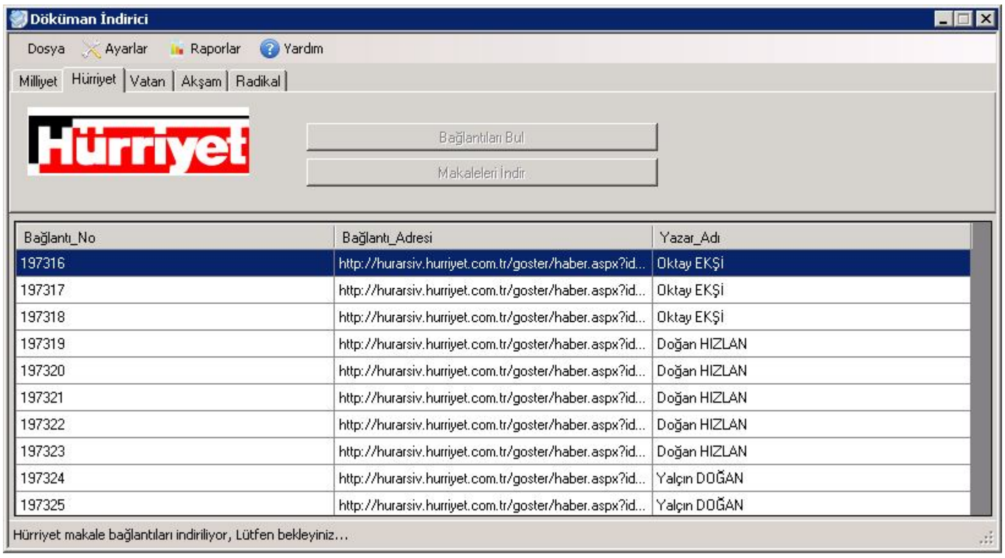


Bağlantı_No	Bağlantı_Adresi	Yazar_Adı
197316	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197317	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197318	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197319	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197320	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197321	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197322	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197323	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197324	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Yalçın DOĞAN
197325	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Yalçın DOĞAN

350 Hürriyet makale bağlantısı bulundu.

Figure 7.5 Result of link finding operations

Second step of the download process is downloading articles that the links are listed. User clicks on the button “Download Articles (*Makaleleri İndir*)” and the download operation is started by the user (Figure 7.6).

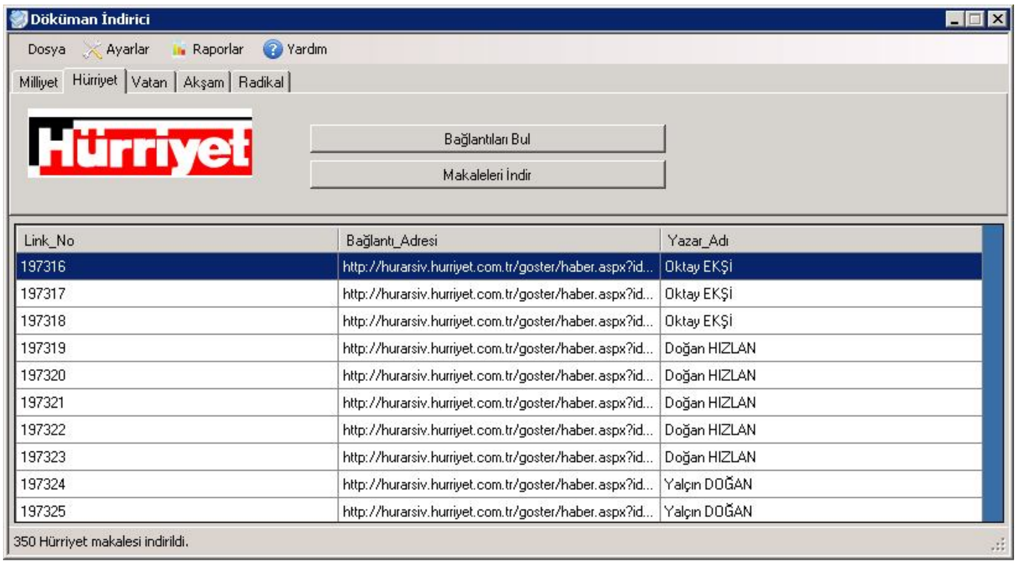


Bağlantı_No	Bağlantı_Adresi	Yazar_Adı
197316	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197317	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197318	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197319	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197320	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197321	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197322	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197323	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197324	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Yalçın DOĞAN
197325	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Yalçın DOĞAN

Hürriyet makale bağlantıları indiriliyor, Lütfen bekleyiniz...

Figure 7.6 Screen statues while downloading article

At the end of the downloading articles operation the number of articles downloaded is written on the status bar (Figure 7.7).

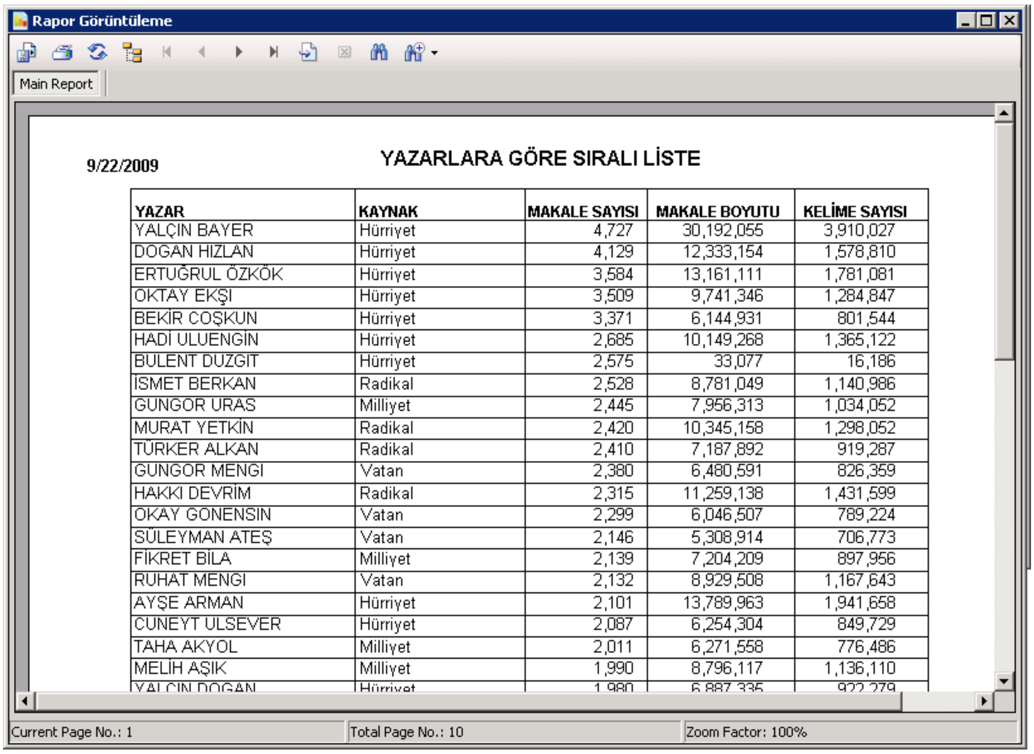


Link_No	Bağlantı_Adresi	Yazar_Adı
197316	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197317	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197318	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Oktay EKŞİ
197319	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197320	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197321	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197322	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197323	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Doğan HIZLAN
197324	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Yalçın DOĞAN
197325	http://hurarsiv.hurriyet.com.tr/goster/haber.aspx?id...	Yalçın DOĞAN

350 Hürriyet makalesi indirildi.

Figure 7.7 Screen statues when download operation completed

Reporting operations of the application is done by clicking on one of the "Reports (*Raporlar*)" on the reports menu, sample report is given in Figure 7.8.



9/22/2009

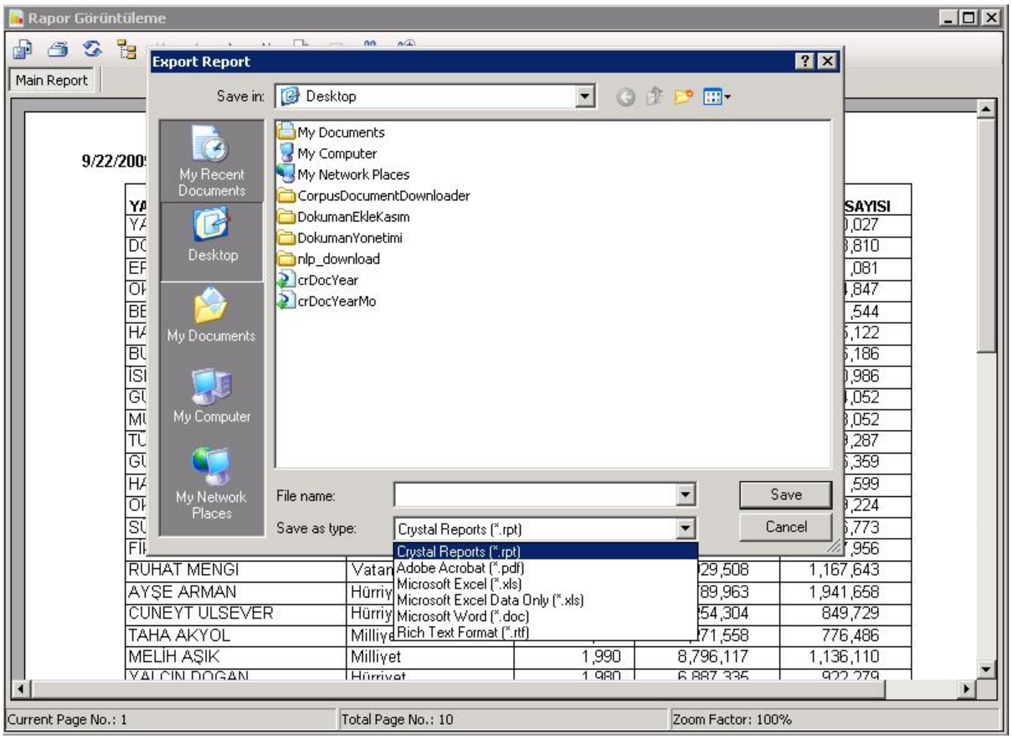
YAZARLARA GÖRE SIRALI LİSTE

YAZAR	KAYNAK	MAKALE SAYISI	MAKALE BOYUTU	KELİME SAYISI
YALÇIN BAYER	Hürriyet	4,727	30,192,055	3,910,027
DOGAN HIZLAN	Hürriyet	4,129	12,333,154	1,578,810
ERTUĞRUL ÖZKÖK	Hürriyet	3,584	13,161,111	1,781,081
OKTAY EKŞI	Hürriyet	3,509	9,741,346	1,284,847
BEKİR COŞKUN	Hürriyet	3,371	6,144,931	801,544
HADİ ULUENGİN	Hürriyet	2,685	10,149,268	1,365,122
BULENT DUZGIT	Hürriyet	2,575	33,077	16,186
İSMET BERKAN	Radikal	2,528	8,781,049	1,140,986
GUNGOR URAS	Milliyet	2,445	7,956,313	1,034,052
MURAT YETKİN	Radikal	2,420	10,345,158	1,298,052
TÜRKER ALKAN	Radikal	2,410	7,187,892	919,287
GUNGOR MENGI	Vatan	2,380	6,480,591	826,359
HAKKI DEVRİM	Radikal	2,315	11,259,138	1,431,599
OKAY GONENSIN	Vatan	2,299	6,046,507	789,224
SÜLEYMAN ATEŞ	Vatan	2,146	5,308,914	706,773
FİKRET BİLA	Milliyet	2,139	7,204,209	897,956
RUHAT MENGI	Vatan	2,132	8,929,508	1,167,643
AYŞE ARMAN	Hürriyet	2,101	13,789,963	1,941,658
CUNEYT ULSEVER	Hürriyet	2,087	6,254,304	849,729
TAHA AKYOL	Milliyet	2,011	6,271,558	776,486
MELİH AŞIK	Milliyet	1,990	8,796,117	1,136,110
YALCIN DOGAN	Hürriyet	1,980	6,887,335	922,279

Current Page No.: 1 | Total Page No.: 10 | Zoom Factor: 100%

Figure 7.8 Sample report

Reports can be exported as several file types: Crystal Reports (rpt), Adobe Acrobat (pdf), Microsoft Excel (xls), Microsoft Excel Data Only (xls), Microsoft Word (doc), and Rich Text Format (rtf). Exporting report can be done by clicking on the button with disk icon. And the save dialog is opened (Figure 7.9).



Rapor Görüntüleme

Main Report

9/22/200

Export Report

Save in: Desktop

My Recent Documents
Desktop
My Documents
My Computer
My Network Places

- My Documents
- My Computer
- My Network Places
- CorpusDocumentDownloader
- DokumanEkleKasım
- DokumanYonetimi
- nlp_download
- crDocYear
- crDocYearMo

File name:

Save as type: Crystal Reports (*.rpt)

- Crystal Reports (*.rpt)
- Adobe Acrobat (*.pdf)
- Microsoft Excel (*.xls)
- Microsoft Excel Data Only (*.xls)
- Microsoft Word (*.doc)
- Rich Text Format (*.rtf)

Save

Cancel

YA				SAYISI
RUHAT MENGI	Vatan		29,508	1,167,643
AYŞE ARMAN	Hürriy		89,963	1,941,658
CUNEYT ULSEVER	Hürriy		54,304	849,729
TAHA AKYOL	Milliye		71,558	776,486
MELİH AŞIK	Milliyet	1,990	8,796,117	1,136,110
YALCIN DOGAN	Hürriyet	1,980	6,887,335	922,279

Current Page No.: 1 | Total Page No.: 10 | Zoom Factor: 100%

Figure 7.9 Save dialog box

Since the URL links of the articles in the newspapers are different, different downlading interface was designed for the newspapers Milliyet, Hürriyet, Vatan, Akşam and Radikal.

Milliyet newspaper’s downloading tab is given in Figure 7.10.

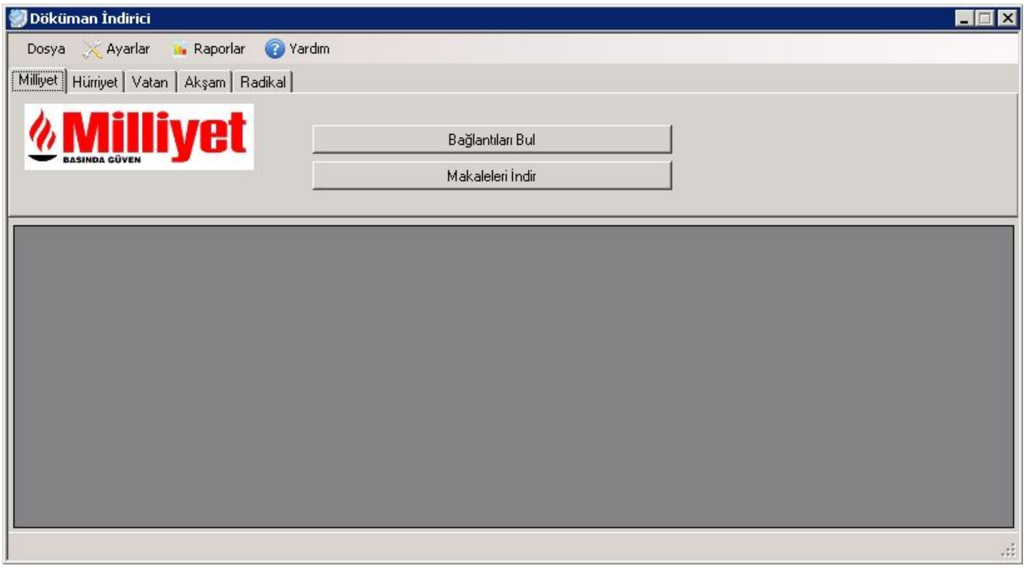


Figure 7.10 Milliyet newspaper’s downloading tab

Hürriyet newspaper's downloading tab is given in Figure 7.11.

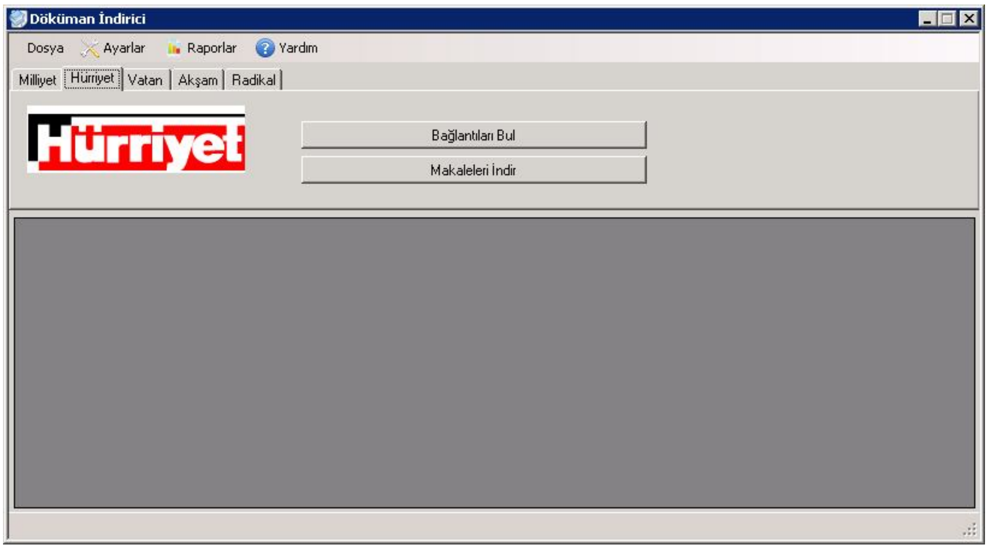


Figure 7.11 Hürriyet newspaper's downloading tab

Vatan newspaper's downloading tab is given in Figure 7.12.

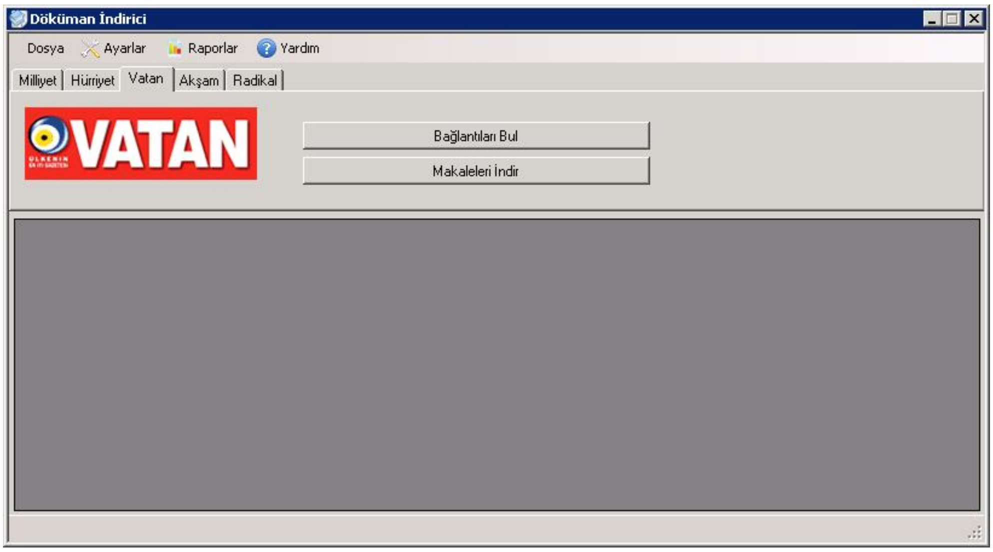


Figure 7.12 Vatan newspaper's downloading tab

Akşam newspaper's downloading tab is given in Figure 7.13.

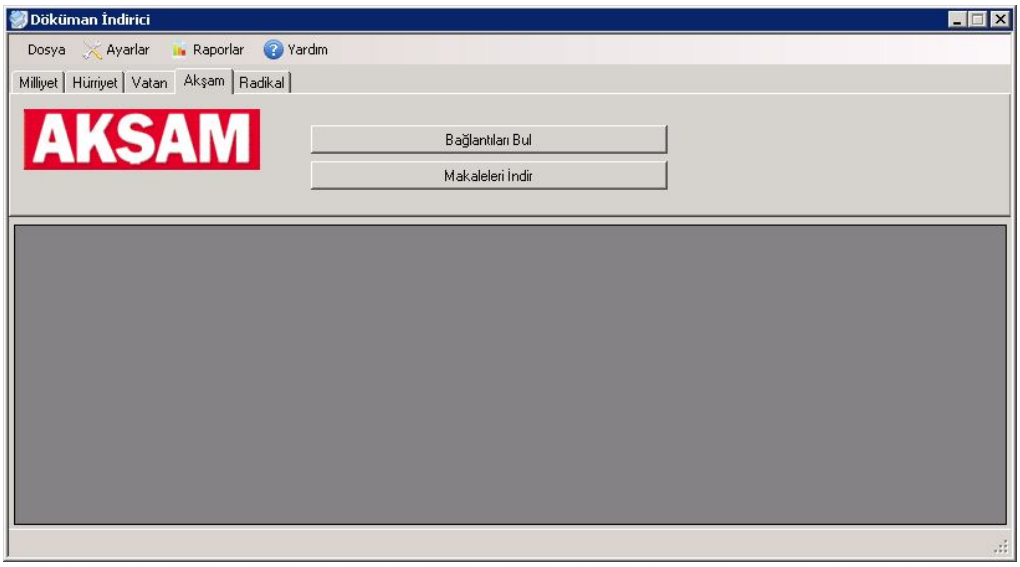


Figure 7.13 Akşam newspaper’s downloading tab

Radikal newspaper’s downloading tab is given in Figure 7.14.

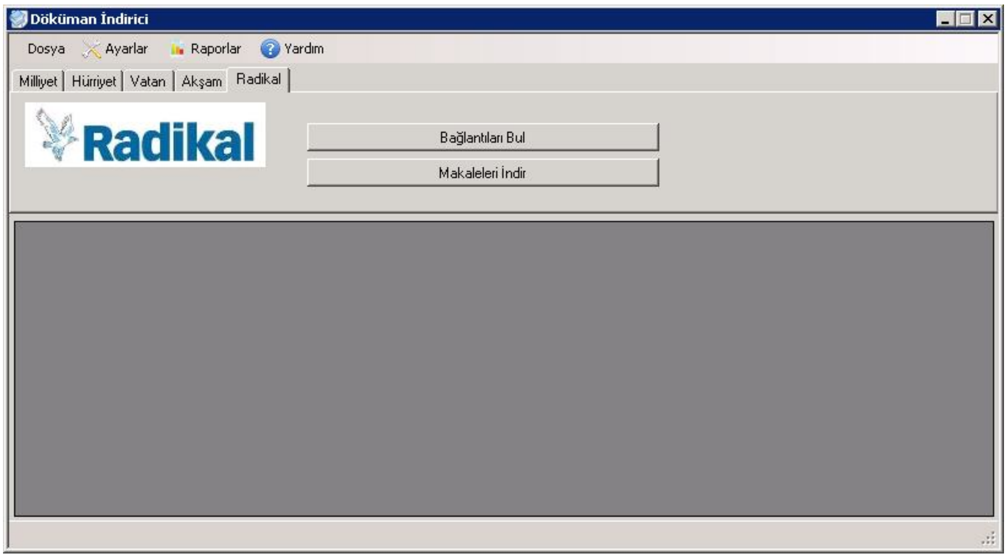


Figure 7.14 Radikal newspaper’s downloading tab

7.2 Automatic Corpus Generation

User interface of “Rule-Based Automatic Corpus Generation (*Otomatik Kural-Tabanlı Derlem Oluşturma*) (RBCorGen)” application is implemented in Turkish (Figure 7.15).

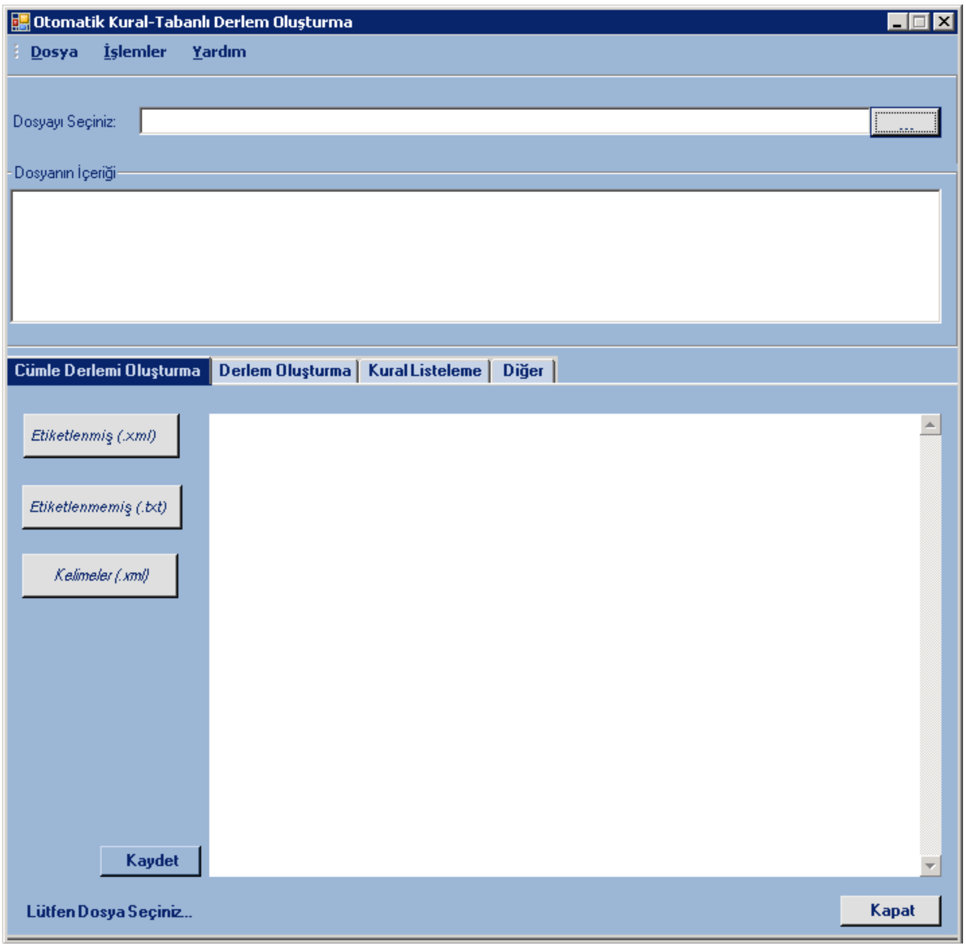


Figure 7.15 Initial Screen of RBCorGen

This interface has a main menu on the top, file loading and screening part and four tabs, which contains all processes in RBCorGen.

Firstly, a document, which will be analyzed, is loaded into the application by using load button.

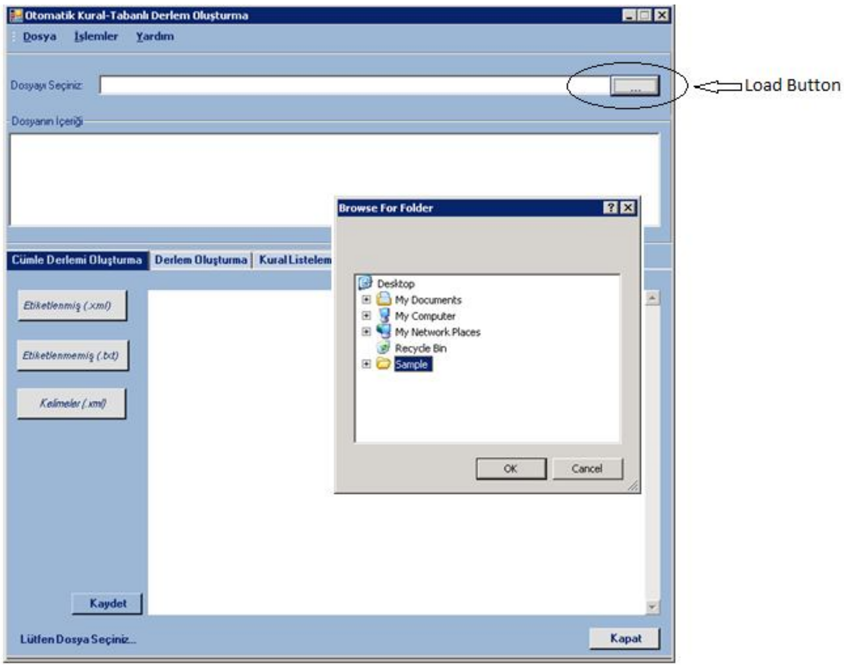


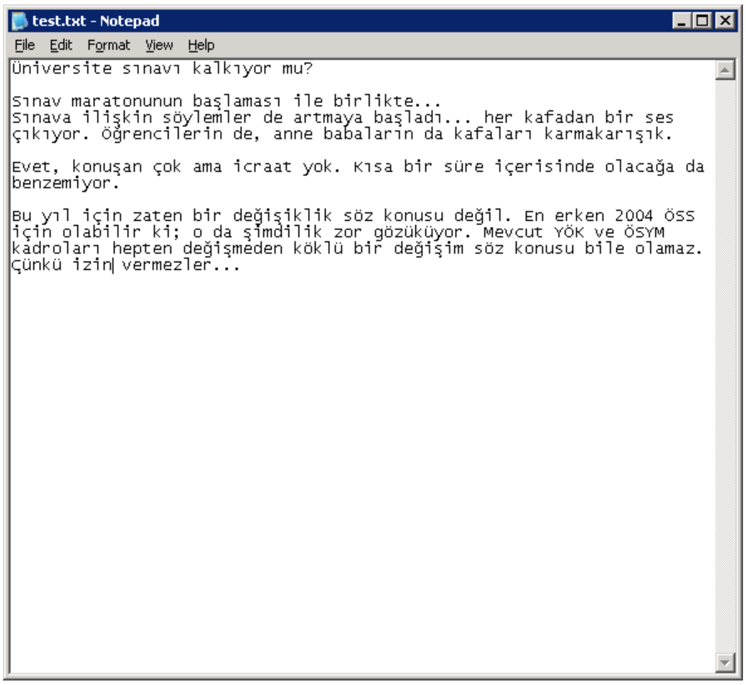
Figure 7.16 Loading a document into RBCorGen

Then, any process is chosen by using the main menu or tabs.

7.2.1 Generating Sentence Corpus

"Cümle Derlemi Oluşturma" tab in the main screen is used for generating sentence corpus in three ways as tagged in XML format, not tagged and stored in text file or tagged in XML format with splitted wordforms.

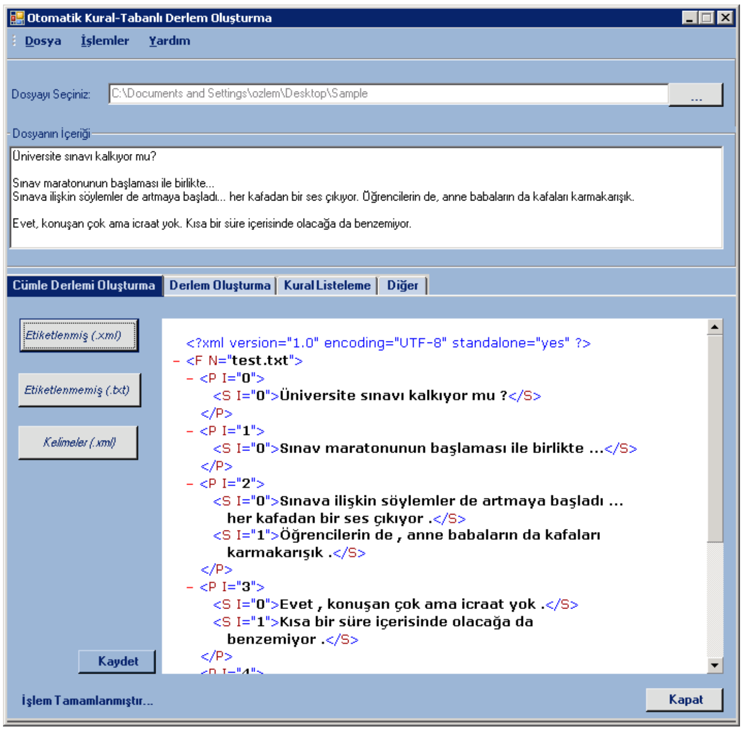
For example, the text, which was written by Abbas Güçlü named as "abbasguclu.txt", from the newspaper "Milliyet" (Figure 7.17), is loaded into the application.



```
Üniversite sınavı kalkıyor mu?

Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses
çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da
benzemiyor.

Bu yıl için zaten bir değişiklik söz konusu değil. En erken 2004 ÖSS
için olabilir ki; o da şimdilik zor gözüküyor. Mevcut YÖK ve ÖSYM
kadroları hepten değişmeden köklü bir değişim söz konusu bile olamaz.
Çünkü izin vermezler...
```

Figure 7.17 Text will be analyzed

If the “*Etiketlenmiş (.xml)*” is chosen, the text is splitted into sentences, stored in XML formatted file and shown in the application (Figure 7.18).



Dosyayı Seçiniz: C:\Documents and Settings\ozlem\Desktop\Sample

Dosyanın İçeriği

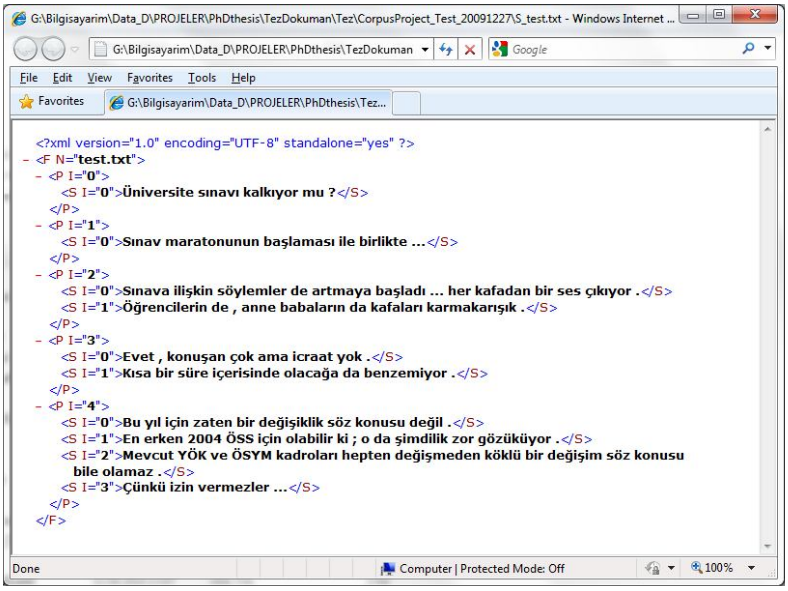
```
Üniversite sınavı kalkıyor mu?

Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da benzemiyor.
```

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <F N="test.txt">
  - <P I="0">
      <S I="0">Üniversite sınavı kalkıyor mu ?</S>
    </P>
  - <P I="1">
      <S I="0">Sınav maratonunun başlaması ile birlikte ...</S>
    </P>
  - <P I="2">
      <S I="0">Sınava ilişkin söylemler de artmaya başladı ...
        her kafadan bir ses çıkıyor .</S>
      <S I="1">Öğrencilerin de , anne babaların da kafaları
        karmakarışık .</S>
    </P>
  - <P I="3">
      <S I="0">Evet , konuşan çok ama icraat yok .</S>
      <S I="1">Kısa bir süre içerisinde olacağa da
        benzemiyor .</S>
    </P>
    <P I="4">
```

İşlem Tamamlanmıştır...

Figure 7.18 XML tagged output of text in Figure 7.17

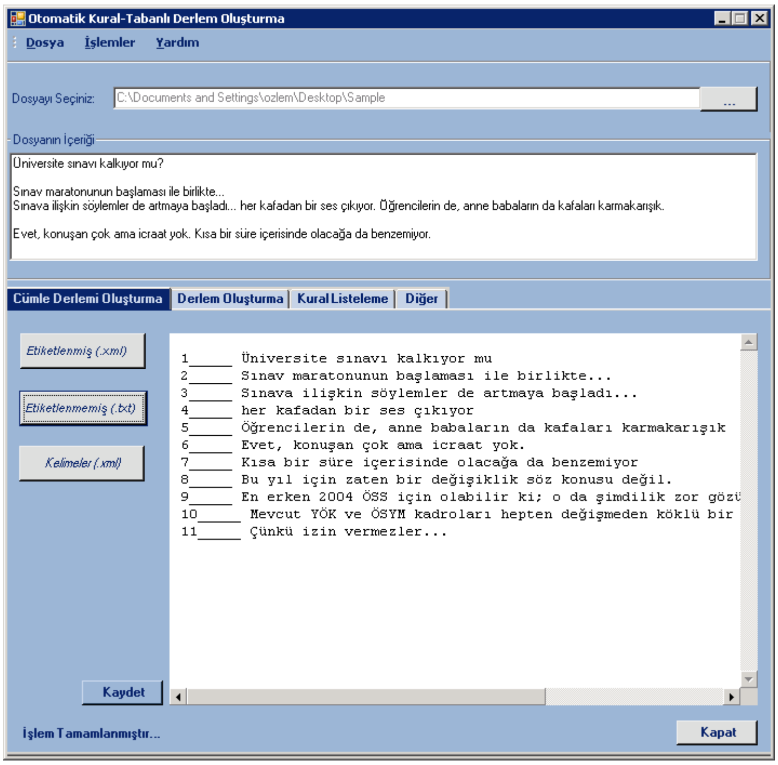
The output file is named as “S_AnalyzedFileName.xml” (Figure 7.19).


G:\Bilgisayarim\Data_D\PROJELER\PhDthesis\TezDokuman\Tez\CorpusProject_Test_20091227\S_test.txt - Windows Internet ...
G:\Bilgisayarim\Data_D\PROJELER\PhDthesis\TezDokuman
Google
File
Edit
View
Favorites
Tools
Help
Favorites
G:\Bilgisayarim\Data_D\PROJELER\PhDthesis\Tez...
Done
Computer | Protected Mode: Off
100%

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <F N="test.txt">
  - <P I="0">
      <S I="0">Üniversite sınavı kalkıyor mu ?</S>
    </P>
  - <P I="1">
      <S I="0">Sınav maratonunun başlaması ile birlikte ...</S>
    </P>
  - <P I="2">
      <S I="0">Sınava ilişkin söylemler de artmaya başladı ... her kafadan bir ses çıkıyor .</S>
      <S I="1">Öğrencilerin de , anne babaların da kafaları karmakarışık .</S>
    </P>
  - <P I="3">
      <S I="0">Evet , konuşan çok ama icraat yok .</S>
      <S I="1">Kısa bir süre içerisinde olacağa da benzemiyor .</S>
    </P>
  - <P I="4">
      <S I="0">Bu yıl için zaten bir değişiklik söz konusu değil .</S>
      <S I="1">En erken 2004 ÖSS için olabilir ki ; o da şimdilik zor gözüküyor .</S>
      <S I="2">Mevcut YÖK ve ÖSYM kadroları hepten değişmeden köklü bir değişim söz konusu
        bile olamaz .</S>
      <S I="3">Çünkü izin vermezler ...</S>
    </P>
  </F>
```

Figure 7.19 XML tagged output of text in Figure 7.17, named as “S_abbasguclu.xml”

If the “*Etiketlenmemiş (.txt)*” is chosen, the text is splitted into sentences and written in a text file (Figure 7.20).



Dosyayı Seçiniz: C:\Documents and Settings\ozlem\Desktop\Sample

Dosyanın İçeriği:

Üniversite sınavı kalkıyor mu?

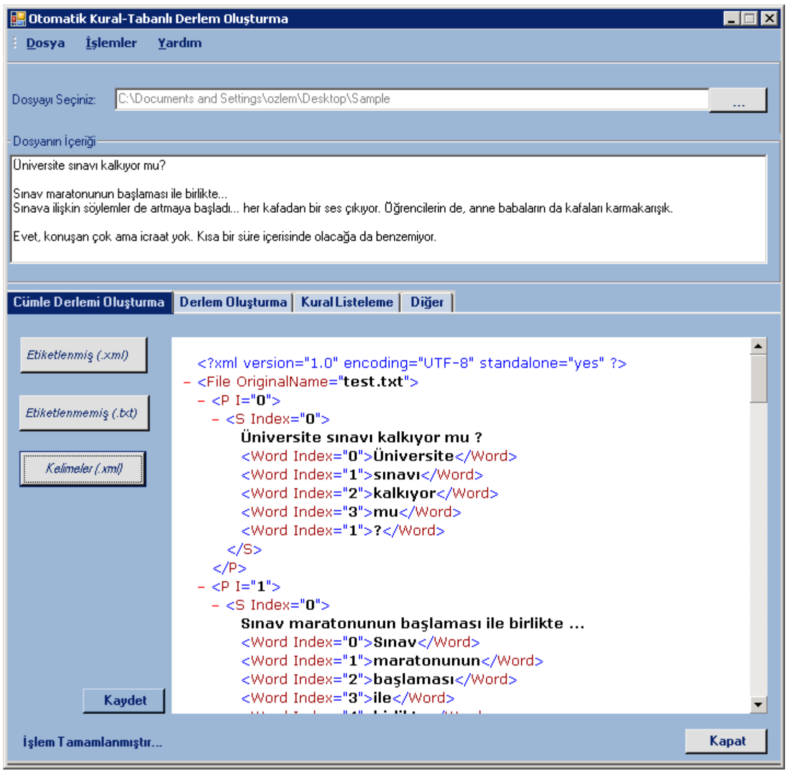
Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da benzemiyor.

```
1______  Üniversite sınavı kalkıyor mu
2______  Sınav maratonunun başlaması ile birlikte...
3______  Sınava ilişkin söylemler de artmaya başladı...
4______  her kafadan bir ses çıkıyor
5______  Öğrencilerin de, anne babaların da kafaları karmakarışık
6______  Evet, konuşan çok ama icraat yok.
7______  Kısa bir süre içerisinde olacağa da benzemiyor
8______  Bu yıl için zaten bir değişiklik söz konusu değil.
9______  En erken 2004 ÖSS için olabilir ki; o da şimdilik zor gözü
10______  Mevcut YÖK ve ÖSYM kadroları hepten değişmeden köklü bir
11______  Çünkü izin vermezler...
```

Figure 7.20 Not tagged output of text in Figure 7.17

If the "*Kelimeler (.xml)*" is chosen, the text is splitted into sentences, sentences splitted into wordforms and stored in XML formatted file (Figure 7.21).



Dosyayı Seçiniz: C:\Documents and Settings\ozlem\Desktop\Sample

Dosyanın İçeriği

Üniversite sınavı kalkıyor mu?

Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

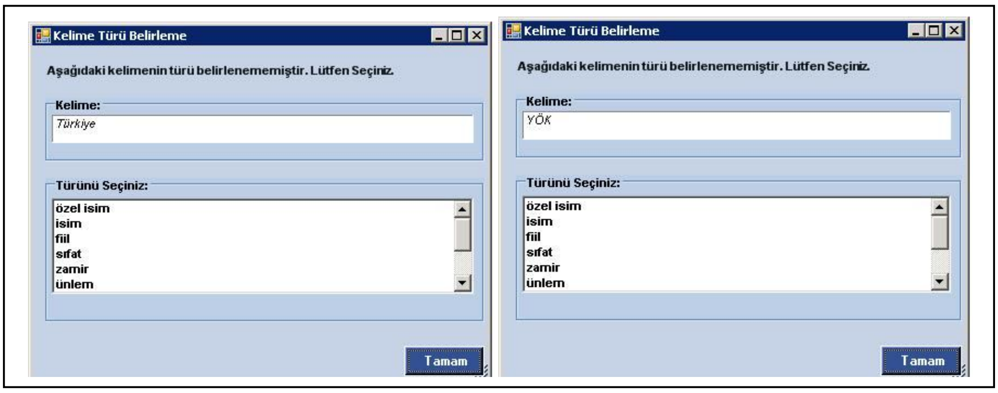
Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da benzemiyor.

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <File OriginalName="test.txt">
  - <P I="0">
    - <S Index="0">
        Üniversite sınavı kalkıyor mu ?
        <Word Index="0">Üniversite</Word>
        <Word Index="1">sınavı</Word>
        <Word Index="2">kalkıyor</Word>
        <Word Index="3">mu</Word>
        <Word Index="1">?</Word>
      </S>
    </P>
  - <P I="1">
    - <S Index="0">
        Sınav maratonunun başlaması ile birlikte ...
        <Word Index="0">Sınav</Word>
        <Word Index="1">maratonunun</Word>
        <Word Index="2">başlaması</Word>
        <Word Index="3">ile</Word>
```

İşlem Tamamlanmıştır...

Figure 7.21 XML tagged output of text with wordforms in Figure 7.17

The words, which start with capital letters, are firstly asked to the user if they are proper noun or not (Figure 7.22).



Aşağıdaki kelimenin türü belirlenememiştir. Lütfen Seçiniz.

Kelime: Türkiye

Türünü Seçiniz:
özel isim
isim
fiil
sıfat
zamir
ünlem

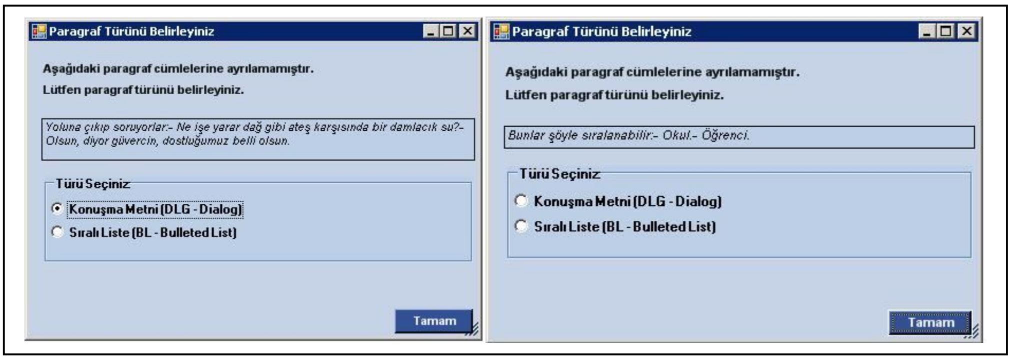
Aşağıdaki kelimenin türü belirlenememiştir. Lütfen Seçiniz.

Kelime: YÖK

Türünü Seçiniz:
özel isim
isim
fiil
sıfat
zamir
ünlem

Figure 7.22 Screenshot of the module asking to user the type of the word

Since bulleted sentences cause ambiguity, and they can not be separated from conversation texts since conversations are indicated by special character, “-” (hypen), after the character “:”, which is also used for bulleting. The sentences come after the “:” character assumed as belong to one sentence; all lines are read and combined together as one sentence, and asked to the user to determine the type of them (Figure 7.23).



Aşağıdaki paragraf cümlelerine ayrılamamıştır.
Lütfen paragraf türünü belirleyiniz.

Yoluna çıkıp soruyorlar:- Ne işe yarar dağ gibi ateş karşısında bir damlacık su?- Olsun, diyor güvercin, dostluğumuz belli olsun.

Türü Seçiniz
- ● Konuşma Metni (DLG - Dialog)
- ○ Sıralı Liste (BL - Bulleted List)

Aşağıdaki paragraf cümlelerine ayrılamamıştır.
Lütfen paragraf türünü belirleyiniz.

Bunlar şöyle sıralanabilir:- Okul.- Öğrenci.

Türü Seçiniz
- ○ Konuşma Metni (DLG - Dialog)
- ○ Sıralı Liste (BL - Bulleted List)

Figure 7.23 Sample of undetermined blocks asked to the user

User defines the type of the sentence block as “conversation sentences”, tagged as DLG (Dialog), or “bulleting text”, tagged as BL (Bulleted List).

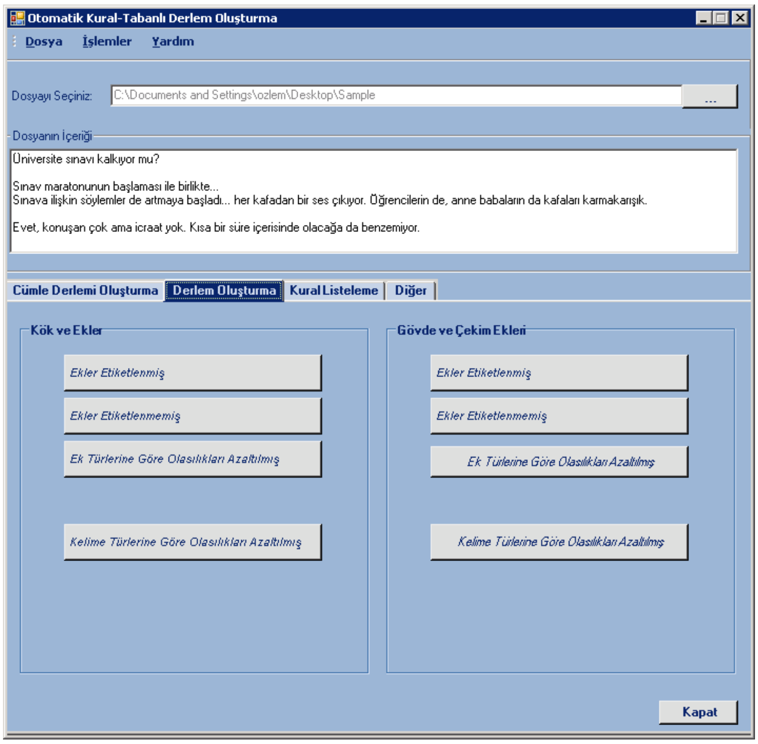
7.2.2 Corpus Generation

“Derlem Oluşturma” tab in the main screen is used for finding stem / root and suffixes of the wordforms (Figure 7.24).

There are two main parts: “Kök ve Ekler (Roots and Suffixes)” and “Gövde ve Çekim Ekleri (Stems and Inflexional Suffixes)”. The roots / stems and suffixes are found in four ways according to the needs;

- *Ekler Etiketlenmiş:* All possible of the roots / stems and suffixes, which are tagged and stored in XML formatted file, named as “AnalyzedFileName_MA_ST.xml”, in which *MA_ST* stands for *Morphological Analysis_Suffixes are Tagged.*

- *Ekler Etiketlenmemiş:* All possible of the roots / stems and suffixes, suffixes are not tagged in details, and results are stored in XML formatted file, named as “AnalyzedFileName_MA_SNT.xml” , in which *MA_SNT* stands for *Morphological Analysis_Suffixes are not Tagged.*
- *Ek Türlerine Göre Olasılıkları Azaltılmış:* All possible of the roots / stems and suffixes, possible suffixes combinations are eliminated according to the type of them, are are stored in XML formatted file, named as “AnalyzedFileName_MA_SE.xml”, in which *MA_SE* stands for *Morphological Analysis_Suffixes are Eliminated.*
- *Kelime Türlerine Göre Olasılıkları Azaltılmış:* All possible of the roots / stems and suffixes are stored in XML formatted file. Possible roots / suffixes are eliminated according to the word order rules by using the Rule-Based POS tagging module, named as “AnalyzedFileName_POS.xml”, in which *POS* stands for *POS Tagging.*


Otomatik Kural-Tabanlı Derlem Oluşturma
Dosya
İşlemler
Yardım
...
Dosyanın İçeriği
Cümle Derlemi Oluşturma
Derlem Oluşturma
Kural Listeleme
Diğer
Kök ve Ekler
Ekler Etiketlenmiş
Ekler Etiketlenmemiş
Ek Türlerine Göre Olasılıkları Azaltılmış
Kelime Türlerine Göre Olasılıkları Azaltılmış
Gövde ve Çekim Ekleri
Ekler Etiketlenmiş
Ekler Etiketlenmemiş
Ek Türlerine Göre Olasılıkları Azaltılmış
Kelime Türlerine Göre Olasılıkları Azaltılmış
Kapat

Dosyayı Seçiniz: C:\Documents and Settings\ozlem\Desktop\Sample

Üniversite sınavı kalkıyor mu?

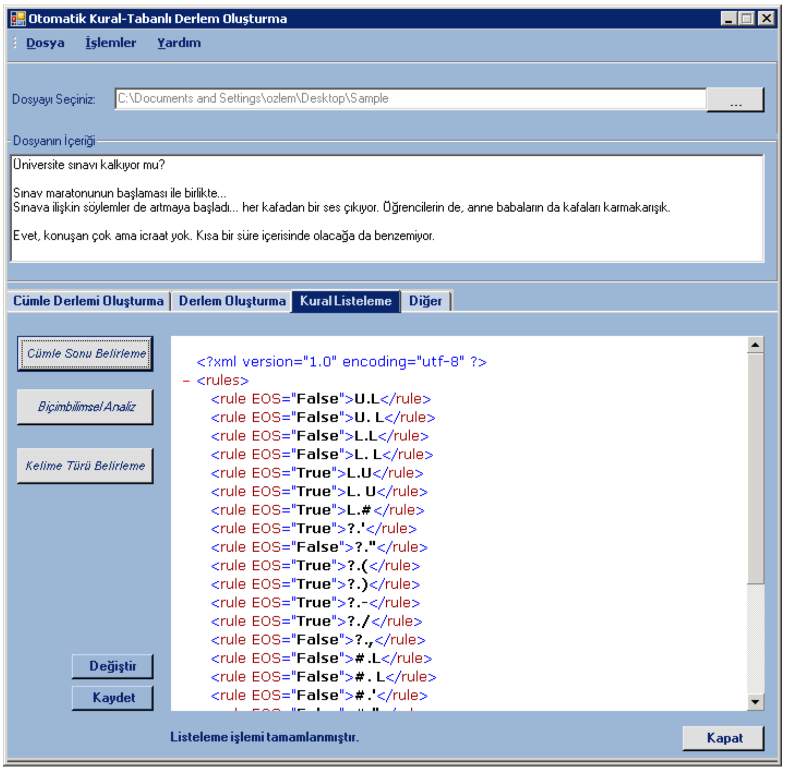
Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da benzemiyor.

Figure 7.24 “Derlem Oluşturma” tab in the application

7.2.3 Rule Lists

“Kural Listeleme” tab in the main screen is used for listing the rules, used in the application, edit and change them (Figure 7.25).



Dosyayı Seçiniz: C:\Documents and Settings\ozlem\Desktop\Sample

Üniversite sınavı kalkıyor mu?

Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da benzemiyor.

```
  <?xml version="1.0" encoding="utf-8" ?>
- <rules>
    <rule EOS="False">U.L</rule>
    <rule EOS="False">U. L</rule>
    <rule EOS="False">L.L</rule>
    <rule EOS="False">L. L</rule>
    <rule EOS="True">L.U</rule>
    <rule EOS="True">L. U</rule>
    <rule EOS="True">L.#</rule>
    <rule EOS="True">?.'</rule>
    <rule EOS="False">?."</rule>
    <rule EOS="True">?.(</rule>
    <rule EOS="True">?.)</rule>
    <rule EOS="True">?.-</rule>
    <rule EOS="True">?./</rule>
    <rule EOS="False">?.,</rule>
    <rule EOS="False">#.L</rule>
    <rule EOS="False"># . L</rule>
    <rule EOS="False"># .'</rule>
```

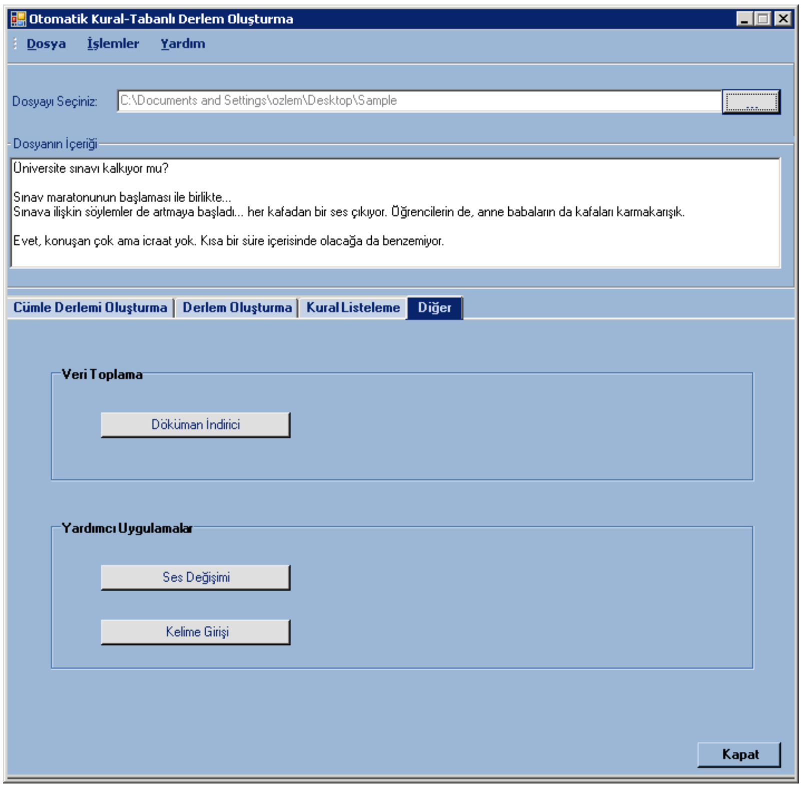
Listeleme işlemi tamamlanmıştır.

Figure 7.25 “Kural Listeleme” tab in the application

“Cümle Sonu Belirleme” button is used to show the rules in the “Rules.xml” file, which is used for the sentence boundary detection process, whereas the “Biçimbilimsel Analiz” button is used to show the rules used in morphological analysis and the “Kelime Türü Belirleme” button is used to show the rules in POS Tagging processes. All rules can be edited by using the “Değiştir” button.

7.2.4 Other Operations

“Diğer” tab in the main screen is used for running some applications that are used to help the main application, such as getting and storing data (“Döküman İndirici” button), entering words in lexicon (“Kelime Girişi” button) or changing the character of the words in lexicon according to vowel changing rule in Turkish ("Ses Değişimi" button) (Figure 7.26).



Dosyayı Seçiniz: C:\Documents and Settings\ozlem\Desktop\Sample

Üniversite sınavı kalkıyor mu?

Sınav maratonunun başlaması ile birlikte...
Sınava ilişkin söylemler de artmaya başladı... her kafadan bir ses çıkıyor. Öğrencilerin de, anne babaların da kafaları karmakarışık.

Evet, konuşan çok ama icraat yok. Kısa bir süre içerisinde olacağa da benzemiyor.

Figure 7.26 "Diğer" tab in the application.

"Döküman İndirici (Document Downloader)" is used to run the *Document Downloader* application, which details are told before.

CHAPTER EIGHT
CONCLUSION

8.1 Conclusion

In order to make analysis on a spoken language, a large scale corpus that includes varied sample of text documents is needed. Effective corpora have been generated and used for NLP applications on many languages, such as English, German, Czech, etc, but any large scale Turkish corpora that involve all properties of the language cannot be generated until now.

The main goal of this study is to develop an infrastructure with rule-based approach to generate large scale Turkish corpus, and to develop appropriate methods that find the sentences, root and suffixes of the Turkish words in an efficient way, while generating large scale corpus. Because of grammatical rule-based structure of Turkish, rule-based method was chosen to develop the infrastructure.

To generate a large scale corpus, at first documents must be collected. Variation of authors and types of documents; such as newspaper, book, magazine; increase the studies on it. Text documents which plays a critical role to generate corpus, must be collocated in a systematic way. Therefore, an application called Document Downloader, which was generated for collecting electronic data to develop large scale corpus, was used to generate a dataset. 195.256 articles, which include 93.228.892 wordforms, from 5 different Turkish newspapers "*Milliyet*", "*Hürriyet*", "*Radikal*", "*Vatan*", "*Akşam*" were downloaded , stored in a storage media and also metadata of these documents, such as URL of document, header of document, size of document, etc., were stored in database.

In order to test the *Rule-Based Automatic Corpus Generation (RB-CorGen)*, at first, the roots, stems and suffixes were collected. The root and stem lists were collected by co-operation with Turkish Linguistic Association (Türk Dil Kurumu, TDK). These lists were modified according to morphophonemic processes vowel and consonant harmonies in Turkish and stored. The suffix list was collected by linguists in Dokuz Eylul University, College of Social Science and Literature Linguistic Department. After that, the tags that would be used in the corpus were created by them. The grammatical rules for sentence boundary detection, suffixes and word ordering were also collected by linguists. All lists were stored in XML format to make the system more flexible and scalable.

Although the punctuation marks, such as ., …, !, and ?, are used to terminate sentences, they may also be used in anywhere else in a sentence, such as using ". (dot)" mark for an abbreviation, as a decimal point in a number, in an e-mail addresses etc, and cause ambiguities, which make harder the process of the determining sentence boundaries. In Turkish, there are some ambiguities in finding sentence boundaries like in any other languages. Ambiguities in sentence boundary detection process, which are caused by abbreviations, enumerations, web and e-mail addresses, were solved in Rule-Based Sentence Boundary Detection (RB-SBD) method by using abbreviation and rule lists.

RB-SBD method was tested on two different test sets generated from randomly selected columns of two Turkish newspapers *Milliyet* and *Yeni Asır*, which included typing faults and ambiguities, and the results were successive. The success rates were determined as 99.60% (99.76% without typing faults) and 99.66% (99.80% without typing faults) in these test sets. The average success rate of the algorithm was 99.78% when typing faults were discarded. Also, 10-fold cross validation check was applied to this method by choosing test sets 10 times randomly from the collected data by the Document Downloader program. As a result, average values of ACC and AUC were calculated as 99.66% and 94.84%, which are the highest values for sentence boundary detection process in Turkish.

The available roots and stems were used in the morphological analysis part of the project. In this process, the output of the sentence boundary detection process was taken as input and all wordforms were analysed. The possible root/stem and suffix combinations were determined and in an XML formatted file as an output.

After finding all possible roots and stems with their word types, the Rule-Based Part-of-Speech Tagging (RB-POST) method was applied for word-category disambigution. Instead of a built-in lexicon, the output of the morphological analysis process and contains possible roots / stems and suffix combinations for each word in analyzed document is used as an input in RB-POST, so there is not any time consuming for looking up a word in a lexicon or database. Besides, the method is disk-imperceptible and also independent from lexicon and database.

This Rule-Based POS tagger was tested on 140 randomly chosen Turkish sentences which were taken from articles of "Milliyet" newspaper in test set. Each sentence has at least one word with more than one possible word type. The success rate was determined as 92%.

The success rates of the new developed methods, RB-SBD, RB-WD and RB-POST, could not be checked against with any of the works carried out by different researchers, since data sets or test sets used in the success rate determination of these works could not be obtained from the developers of the methods. Therefore, the new developed methods were carried out on the randomly chosen data collected by the Document Downloader program.

Since the language structure is commonly the same with other agglutinative languages, such as Turkic languages including Turkmen, Azerbaijani (Azeri), Kazakh, Kyrgyz and Uzbek, the portable infrastructure, RB-CorGen, may be easily adapted and used for them to apply the methods and generate a corpus by only giving the rules, abbreviation lists, root and suffixes lists.

8.2 Future Works

All processes in rule-based corpus generation need well-defined and organized rules to use and give successive results. It was seen that the accuracy of the RB-CorGen increased with the increasing number of rules. In the future, new rules may be added ino the system, the number of rules, and also the accuracy of the system may be increased continuously.

Besides, the process time of the RB-CorGen may be decreased by making refinements and improvements on the developed methods for corpus generation.

All of the ambiguities in corpus generation processes cannot be solved in RB-CorGen. New techniques may also be developed in phrase structure grammar and word sense disambiguation to solve some ambiguities, and integrated into RB-CorGen easily. Consequently, the number of possible root/stem and suffixes combinations, and also the size of the output on disk may be decreased.

REFERENCES

Aberdeen, J., Burger, J., Day, D., Hirschman, L., Robinson, P. & Vilain M. (1995). Mitre: Description of the alembic system used for muc-6. *Proceedings of the Sixth Message Under-standing Conference (MUC-6)*, Columbia, Maryland.

Alkım, E., Aktaş, Ö., & Çebi, Y. (2009). *Türk Dilleri Arası Çeviri Altyapısı.* Uluslararası Dünya Dili Türkçe Sempozyumu. Lefkoşa, K.K.T.C.

Alpkoçak, A., Kut, A., & Özkarahan, E. (1995). Bilgi bulma sistemleri için otomatik türkçe dizinleme yöntemi. *Bilişim Bildirileri*, Dokuz Eylül University, İzmir, Turkey.

Altinyurt, L., Orhan, Z., & Güngör, T. (2006). A composite approach for part of speech tagging in Turkish. *Proceedings of Third International Bulgarian-Turkish Conference on Computer Science*. 19-24. Istanbul.

Antworth, E. L. (1990). PC-KIMMO: A two-level processor for morphological analysis. *Occasional Publications in Academic Computing, Summer Institute of Linguistics*. 16. Dallas, Texas.

Asliyan, R., Günel, K., & Yakhno, T. (2007). Detecting misspelled words in Turkish text using syllable n-gram frequencies. Pattern Recognition and Machine Intelligence PReMI, A. Ghosh, R.K. De, and S.K. Pal (Ed.), LNCS 4815, 553–559. Springer-Verlag Berlin Heidelberg.

Birant, Çağdaş Can (2008). *Root-Suffix seperation of Turkish words.* Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi. İzmir, Turkey.

Black, A.W. & Taylor, P. (1994). CHATR: A generic speech synthesis system, *Proceedings of International Conference on Computational Linguistics (COLING '94)*, 2, 983-986. Kyoto, Japan.

Brill E. (1992). A simple rule-based part of speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, 152-155. Trento, Italy.

Brill E. (1994). Some advances in transformation based part of speech tagging. *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. Cambridge, MA, USA: MIT Press.

Brown Corpus, (n.d.). Retrieved March 3, 2010, from http://www.helsinki.fi/varieng/CoRD/corpora/BROWN/index.html

Burnard, L. (2000). Reference guide for the British National Corpus (World Edition). Retrieved March 3, 2010, from, www.natcorp.ox.ac.uk/archive/worldURG/urg.pdf.

Cebiroğlu, G., &Adalı, E. (2002). Sözlüksüz köke ulaşma yöntemi. *Proceedings of 19th TBD Bilişim Kurultayi*, 155-160, Istanbul.

Charniak, E. (1993). Statistical language learning. *Computational Linguistics*, *21*(1), 103-111. Cambridge, MA, USA: MIT Press.

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

Church, K., & Gale, W. (1991). Probability scoring for spelling correction. *Statistics and Computing, 1* (2), 93-103.

CORD The Brown Corpus. (n.d.) a. Retrieved March 3, 2010, from http://www.helsinki.fi/varieng/CoRD/corpora/ BROWN/basic.html

CORD The Brown Corpus. (n.d.) b. Retrieved March 3, 2010, from http://www.helsinki.fi/varieng/CoRD/corpora/BROWN/tags.html.

COSMAS, German Corpus. (n.d.). Retrieved March 3, 2010, from http://www.intute.ac.uk/cgi-bin/fullrecord.pl?handle=humbul3534.

Croatian National Corpus: Home Page. (n.d.). Retrieved March 3, 2010, from http://www.hnk.ffzg.hr/cnc.htm.

Crystal,D. (1991). *A dictionary of linguistics and phonetics* (3rd Edition). Oxford: Blackwell Publishing.

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

Czech National Corpus, (n.d.). Retrieved March 3, 2010, from http://ucnk.ff.cuni.cz/english/index.php.

Çebi, Y. & Dalkılıç, G. (2004). *Turkish word n-gram analyzing algorithms for a large scale Turkish corpus - TurCo*, IEEE International Conference on Information Technology ITCC 2004, 2, 236-240.

Çebi, Y., Aktaş, Ö. & Birant, Ç. C. (2006). *Türkçe Derlem Olusturmada Otomasyon ve Karşılaşılan Zorluklar.* VI. Türk Dünyasi Ekonomi, Dil ve Bilisim Is Birligi Forumu, TDK-TBD. Biskek, Kirgizistan.

Çiçekli, İ. & Korkmaz, T. (1998). Generation of Simple Turkish Sentences with Systemic-Functional Grammar, *Proceedings of the 3rd International Conference on New Methods in Language Processing (NeMLaP-3)*, Sydney, Australia, January 1998, 165-174.

Dalkılıç, G. (2001). *Some statistical properties of contemporary printed Turkish and a text compression application.* MSc Thesis. International Computing Institute, Ege University. Izmir, Turkey.

Dalkılıç, M.E., & Dalkılıç, G. (2001). Some measurable language characteristics of printed Turkish. *Proceedings of the XVI. International Symposium on Computer and Information Sciences*, 217-224. Antalya, Turkey.

Dalkilic, G. & Cebi, Y. (2002). *A 300 MB Turkish corpus and word analysis.* LNCS 2002, 2457/2002, 205-212. Springer Berlin / Heidelberg.

DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics, 14* (1), 31-39. Cambridge, MA, USA: MIT Press.

Dinçer, B. T., Karaoğlan, B. (2004). Sentence boundary detection in Turkish. *Proceedings of Advances in Information Systems - ADVIS 2004*, LNCS 3261, 255–262, Springer-Verlag Berlin Heidelberg.

Diri, B. (2000). A text compression system based on the morphology of Turkish Language. *Proceedings of the XV International Symposium on Computer and Information Sciences*, 12-23. Istanbul, Turkey.

Diri, B., & Amasyalı, M. F. (2003). *Automatic author detection for Turkish text.* 13th International Conference on Artificial Neural Network and 10th International Conference on Neural Information Processing.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics, 19* (1): 61–74, Cambridge, MA, USA: MIT Press.

Encoding the British National Corpus. (n.d.). Retrieved March 3, 2010, from, http://xml.coverpages.org/bnc-encoding2.html.

Garside R., Leech G. & Sampson G. (Ed.) (1987). *The computational analysis of English: A corpus-based approach.* U.S.A.: Longman Group .

Garside, R., Leech, G. & McEnery, A. (1997). *Corpus annotation: Linguistic information from computer text corpora.* New York: Addison Wesley Longman Inc.

Greene B. B., & Rubin G. M. (1971). *Automatic Grammatical Tagging of English.* Providence, Rhode Island: Brown University Press, Department of Linguistics.

Greenwood, A.R. (1997). Articulatory speech synthesis using diphone units. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing,* 1635-1638. Munich.

Guilder, L. V. (1995). *Automated part of speech tagging: A brief overview.* Retrieved January 25, 2010, from http://ccl.pku.edu.cn/doubtfire/NLP/Lexical_Analysis/Word_Segmentation_Tagging/POS_Tagging_Overview/POS Tagging Overview.htm.

Gündü, H. (2008). *Determination of author characteristics.* B.Sc. Thesis. Dokuz Eylul University, Department of Computer Engineering, İzmir, Turkey.

Güngör, T. (1995). *Computer processing of Turkish: Morphological and lexical investigation.* PhD Thesis. Computer Engineering Department, Boğaziçi University. Istanbul, Turkey.

Güngördü Z. (1993). *A lexical-functional grammar for Turkish.* MSc Thesis. Computer Engineering Department, Bilkent University, Ankara, Turkey.

Hakkani-Tür, D.Z., Oflazer, K., & Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, *36*, 381–410. Netherlands: SpringerLink.

Hallaç, Ü. (2007). *Determination of Turkish word types.* M.Sc. Thesis. Dokuz Eylül University, Graduate School of Natural and Applied Science, Department of Computer Engineering. İzmir, Turkey.

Hennecke, M., Moore, R. & Swan, H. (1997). *Natural language generation*. Retrieved April 4, 2010, from http://www.dfki.de/fluids/Natural_Language_Generation.html.

Huang, X., Acero, A. & Hon, H.W. (2001). *Spoken language processing*. New Jersey: Prentice Hall.

Ide, N. & Suderman, K. (2003). *The American national corpus.* Retrieved March 3, 2010, from http://www.cs.vassar.edu/~ide/papers/anc-lrec04.pdf.

Järvinen, T. (1994). Annotating 200 million words: the Bank of English project. *Proceedings of the 15th conference on Computational linguistics, 1,* 565 - 568. Kyoto, Japan.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, speech recognition, and computational linguistics*. New Jersey: Prentice-Hall.

Karlsson, F., Voutilainen, A., Heikkilä, J., & Anttila, A. (Ed.). (1995). *Constraint grammar: A language-independent system for parsing unrestricted text.* Berlin: Mouton de Gruyter.

Kiss T., & Strunk, J. (2006). Unsupervised multilingual sentence Boundary detection. *Computational Linguistics, 32* (4), 485-525, Cambridge, MA, USA: MIT Press.

Kızılay, F. (2009). *An Infrastructure model for collecting electronic data to develop large scale corpus.* M.Sc. Thesis. Dokuz Eylül University, Graduate School of Natural and Applied Science, Department of Computer Engineering. İzmir, Turkey.

Koltuksuz, A. H. (1995). *Cryptanalitic measures of Turkish for symmetrical cryptosystems.* PhD Thesis, Ege University Department of Computer Engineering, Izmir, Turkey.

Köksal, A. (1975). *Automatic morphological analysis of Turkish.* Ph.D. Thesis, Hacettepe University, Ankara, Turkey.

Kucera, K. (2002). Czech National Corpus: Principles, design, and results. *Literary and Linguistic Computing, 17* (2), 245-257. Oxford: Oxford University Press.

Kukich, K. (1992). Technique for automatically correcting words in text. *ACM Computing Surveys (CSUR), 24* (4), 377-439. NY USA: ACM Press.

Leech, G., Garside, R., & Bryant, M. (1994). CLAWS4: The tagging of the British National Corpus. *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*, 622-628. Kyoto, Japan.

Leech, G., Hundt, M., Mair, C. & Smith, N. (2009). The composition of the Brown Corpus. *In Change in Contemporary English, A Grammatical Study* (273-275). Cambridge: Cambridge University Press.

Lindebjerg, A. (September, 1997). *Brown Corpus manual.* Retrieved March 3, 2010, from icame.uib.no/brown/bcm.html.

Mani, I. (2001). *Automatic summarization.* Amsterdam, The Netherlands: John Benjamins Publishing Company.

Manning, C. D. & Schutze, H. (1999). *Foundations of statistical natural language processing.* Cambridge: MIT Press.

METU Turkish Corpus Project. (n.d.). Retrieved March 3, 2010, from http://ii.metu.edu.tr/tr/research_group/metu-turkish-corpus-project.

Mikheev, A. (2000). Tagging sentence boundaries. *Proceedings of the 1st North American Chapter of the Association for computational linguistics conference*, 264 – 271, New Mexico State.

Modern French Corpus. (n.d.). Retrieved March 3, 2010, from http://catalog.elra.info/product_info.php?products_id=634.

Nadas, A. (1984). Estimation of probabilities in the language model of the IBM speech recognition system. *Proceedings of IEEE Transactions on Acoustics, Speech, and Signal Processing, 32* (4), 859-861.

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

Oflazer K. (2003). Extended finite state approach. *Computational Linguistics, 29* (4), 515-544. Cambridge: MIT Press.

Palmer, D. D., & Hearst, M. A. (1997). Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics, 23* (2), 241-267. Cambridge: MIT Press.

Parker, R., Graff, D., Kong, J., Chen, K., & Maeda, K. (2009). *English Gigaword fourth edition.* Retrieved March 3, 2010, from http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2009T13.

PAROLE CORPUS-Information. (n.d.). Retrieved March 3, 2010, from http://parole.inl.nl/html-eng/main_info.html.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems, 14* (3), 130−137. Bingley, England: Emerald Group Pub. Ltd.

Ray, Erik T. (2003). *Learning XML* (Second Edition). United States of America: O'Reilly Media, Inc.

Reynar, J. C., & Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing (ANLP'97)*, Washington, D.C.

Riley, M.D. (1989). Some applications of tree-based modeling to speech and language indexing. *Proceedings of the DARPA Speech and Natural Language Workshop*, 339-352, Morristown, NJ, USA.

Sagisaka, Y., Iwahashi, N. & Mimura, K. (1992). ATR v-TALK Speech Synthesis System, *Proceedings of the International Conference on Spoken Language Processing (ICSLP),* (1), 483-486. Canada.

Sak, H., Güngör, T. & Safkan, Y. (2006) A corpus-based concatenative speech synthesis system for Turkish. *Turkish Journal of Electrical Engineering and Computer Sciences, 14* (2), 209-223. TUBITAK, Ankara, Turkey.

Say, B., Özge, U., & Oflazer, K. (2002). *Bilgisayar ortamında bir derlem geliştirme çalışması.* Akademik Bilisim Konferansi (AB'02). Selcuk Üniversitesi, Konya, Turkey.

Say, B., Zeyrek, D., Oflazer, K. & Ozge, U. (2002). Development of a corpus and a treebank for present-day written Turkish. *Proceedings of the Eleventh International Conference of Turkish Linguistics (ICTL)*, 183-192. Eastern MediterraneanUniversity, Northern Cyprus.

Say, C., Demir, Ş., Çetinoğlu, Ö., & Öğün, F. (2004). A natural language processing infrastructure for Turkish. *Proceedings of the 20th international conference on Computational Linguistics,* 1385. Retrieved January, 2009, from ACM Digital Library Database.

Sever, H., & Bitirim, Y. (2003). FindStem: Analysis and evaluation of a Turkish stemming algorithm. In M. A. Nascimento, E. S. De Moura, A. L. Oliveira (Ed.). *Proceedings of the 10th String Processing and Information Retrieval*, LNCS, 2857, 238-251. Springer-Verlag, Heidelberg.

Shannon, C.E. (1951). Prediction and entropy of printed English. *The Bell System Technical Journal, 30* (1),50-64.

Sinclair, J. (1991). *Corpus Concordance, Collocation (Describing English Language).* Oxford: Oxford University Press.

Solak, A., & Oflazer, K. (1993). Design and implementation of a spelling checker for Turkish. *Literary and Linguistic Computing, 8* (3), 113-130. Oxford: Oxford University Press.

Solak A., & Can, F. (1994). Effects of stemming on Turkish text retrieval. *Technical report BUCEIS-94-20*, Bilkent University, Ankara, Turkey.

Sperberg-McQueen, C. M. & Burnard, L. (Ed.). (1994). *Guidelines for electronic Text Encoding and Interchange. (TEI P3)*, Oxford, Text Encoding Initiative.

Tantuğ, A. C., Adalı, E., & Oflazer, K. (2006). *A Prototype machine translation system between Turkmen and Turkish.* Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2006). Muğla, Turkey.

Tantuğ, A. C., Adalı, E., & Oflazer, K. (2007). Machine translation between Turkic Languages. *Proceedings of the ACL 2007 Association for Computational Linguistics,* 189–192, Prague.

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

Temizsoy, M., & Çiçekli, İ. (1998). An ontology based approach to parsing Turkish Sentences. *Proceedings of Third Conference of the Association for Machine Translation in the Americas AMTA'98.*, LNCS 1529, 124 – 135, Langhorne, PA, USA: Springer Berlin / Heidelberg.

The Bank of English User Guide. (n.d.). Retrieved March 3, 2010, from http://www.titania.bham.ac.uk/docs/svenguide.html

The British National Corpus: facts and figures. (n.d.). Retrieved March 3, 2010, from http://www.oup.com/elt/catalogue/teachersites/oald7/more_on_dicts/bnc?cc=global.

Voutilainen, A. (1995a). Morphological disambiguation. In Karlsson, F., Voutilainen, A., Heikkilä, J., and Anttila, A. (Ed.), *Constraint Grammar: A language-independent system for parsing unrestricted text,* 165-284. Berlin: Mouton de Gruyter.

Voutilainen, A. (1995b). A syntax-based part of speech analyser. *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics.* 157-164. Dublin.

Wang, H., & Huang, Y. (2003). *Bondec – A Sentence Boundary Detector*, http://nlp.stan-ford.edu/courses/cs224n/2003/fp/huangy/final_project.doc.

Weiss, D. (2005). *Stempelator: A hybrid stemmer for the Polish Language.* Institute of Computing Science, Poznań University of Technology, Poland, Research Report RA-002/05.

What is optical character recognition?. (n.d.). Retrieved March 1, 2010, from http://www.webopedia.com/TERM/O/optical_character_recognition.html.

Yona, S. (2001). *Lingua::EN::Sentence package.* http://cpansearch.perl.org/src/SHLOMOY/Lingua-EN-Sentence-0.25/lib/Lingua/ EN/Sentence.pm.

Zue, V., Cole, R. & Ward W. (1995). *Speech recognition.* Retrieved March 1, 2010, from http://cslu.cse.ogi.edu/HLTsurvey/ch1node4.html.

APPENDICES

APPENDIX A Turkish Grammatical Rules

APPENDIX A.1 Properties of Turkish

Turkish is an agglutinative language like Finnish, Hungarian. It belongs to the southwestern group of Turkic family. Turkic languages are in the Uralic-Altaic language family. In agglutinative languages, words formed by combined root words and morphemes. Word structures can grow by addition of morphemes. Morphemes added to a stem can convert the word from nominal to a verbal structure or viceversa.

Turkish has a very productive morphology. There is a root and several suffixes are combined to this root. It is possible to produce a very high number of words from the same root with suffixes. The lexicon size may grow to unmanageable size.

A popular example of a Turkish word formation is:
OSMANLILAŞTIRAMAYABİLECEKLERİMİZDENMİŞSİNİZCESİNE

This can be broken down into morphemes:
OSMAN+LI+LAŞ+TIR+A+MA+(Y)ABİL+ECEK+LER+İMİZ+DEN+MİŞ+SİNİZ +CESİNE

In this example, one word in Turkish corresponds to a full sentence in English. This example can be translated into English as “as if you were of those whom we might consider not converting into an Ottoman”.

There are 29 letters in Turkish language. The eight of them are vowels and twenty-one of them are consonants. (See Appendix A.5)

The number of vowels is more than many languages. Vowels of Turkish can be classified in three groups according to their properties:

- Front and back,
- Round and unrounded,
- High or low

The vowels can be partitioned as below in detail:

- Back vowels: {a, ı, o, u}
- Front vowels: {e, i, ö, ü}
- Front unrounded vowels: {e, i}
- Front rounded vowels: {ö, ü}
- Back unrounded vowels: {a, ı}
- Back rounded vowels: {o, u}
- High vowels: {ı, i, u, ü}
- Low unrounded vowels: {a, e}

Turkish word formation uses a number of phonetic harmony rules. When a suffix is appended to a stem vowels and consonants change in certain ways.

APPENDIX A.2 Vowel Harmony

Vowel harmony is the best-known morphophonemic process in Turkish. It is most interesting and distinctive feature. Vowel harmony is a left-to-right process. It operates sequentially from syllable to syllable. Vowel harmony processes force certain vowels in suffixes agree with the last vowel in the stems or roots they are being affixed to. When vowels are affixed to a stem, they change according to the vowel harmony rules. The first vowel in the suffix changes according to the last vowel of the stem. Vowel harmony consists of two assimilations: Palatal and Labial Assimilations.

1. *Palatal assimilation*

This is called “major vowel harmony” . This vowel harmony is common to almost Turkic languages. This assimilation is about front/back feature of the language. Back vowels are the set of {a, ı, o, u} and the front vowels are the set of {e, i, ö, ü}.

If the vowels of the following morphemes are back then the vowel of the first morpheme in a word is back, e.g. askı + lar

“lar” is a plural suffix. “ler”, other form of plural suffix, is not used, because the vowels of the stem are back vowels.

If the vowels of the following morphemes are front then the vowel of the first morpheme in a word is front, e.g. ev + ler

Long vowels are “â, û, ô”. These vowels are in words of French origin in general. Examples:

saât+ler (saatler)
gôl+ler (goller)
usûl+ler (usuller)

2. *Labial assimilation*

This is called “minor vowel harmony”. This assimilation is about rounded/unrounded feature of the language. Examples:

çöl + ün
usul + ün (usûl + ün)
topal + ın
defter + im
saat + im (saât + im)

APPENDIX A.3 Consonant Harmony

Consonant harmony is another basic aspect of Turkish phonology. Consonants of Turkish phonology can be classified into two main groups. These are voiceless and voiced. Voiceless consonants are {“ç”, ”f’, ”h”, ”k”, ”p”, ”s”, ”ş”, ”t”}. Voiced consonants are {“b”, ”c”, ”d”, ”g”, ”ğ”, ”j”, ”l”, ”m”, ”n”, ”r”, ”v”, ”y”, ”z”}. Consonant harmony rules doesn’t formulate easily because of irregular character of borrowed and native words. There are some consonant harmony rules in Turkish:

- If the end of the word is one the voiceless consonants (“p”, ”ç”, ”t”, ”k”) then it changes to a corresponding voiced consonants (“b”, ”c”, ”d”, ”ğ”).
    - “p” changes to “b” (kitab + ım).
    - “d” changes to “t” (ta(d)t + tık), but not every “d” changes, such as “önad”, “soyad”, etc.
    - “k” changes to “ğ” (aya(k)ğ + ın).
    - “ç” changes to “c” (ağa(ç)c + ın), but not every “ç” changes, such as “göç”, “aç” ”iç”, etc.

- If a suffix starts with “d”, and if the last consonant of the stem is one of {“ç”, ”f’, ”h”, ”k”, ”p”, ”s”, ”ş”, ”t”}, “d” is replaced with “t” , e.g. yulaf+tan (yulaf + dan)

- If the last consonant of the stem is one of {“ç”, ”f’, “h”, “k”, “p”, “s”, “ş”} and if the suffix begins with the “c” then “c” is resolved as a “ç” , e.g. yaş+ça (yaş +ca)

- If “k” is at the end of the stem and “k” preceded by an “n” then “k” becomes “g” , e.g. çelen(k)g + e

There are some exceptions for this rule, e.g. “bank”.

- If the final character of the stem is “g” and a vowel is beginning of the suffix then “g” becomes “ğ” in foreign origin words, e.g. analo(g)ğ + a

  There are some exceptions for this rule, also, e.g.“lig”, “pedagog”, etc.

  If the final character of the stem is “g” and a consonant is beginning of the suffix then “g” does not become “ğ” , e.g. bumerang + tan

- If the final character of the stem is a vowel, and a vowel is beginning of the suffix then “y” inserted to stem, e.g. akarsu +y +unuz

- When certain suffixes are affixed last consonant is duplicated in Arabic or Persian origin words, e.g. zam + m + ı

- If Arabic origin words ending with a vowel then drops in exception to the general rule, e.g. camii – camisi

There are many numbers of words that have this property, e.g. “mevki”, “cami”, “terfi”, “zayi”, “ikna”, “merci”, etc.

APPENDIX A.4 Root Deformations

Turkish roots are not flexible in normally. There are some cases about various deformations. There are some exception cases:

- Root is observed in personal pronouns

  Examples: ben – bana
  sen – sana

- Wide vowel at the end of the stem is narrowed when the suffix “yor” comes after the verbs ending with the “a,e” , e.g. kapiyor (kapa – i – yor)

- When a suffix is beginning with a vowel comes after some nouns, which has a vowel {ı, i} in its last syllable, this vowel drops. This occurs generally designating parts of the human body, e.g. ağzımız (ağız – ı – mız)

- When the possessive suffix "ıl, il" is affixed to some verbs, and the last vowel of the verb is vowel "ı, i" then this vowel drops, e.g. ayrıl (ayır – ıl)

- If a plural suffix is affixed to a compound words then this suffix coming before the possessive suffix at the end of the stem.
  Example: gözyaşı + lar -> gözyaşları (not gözyaşılar)

APPENDIX A.5 Turkish Alphabet

Lowercase Letters

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	c	ç	d	e	f	g	ğ	h	ı	i	j	k	l	m	n	o

19	20	21	22	23	24	25	26	27	28	29
ö	p	r	s	ş	t	u	ü	v	y	z

Consonants: {b, c, ç, d, f, g, ğ, h, j, k, l, m, n, p, r, s, ş, t, v, y, z}
Vowels: {a, e, ı, i, o, ö, u, ü}

Uppercase Letters

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O

19	20	21	22	23	24	25	26	27	28	29
Ö	P	R	S	Ş	T	U	Ü	V	Y	Z

Consonants: {B, C, Ç, D, F, G, Ğ, H, J, K, L, M, N, P, R, S, Ş, T, V, Y, Z}
Vowels: {A, E, I, İ, O, Ö, U, Ü}

APPENDIX B Rules

APPENDIX B.1 Sentence Boundary Detection Rules

```
<?xml version="1.0" encoding="utf-8" ?>
<rules>
      <rule EOS="False">U.L</rule>
      <rule EOS="False">U. L</rule>
      <rule EOS="False">L.L</rule>
      <rule EOS="False">L. L</rule>
      <rule EOS="True">L.U</rule>
      <rule EOS="True">L. U</rule>
      <rule EOS="True">L.#</rule>
      <rule EOS="True">?.'</rule>
      <rule EOS="False">?."</rule>
      <rule EOS="True">?.(</rule>
      <rule EOS="True">?.)</rule>
      <rule EOS="True">?.-</rule>
      <rule EOS="True">?./</rule>
      <rule EOS="False">?.,</rule>
      <rule EOS="False">#.L</rule>
      <rule EOS="False">#. L</rule>
      <rule EOS="False">#.'</rule>
      <rule EOS="False">#."</rule>
      <rule EOS="False">#.(</rule>
      <rule EOS="False">#.)</rule>
      <rule EOS="False">#.-</rule>
      <rule EOS="False">#.,</rule>
      <rule EOS="False">#.#</rule>
      <rule EOS="False">#.U</rule>
      <rule EOS="False">#. U</rule>
</rules>
```

APPENDIX B.2 Stem / Root Parsing Rules

```
TBAE,TBEE\,DuEC,DuEKip,DuEOlz,DuEZ\DuEG\DuEKip\,K,DuEK,
TBSE,TBEE\,DuEC,DuEKip,DuEOlz,DuEZ\DuEG\DuEKip\,K,DuEK,
TBEE\,DuEC,DuEKip,DuEOlz,DuEZ\DuEG\DuEKip\,K,DuEK,
TBAE,TBEE\,DuEC,DuEKip,DuEOlz,Ytu,K,DuEK,
TBSE,TBEE\,DuEC,DuEKip,DuEOlz,Ytu,K,DuEK,
TBEE\,DuEC,DuEKip,DuEOlz,Ytu,K,DuEK,
TBEA,TBAA\,YS,DuASay,YS,DuAUy,YS,DuADur,
TBSA,TBAA\,YS,DuASay,YS,DuAUy,YS,DuADur,
TBAA\,YS,DuASay,YS,DuAUy,YS,DuADur,
TBAE,TBEE\,DuEC,DuEOlz,Ytu,YS,DuASay,YS,DuAUy,YS,DuADur,
TBSE,TBEE\,DuEC,DuEOlz,Ytu,YS,DuASay,YS,DuAUy,YS,DuADur,
TBEE\,DuEC,DuEOlz,Ytu,YS,DuASay,YS,DuAUy,YS,DuADur,

TB,E,A,S
E,DuEC,DuEKipYet,DuEOlz,YS,DuEZ\DuEG\DuEKip\,YS,K,DuEK,
E,DuEC,DuEKipYet,DuEOlz,YS,Ytu,YS,K,DuEK,
A,YS,DuASay,YS,DuAUy,YS,DuADur,
E,DuEC,DuEOlz,Ytu,YS,DuASay,YS,DuAUy,YS,DuADur,
```

APPENDIX B.3 POS Tagging Rules

```
<?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="v" />
  </Rule>
  <Rule RuleId="6" RuleType="sözdizim" RuleState="true">
    <Item ItemType="n" />
  </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="v" />
    <Item ItemType="isim" />
  </Rule>
  <Rule RuleId="11" RuleType="sözdizim" RuleState="true">
    <Item ItemType="fiil" />
    <Item ItemType="n" />
  </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>
  <Rule RuleId="15" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
  </Rule>
```

APPENDIX B.3 (Cont’d.)

```
<Rule RuleId="16" RuleType="sözdizim" RuleState="false">
  <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>
```

APPENDIX B.3 (Cont'd.)

```
  <Rule RuleId="28" RuleType="sözdizim" RuleState="true">
    <Item ItemType="zarf" />
    <Item ItemType="zarf" />
    <Item ItemType="fiil" />
  </Rule>
  <Rule RuleId="29" RuleType="sözdizim" RuleState="false">
    <Item ItemType="sıfat" />
    <Item ItemType="v" />
    <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="n" />
  </Rule>
  <Rule RuleId="32" RuleType="sözdizim" RuleState="true">
    <Item ItemType="sıfat" />
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
    <Item ItemType="n" />
  </Rule>
  <Rule RuleId="33" RuleType="sözdizim" RuleState="false">
    <Item ItemType="sıfat" />
    <Item ItemType="n2" />
    <Item ItemType="isim" />
  </Rule>
  <Rule RuleId="34" RuleType="sözdizim" RuleState="false">
    <Item ItemType="sıfat" />
    <Item ItemType="nv" />
    <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="n" />
  </Rule>
  <Rule RuleId="37" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
    <Item ItemType="n" />
  </Rule>
  <Rule RuleId="38" RuleType="sözdizim" RuleState="true">
    <Item ItemType="isim" />
    <Item ItemType="zarf" />
    <Item ItemType="isim" />
    <Item ItemType="fiil" />
  </Rule>
```

APPENDIX B.3 (Cont'd.)

```
<Rule RuleId="39" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
  <Item ItemType="n" />
</Rule>
<Rule RuleId="40" RuleType="sözdizim" RuleState="true">
  <Item ItemType="edat" />
</Rule>
<Rule RuleId="41" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <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="v" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="46" RuleType="sözdizim" RuleState="false">
  <Item ItemType="edat" />
  <Item ItemType="v" />
  <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="n" />
</Rule>
<Rule RuleId="48" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
  <Item ItemType="n" />
</Rule>
```

APPENDIX B.3 (Cont’d.)

```
<Rule RuleId="49" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
  <Item ItemType="n" />
</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="n" />
</Rule>
<Rule RuleId="51" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="isim" />
  <Item ItemType="edat" />
  <Item ItemType="fiil" />
  <Item ItemType="n" />
</Rule>
<Rule RuleId="52" RuleType="sözdizim" RuleState="true">
  <Item ItemType="isim" />
  <Item ItemType="zarf" />
  <Item ItemType="isim" />
  <Item ItemType="fiil" />
  <Item ItemType="n" />
</Rule>
<Rule RuleId="53" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zamir" />
  <Item ItemType="v" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
</Rule>
<Rule RuleId="54" RuleType="sözdizim" RuleState="true">
  <Item ItemType="zamir" />
  <Item ItemType="v" />
  <Item ItemType="zarf" />
  <Item ItemType="fiil" />
  <Item ItemType="n" />
</Rule>
<Rule RuleId="55" RuleType="sözdizim" RuleState="true">
  <Item ItemType="fiil" />
  <Item ItemType="fiil" />
</Rule>
```

APPENDIX C Lists

APPENDIX C.1 Abbreviation List

```
 <abbr>A</abbr>
 <abbr>AA</abbr>
 <abbr>AAFSE</abbr>
 <abbr>AAM</abbr>
 <abbr>AB</abbr>
 <abbr>ABD</abbr>
 <abbr>ABS</abbr>
 <abbr>ADSL</abbr>
 <abbr>AET</abbr>
 <abbr>AFP</abbr>
 <abbr>AGİK</abbr>
 <abbr>AGİT</abbr>
 <abbr>AI</abbr>
 <abbr>AID</abbr>
 <abbr>AIDS</abbr>
 <abbr>AİHM</abbr>
 <abbr>AİHS</abbr>
 <abbr>AK</abbr>
 <abbr>AKDTYK</abbr>
 <abbr>AKM</abbr>
 <abbr>AKPM</abbr>
 <abbr>Alb</abbr>
 <abbr>Alm</abbr>
 <abbr>AO</abbr>
 <abbr>AOÇ</abbr>
 <abbr>AÖF</abbr>
 <abbr>AP</abbr>
 <abbr>APS</abbr>
 <abbr>Apt</abbr>
 <abbr>ARGE</abbr>
 <abbr>Arş</abbr>
 <abbr>Arş.Gör</abbr>
 <abbr>Arş. Gör</abbr>
 <abbr>As</abbr>
 <abbr>ASELSAN</abbr>
 <abbr>As.İz</abbr>
 <abbr>As. İz</abbr>
 <abbr>ASKİ</abbr>
 <abbr>ASO</abbr>
 <abbr>AŞ</abbr>
 <abbr>A.Ş</abbr>
<abbr>ATM</abbr>
…
…
```

APPENDIX C.2 Root and Stem Lists

APPENDIX C.2.1 Sample Roots

ID	Root	Name	Verb
1	ab	TRUE	FALSE
2	aba	TRUE	FALSE
3	abadî	TRUE	FALSE
4	abajur	TRUE	FALSE
5	abaküs	TRUE	FALSE
6	abandone	TRUE	FALSE
7	abanî	TRUE	FALSE
8	abanoz	TRUE	FALSE
9	abaşo	TRUE	FALSE
10	abat	TRUE	FALSE
11	Abaza	TRUE	FALSE
12	abazan	TRUE	FALSE
13	Abbasî	TRUE	FALSE
14	abd	TRUE	FALSE
15	abdal	TRUE	FALSE
16	aberasyon	TRUE	FALSE
17	abes	TRUE	FALSE
18	abide	TRUE	FALSE
19	abis	TRUE	FALSE
20	abiye	TRUE	FALSE
21	abla	TRUE	FALSE
22	ablak	TRUE	FALSE
23	ablâtif	TRUE	FALSE
24	ablatya	TRUE	FALSE
25	abli	TRUE	FALSE
26	abluka	TRUE	FALSE
27	abone	TRUE	FALSE
28	abonman	TRUE	FALSE
29	aborda	TRUE	FALSE
30	abra	TRUE	TRUE
31	abraş	TRUE	FALSE
32	abril	TRUE	FALSE
33	abstraksiyonizm	TRUE	FALSE
34	abstre	TRUE	FALSE
35	absürt	TRUE	FALSE
36	abu	TRUE	FALSE
37	abuli	TRUE	FALSE
38	abullabut	TRUE	FALSE
39	abus	TRUE	FALSE

APPENDIX C.2.2 Sample Stems

ID	Gov	Ozelisim	isim	Fiil	Sifat	Zamir	Unlem	Edat	Zarf	Baglac	YFiil	Birlesik	Orjinal
1	a	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	ab	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	aba	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	aba güreşi	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
5	abacı	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	abacılık	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	abadi	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8	abajur	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	abajurcu	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	abajurculuk	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
11	abajurlu	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
12	abajursuz	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	abaküs	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	abalı	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	aban	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
16	abandır	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
17	abandırma	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
18	abandone	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
19	abani	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
20	abanma	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
21	abanoz	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
22	abanozgiller	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
23	abanozlaş	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
24	abanozlaşma	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
25	abart	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

APPENDIX C.2.2 (Cont'd.)

ID	Gov	Ozelisim	isim	Fiil	Sifat	Zamir	Unlem	Edat	Zarf	Baglac	YFiil	Birlesik	Orjinal
26	abartı	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
27	abartıcı	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
28	abartıcılık	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
29	abartıl	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
30	abartılı	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31	abartılma	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
32	abartısız	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
33	abartısızlık	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
34	abartış	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
35	abartma	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
36	abartmacı	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
37	abartmacılık	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
38	abartmalı	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
39	abartmasız	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
40	abasız	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

APPENDIX C.2.3 Sample Modified Roots / Stems According to Morphophonemic Processes

Kelime	Ozelisim	isim	Fiil	Sifat	Zamir	Unlem	Edat	Zarf	Baglac	YFiil	Birlesik	Orjinal	Link	İsim_kok	Fiil_kok	Turetme
acı\ kuvved	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	kuvvet-> kuvved
açacağ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	açacak->açacağ
açık\ hesab	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	hesap-> hesab
açık\ kard	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	kart-> kard
açık\ sened	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	senet->\ sened
adağ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	adak->adağ
adab	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	adap->adab
adaved	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	adavet->adaved
aded	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	adet->aded
afad	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	afat->afad
afed	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	afet->afed
afiyed	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	afiyet->afiyed
ağ\ yatağ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	yatak-> yatağ
ağac	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	ağaç->ağac
ağbeneğ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	ağbenek->ağbeneğ
ağılı\ böceğ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	böcek->böceğ
ağır\ aksağ	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	aksak->aksağ
ağır\ arac	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	araç->arac
ağırayağ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	ağırayak->ağırayağ
ahbab	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	ahbap->ahbab
aheng̃	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	ahenk->aheng̃
aheng	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	ahenk->aheng
ahfad	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	ahfat->ahfad
ak\ beneğ	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	ak benek->ak\ beneğ
akağac	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	akağaç->akağac

APPENDIX C.3 Tags

Suffix	Tag	Expression
Ad	**A**	**Noun**
Özel ad	AOz	Proper Noun
Genel ad	AG	Common Noun
Ad Öbeği	AO	Noun Phrase
Eylem	**E**	**Verb**
Geçişsiz Eylem	EGs	Intransitive Verb
Geçişli Eylem	EGl	Transitive Verb
Eylem Öbeği	EO	Verb Phrase
Sıfat	**S**	Adjective
Sıfat Öbeği	SO	Adjective Phrase
Niteleme sıfatı	SNit	Qualifying Adjective
Belirtme Sıfatı	SBel	Descriptive Adjectives
- Gösterme	SBelGos	- Demonstrative
- Sayı	SBelSay	- Numerical
- Belgisiz	SBelBlg	- Indefinite
- Soru	SBelSor	- Interrogative
Unvan sıfatı	SUnv	Honorific Adjective
Pekiştirme sıfatı	SPek	Intensifier Adjective
Küçültme	SKuc	Dimunitive Adjective
Belirteç	**B**	**Adverb**
Zaman Belirteci	BZam	Adverb of Time
Yer Belirteci	BYer	Adverb of Place
Durum Belirteci	BDur	Adverb of Manner
Nicelik Belirteci	BNic	Adverb of Quantity
Soru Belirteci	BSor	Interrogative Adverb
Belirteç Öbeği	BO	Adverb Phrase
Bağlaç	**Bag**	**Conjunction**
Ekleme	BagEk	Addition
Genişletme	BagGen	Expansion
Seçenek	BagSec	Option
Karşıtlık	BagKar	Contrast
Neden Sonuç	BagNs	Cause and Effect
İlgeç	**I**	**Postposition**
İlgeç Öbeği	IO	Postposition Phrase
Adıl	**Adil**	**Pronoun**
Kişi	AdilK	Personal
Gösterme	AdilGos	Demonstrative
Belgisiz	AdilBlg	Indefinite
Soru	AdilSor	Interrogative
İlgi	AdilIlgi	Relative
İyelik	AdilIye	Possesive

APPENDIX C.3 (Cont'd.)

Ad Çekim Biçimbirimleri (Ad Ulamları) ***(Nominal Inflection Morphemes (Noun Grammatical Categories))***		
Suffix	**Tag**	**Expression**
Sayı	**DuASay**	
{-lAr}	DuASayC	Adlara eklenen çoğul eki (Plural Suffix)
{-Ø}	DuASayØ	Tekil adlar (Singular Suffix)
Uyum	**DuAUy**	**Possesive Suffixes**
{-(I)m}	DuAUyKT1	1. Tekil kişi iyelik eki (*1st person singular*)
{-(I)n} / {-(I)nIz}	DuAUyKT2	2. Tekil kişi iyelik eki (*2nd person singular*)
{-(s)I(n)}	DuAUyKT3	3. Tekil kişi iyelik eki (*3rd person singular*)
{-(I)mIz}	DuAUyKC1	1. Çoğul kişi iyelik eki (*1st person plural*)
{-(I)nIz}	DuAUyKC2	2. Çoğul kişi iyelik eki (*2nd person plural*)
{-lArI(n)}	DuAUyKC3	3. Çoğul kişi iyelik eki (*3rd person plural*)
Durum	**DuADur**	**Case**
{-Ø}	DuADurYal	Yalın Durum (*Nominative Case*)
{-(y)I}	DuADurBel	Belirtme durumu (*Accusative Case*)
{-(y)A}	DuADurYon	Yönelme durumu (*Dative Case*)
{-DA}	DuADurBul	Bulunma durumu (*Locative Case*)
{-Dan}	DuADurCik	Çıkma durumu (*Ablative Case*)
{-(n)In/-Im}	DuADurTam	Tamlayan durumu (*Genitive Case*)
Eylem Çekim Biçimbirimleri – (EYLEM ULAMLARI) ***(Verbal Inflection Morphemes– (Verb Categories))***		
Zaman	**DuEZ**	
{-DI}	DuEZGD	Di'li Geçmiş Zaman (*Past Tense with Dİ*)
{-mIş}	DuEZGM	Miş'li Geçmiş Zaman (*Past Tense with MİŞ*)
{-(A, I)r)}	DuEZGen	Geniş Zaman (*Aorist*)
{-(i)yor}	DuEZSim	Şimdiki Zaman (*Present Tense*)
{-EcEK}	DuEZGel	Gelecek Zaman (*Future Tense*)
Görünüş	**DuEG**	
{-DI}	DuEGBitD	Bitmişlik Görünüşü Di'li (Perfect aspect - DI)
{-mIş}	DuEGBitM	Bitmişlik Görünüşü Miş'li (Perfect aspect - MIŞ)
{-(A, I)r)}	DuEGBtmsR	Bitmemişlik Görünüşü (Non-perfect)
{-(i)yor}	DuEGSur	Sürerlik Görünüşü (Progressive)
{-EcEK}	DuEGBtmsE	Bitmemişlik Görünüşü (Non-perfect)
Kiplik	**DuEKip**	
{-sA}	DuEKipSa	Dilek kipi (Subjunctive)
{-A}	DuEKipA	İstek Kipi (Optative)
{-mAlI}	DuEKipMali	Gereklilik Kipi (Necessitative)
{-Ø}	DuEKipEmir	Emir Kipi (Imperative)
Çatı	**DuEC**	**Voice**
Ø	DuECEt	Etken (Active)
{-Il}, {-(I)n}	DuECEdil	Edilgen (Passive)
{-(I)n}, ***{-Il}***	DuECDonus	Dönüşlü (Reflexive)
{-(I)ş}	DuECIstes	İşteş (Reciprocal)
{(A, I) -r/-t/-rt}, {-DIr}	DuECEttir	Ettirgen (Causative)

APPENDIX C.3 (Cont'd.)

Eylem Çekim Biçimbirimleri – (EYLEM ULAMLARI) (Cont'd.) ***(Verbal Inflection Morphemes– (Verb Categories))***		
Suffix	**Tag**	**Expression**
Olumsuzluk	**DuEOlz**	**Negation**
{-mA/-m}	DuEOlz	
Kişi 1. Grup (*Person Group 1*)	**DuEKGr1**	**DuEZGD ve DuEKipSa'dan sonra (*After DuEZGD and DuEKipSa*)**
{-m}	DuEKGr1T1	1. Tekil kişi (*1st person singular*)
{-n}	DuEKGr1T2	2. Tekil kişi (*2nd person singular*)
{-Ø}	DuEKGr1T3	3. Tekil kişi (*3rd person singular*)
{-k}	DuEKGr1C1	1. Çoğul kişi (*1st person plural*)
{-n-Iz}	DuEKGr1C2	2. Çoğul kişi (*2nd person plural*)
{-lAr}	DuEKGr1C3	3. Çoğul kişi (*3rd person plural*)
Kişi 2. Grup (*Person Group 2*)	**DuEKGr2**	**DuEZGen / DuEZSim / DuEZGel / DuEKipMali'dan sonra (*After DuEZGen / DuEZSim / DuEZGel / DuEKipMali*)**
{-(y)Im}	DuEKGr2T1	1. Tekil kişi (*1st person singular*)
{-sIn}	DuEKGr2T2	2. Tekil kişi (*2nd person singular*)
{-Ø}	DuEKGr2T3	3. Tekil kişi (*3rd person singular*)
{-(y)Iz}	DuEKGr2C1	1. Çoğul kişi (*1st person plural*)
{-sIn-Iz}	DuEKGr2C2	2. Çoğul kişi (*2nd person plural*)
{-lAr}	DuEKGr2C3	3. Çoğul kişi (*3rd person plural*)
Kişi 3. Grup	**DuEKGr3**	**DuEKipA'dan sonra**
{-(y)Im}	DuEKGr3T1	1. Tekil kişi (*1st person singular*)
{-sIn}	DuEKGr3T2	2. Tekil kişi (*2nd person singular*)
{-Ø}	DuEKGr3T3	3. Tekil kişi (*3rd person singular*)
{-lIm}	DuEKGr3C1	1. Çoğul kişi (*1st person plural*)
{-sIn-Iz}	DuEKGr3C2	2. Çoğul kişi (*2nd person plural*)
{-lAr}	DuEKGr3C3	3. Çoğul kişi (*3rd person plural*)
Kişi 4. Grup	**DuEKGr4**	**DuEKipEmir'den sonra**
—	DuEKGr3T1	1. Tekil kişi (*1st person singular*)
{-Ø}	DuEKGr3T2	2. Tekil kişi (*2nd person singular*)
{-sIn}	DuEKGr3T3	3. Tekil kişi (*3rd person singular*)
—	DuEKGr3C1	1. Çoğul kişi (*1st person plural*)
{-In}, {-In-Iz}	DuEKGr3C2	2. Çoğul kişi (*2nd person plural*)
{-sIn-lAr}	DuEKGr3C3	3. Çoğul kişi (*3rd person plural*)
{-EcEK}	DuEKGr3T1	Gelecek Zaman 1. Tekil kişi (*1st person singular in Future Tense*)

APPENDIX C.3 (Cont'd.)

Türetim Biçimbirimleri ***(Yapım Ekleri)*** ***(Derivational Morphemes (Derivation Suffixes))***		
Suffix	**Tag**	**Expression**
-leyin	TBAB:leyin	Addan Belirteç (*Noun to Adverb*) (sabah → sabahleyin)
-A/E	TBAE:a	Addan Eylem (*Noun to Verb*) (boş → boşa-)
-A/E	TBEA:a	Eylemden Ad (*Verb to Noun*) (yar- → yara)
-AcAk	TBEA:acak	Eylemden Ad (*Verb to Noun*) (iç- → içecek)
-AcAn	TBAS:acan	Addan Sıfat (*Noun to Adjective*) (baba →babacan)
-AcAn	TBES:acan	Eylem Sıfat (*Verb to Adjective*) (sev- →sevecen)
-AğAn	TBES:agan	Eylemden Sıfat (*Verb to Adjective*) (dur- → durağan)
-AlA	TBEE:ala	Eylemden Eylem (*Verb to Verb*) (serp- → serpele-)
-AlgA	TBEA:alga	Eylemden Ad (*Verb to Noun*) (çiz- → çizelge)
-AmAk	TBEA:amak	Eylemden Ad (*Verb to Noun*) (bas-→ basamak)
-An	TBEA:an	Eylemden Ad (*Verb to Noun*) (çarp- → çarpan)
-AnAk	TBEA:anak	Eylemden Ad (*Verb to Noun*) (yet- → yetenek)
-CA	TBAA:ca	Addan Ad (*Noun to Noun*) (çekme → çekmece)
-CA	TBEA:ca	Eylemden Ad (*Verb to Noun*) (düşün- → düşünce)
-CA	TBSS:ca	Sıfattan Sıfat (*Adjective to Adjective*) (yavaş →yavaşça)
-CAk	TBAA:cak	Addan Ad (*Noun to Noun*) (yavru →yavrucak)
-CI	TBAA:ci	Addan Ad (*Noun to Noun*) (emek →emekçi)
-CIk	TBAA:cik	Addan Ad (*Noun to Noun*) (ada → adacık)
-CIl	TBAA:cil	Addan Ad (*Noun to Noun*) (balık →balıkçıl)
-(A)Ç	TBAA:ac	Addan Ad (*Noun to Noun*) (ana →anaç)
-(A)Ç	TBEA:ac	Eylemden Ad (*Verb to Noun*) (bağla →bağlaç)
-DA	TBBE:da	Yansıma (*Onomatopoeic*) (şapır →şapırda)
-DA	TBAA:da	Addan Ad (*Noun to Noun*) (göz →gözde)
-DAm	TBAA:dam	Addan Ad (*Noun to Noun*) (yön →yöntem)
-Dan	TBAS:dan	Addan Sıfat (*Noun to Adjective*) (iç →içten)
-DAş	TBAA:das	Addan Sıfat (*Noun to Adjective*) (çağ →çağdaş)
-DI	TBEA:di	Eylemden Ad (*Verb to Noun*) (uy- →uydu)
-DIk	TBEA:dik	Eylemden Ad (*Verb to Noun*) (tanı- →tanıdık)
-GA	TBEA:ga	Eylemden Ad (*Verb to Noun*) (diz- →dizge)
-GAç	TBEA:gac	Eylemden Ad (*Verb to Noun*) (süz- →süzgeç)
-GAç	TBES:gac	Eylemden Sıfat (*Verb to Adjective*) (utan- utangaç)
-GAn	TBES:gan	Eylemden Sıfat (*Verb to Adjective*) (atıl- atılgan)
-GI	TBEA:gi	Eylemden Ad (*Verb to Noun*) (sil- →silgi)
-GIç	TBEA:gic	Eylemden Ad (*Verb to Noun*) (dal- dalgıç)
-GIç	TBES:gic	Eylemden Sıfat (*Verb to Adjective*) (bil- →bilgiç)
-GIn	TBES:gin	Eylemden Sıfat (*Verb to Adjective*) (sar- →sargın)
-I	TBEA:i	Eylemden Ad (*Verb to Noun*) (yap- →yapı)
-I	TBEE:i	Eylemden Eylem (kaz- →kaz-ı-)
-ICI	TBEA:ici	Eylemden Ad (*Verb to Noun*) (koş- →koşucu)
-ICI	TBES:ici	Eylemden Sıfat (*Verb to Adjective*) (üz- →üzücü)
-IlI	TBES:ili	Eylemden Sıfat (*Verb to Adjective*) (as- →asılı)
-(y)Iş	TBEA:is	Eylemden Ad (*Verb to Noun*) (yağ- →yağış)
-Iz	TBAA:iz	Addan Ad (*Noun to Noun*) (yavrucak →yavrucağız)
-(I)K	TBAE:k	Addan Eylem (göz →gözük-)
-(A)K	TBAA:k	Addan Ad (*Noun to Noun*) (sol →solak)
-(A)K	TBEA:k	Eylemden Ad (*Verb to Noun*) (otla- →otlak)
-Kır	TBAA:kir	Yansıma Addan Ad (*Noun to Noun)* (fiş →fışkır-)
-(A)l	TBAE:l	Addan Eylem (*Noun to Verb*) (koca →kocal-)
-(A)l	TBEA:l	Eylemden Ad (*Verb to Noun*) (oku- →okul)

APPENDIX C.3 (Cont'd.)

Türetim Biçimbirimleri *(Yapım Ekleri)*		
Suffix	**Tag**	**Expression**
-(A)l	TBAA:l	Addan Ad (*Noun to Noun*) (kum →kumul)
-lA	TBAE:la	Addan Eylem (*Noun to Verb*) (ak →akla-)
-LI	TBAS:li	Addan Sıfat (*Noun to Adjective*) (ün →ünlü)
-sIz	TBAS:siz	Addan Sıfat (*Noun to Adjective*) (ün →ünsüz)
-lIK	TBAA:lik	Addan Ad (*Noun to Noun*) (taş →taşlık)
-(A/I)M	TBEA:am	Eylemden Ad (*Verb to Noun*) (düzle- →düzlem)
-mA	TBEA:ma	Eylemden Ad (*Verb to Noun*) (yaz- →yazma)
-mAcA	TBEA:maca	Eylemden Ad (*Verb to Noun*) (düz- → düzmece)
-mAç	TBEA:mac	Eylemden Ad (*Verb to Noun*) (de- →demeç)
-mAk	TBEA:ma	Eylemden Ad (*Verb to Noun*) (ye- →yemek)
-mAn	TBEA:man	Eylemden Ad (*Verb to Noun*) (az- →azman)
-mAn	TBAS:man	Addan Ad (*Noun to Noun*) (uz →uzman)
-mAz	TBEA:maz	Eylemden Ad (*Verb to Noun*) (aç- →açmaz)
-mIk	TBEA:mik	Eylemden Ad (*Verb to Noun*) (kıy- →kıymık)
-mIş	TBEA:mis	Eylemden Ad (*Verb to Noun*) (er- →ermiş)
-msA	TBAE:msa	Addan Eylem (*Noun to Verb*) (ben →benimse)
-msA	TBSE:msa	Sıfattan Eylem (*Adjective to Verb*) (az →azımsa)
-msA	TBEE:msa	Eylemden Eylem (*Verb to Verb*) (gül- →gülümse-)
-msI	TBSS:msi	Sıfattan Sıfat (*Adjective to Adjective*) (sarı →sarımsı)
-msI	TBAS:msi	Addan Sıfat (*Noun to Adjective*) (hamur →hamurumsu)
-mtrak	TBSS:mtrak	Sıfattan Sıfat (*Adjective to Adjective*) (acı →acımtrak)
-(A/I)n	TBEA:in	Eylemden Ad (*Verb to Noun*) (tüt →tütün)
-(A/I)n	TBAA:in	Addan Ad (*Noun to Noun*) (kök →köken)
-(I)ncI	TBAS:inci	Addan Sıfat (*Noun to Adjective*) (bir →birinci)
-(I)nç	TBEA:nc	Eylemden Ad (*Verb to Noun*) (bas-basınç)
-(I)ntI	TBEA:nti	Eylemden Ad (*Verb to Noun*) (yay- →yayıntı)
-(A/I)r	TBAE:r	Addan Eylem (*Noun to Verb*) (deli →delir-)
-(A/I)r	TBEA:r	Eylemden Ad (*Verb to Noun*) (dön- →döner)
-rA	TBAB:ra	Addan belirteç (*Noun to Adverb*) (son →sonra)
-rAk	TBAS:rak	Addan Sıfat (*Noun to Adjective*) (küçük →küçürek)
-sA	TBAE:sa	Addan Eylem (*Noun to Verb*) (su →susa)
-sAk	TBEA:sak	Eylemden Ad (*Verb to Noun*) (tut- →tutsak)
-sAk	TBAS:sak	Addan Sıfat (*Noun to Adjective*) (ırak →ıraksak)
-sAl	TBEA:sal	Eylemden Ad (*Verb to Noun*) (uy- →uysal)
-sAl	TBAS:sal	Addan Sıfat (*Noun to Adjective*) (bölge →bölgesel)
-sI	TBEA:si	Eylemden Ad (*Verb to Noun*) (tüt- →ütsü)
-sI	TBAS:si	Addan Sıfat (*Noun to Adjective*) (diken →dikensi)
-sI	TBEE:si	Eylemden Eylem (*Verb to Verb*) (yan- →yansı-)
-(A/I)t	TBEA:t	Eylemden Ad (*Verb to Noun*) (um- →umut)
-(A/I)t	TBAS:t	Addan Sıfat (*Noun to Adjective*) (yaş →yaşıt)
-tAy	TBEA:tay	Eylemden Ad (*Verb to Noun*) (danış- →danıştay)
-tAy	TBAA:tay	Addan Ad (*Noun to Noun*) (kurul kurultay)
-tI	TBEA:ti	Eylemden Ad (*Verb to Noun*) (doğrul- → doğrultu)
-(A)v	TBEA:av	Eylemden Ad (*Verb to Noun*) (işle- →işlev)
-(A)y	TBEA:ay	Eylemden Ad (*Verb to Noun*) (dene- → deney)
-(A)y	TBAA:ay	Addan Ad (*Noun to Noun*) (yüz → yüzey)
-(I)z	TBAA:z	Addan Ad (*Noun to Noun*) (iki →ikiz)

APPENDIX C.4 Sample Outputs

APPENDIX C.4.1 Sentence Boundary Detection

APPENDIX C.4.1.1 Sample Document

Hayat bazen festival gibi... Etrafa bir bakıyorsunuz ki... Oooo! Tam bir festival havası. Her kafadan bir ses çıkıyor. Dünyanın bir ucunda da aynı, burnunuzun dibinde de... Festival denince aklınıza karnaval havası, havai fişekler, günlerce süren şarkılar, türküler, tiyatrolar geliyor değil mi? Hayat da böyle işte. Tek fark, katılmak istesek de istemesek de festival alayının içindeyiz biz de! Tarihte de festivaller işte böyle hayat bağlantısıyla doğmuş zaten. Doğumu, yeniden canlanmayı simgeleyen bahar aylarında ve ölümü simgeleyen kış aylarında başlarmış Eski Yunan'da... Ondan önce ise ilk insan döneminde av dönüşü yapılan ritüeller de tiyatronun doğuşuyla birlikte ilk görüldüğü dönemler. Zamanla değişe değişe günümüze kadar yol almış bu festivaller. Rio Karnavalı'ndan sarımsak, karpuz, kavun festivaline kadar da şekil değiştirerek, farklılık göstererek hem de... Tarihin ve mitolojinin bize söylediklerine dönecek olursak... Eski Yunan'da ölümsüz tanrıların pek faydalı yaratıklar olduğuna inanılmazdı. Zeus; korkunç şimşeğini düşüncesizce kullanan, genç kızların peşine düşen bir tanrıydı. Ares; savaştan, kan dökülmesinden hoşlanırdı. Hera; kıskanç olmaya görsün, adalet diye bir şey tanımazdı. Athena da çarpışmaları severdi; Aphrodite tuzak kurmakta, ağını atmakta pek ustaydı doğrusu. Bu açıdan ele alınınca ötekilerden ayrılan iki tanrı vardı; insanoğlunun en iyi arkadaşıydı onlar: Kronos'la Rhea'nın kızları, Bereket, Başak Tanrıçası Demeter'le Şarap Tanrısı Dionysos. Demeter, Dionysos'tan daha yaşlıydı. Buğdaylar, altın üzümler toplandıktan sonra ne olur? Görünürde başaklar, asmalar kalmayınca ne olur? Tarlaların yeşilliğinin yerini kara kırağı alınca ne olur? İnsanlar, kendi kendilerine bu soruları sorarlardı işte. Günler, geceler, mevsimler geçer, yıldızlar döner, bu olay hep tekrarlanırdı.Demeter'le Dionysos hasat günlerinin mutlu tanrılarıydılar, ama ya kışın ne yaparlardı? Kışın acı çekerdi onlar, toprak da üzüntülere gömülürdü. Bunun neden böyle olduğunu araştıranlar, kaynağı bazı öykülerde bulmuşlar. Sonuçta tanrının acılarını ve sevinçlerini canlandıran dinsel bayramlar ortaya çıkmış.Dionysos törenleri, insanlara yalnız mutluluk içinde yaşamayı değil, iyi bir umutla ölmeyi de öğretirmiş. Hiçbir bayram ve törenle karşılaştırılmayacak olan bu şölenler asmalar yeşermeye yüz tutunca başlar ve beş gün sürermiş. Barış ve kardeşlik havası eser, tutsaklar salıverilirmiş. Halk açık havada, bir tiyatroda toplanır, oynanan oyunları izlermiş. Burası Ege... Mitoloji kahramanları buradan da geçmiş. Tıpkı Dionysos gibi. Kaynaklar, Lade Deniz Savaşı'nı yöneten komutan Dionysos'un Phokaialı yani Foçalı olduğunu söylüyor. Bu komutanın da ismini mitolojinin en büyük kahramanlarından "Şarap Tanrısı" Dionysos'tan aldığını...

APPENDIX C.4.1.1 (Cont'd.)

Dolayısıyla Dionysos'un Foçalı olduğunu, festival alaylarının ilk buralardan da geçtiğini tahmin edebiliriz biz de! Sözü artık Foça Festivali'ne bağlayabilirim... Bugün başlayacak Ağustos'ta sona erecek. Resim sergisinden şan dinletisine, söyleşiden Foça kazıları gezisine, folklor gösterilerinden panele, şiir dinletisine, spor karşılaşmalarına ve Funda Arar, Ferhat Göçer, Edip Akbayram konserine kadar onlarca etkinlik var. Bir başka festival ise Ayvalık'ta... 22-30 Ağustos arasındaki kültür sanat günlerinde her türlü sanatsal beğeniye uygun etkinlik programda düşünülmüş. Ayşe Kulin'le ve İnci Aral'la söyleşiden İdil Biret konserine kadar... Sergi, şiir dinletisi hatta Yol Arkadaşım dizisi oyuncularıyla sohbet imkanı bile. Emre Kınay Tiyatrosu'nun "Aşk Her Yerde"si, BKM'nin "Çok Güzel Hareketler Bunlar"ı, Sunay Akın'ın tek kişilik gösterisi ve Kedi Tiyatrosu'nun "Kibarlık Budalası" da festivalde. Hatta Sezen Aksu, Bengü ve Onur Akın konseri de... Etrafta festival havası var dememiş miydim!

APPENDIX C.4.1.2 Text Output

```
1_____ Hayat bazen festival gibi....
2_____ Etrafa bir bakıyorsunuz ki....
3_____ Oooo!
4_____ Tam bir festival havası.
5_____ Her kafadan bir ses çıkıyor.
6_____ Dünyanın bir ucunda da aynı, burnunuzun dibinde de....
7_____ Festival denince aklınıza karnaval havası, havai
fişekler, günlerce süren şarkılar, türküler, tiyatrolar
geliyor değil mi?
8_____ Hayat da böyle işte.
9_____ Tek fark, katılmak istesek de istemesek de festival
alayının içindeyiz biz de!
10_____ Tarihte de festivaller işte böyle hayat bağlantısıyla
doğmuş zaten.
11_____ Doğumu, yeniden canlanmayı simgeleyen bahar aylarında
ve ölümü simgeleyen kış aylarında başlarmış Eski Yunan'da....
12_____ Ondan önce ise ilk insan döneminde av dönüşü yapılan
ritüeller de tiyatronun doğuşuyla birlikte ilk görüldüğü
dönemler.
13_____ Zamanla değişe değişe günümüze kadar yol almış bu
festivaller.
14_____ Rio Karnavalı'ndan sarımsak, karpuz, kavun festivaline
kadar da şekil değiştirerek, farklılık göstererek hem de....
15_____ Tarihin ve mitolojinin bize söylediklerine dönecek
olursak....
16_____ Eski Yunan'da ölümsüz tanrıların pek faydalı
yaratıklar olduğuna inanılmazdı.
17_____ Zeus; korkunç şimşeğini düşüncesizce kullanan, genç
kızların peşine düşen bir tanrıydı.
18_____ Ares; savaştan, kan dökülmesinden hoşlanırdı.
19_____ Hera; kıskanç olmaya görsün, adalet diye bir şey
tanımazdı.
20_____ Athena da çarpışmaları severdi; Aphrodite tuzak
kurmakta, ağını atmakta pek ustaydı doğrusu.
21_____ Bu açıdan ele alınınca ötekilerden ayrılan iki tanrı
vardı; insanoğlunun en iyi arkadaşıydı onlar: Kronos'la
Rhea'nın kızları, Bereket, Başak Tanrıçası Demeter'le Şarap
Tanrısı Dionysos.
22_____ Demeter, Dionysos'tan daha yaşlıydı.
23_____ Buğdaylar, altın üzümler toplandıktan sonra ne olur?
24_____ Görünürde başaklar, asmalar kalmayınca ne olur?
25_____ Tarlaların yeşilliğinin yerini kara kırağı alınca ne
olur?
26_____ İnsanlar, kendi kendilerine bu soruları sorarlardı
işte.
27_____ Günler, geceler, mevsimler geçer, yıldızlar döner, bu
olay hep tekrarlanırdı.
28_____ Demeter'le Dionysos hasat günlerinin mutlu
tanrılarıydılar, ama ya kışın ne yaparlardı?
29_____ Kışın acı çekerdi onlar, toprak da üzüntülere
gömülürdü.
30_____ Bunun neden böyle olduğunu araştıranlar, kaynağı bazı
öykülerde bulmuşlar.
```

APPENDIX C.4.1.2 (Cont'd.)

31_____ Sonuçta tanrının acılarını ve sevinçlerini canlandıran dinsel bayramlar ortaya çıkmış.

32_____ Dionysos törenleri, insanlara yalnız mutluluk içinde yaşamayı değil, iyi bir umutla ölmeyi de öğretirmiş.

33_____ Hiçbir bayram ve törenle karşılaştırılmayacak olan bu şölenler asmalar yeşermeye yüz tutunca başlar ve beş gün sürermiş.

34_____ Barış ve kardeşlik havası eser, tutsaklar salıverilirmiş.

35_____ Halk açık havada, bir tiyatroda toplanır, oynanan oyunları izlermiş.

36_____ Burası Ege....

37_____ Mitoloji kahramanları buradan da geçmiş.

38_____ Tıpkı Dionysos gibi.

39_____ Kaynaklar, Lade Deniz Savaşı'nı yöneten komutan Dionysos'un Phokaialı yani Foçalı olduğunu söylüyor.

40_____ Bu komutanın da ismini mitolojinin en büyük kahramanlarından "Şarap Tanrısı" Dionysos'tan aldığını....

41_____ Dolayısıyla Dionysos'un Foçalı olduğunu, festival alaylarının ilk buralardan da geçtiğini tahmin edebiliriz biz de!

42_____ Sözü artık Foça Festivali'ne bağlayabilirim....

43_____ Bugün başlayacak Ağustos'ta sona erecek.

44_____ Resim sergisinden şan dinletisine, söyleşiden Foça kazıları gezisine, folklor gösterilerinden panele, şiir dinletisine, spor karşılaşmalarına ve Funda Arar, Ferhat Göçer, Edip Akbayram konserine kadar onlarca etkinlik var.

45_____ Bir başka festival ise Ayvalık'ta....

46_____ 22-30 Ağustos arasındaki kültür sanat günlerinde her türlü sanatsal beğeniye uygun etkinlik programda düşünülmüş.

47_____ Ayşe Kulin'le ve İnci Aral'la söyleşiden İdil Biret konserine kadar....

48_____ Sergi, şiir dinletisi hatta Yol Arkadaşım dizisi oyuncularıyla sohbet imkanı bile.

49_____ Emre Kınay Tiyatrosu'nun "Aşk Her Yerde"si, BKM'nin "Çok Güzel Hareketler Bunlar"ı, Sunay Akın'ın tek kişilik gösterisi ve Kedi Tiyatrosu'nun "Kibarlık Budalası" da festivalde.

50_____ Hatta Sezen Aksu, Bengü ve Onur Akın konseri de....

51_____ Etrafta festival havası var dememiş miydim!

APPENDIX C.4.1.3 XML Output

```
- <F N="MD_Banu Şen_2008.08.28_31068.txt">
- <P I="0">
  <S I="0">Hayat bazen festival gibi ...</S>
  <S I="1">Etrafa bir bakıyorsunuz ki ...</S>
  <S I="2">Oooo !</S>
  <S I="3">Tam bir festival havası .</S>
  <S I="4">Her kafadan bir ses çıkıyor .</S>
  <S I="5">Dünyanın bir ucunda da aynı , burnunuzun dibinde de
...</S>
  <S I="6">Festival denince aklınıza karnaval havası , havai
fişekler , günlerce süren şarkılar , türküler , tiyatrolar
geliyor değil mi ?</S>
  <S I="7">Hayat da böyle işte .</S>
  <S I="8">Tek fark , katılmak istesek de istemesek de
festival alayının içindeyiz biz de !</S>
  <S I="9">Tarihte de festivaller işte böyle hayat
bağlantısıyla doğmuş zaten .</S>
  <S I="10">Doğumu , yeniden canlanmayı simgeleyen bahar
aylarında ve ölümü simgeleyen kış aylarında başlarmış Eski
Yunan'da ...</S>
  <S I="11">Ondan önce ise ilk insan döneminde av dönüşü
yapılan ritüeller de tiyatronun doğuşuyla birlikte ilk
görüldüğü dönemler .</S>
  <S I="12">Zamanla değişe değişe günümüze kadar yol almış bu
festivaller .</S>
  <S I="13">Rio Karnavalı'ndan sarımsak , karpuz , kavun
festivaline kadar da şekil değiştirerek , farklılık göstererek
hem de ...</S>
  <S I="14">Tarihin ve mitolojinin bize söylediklerine dönecek
olursak ...</S>
  <S I="15">Eski Yunan'da ölümsüz tanrıların pek faydalı
yaratıklar olduğuna inanılmazdı .</S>
  <S I="16">Zeus ; korkunç şimşeğini düşüncesizce kullanan ,
genç kızların peşine düşen bir tanrıydı .</S>
  <S I="17">Ares ; savaştan , kan dökülmesinden hoşlanırdı .
</S>
  <S I="18">Hera ; kıskanç olmaya görsün , adalet diye bir şey
tanımazdı .</S>
  <S I="19">Athena da çarpışmaları severdi ; Aphrodite tuzak
kurmakta , ağını atmakta pek ustaydı doğrusu .</S>
  <S I="20">Bu açıdan ele alınınca ötekilerden ayrılan iki
tanrı vardı ; insanoğlunun en iyi arkadaşıydı onlar: Kronos'la
Rhea'nın kızları , Bereket , Başak Tanrıçası Demeter'le Şarap
Tanrısı Dionysos .</S>
  <S I="21">Demeter , Dionysos'tan daha yaşlıydı .</S>
  <S I="22">Buğdaylar , altın üzümler toplandıktan sonra ne
olur ?</S>
  <S I="23">Görünürde başaklar , asmalar kalmayınca ne olur
?</S>
  <S I="24">Tarlaların yeşilliğinin yerini kara kırağı alınca
ne olur ?</S>
  <S I="25">İnsanlar , kendi kendilerine bu soruları
sorarlardı işte .</S>
```

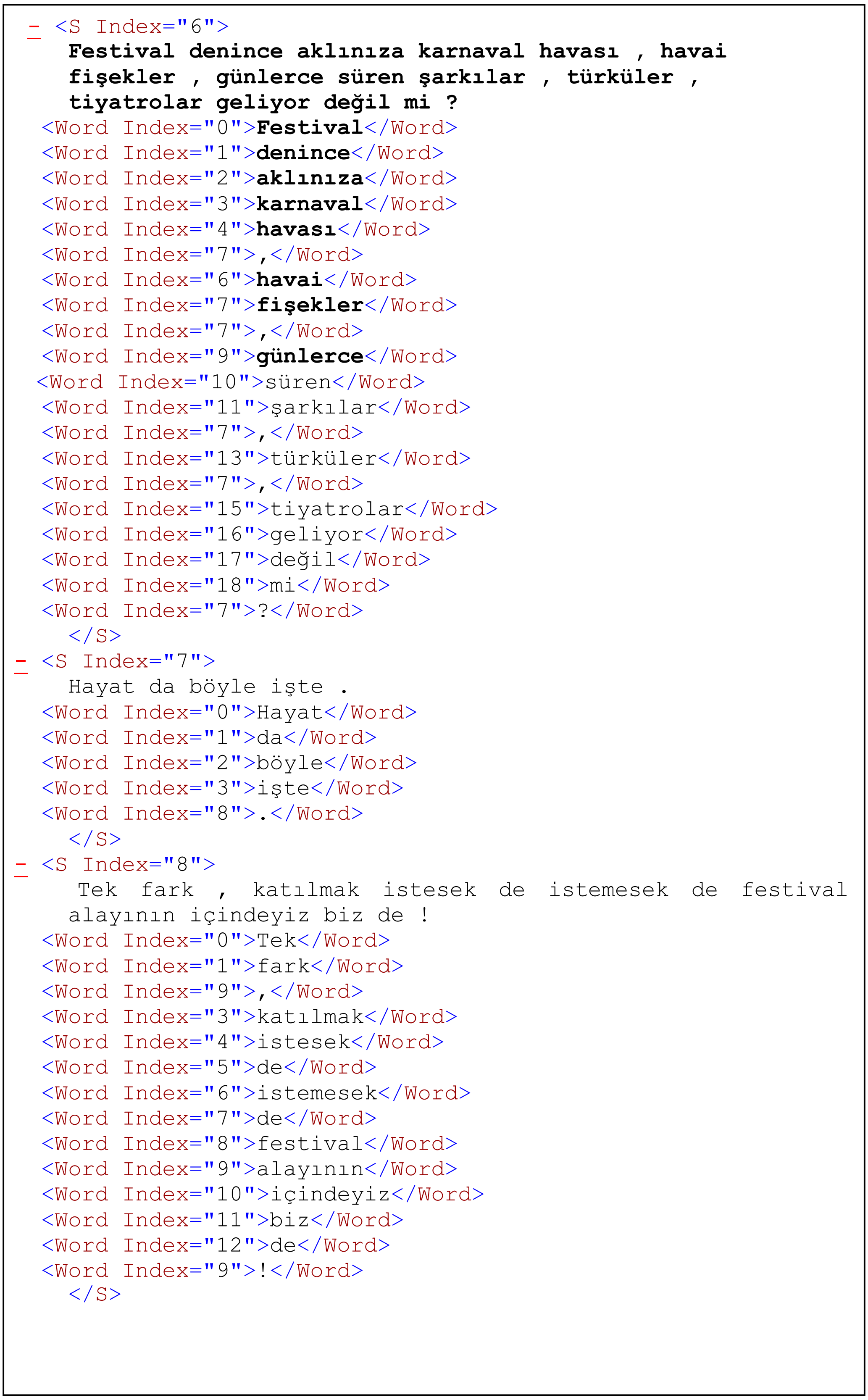
APPENDIX C.4.1.3 (Cont'd.)

```
  <S I="26">Günler , geceler , mevsimler geçer , yıldızlar
döner , bu olay hep tekrarlanırdı .</S>
  <S I="27">Demeter'le Dionysos hasat günlerinin mutlu
tanrılarıydılar , ama ya kışın ne yaparlardı ?</S>
  <S I="28">Kışın acı çekerdi onlar , toprak da üzüntülere
gömülürdü .</S>
  <S I="29">Bunun neden böyle olduğunu araştıranlar , kaynağı
bazı öykülerde bulmuşlar .</S>
  <S I="30">Sonuçta tanrının acılarını ve sevinçlerini
canlandıran dinsel bayramlar ortaya çıkmış .</S>
  <S I="31">Dionysos törenleri , insanlara yalnız mutluluk
içinde yaşamayı değil , iyi bir umutla ölmeyi de öğretirmiş
.</S>
  <S I="32">Hiçbir bayram ve törenle karşılaştırılmayacak olan
bu şölenler asmalar yeşermeye yüz tutunca başlar ve beş gün
sürermiş .</S>
  <S I="33">Barış ve kardeşlik havası eser , tutsaklar
salıverilirmiş .</S>
  <S I="34">Halk açık havada , bir tiyatroda toplanır ,
oynanan oyunları izlermiş .</S>
  <S I="35">Burası Ege ...</S>
  <S I="36">Mitoloji kahramanları buradan da geçmiş .</S>
  <S I="37">Tıpkı Dionysos gibi .</S>
  <S I="38">Kaynaklar , Lade Deniz Savaşı'nı yöneten komutan
Dionysos'un Phokaialı yani Foçalı olduğunu söylüyor .</S>
  <S I="39">Bu komutanın da ismini mitolojinin en büyük
kahramanlarından Şarap Tanrısı Dionysos'tan aldığını ...</S>
  <S I="40">Dolayısıyla Dionysos'un Foçalı olduğunu , festival
alaylarının ilk buralardan da geçtiğini tahmin edebiliriz biz
de !</S>
  <S I="41">Sözü artık Foça Festivali'ne bağlayabilirim ...
</S>
  <S I="42">Bugün başlayacak Ağustos'ta sona erecek .</S>
  <S I="43">Resim sergisinden şan dinletisine , söyleşiden
Foça kazıları gezisine , folklor gösterilerinden panele , şiir
dinletisine , spor karşılaşmalarına ve Funda Arar , Ferhat
Göçer , Edip Akbayram konserine kadar onlarca etkinlik var .
</S>
  <S I="44">Bir başka festival ise Ayvalık'ta ... </S>
  <S I="45">22 30 Ağustos arasındaki kültür sanat günlerinde
her türlü sanatsal beğeniye uygun etkinlik programda
düşünülmüş . </S>
  <S I="46">Ayşe Kulin'le ve İnci Aral'la söyleşiden İdil
Biret konserine kadar ... </S>
  <S I="47">Sergi , şiir dinletisi hatta Yol Arkadaşım dizisi
oyuncularıyla sohbet imkanı bile .</S>
  <S I="48">Emre Kınay Tiyatrosu'nun Aşk Her Yerde si ,
BKM'nin Çok Güzel Hareketler Bunlar ı , Sunay Akın'ın tek
kişilik gösterisi ve Kedi Tiyatrosu'nun Kibarlık Budalası da
festivalde .</S>
  <S I="49">Hatta Sezen Aksu , Bengü ve Onur Akın konseri
de... </S>
  <S I="50">Etrafta festival havası var dememiş miydim !</S>
  </P>
  </F>
```

APPENDIX C.4.1.4   Parsed with Wordforms

```
- <File OriginalName="MD_Banu Şen_2008.08.28_31068.txt">
- <P I="0">
- <S Index="0">
    Hayat bazen festival gibi ...
  <Word Index="0">Hayat</Word>
  <Word Index="1">bazen</Word>
  <Word Index="2">festival</Word>
  <Word Index="3">gibi</Word>
  <Word Index="1">...</Word>
  <Word Index="5" />
    </S>
- <S Index="1">
    Etrafa bir bakıyorsunuz ki ...
  <Word Index="0">Etrafa</Word>
  <Word Index="1">bir</Word>
  <Word Index="2">bakıyorsunuz</Word>
  <Word Index="3">ki</Word>
  <Word Index="2">...</Word>
  <Word Index="5" />
    </S>
- <S Index="2">
    Oooo !
  <Word Index="0">Oooo</Word>
  <Word Index="3">!</Word>
    </S>
- <S Index="3">
    Tam bir festival havası .
  <Word Index="0">Tam</Word>
  <Word Index="1">bir</Word>
  <Word Index="2">festival</Word>
  <Word Index="3">havası</Word>
  <Word Index="4">.</Word>
    </S>
 - <S Index="4">
    Her kafadan bir ses çıkıyor .
  <Word Index="0">Her</Word>
  <Word Index="1">kafadan</Word>
  <Word Index="2">bir</Word>
  <Word Index="3">ses</Word>
  <Word Index="4">çıkıyor</Word>
  <Word Index="5">.</Word>
    </S>
- <S Index="5">
    Dünyanın bir ucunda da aynı , burnunuzun dibinde de ...
  <Word Index="0">Dünyanın</Word>
  <Word Index="1">bir</Word>
  <Word Index="2">ucunda</Word>
  <Word Index="3">da</Word>
  <Word Index="4">aynı</Word>
  <Word Index="6">,</Word>
  <Word Index="6">burnunuzun</Word>
  <Word Index="7">dibinde</Word>
  <Word Index="8">de</Word>
  <Word Index="6">...</Word>
  <Word Index="10" />
    </S>
```

APPENDIX C.4.1.4 (Cont'd.)



```
 - <S Index="6">
     Festival denince aklınıza karnaval havası , havai
     fişekler , günlerce süren şarkılar , türküler ,
     tiyatrolar geliyor değil mi ?
   <Word Index="0">Festival</Word>
   <Word Index="1">denince</Word>
   <Word Index="2">aklınıza</Word>
   <Word Index="3">karnaval</Word>
   <Word Index="4">havası</Word>
   <Word Index="7">,</Word>
   <Word Index="6">havai</Word>
   <Word Index="7">fişekler</Word>
   <Word Index="7">,</Word>
   <Word Index="9">günlerce</Word>
  <Word Index="10">süren</Word>
   <Word Index="11">şarkılar</Word>
   <Word Index="7">,</Word>
   <Word Index="13">türküler</Word>
   <Word Index="7">,</Word>
   <Word Index="15">tiyatrolar</Word>
   <Word Index="16">geliyor</Word>
   <Word Index="17">değil</Word>
   <Word Index="18">mi</Word>
   <Word Index="7">?</Word>
     </S>
- <S Index="7">
     Hayat da böyle işte .
   <Word Index="0">Hayat</Word>
   <Word Index="1">da</Word>
   <Word Index="2">böyle</Word>
   <Word Index="3">işte</Word>
   <Word Index="8">.</Word>
     </S>
- <S Index="8">
      Tek  fark  ,  katılmak  istesek  de  istemesek  de  festival
     alayının içindeyiz biz de !
   <Word Index="0">Tek</Word>
   <Word Index="1">fark</Word>
   <Word Index="9">,</Word>
   <Word Index="3">katılmak</Word>
   <Word Index="4">istesek</Word>
   <Word Index="5">de</Word>
   <Word Index="6">istemesek</Word>
   <Word Index="7">de</Word>
   <Word Index="8">festival</Word>
   <Word Index="9">alayının</Word>
   <Word Index="10">içindeyiz</Word>
   <Word Index="11">biz</Word>
   <Word Index="12">de</Word>
   <Word Index="9">!</Word>
     </S>
```

APPENDIX C.4.1.4 (Cont'd.)

```
- <S Index="9">
     Tarihte de festivaller işte böyle hayat bağlantısıyla
    doğmuş zaten .
  <Word Index="0">Tarihte</Word>
  <Word Index="1">de</Word>
  <Word Index="2">festivaller</Word>
  <Word Index="3">işte</Word>
  <Word Index="4">böyle</Word>
  <Word Index="5">hayat</Word>
  <Word Index="6">bağlantısıyla</Word>
  <Word Index="7">doğmuş</Word>
  <Word Index="8">zaten</Word>
  <Word Index="10">.</Word>
    </S>
- <S Index="10">
     Doğumu , yeniden canlanmayı simgeleyen bahar aylarında
    ve ölümü simgeleyen kış aylarında başlarmış Eski
    Yunan'da ...
  <Word Index="0">Doğumu</Word>
  <Word Index="11">,</Word>
  <Word Index="2">yeniden</Word>
  <Word Index="3">canlanmayı</Word>
  <Word Index="4">simgeleyen</Word>
  <Word Index="5">bahar</Word>
  <Word Index="6">aylarında</Word>
  <Word Index="7">ve</Word>
  <Word Index="8">ölümü</Word>
  <Word Index="9">simgeleyen</Word>
  <Word Index="10">kış</Word>
  <Word Index="11">aylarında</Word>
  <Word Index="12">başlarmış</Word>
  <Word Index="13">Eski</Word>
  <Word Index="14">Yunan'da</Word>
  <Word Index="11">...</Word>
  <Word Index="16" />
    </S>
 ...
 ...
```

APPENDIX C.4.2 Word Detection

APPENDIX C.4.2.1 Tagged Output (Sentence 1)

```
- <File OriginalName="test.txt">
- <P I="0">
- <S Index="0">
    Doğru söyleyeni dokuz köyden kovarlar .
- <Word Index="0" Value="Doğru">
- <R I="0" V="Do" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBEA-k>ğ</TBEA-k>
  <TBEA-r>r</TBEA-r>
  <TBEA-i>u</TBEA-i>
    </Sx>
- <Sx I="1">
  <TBAE-k>ğ</TBAE-k>
  <YtuUR1>r</YtuUR1>
  <DuAUyKT3>u</DuAUyKT3>
    </Sx>
- <Sx I="2">
  <TBAE-k>ğ</TBAE-k>
  <DuECEttir>r</DuECEttir>
  <DuAUyKT3>u</DuAUyKT3>
    </Sx>
- <Sx I="3">
  <TBAE-k>ğ</TBAE-k>
  <YtuUR1>r</YtuUR1>
  <DuADurBel>u</DuADurBel>
    </Sx>
- <Sx I="4">
  <TBAE-k>ğ</TBAE-k>
  <DuECEttir>r</DuECEttir>
  <DuADurBel>u</DuADurBel>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="Doğ" T="fiil">
- <Suffixes>
- <Sx I="0">
  <TBEA-r>r</TBEA-r>
  <TBEA-i>u</TBEA-i>
    </Sx>
- <Sx I="1">
  <TBAE-r>r</TBAE-r>
  <TBEE-i>u</TBEE-i>
    </Sx>
- <Sx I="2">
  <YtuUR1>r</YtuUR1>
  <DuAUyKT3>u</DuAUyKT3>
    </Sx>
```

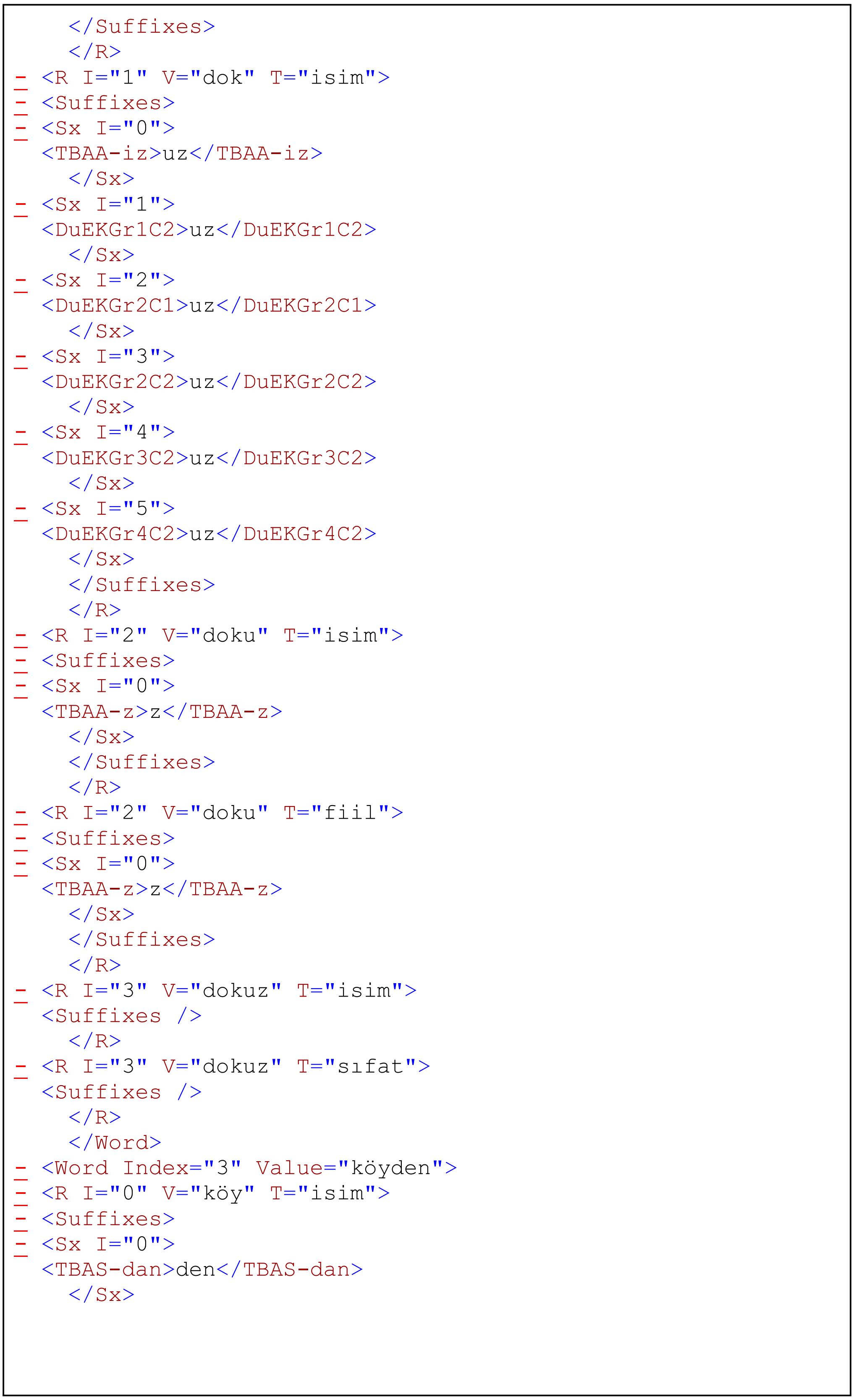
APPENDIX C.4.2.1 (Cont'd.)

```
- <Sx I="3">
  <DuECEttir>r</DuECEttir>
  <DuAUyKT3>u</DuAUyKT3>
    </Sx>
- <Sx I="4">
  <TBAE-r>r</TBAE-r>
  <DuAUyKT3>u</DuAUyKT3>
    </Sx>
- <Sx I="5">
  <TBEA-r>r</TBEA-r>
  <DuAUyKT3>u</DuAUyKT3>
    </Sx>
- <Sx I="6">
  <YtuUR1>r</YtuUR1>
  <DuADurBel>u</DuADurBel>
    </Sx>
- <Sx I="7">
  <DuECEttir>r</DuECEttir>
  <DuADurBel>u</DuADurBel>
    </Sx>
- <Sx I="8">
  <TBAE-r>r</TBAE-r>
  <DuADurBel>u</DuADurBel>
    </Sx>
- <Sx I="9">
  <TBEA-r>r</TBEA-r>
  <DuADurBel>u</DuADurBel>
    </Sx>
    </Suffixes>
    </R>
- <R I="2" V="Doğru" T="sıfat">
  <Suffixes />
    </R>
    </Word>
- <Word Index="1" Value="söyleyeni">
- <R I="0" V="söyle" T="fiil">
- <Suffixes>
- <Sx I="0">
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>e</TBEA-a>
  <TBEA-in>n</TBEA-in>
  <TBEA-i>i</TBEA-i>
    </Sx>
- <Sx I="1">
  <YS>y</YS>
  <TBEA-a>e</TBEA-a>
  <TBEA-in>n</TBEA-in>
  <TBEA-i>i</TBEA-i>
    </Sx>
- <Sx I="2">
  <TBEA-ay>y</TBEA-ay>
  <TBEA-an>en</TBEA-an>
  <TBEA-i>i</TBEA-i>
    </Sx>
```

APPENDIX C.4.2.1 (Cont'd.)

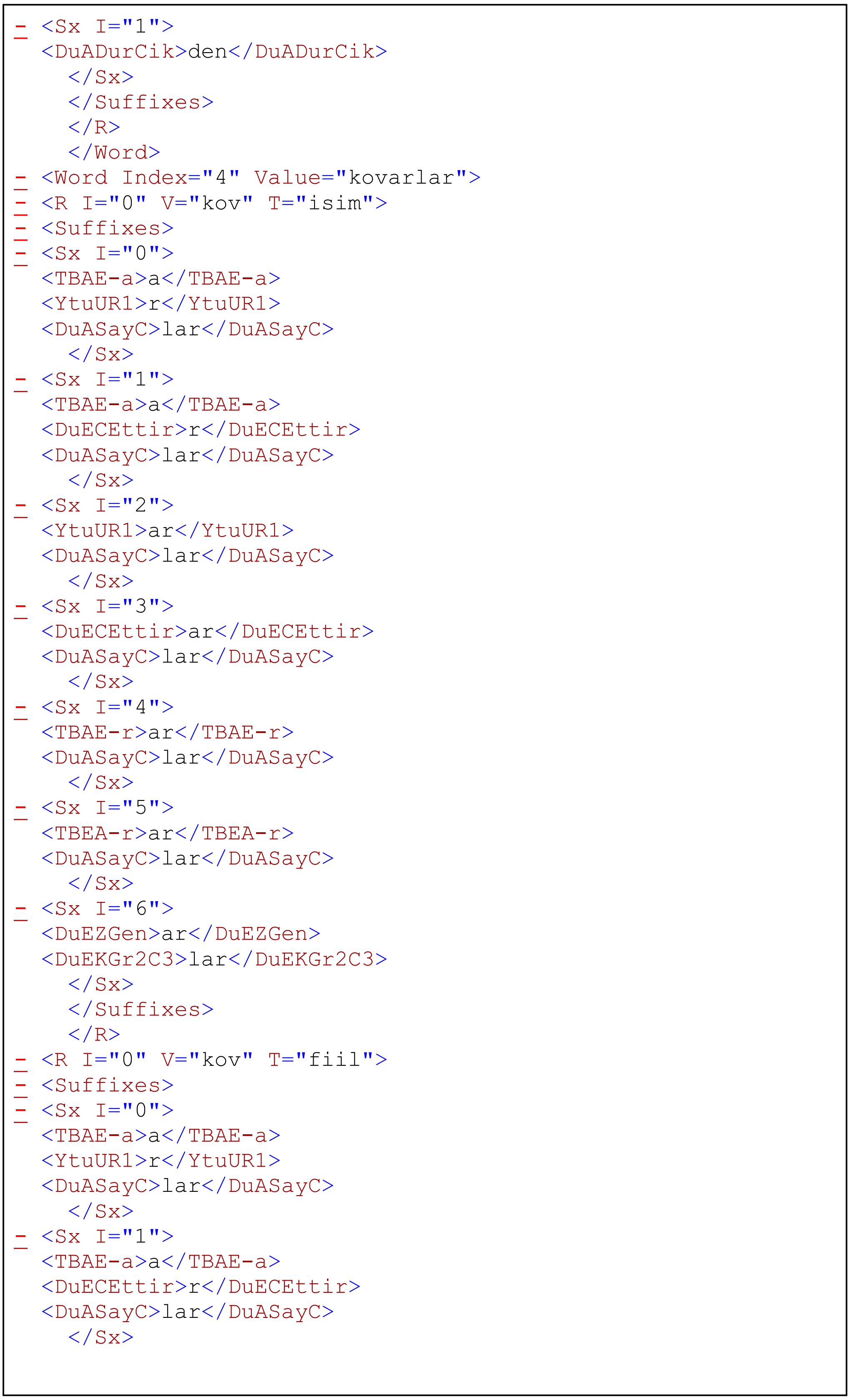
```
- <Sx I="3">
  <YS>y</YS>
  <TBEA-an>en</TBEA-an>
  <TBEA-i>i</TBEA-i>
    </Sx>
- <Sx I="4">
  <TBEA-ay>y</TBEA-ay>
  <TBEA-in>en</TBEA-in>
  <TBEA-i>i</TBEA-i>
    </Sx>
- <Sx I="5">
  <YS>y</YS>
  <TBEA-in>en</TBEA-in>
  <TBEA-i>i</TBEA-i>
    </Sx>
- <Sx I="6">
  <TBEA-ay>y</TBEA-ay>
  <TBAA-in>en</TBAA-in>
  <DuAUyKT3>i</DuAUyKT3>
    </Sx>
- <Sx I="7">
  <TBAA-ay>y</TBAA-ay>
  <TBAA-in>en</TBAA-in>
  <DuAUyKT3>i</DuAUyKT3>
    </Sx>
- <Sx I="8">
  <TBEA-ay>y</TBEA-ay>
  <TBAA-in>en</TBAA-in>
  <DuADurBel>i</DuADurBel>
    </Sx>
- <Sx I="9">
  <TBAA-ay>y</TBAA-ay>
  <TBAA-in>en</TBAA-in>
  <DuADurBel>i</DuADurBel>
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="2" Value="dokuz">
- <R I="0" V="do" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBEA-k>k</TBEA-k>
  <TBEA-i>u</TBEA-i>
  <TBAA-z>z</TBAA-z>
    </Sx>
- <Sx I="1">
  <TBAA-k>k</TBAA-k>
  <TBAA-iz>uz</TBAA-iz>
    </Sx>
- <Sx I="2">
  <TBEA-k>k</TBEA-k>
  <TBAA-iz>uz</TBAA-iz>
    </Sx>
```

APPENDIX C.4.2.1 (Cont'd.)



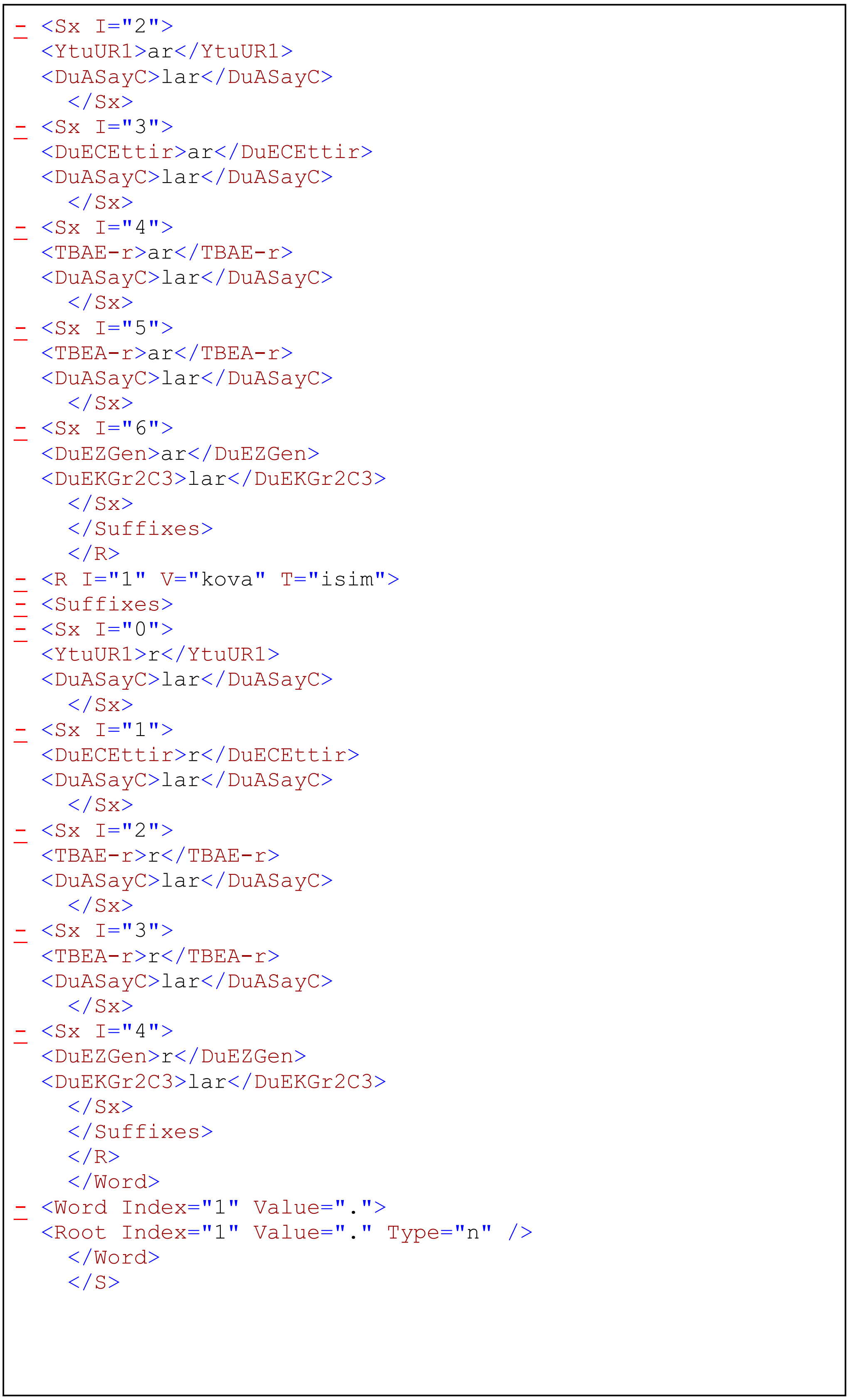
```
    </Suffixes>
    </R>
- <R I="1" V="dok" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBAA-iz>uz</TBAA-iz>
    </Sx>
- <Sx I="1">
  <DuEKGr1C2>uz</DuEKGr1C2>
    </Sx>
- <Sx I="2">
  <DuEKGr2C1>uz</DuEKGr2C1>
    </Sx>
- <Sx I="3">
  <DuEKGr2C2>uz</DuEKGr2C2>
    </Sx>
- <Sx I="4">
  <DuEKGr3C2>uz</DuEKGr3C2>
    </Sx>
- <Sx I="5">
  <DuEKGr4C2>uz</DuEKGr4C2>
    </Sx>
    </Suffixes>
    </R>
- <R I="2" V="doku" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBAA-z>z</TBAA-z>
    </Sx>
    </Suffixes>
    </R>
- <R I="2" V="doku" T="fiil">
- <Suffixes>
- <Sx I="0">
  <TBAA-z>z</TBAA-z>
    </Sx>
    </Suffixes>
    </R>
- <R I="3" V="dokuz" T="isim">
  <Suffixes />
    </R>
- <R I="3" V="dokuz" T="sıfat">
  <Suffixes />
    </R>
    </Word>
- <Word Index="3" Value="köyden">
- <R I="0" V="köy" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBAS-dan>den</TBAS-dan>
    </Sx>
```

APPENDIX C.4.2.1 (Cont'd.)



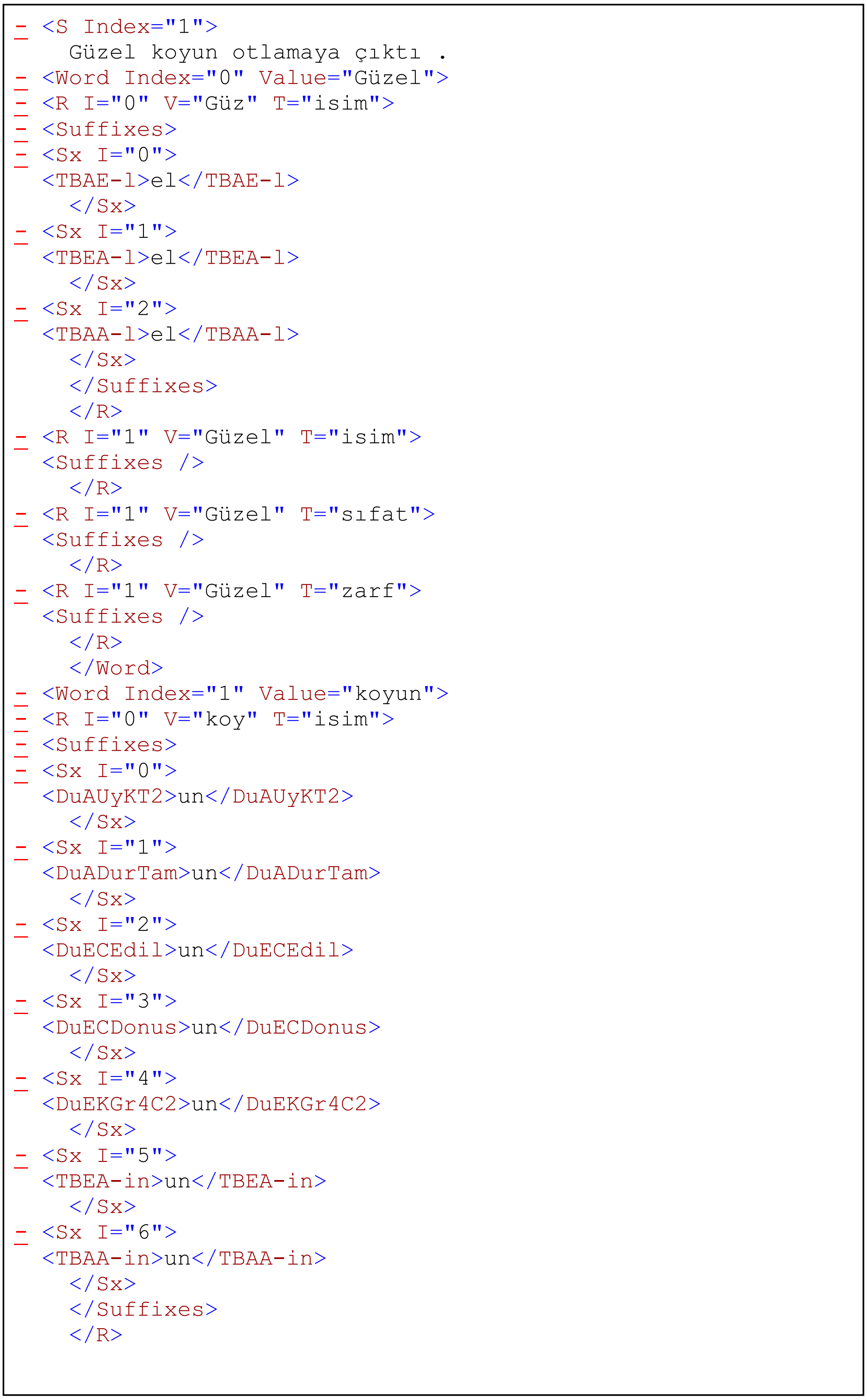
```
- <Sx I="1">
  <DuADurCik>den</DuADurCik>
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="4" Value="kovarlar">
- <R I="0" V="kov" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBAE-a>a</TBAE-a>
  <YtuUR1>r</YtuUR1>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="1">
  <TBAE-a>a</TBAE-a>
  <DuECEttir>r</DuECEttir>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="2">
  <YtuUR1>ar</YtuUR1>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="3">
  <DuECEttir>ar</DuECEttir>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="4">
  <TBAE-r>ar</TBAE-r>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="5">
  <TBEA-r>ar</TBEA-r>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="6">
  <DuEZGen>ar</DuEZGen>
  <DuEKGr2C3>lar</DuEKGr2C3>
    </Sx>
    </Suffixes>
    </R>
- <R I="0" V="kov" T="fiil">
- <Suffixes>
- <Sx I="0">
  <TBAE-a>a</TBAE-a>
  <YtuUR1>r</YtuUR1>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="1">
  <TBAE-a>a</TBAE-a>
  <DuECEttir>r</DuECEttir>
  <DuASayC>lar</DuASayC>
    </Sx>
```

APPENDIX C.4.2.1 (Cont'd.)



```
- <Sx I="2">
  <YtuUR1>ar</YtuUR1>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="3">
  <DuECEttir>ar</DuECEttir>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="4">
  <TBAE-r>ar</TBAE-r>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="5">
  <TBEA-r>ar</TBEA-r>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="6">
  <DuEZGen>ar</DuEZGen>
  <DuEKGr2C3>lar</DuEKGr2C3>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="kova" T="isim">
- <Suffixes>
- <Sx I="0">
  <YtuUR1>r</YtuUR1>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="1">
  <DuECEttir>r</DuECEttir>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="2">
  <TBAE-r>r</TBAE-r>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="3">
  <TBEA-r>r</TBEA-r>
  <DuASayC>lar</DuASayC>
    </Sx>
- <Sx I="4">
  <DuEZGen>r</DuEZGen>
  <DuEKGr2C3>lar</DuEKGr2C3>
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="1" Value=".">
  <Root Index="1" Value="." Type="n" />
    </Word>
    </S>
```

APPENDIX C.4.2.2 Tagged Output (Sentence 2)

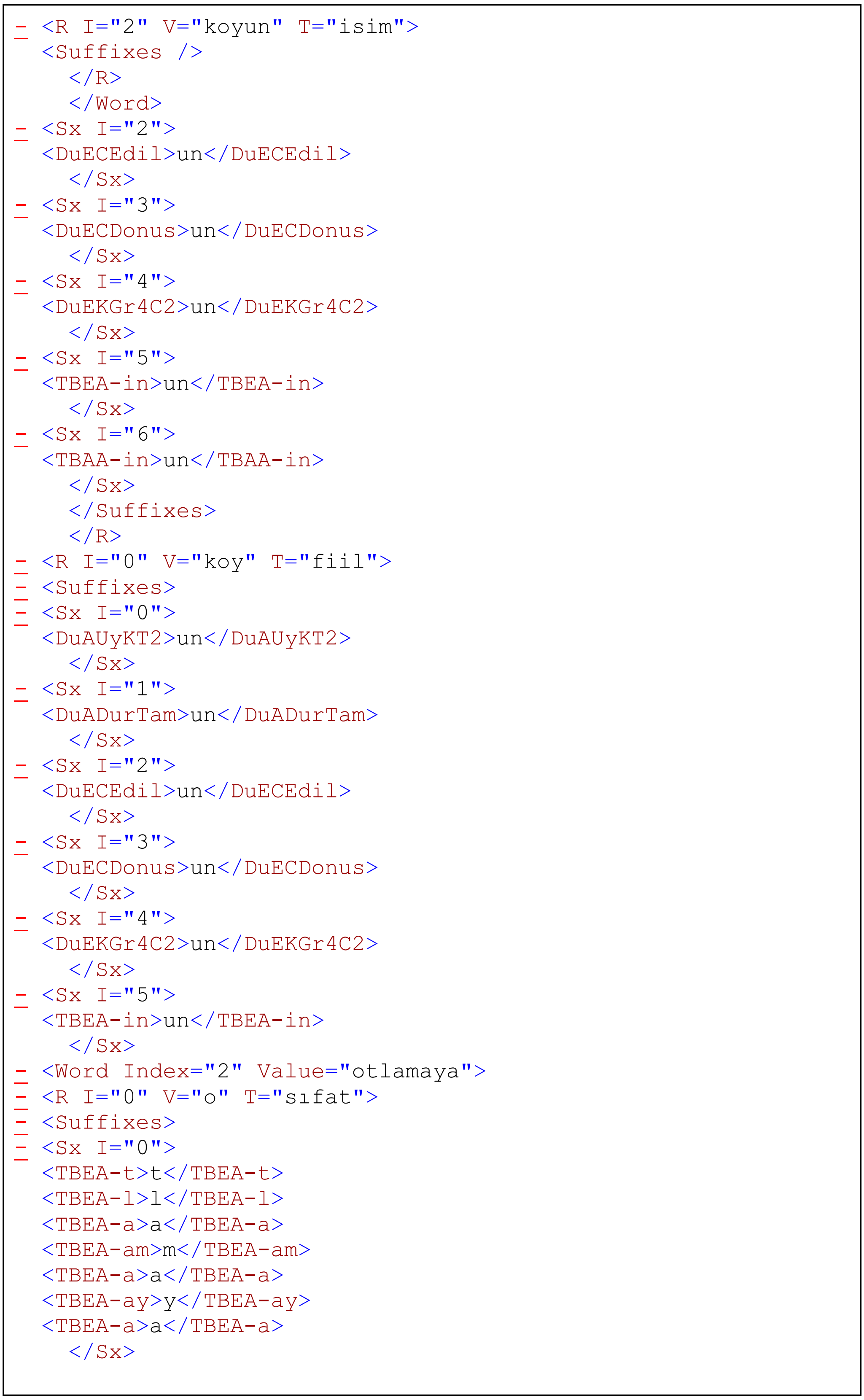


```
- <S Index="1">
    Güzel koyun otlamaya çıktı .
- <Word Index="0" Value="Güzel">
- <R I="0" V="Güz" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBAE-l>el</TBAE-l>
    </Sx>
- <Sx I="1">
  <TBEA-l>el</TBEA-l>
    </Sx>
- <Sx I="2">
  <TBAA-l>el</TBAA-l>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="Güzel" T="isim">
  <Suffixes />
    </R>
- <R I="1" V="Güzel" T="sıfat">
  <Suffixes />
    </R>
- <R I="1" V="Güzel" T="zarf">
  <Suffixes />
    </R>
    </Word>
- <Word Index="1" Value="koyun">
- <R I="0" V="koy" T="isim">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2>un</DuAUyKT2>
    </Sx>
- <Sx I="1">
  <DuADurTam>un</DuADurTam>
    </Sx>
- <Sx I="2">
  <DuECEdil>un</DuECEdil>
    </Sx>
- <Sx I="3">
  <DuECDonus>un</DuECDonus>
    </Sx>
- <Sx I="4">
  <DuEKGr4C2>un</DuEKGr4C2>
    </Sx>
- <Sx I="5">
  <TBEA-in>un</TBEA-in>
    </Sx>
- <Sx I="6">
  <TBAA-in>un</TBAA-in>
    </Sx>
    </Suffixes>
    </R>
```

APPENDIX C.4.2.2 (Cont'd.)

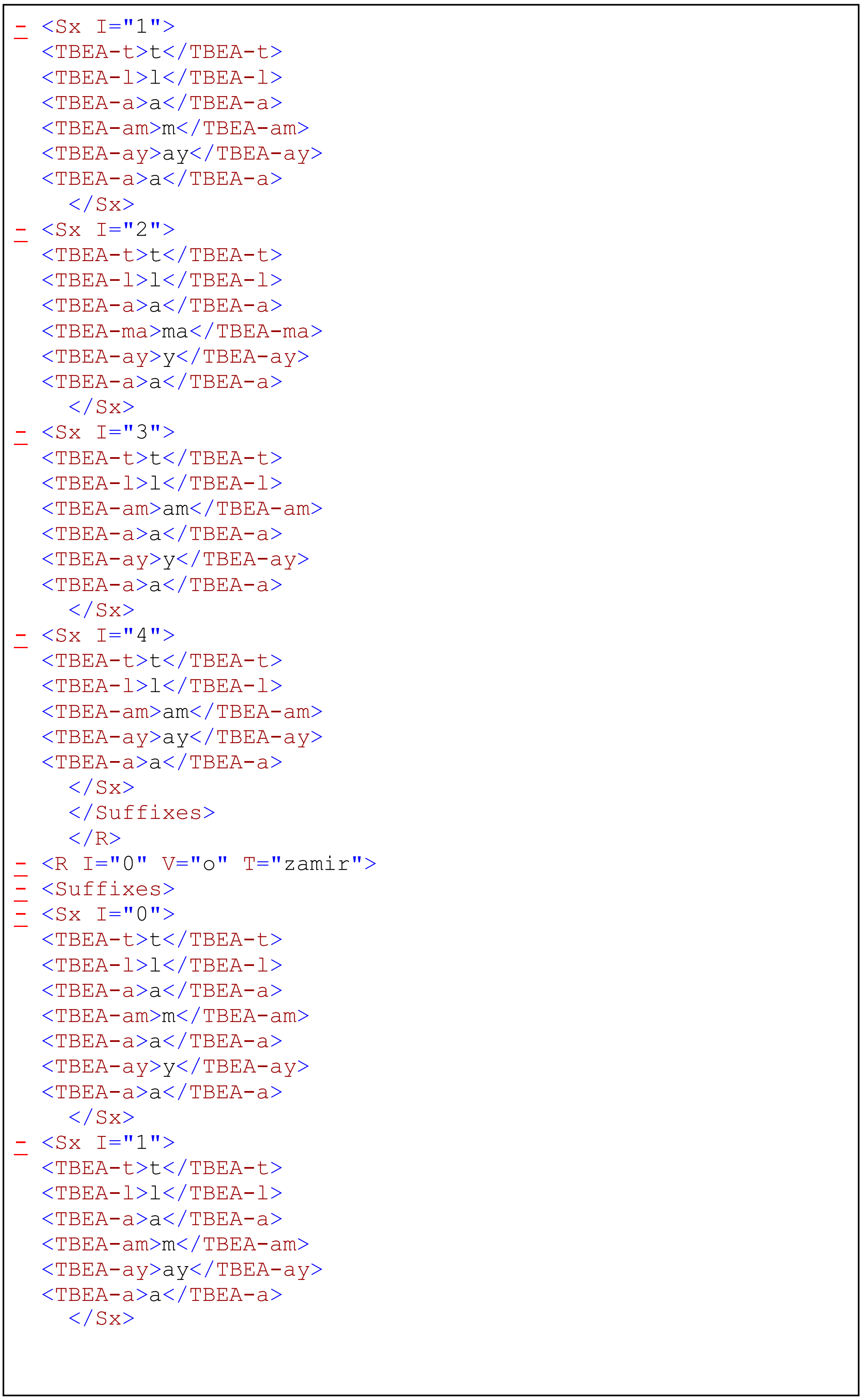
```
- <R I="0" V="koy" T="fiil">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2>un</DuAUyKT2>
    </Sx>
- <Sx I="1">
  <DuADurTam>un</DuADurTam>
    </Sx>
- <Sx I="2">
  <DuECEdil>un</DuECEdil>
    </Sx>
- <Sx I="3">
  <DuECDonus>un</DuECDonus>
    </Sx>
- <Sx I="4">
  <DuEKGr4C2>un</DuEKGr4C2>
    </Sx>
- <Sx I="5">
  <TBEA-in>un</TBEA-in>
    </Sx>
- <Sx I="6">
  <TBAA-in>un</TBAA-in>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="koyu" T="sıfat">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2>n</DuAUyKT2>
    </Sx>
- <Sx I="1">
  <DuECEdil>n</DuECEdil>
    </Sx>
- <Sx I="2">
  <DuECDonus>n</DuECDonus>
    </Sx>
- <Sx I="3">
  <DuEKGr1T2>n</DuEKGr1T2>
    </Sx>
- <Sx I="4">
  <TBEA-in>n</TBEA-in>
    </Sx>
- <Sx I="5">
  <TBAA-in>n</TBAA-in>
    </Sx>
- <Sx I="6">
  <YS>n</YS>
    </Sx>
    </Suffixes>
    </R>
```

APPENDIX C.4.2.2 (Cont'd.)



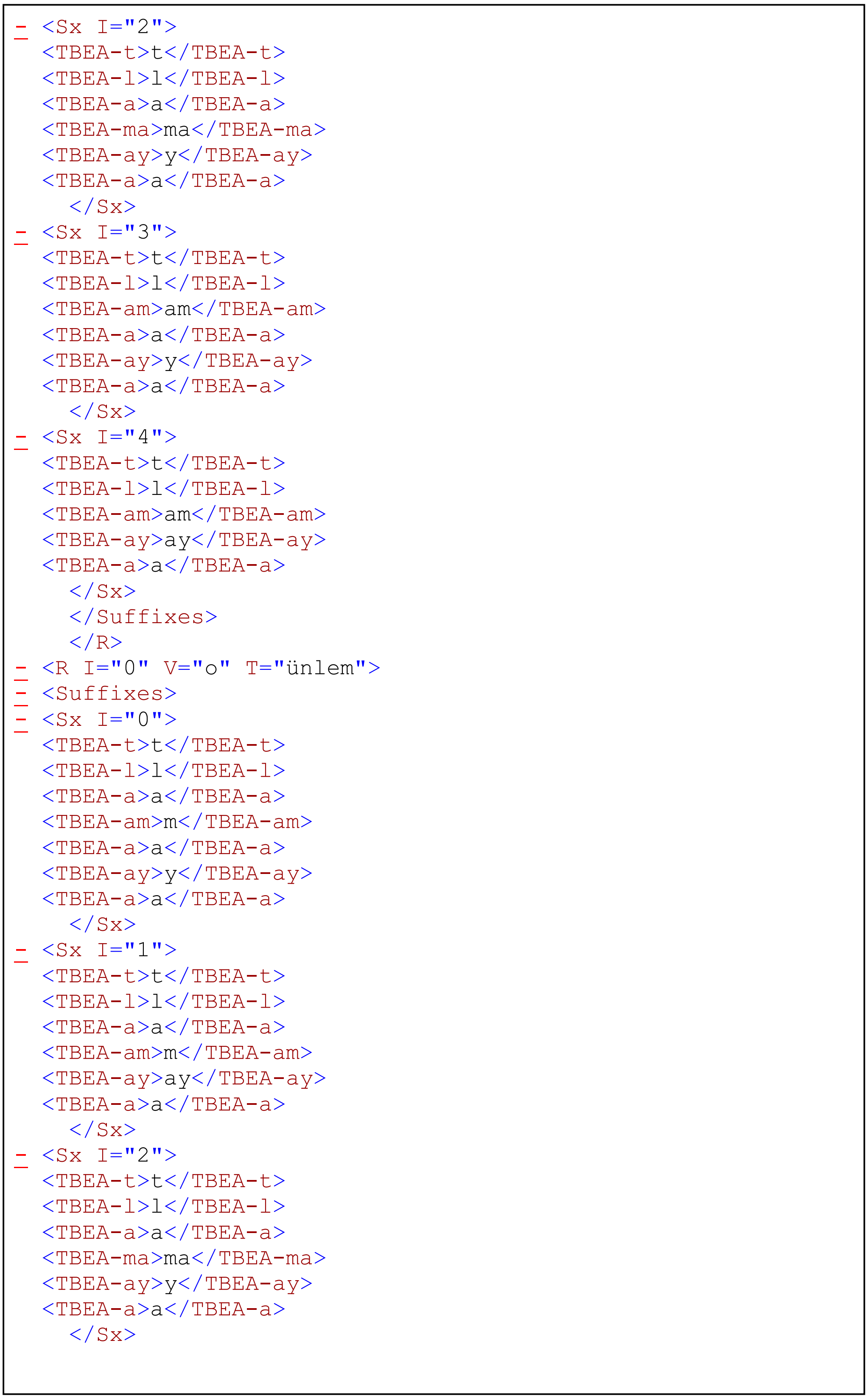
```
- <R I="2" V="koyun" T="isim">
  <Suffixes />
    </R>
    </Word>
- <Sx I="2">
  <DuECEdil>un</DuECEdil>
    </Sx>
- <Sx I="3">
  <DuECDonus>un</DuECDonus>
    </Sx>
- <Sx I="4">
  <DuEKGr4C2>un</DuEKGr4C2>
    </Sx>
- <Sx I="5">
  <TBEA-in>un</TBEA-in>
    </Sx>
- <Sx I="6">
  <TBAA-in>un</TBAA-in>
    </Sx>
    </Suffixes>
    </R>
- <R I="0" V="koy" T="fiil">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2>un</DuAUyKT2>
    </Sx>
- <Sx I="1">
  <DuADurTam>un</DuADurTam>
    </Sx>
- <Sx I="2">
  <DuECEdil>un</DuECEdil>
    </Sx>
- <Sx I="3">
  <DuECDonus>un</DuECDonus>
    </Sx>
- <Sx I="4">
  <DuEKGr4C2>un</DuEKGr4C2>
    </Sx>
- <Sx I="5">
  <TBEA-in>un</TBEA-in>
    </Sx>
- <Word Index="2" Value="otlamaya">
- <R I="0" V="o" T="sıfat">
- <Suffixes>
- <Sx I="0">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)



```
- <Sx I="1">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="2">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-ma>ma</TBEA-ma>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="3">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="4">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
    </Suffixes>
    </R>
- <R I="0" V="o" T="zamir">
- <Suffixes>
- <Sx I="0">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="1">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)

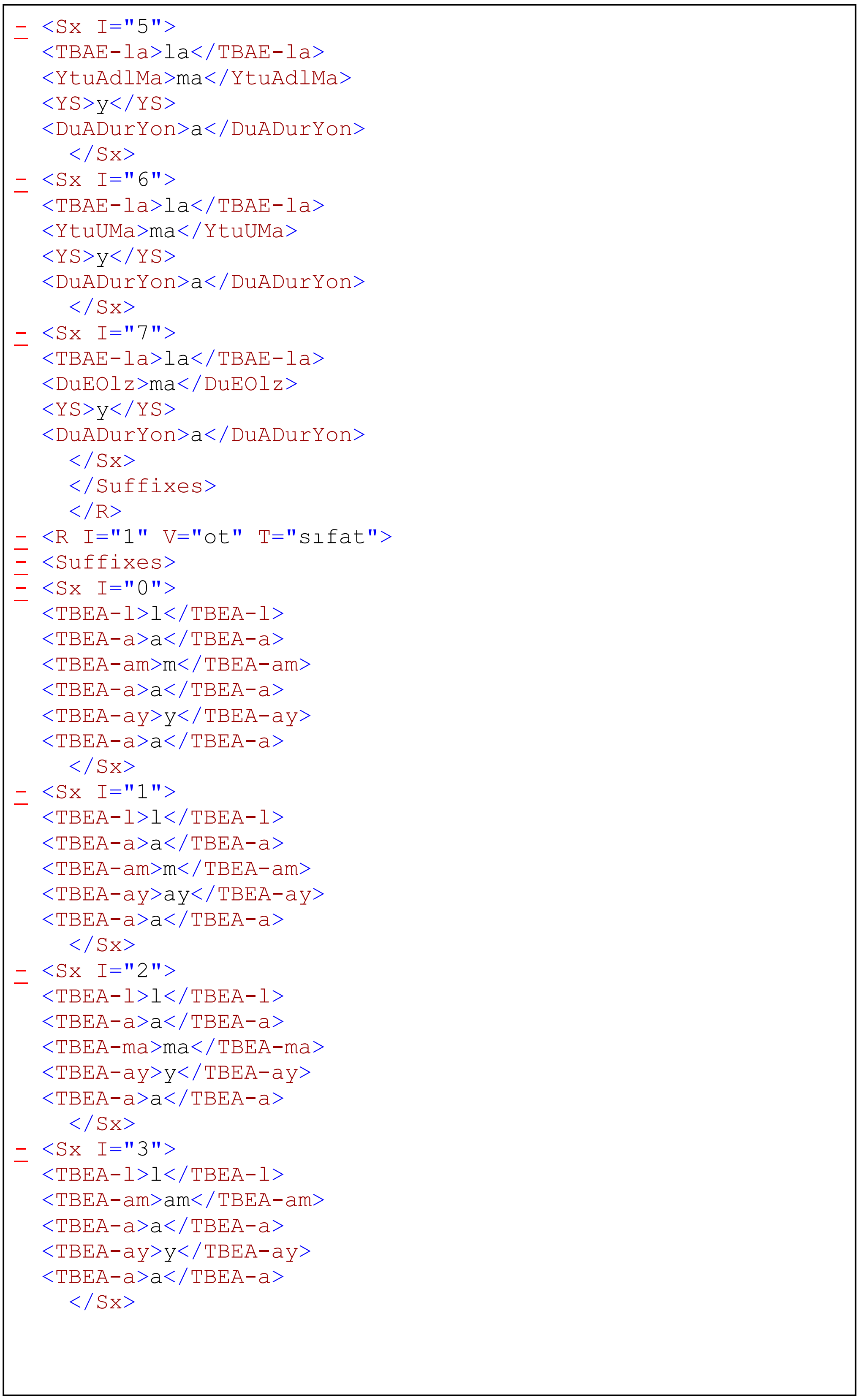


```
- <Sx I="2">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-ma>ma</TBEA-ma>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="3">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="4">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
    </Suffixes>
    </R>
- <R I="0" V="o" T="ünlem">
- <Suffixes>
- <Sx I="0">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="1">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="2">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-ma>ma</TBEA-ma>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)

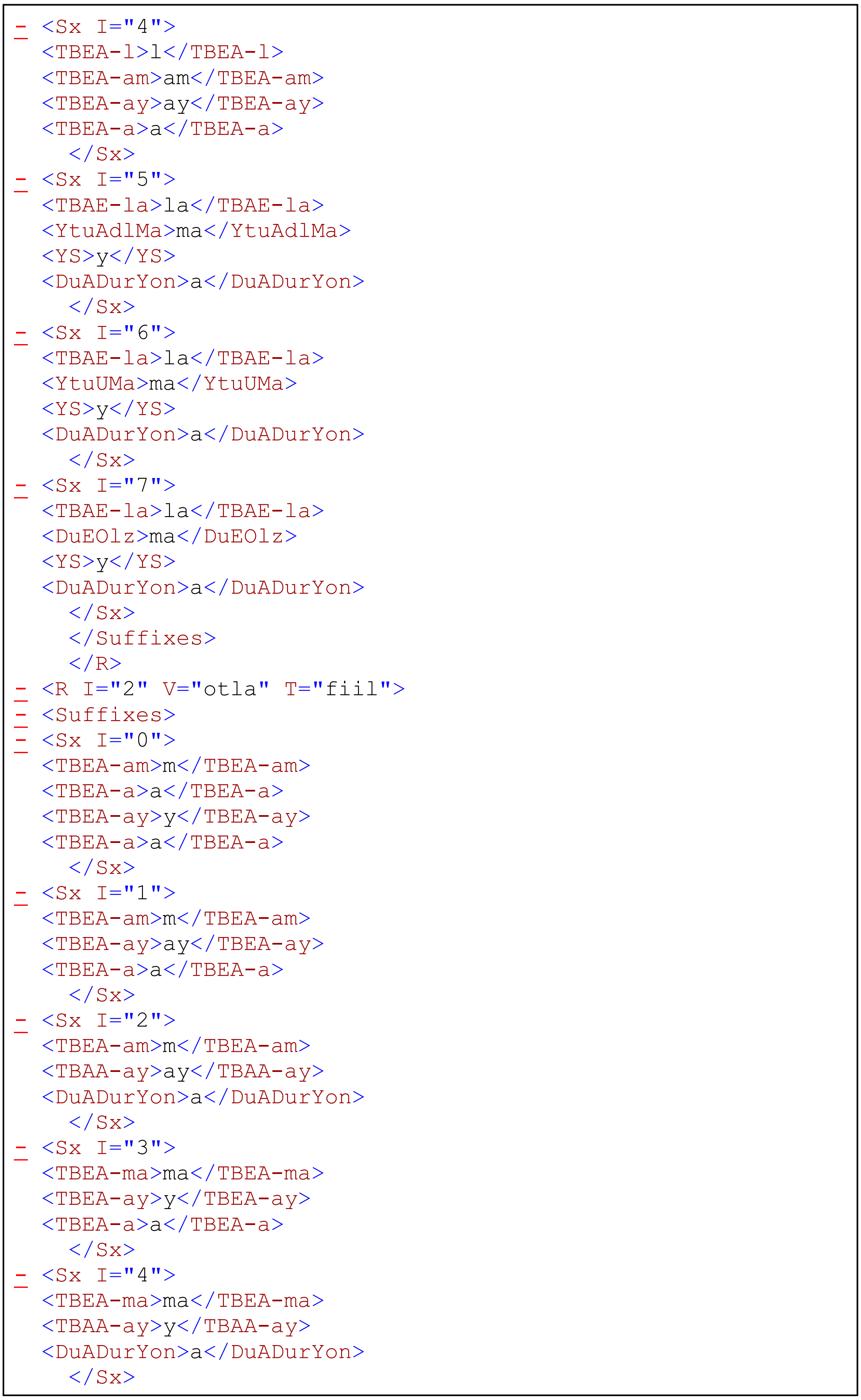
```
- <Sx I="3">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="4">
  <TBEA-t>t</TBEA-t>
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="ot" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="1">
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="2">
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-ma>ma</TBEA-ma>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="3">
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="4">
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)



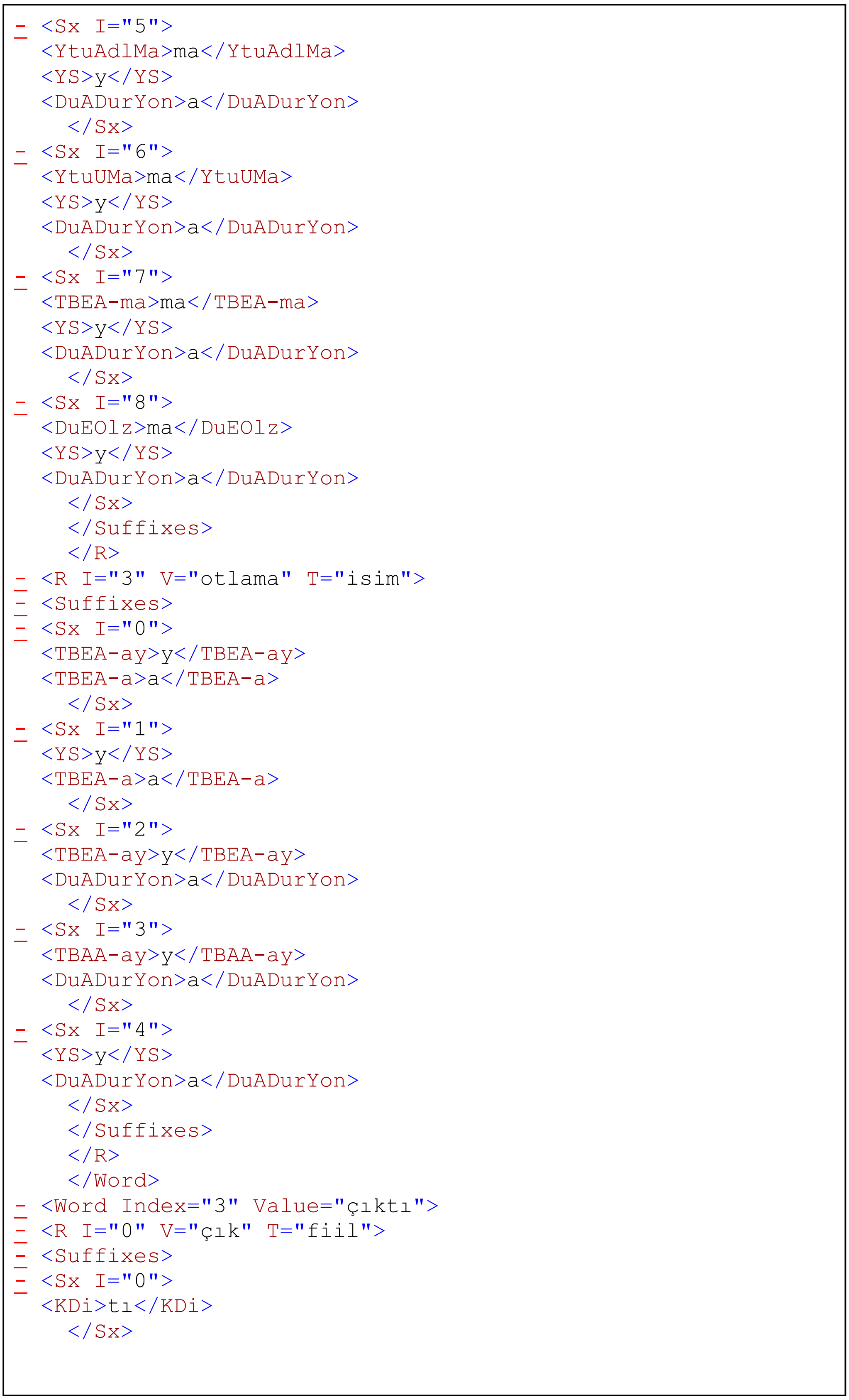
```
- <Sx I="5">
  <TBAE-la>la</TBAE-la>
  <YtuAdlMa>ma</YtuAdlMa>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="6">
  <TBAE-la>la</TBAE-la>
  <YtuUMa>ma</YtuUMa>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="7">
  <TBAE-la>la</TBAE-la>
  <DuEOlz>ma</DuEOlz>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="ot" T="sıfat">
- <Suffixes>
- <Sx I="0">
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="1">
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-am>m</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="2">
  <TBEA-l>l</TBEA-l>
  <TBEA-a>a</TBEA-a>
  <TBEA-ma>ma</TBEA-ma>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="3">
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)



```
- <Sx I="4">
  <TBEA-l>l</TBEA-l>
  <TBEA-am>am</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="5">
  <TBAE-la>la</TBAE-la>
  <YtuAdlMa>ma</YtuAdlMa>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="6">
  <TBAE-la>la</TBAE-la>
  <YtuUMa>ma</YtuUMa>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="7">
  <TBAE-la>la</TBAE-la>
  <DuEOlz>ma</DuEOlz>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
    </Suffixes>
    </R>
- <R I="2" V="otla" T="fiil">
- <Suffixes>
- <Sx I="0">
  <TBEA-am>m</TBEA-am>
  <TBEA-a>a</TBEA-a>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="1">
  <TBEA-am>m</TBEA-am>
  <TBEA-ay>ay</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="2">
  <TBEA-am>m</TBEA-am>
  <TBAA-ay>ay</TBAA-ay>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="3">
  <TBEA-ma>ma</TBEA-ma>
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="4">
  <TBEA-ma>ma</TBEA-ma>
  <TBAA-ay>y</TBAA-ay>
  <DuADurYon>a</DuADurYon>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)

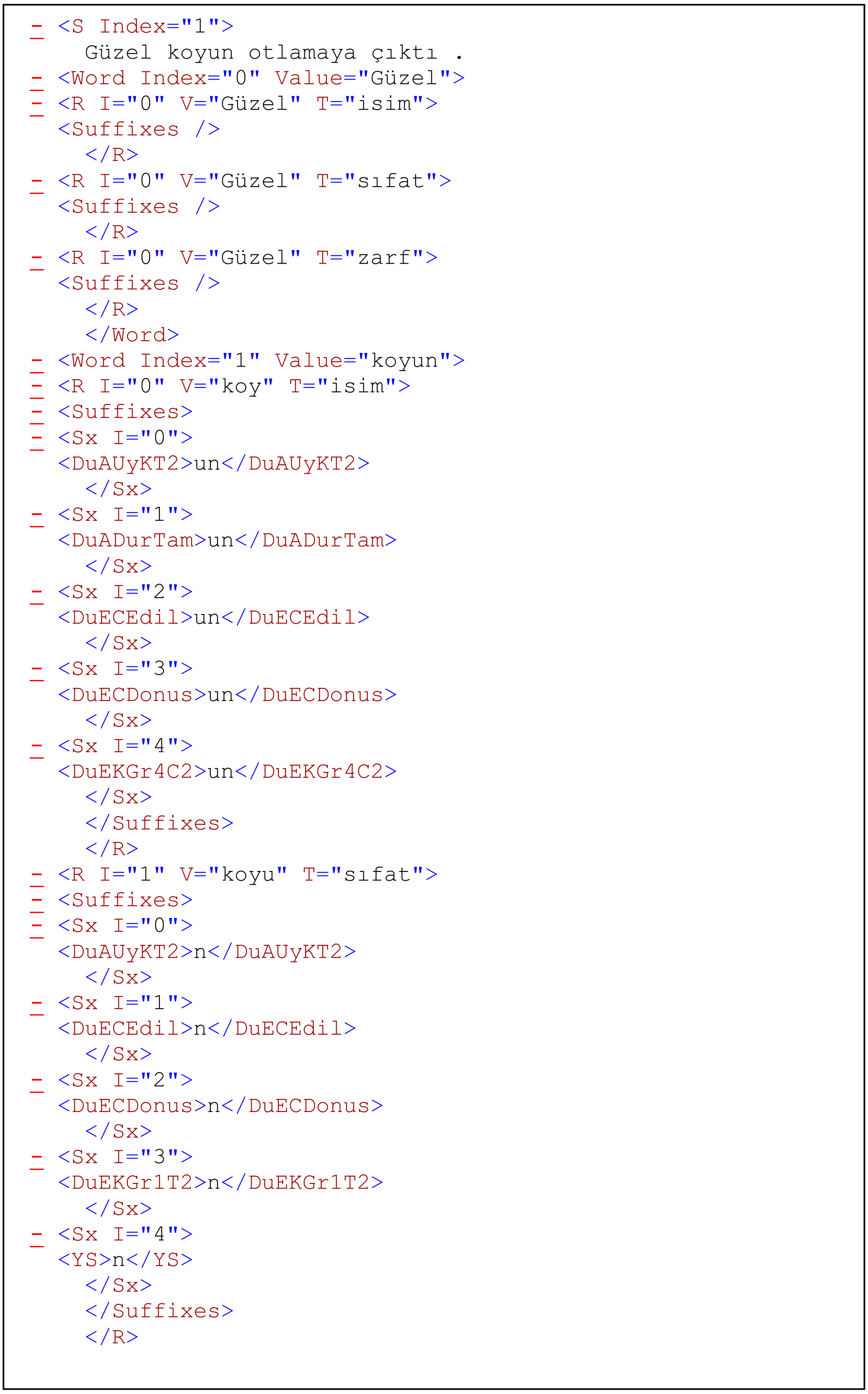


```
- <Sx I="5">
  <YtuAdlMa>ma</YtuAdlMa>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="6">
  <YtuUMa>ma</YtuUMa>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="7">
  <TBEA-ma>ma</TBEA-ma>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="8">
  <DuEOlz>ma</DuEOlz>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
    </Suffixes>
    </R>
- <R I="3" V="otlama" T="isim">
- <Suffixes>
- <Sx I="0">
  <TBEA-ay>y</TBEA-ay>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="1">
  <YS>y</YS>
  <TBEA-a>a</TBEA-a>
    </Sx>
- <Sx I="2">
  <TBEA-ay>y</TBEA-ay>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="3">
  <TBAA-ay>y</TBAA-ay>
  <DuADurYon>a</DuADurYon>
    </Sx>
- <Sx I="4">
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="3" Value="çıktı">
- <R I="0" V="çık" T="fiil">
- <Suffixes>
- <Sx I="0">
  <KDi>tı</KDi>
    </Sx>
```

APPENDIX C.4.2.2 (Cont'd.)

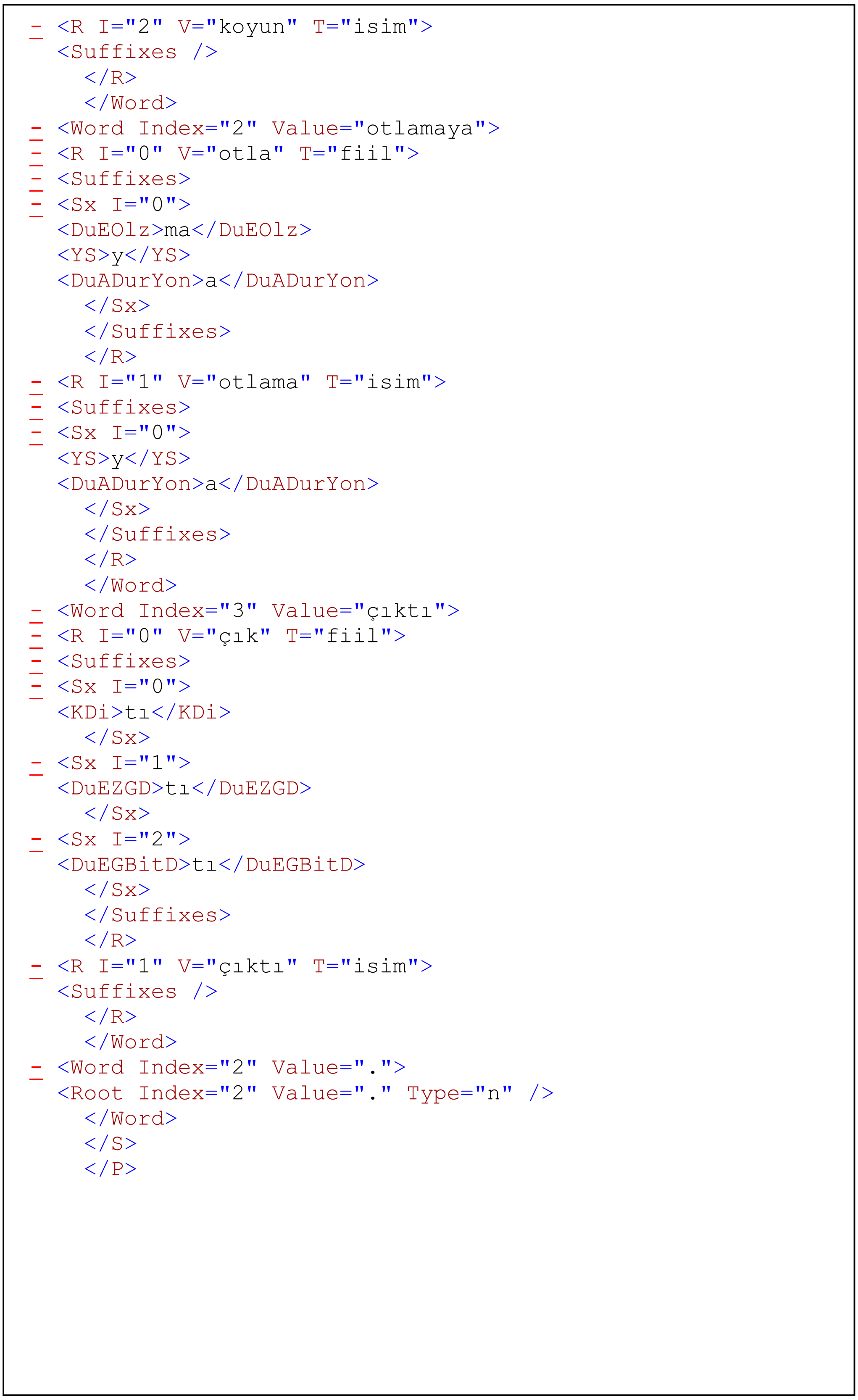
```
- <Sx I="1">
  <YtuUDI1>tı</YtuUDI1>
    </Sx>
- <Sx I="2">
  <TBEA-di>tı</TBEA-di>
    </Sx>
- <Sx I="3">
  <DuEZGD>tı</DuEZGD>
    </Sx>
- <Sx I="4">
  <DuEGBitD>tı</DuEGBitD>
    </Sx>
- <Sx I="5">
  <TBEA-ti>tı</TBEA-ti>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="çıktı" T="isim">
  <Suffixes />
    </R>
    </Word>
- <Word Index="2" Value=".">
  <Root Index="2" Value="." Type="n" />
    </Word>
    </S>
    </P>
    </File>
```

APPENDIX C.4.2.3 Eliminated Output (Sentence 2)



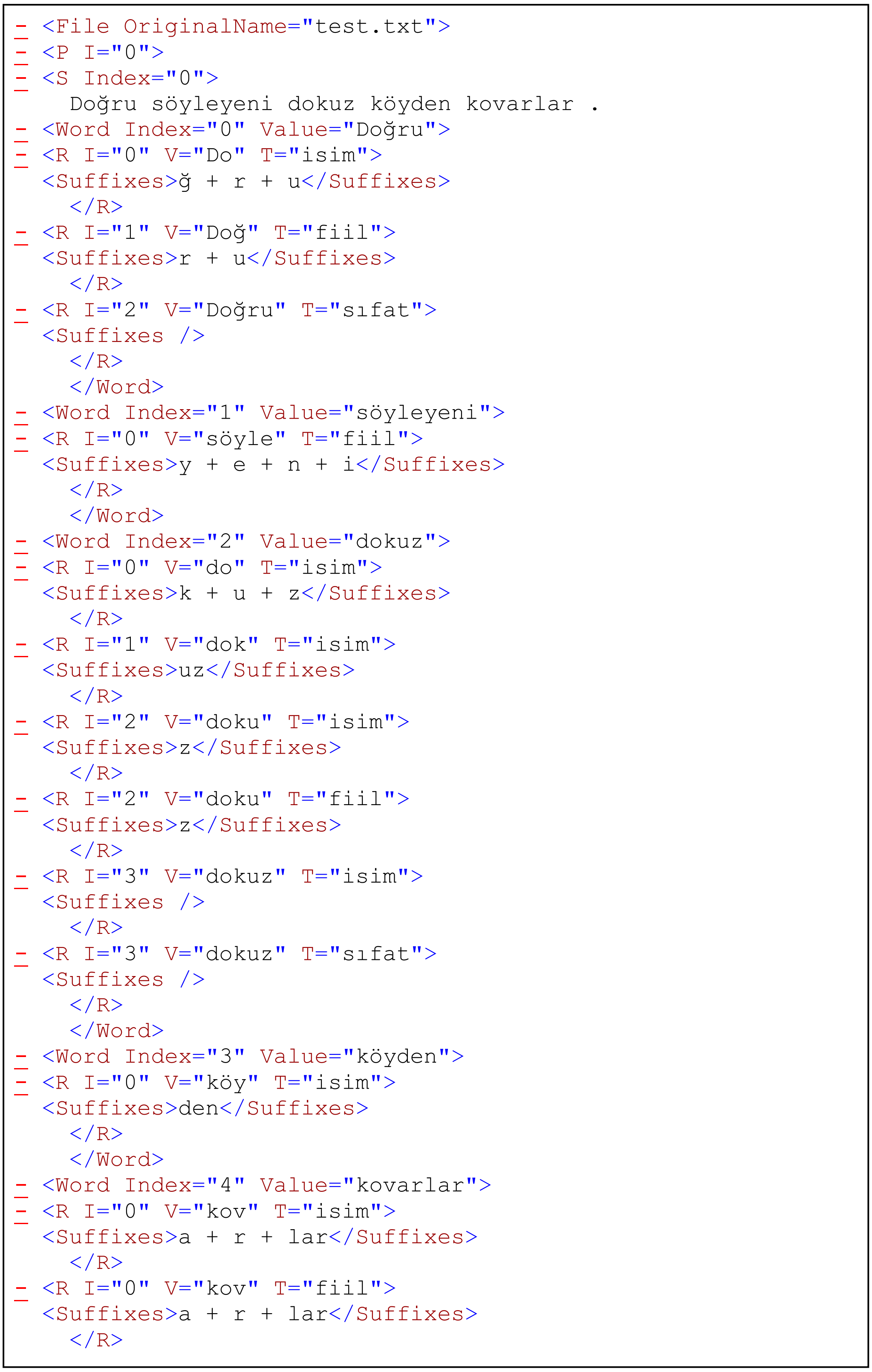
```
- <S Index="1">
    Güzel koyun otlamaya çıktı .
- <Word Index="0" Value="Güzel">
- <R I="0" V="Güzel" T="isim">
  <Suffixes />
    </R>
- <R I="0" V="Güzel" T="sıfat">
  <Suffixes />
    </R>
- <R I="0" V="Güzel" T="zarf">
  <Suffixes />
    </R>
    </Word>
- <Word Index="1" Value="koyun">
- <R I="0" V="koy" T="isim">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2>un</DuAUyKT2>
    </Sx>
- <Sx I="1">
  <DuADurTam>un</DuADurTam>
    </Sx>
- <Sx I="2">
  <DuECEdil>un</DuECEdil>
    </Sx>
- <Sx I="3">
  <DuECDonus>un</DuECDonus>
    </Sx>
- <Sx I="4">
  <DuEKGr4C2>un</DuEKGr4C2>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="koyu" T="sıfat">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2>n</DuAUyKT2>
    </Sx>
- <Sx I="1">
  <DuECEdil>n</DuECEdil>
    </Sx>
- <Sx I="2">
  <DuECDonus>n</DuECDonus>
    </Sx>
- <Sx I="3">
  <DuEKGr1T2>n</DuEKGr1T2>
    </Sx>
- <Sx I="4">
  <YS>n</YS>
    </Sx>
    </Suffixes>
    </R>
```

APPENDIX C.4.2.3 (Cont'd.)



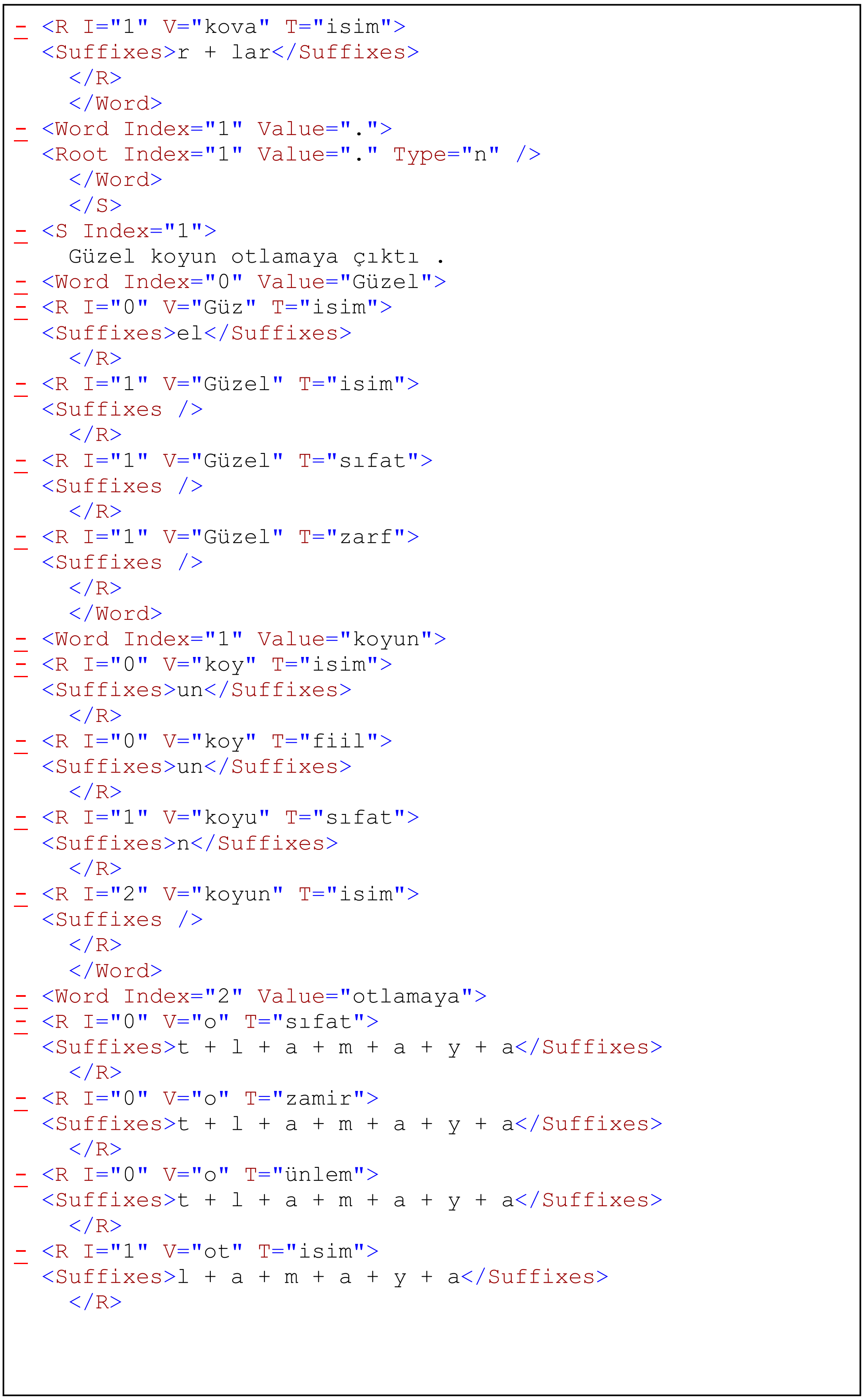
```
- <R I="2" V="koyun" T="isim">
  <Suffixes />
    </R>
    </Word>
- <Word Index="2" Value="otlamaya">
- <R I="0" V="otla" T="fiil">
- <Suffixes>
- <Sx I="0">
  <DuEOlz>ma</DuEOlz>
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="otlama" T="isim">
- <Suffixes>
- <Sx I="0">
  <YS>y</YS>
  <DuADurYon>a</DuADurYon>
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="3" Value="çıktı">
- <R I="0" V="çık" T="fiil">
- <Suffixes>
- <Sx I="0">
  <KDi>tı</KDi>
    </Sx>
- <Sx I="1">
  <DuEZGD>tı</DuEZGD>
    </Sx>
- <Sx I="2">
  <DuEGBitD>tı</DuEGBitD>
    </Sx>
    </Suffixes>
    </R>
- <R I="1" V="çıktı" T="isim">
  <Suffixes />
    </R>
    </Word>
- <Word Index="2" Value=".">
  <Root Index="2" Value="." Type="n" />
    </Word>
    </S>
    </P>
```

APPENDIX C.4.2.4 Suffixes Not Tagged



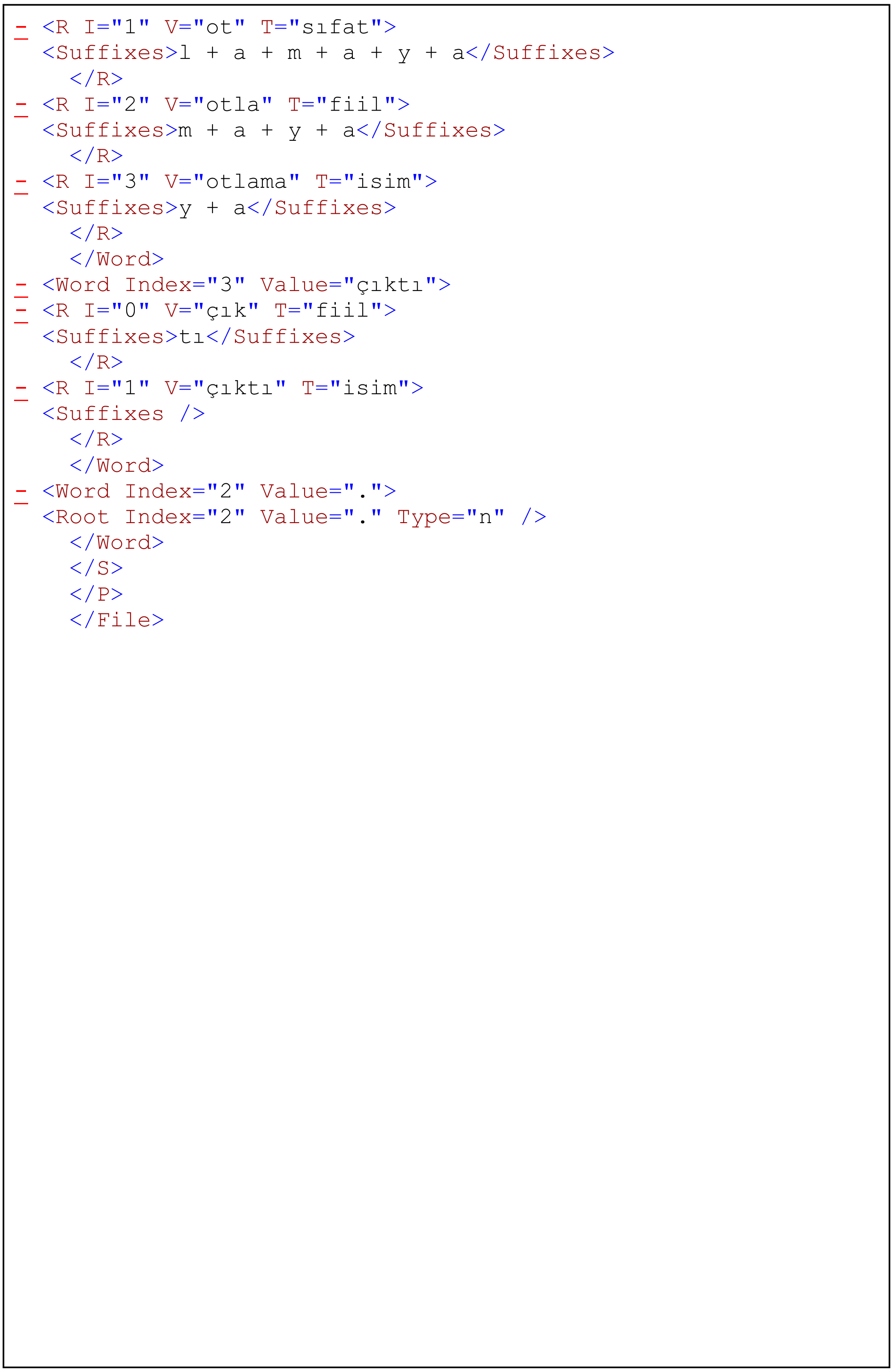
```
- <File OriginalName="test.txt">
- <P I="0">
- <S Index="0">
    Doğru söyleyeni dokuz köyden kovarlar .
- <Word Index="0" Value="Doğru">
- <R I="0" V="Do" T="isim">
  <Suffixes>ğ + r + u</Suffixes>
    </R>
- <R I="1" V="Doğ" T="fiil">
  <Suffixes>r + u</Suffixes>
    </R>
- <R I="2" V="Doğru" T="sıfat">
  <Suffixes />
    </R>
    </Word>
- <Word Index="1" Value="söyleyeni">
- <R I="0" V="söyle" T="fiil">
  <Suffixes>y + e + n + i</Suffixes>
    </R>
    </Word>
- <Word Index="2" Value="dokuz">
- <R I="0" V="do" T="isim">
  <Suffixes>k + u + z</Suffixes>
    </R>
- <R I="1" V="dok" T="isim">
  <Suffixes>uz</Suffixes>
    </R>
- <R I="2" V="doku" T="isim">
  <Suffixes>z</Suffixes>
    </R>
- <R I="2" V="doku" T="fiil">
  <Suffixes>z</Suffixes>
    </R>
- <R I="3" V="dokuz" T="isim">
  <Suffixes />
    </R>
- <R I="3" V="dokuz" T="sıfat">
  <Suffixes />
    </R>
    </Word>
- <Word Index="3" Value="köyden">
- <R I="0" V="köy" T="isim">
  <Suffixes>den</Suffixes>
    </R>
    </Word>
- <Word Index="4" Value="kovarlar">
- <R I="0" V="kov" T="isim">
  <Suffixes>a + r + lar</Suffixes>
    </R>
- <R I="0" V="kov" T="fiil">
  <Suffixes>a + r + lar</Suffixes>
    </R>
```

APPENDIX C.4.2.4 (Cont'd.)



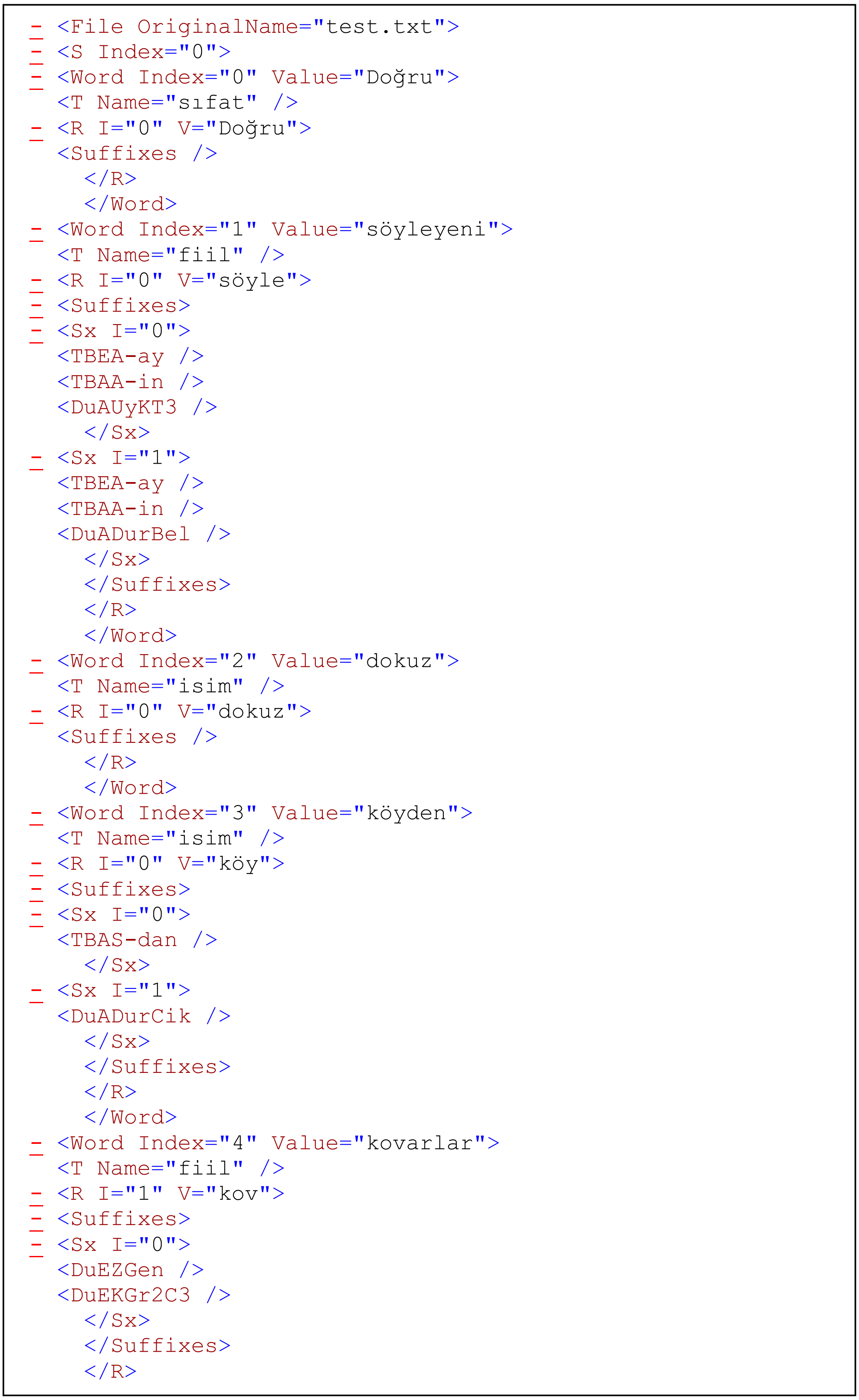
```
- <R I="1" V="kova" T="isim">
  <Suffixes>r + lar</Suffixes>
   </R>
   </Word>
- <Word Index="1" Value=".">
  <Root Index="1" Value="." Type="n" />
   </Word>
   </S>
- <S Index="1">
   Güzel koyun otlamaya çıktı .
- <Word Index="0" Value="Güzel">
- <R I="0" V="Güz" T="isim">
  <Suffixes>el</Suffixes>
   </R>
- <R I="1" V="Güzel" T="isim">
  <Suffixes />
   </R>
- <R I="1" V="Güzel" T="sıfat">
  <Suffixes />
   </R>
- <R I="1" V="Güzel" T="zarf">
  <Suffixes />
   </R>
   </Word>
- <Word Index="1" Value="koyun">
- <R I="0" V="koy" T="isim">
  <Suffixes>un</Suffixes>
   </R>
- <R I="0" V="koy" T="fiil">
  <Suffixes>un</Suffixes>
   </R>
- <R I="1" V="koyu" T="sıfat">
  <Suffixes>n</Suffixes>
   </R>
- <R I="2" V="koyun" T="isim">
  <Suffixes />
   </R>
   </Word>
- <Word Index="2" Value="otlamaya">
- <R I="0" V="o" T="sıfat">
  <Suffixes>t + l + a + m + a + y + a</Suffixes>
   </R>
- <R I="0" V="o" T="zamir">
  <Suffixes>t + l + a + m + a + y + a</Suffixes>
   </R>
- <R I="0" V="o" T="ünlem">
  <Suffixes>t + l + a + m + a + y + a</Suffixes>
   </R>
- <R I="1" V="ot" T="isim">
  <Suffixes>l + a + m + a + y + a</Suffixes>
   </R>
```

APPENDIX C.4.2.4 (Cont'd.)



```
- <R I="1" V="ot" T="sıfat">
  <Suffixes>l + a + m + a + y + a</Suffixes>
    </R>
- <R I="2" V="otla" T="fiil">
  <Suffixes>m + a + y + a</Suffixes>
    </R>
- <R I="3" V="otlama" T="isim">
  <Suffixes>y + a</Suffixes>
    </R>
    </Word>
- <Word Index="3" Value="çıktı">
- <R I="0" V="çık" T="fiil">
  <Suffixes>tı</Suffixes>
    </R>
- <R I="1" V="çıktı" T="isim">
  <Suffixes />
    </R>
    </Word>
- <Word Index="2" Value=".">
  <Root Index="2" Value="." Type="n" />
    </Word>
    </S>
    </P>
    </File>
```

APPENDIX C.4.3 POS Tagging Module



```
- <File OriginalName="test.txt">
- <S Index="0">
- <Word Index="0" Value="Doğru">
  <T Name="sıfat" />
- <R I="0" V="Doğru">
  <Suffixes />
    </R>
    </Word>
- <Word Index="1" Value="söyleyeni">
  <T Name="fiil" />
- <R I="0" V="söyle">
- <Suffixes>
- <Sx I="0">
  <TBEA-ay />
  <TBAA-in />
  <DuAUyKT3 />
    </Sx>
- <Sx I="1">
  <TBEA-ay />
  <TBAA-in />
  <DuADurBel />
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="2" Value="dokuz">
  <T Name="isim" />
- <R I="0" V="dokuz">
  <Suffixes />
    </R>
    </Word>
- <Word Index="3" Value="köyden">
  <T Name="isim" />
- <R I="0" V="köy">
- <Suffixes>
- <Sx I="0">
  <TBAS-dan />
    </Sx>
- <Sx I="1">
  <DuADurCik />
    </Sx>
    </Suffixes>
    </R>
    </Word>
- <Word Index="4" Value="kovarlar">
  <T Name="fiil" />
- <R I="1" V="kov">
- <Suffixes>
- <Sx I="0">
  <DuEZGen />
  <DuEKGr2C3 />
    </Sx>
    </Suffixes>
    </R>
```

APPENDIX C.4.3 (Cont'd.)

```
  <T Name="isim" />
- <R I="0" V="kov">
- <Suffixes>
- <Sx I="0">
  <TBAE-r />
  <DuASayC />
    </Sx>
    </Suffixes>
    </R>
- <R I="2" V="kova">
- <Suffixes>
- <Sx I="0">
  <TBAE-r />
  <DuASayC />
    </Sx>
    </Suffixes>
    </R>
    </Word>
  <Word Index="1" Value="." />
    </S>
- <S Index="1">
- <Word Index="0" Value="Güzel">
  <T Name="isim" />
- <R I="0" V="Güzel">
  <Suffixes />
    </R>
    </Word>
- <Word Index="1" Value="koyun">
  <T Name="isim" />
- <R I="0" V="koy">
- <Suffixes>
- <Sx I="0">
  <DuAUyKT2 />
    </Sx>
- <Sx I="1">
  <DuADurTam />
    </Sx>
    </Suffixes>
    </R>
- <R I="2" V="koyun">
  <Suffixes />
    </R>
    </Word>
- <Word Index="2" Value="otlamaya">
  <T Name="isim" />
- <R I="1" V="otlama">
- <Suffixes>
- <Sx I="0">
  <TBAA-ay />
  <DuADurYon />
    </Sx>
- <Sx I="1">
  <YS />
  <DuADurYon />
    </Sx>
```

APPENDIX C.4.3 (Cont'd.)

```
    </R>
    </Word>
- <Word Index="3" Value="çıktı">
  <T Name="fiil" />
- <R I="0" V="çık">
- <Suffixes>
- <Sx I="0">
  <KDi />
    </Sx>
- <Sx I="1">
  <DuEZGD />
    </Sx>
- <Sx I="2">
  <DuEGBitD />
    </Sx>
    </Suffixes>
    </R>
    </Word>
  <Word Index="2" Value="." />
    </S>
    </File>
```

APPENDIX C.4.4 Sample Output 2

Text File:

```
Gelecek yılın müfredatı hazırlandı.
```

Output of “*Sentence Boundary Detection*” Module:

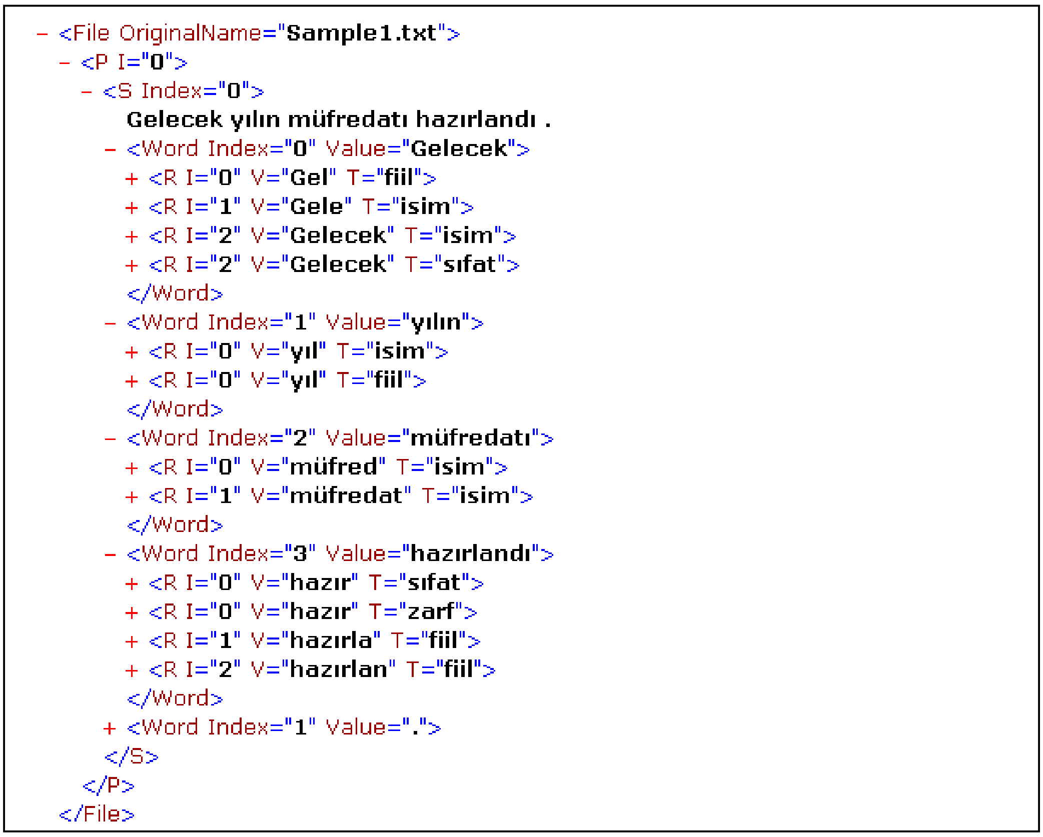
```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <F N="sample2.txt">
      - <P I="0">
              <S I="0">Gelecek yılın müfredatı hazırlandı .</S>
        </P>
  </F>
```

Output of “*Sentence Boundary Detection – with words*” Module (Input of the “Finding Stem/Root” Module):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <File OriginalName="sample2.txt">
      - <P I="0">
            - <S Index="0">
              Gelecek yılın müfredatı hazırlandı .
              <Word Index="0">Gelecek</Word>
              <Word Index="1">yılın</Word>
              <Word Index="2">müfredatı</Word>
              <Word Index="3">hazırlandı</Word>
              <Word Index="1">.</Word>
              </S>
        </P>
  </File>
```

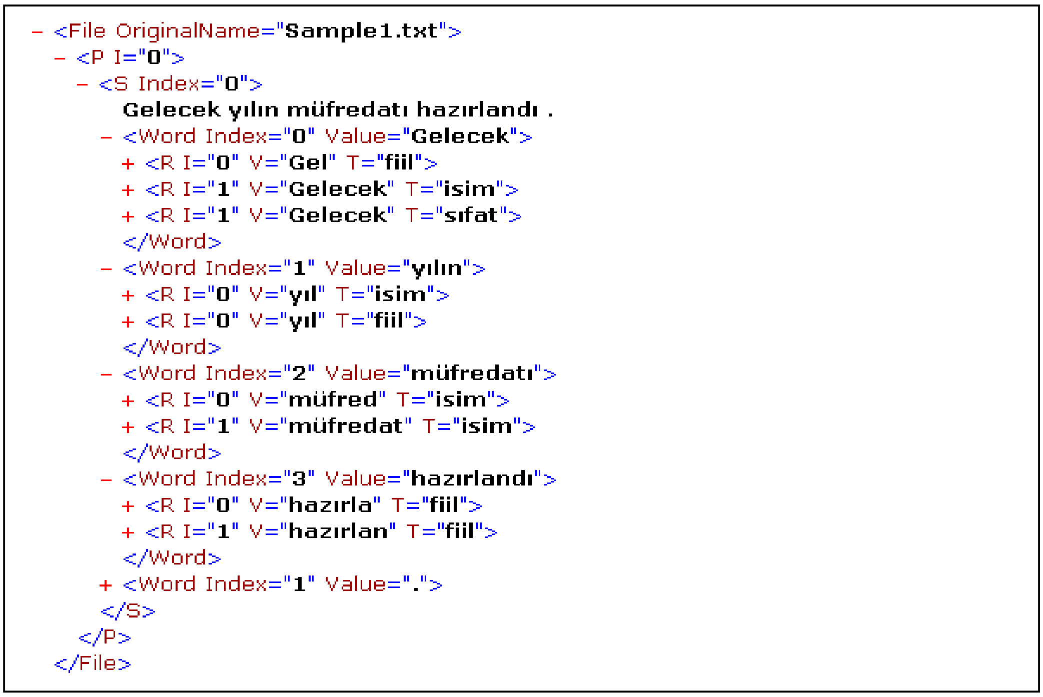
APPENDIX C.4.4 (Cont’d.)

Output of “*Finding Roots*” Module:



```
- <File OriginalName="Sample1.txt">
  - <P I="0">
    - <S Index="0">
        Gelecek yılın müfredatı hazırlandı .
      - <Word Index="0" Value="Gelecek">
        + <R I="0" V="Gel" T="fiil">
        + <R I="1" V="Gele" T="isim">
        + <R I="2" V="Gelecek" T="isim">
        + <R I="2" V="Gelecek" T="sıfat">
        </Word>
      - <Word Index="1" Value="yılın">
        + <R I="0" V="yıl" T="isim">
        + <R I="0" V="yıl" T="fiil">
        </Word>
      - <Word Index="2" Value="müfredatı">
        + <R I="0" V="müfred" T="isim">
        + <R I="1" V="müfredat" T="isim">
        </Word>
      - <Word Index="3" Value="hazırlandı">
        + <R I="0" V="hazır" T="sıfat">
        + <R I="0" V="hazır" T="zarf">
        + <R I="1" V="hazırla" T="fiil">
        + <R I="2" V="hazırlan" T="fiil">
        </Word>
      + <Word Index="1" Value=".">
      </S>
    </P>
  </File>
```

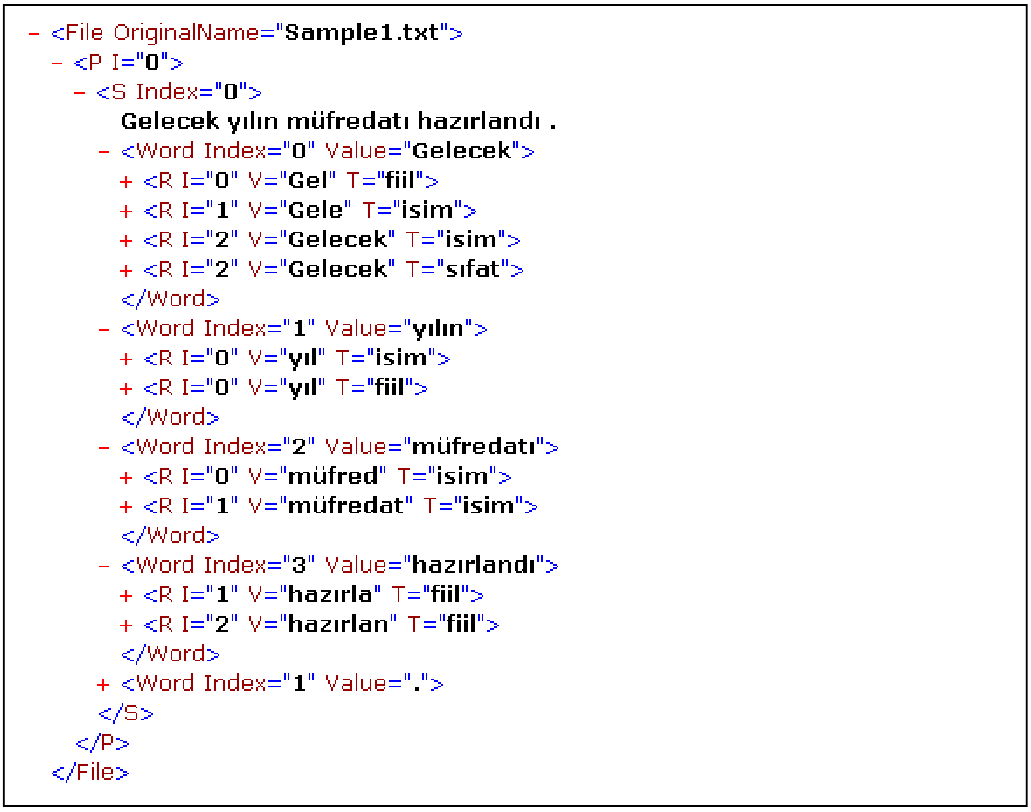
Output of “*Finding Stems*” Module:



```
- <File OriginalName="Sample1.txt">
  - <P I="0">
    - <S Index="0">
        Gelecek yılın müfredatı hazırlandı .
      - <Word Index="0" Value="Gelecek">
        + <R I="0" V="Gel" T="fiil">
        + <R I="1" V="Gelecek" T="isim">
        + <R I="1" V="Gelecek" T="sıfat">
        </Word>
      - <Word Index="1" Value="yılın">
        + <R I="0" V="yıl" T="isim">
        + <R I="0" V="yıl" T="fiil">
        </Word>
      - <Word Index="2" Value="müfredatı">
        + <R I="0" V="müfred" T="isim">
        + <R I="1" V="müfredat" T="isim">
        </Word>
      - <Word Index="3" Value="hazırlandı">
        + <R I="0" V="hazırla" T="fiil">
        + <R I="1" V="hazırlan" T="fiil">
        </Word>
      + <Word Index="1" Value=".">
      </S>
    </P>
  </File>
```

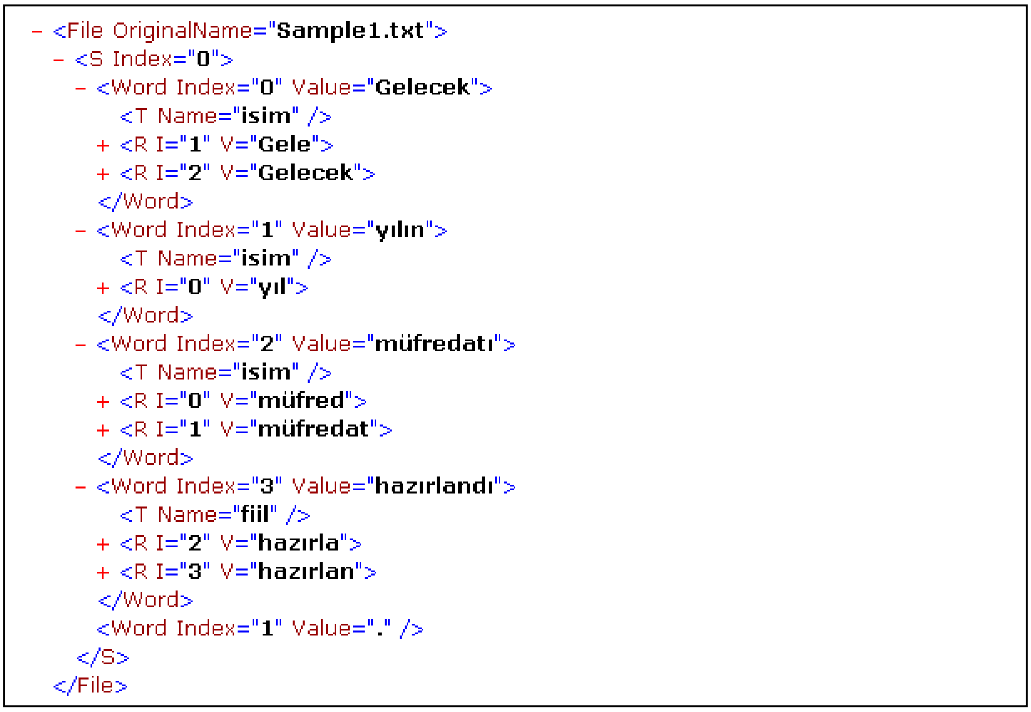
APPENDIX C.4.4 (Cont'd.)

Output of "*Eliminating Root/Stem and Suffixes according to the Type of Suffixes*" Module:



```
- <File OriginalName="Sample1.txt">
  - <P I="0">
    - <S Index="0">
        Gelecek yılın müfredatı hazırlandı .
      - <Word Index="0" Value="Gelecek">
        + <R I="0" V="Gel" T="fiil">
        + <R I="1" V="Gele" T="isim">
        + <R I="2" V="Gelecek" T="isim">
        + <R I="2" V="Gelecek" T="sıfat">
        </Word>
      - <Word Index="1" Value="yılın">
        + <R I="0" V="yıl" T="isim">
        + <R I="0" V="yıl" T="fiil">
        </Word>
      - <Word Index="2" Value="müfredatı">
        + <R I="0" V="müfred" T="isim">
        + <R I="1" V="müfredat" T="isim">
        </Word>
      - <Word Index="3" Value="hazırlandı">
        + <R I="1" V="hazırla" T="fiil">
        + <R I="2" V="hazırlan" T="fiil">
        </Word>
      + <Word Index="1" Value=".">
      </S>
    </P>
  </File>
```

Output of "POS Tagging" Module:



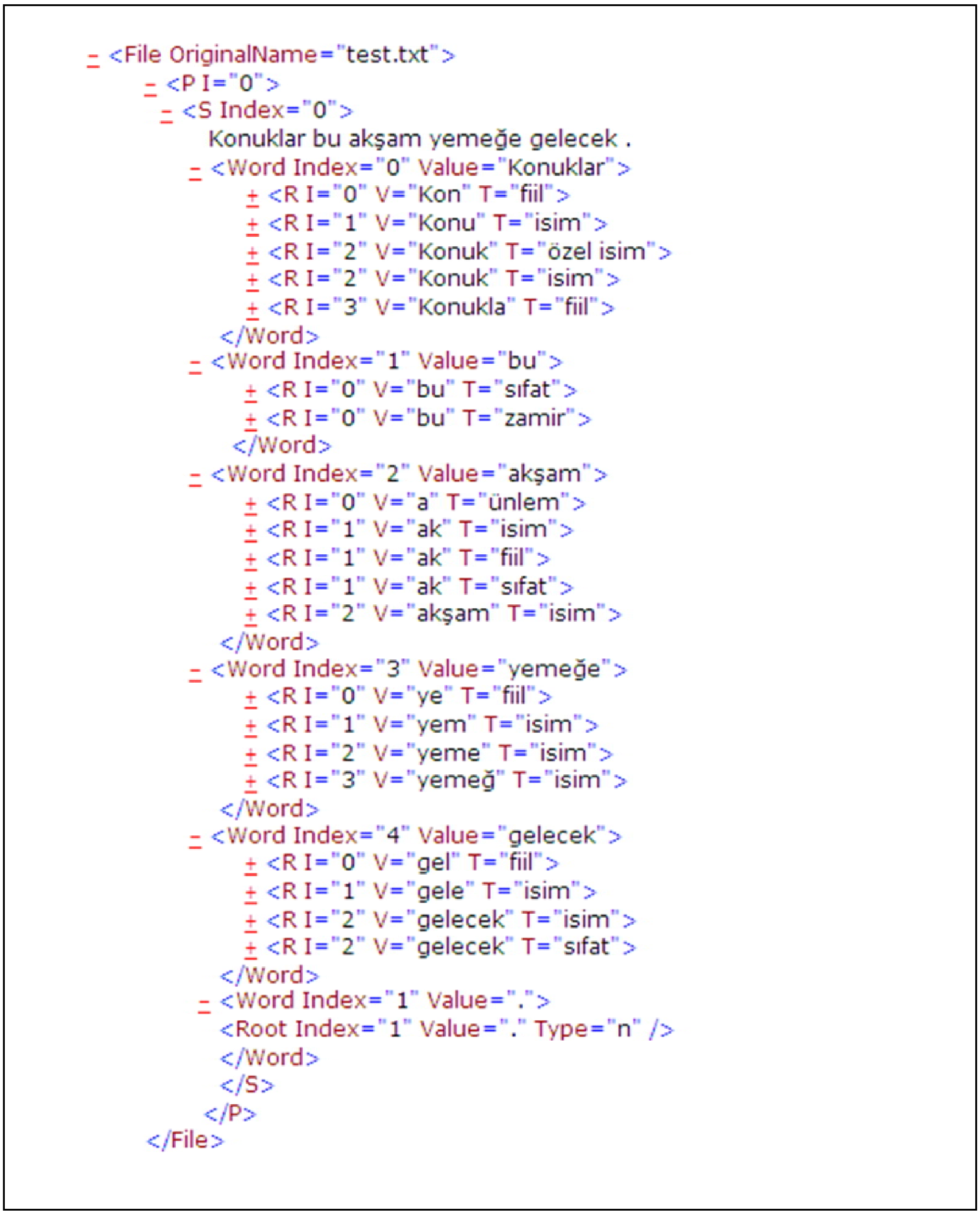
```
- <File OriginalName="Sample1.txt">
  - <S Index="0">
    - <Word Index="0" Value="Gelecek">
        <T Name="isim" />
      + <R I="1" V="Gele">
      + <R I="2" V="Gelecek">
      </Word>
    - <Word Index="1" Value="yılın">
        <T Name="isim" />
      + <R I="0" V="yıl">
      </Word>
    - <Word Index="2" Value="müfredatı">
        <T Name="isim" />
      + <R I="0" V="müfred">
      + <R I="1" V="müfredat">
      </Word>
    - <Word Index="3" Value="hazırlandı">
        <T Name="fiil" />
      + <R I="2" V="hazırla">
      + <R I="3" V="hazırlan">
      </Word>
      <Word Index="1" Value="." />
    </S>
  </File>
```

APPENDIX C.4.5 Sample Output 3

Text File:

```
Konuklar bu akşam yemeğe gelecek.
```

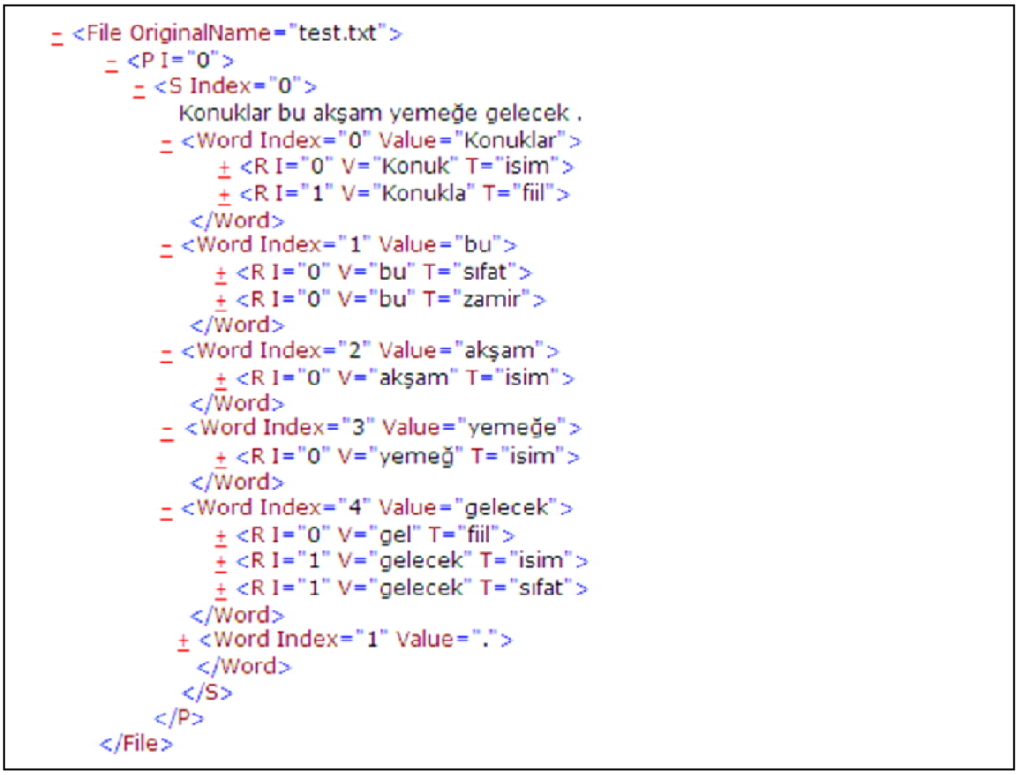
Output of "*Finding All Roots, Stems and Suffixes*" Module:



```
- <File OriginalName="test.txt">
   - <P I="0">
     - <S Index="0">
         Konuklar bu akşam yemeğe gelecek .
       - <Word Index="0" Value="Konuklar">
           + <R I="0" V="Kon" T="fiil">
           + <R I="1" V="Konu" T="isim">
           + <R I="2" V="Konuk" T="özel isim">
           + <R I="2" V="Konuk" T="isim">
           + <R I="3" V="Konukla" T="fiil">
         </Word>
       - <Word Index="1" Value="bu">
           + <R I="0" V="bu" T="sıfat">
           + <R I="0" V="bu" T="zamir">
           </Word>
       - <Word Index="2" Value="akşam">
           + <R I="0" V="a" T="ünlem">
           + <R I="1" V="ak" T="isim">
           + <R I="1" V="ak" T="fiil">
           + <R I="1" V="ak" T="sıfat">
           + <R I="2" V="akşam" T="isim">
         </Word>
       - <Word Index="3" Value="yemeğe">
           + <R I="0" V="ye" T="fiil">
           + <R I="1" V="yem" T="isim">
           + <R I="2" V="yeme" T="isim">
           + <R I="3" V="yemeğ" T="isim">
         </Word>
       - <Word Index="4" Value="gelecek">
           + <R I="0" V="gel" T="fiil">
           + <R I="1" V="gele" T="isim">
           + <R I="2" V="gelecek" T="isim">
           + <R I="2" V="gelecek" T="sıfat">
         </Word>
        - <Word Index="1" Value=".">
          <Root Index="1" Value="." Type="n" />
          </Word>
          </S>
        </P>
   </File>
```

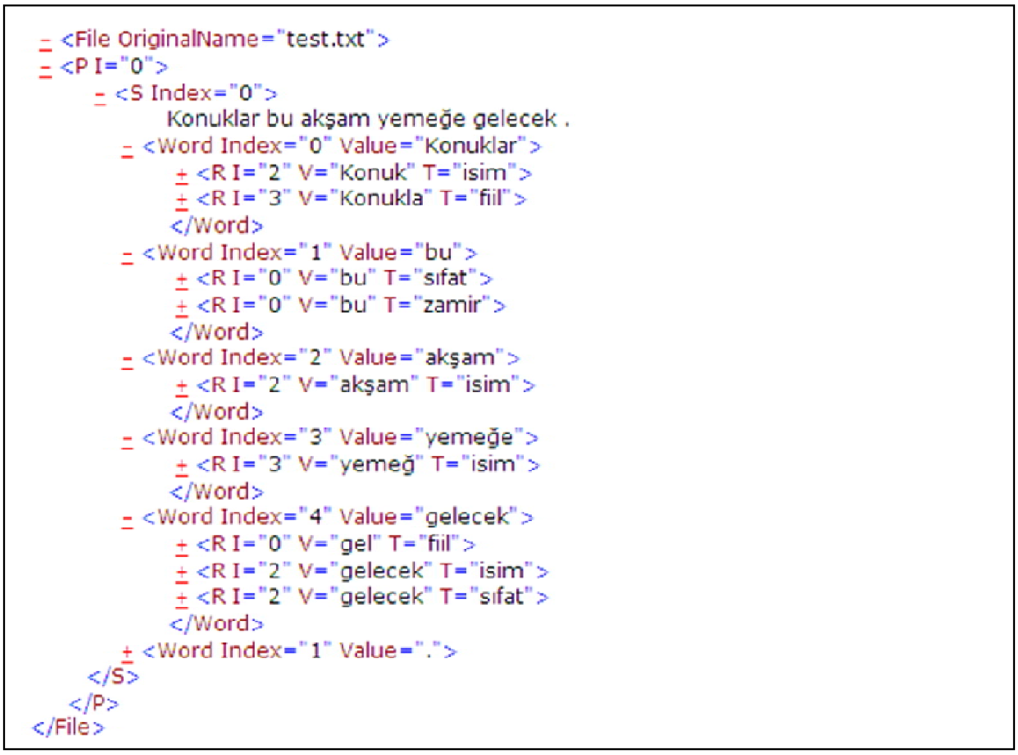
APPENDIX C.4.5 (Cont’d.)

Output of “*Finding Stems and Suffixes*” Module:



```
- <File OriginalName="test.txt">
   - <P I="0">
     - <S Index="0">
         Konuklar bu akşam yemeğe gelecek .
       - <Word Index="0" Value="Konuklar">
           + <R I="0" V="Konuk" T="isim">
           + <R I="1" V="Konukla" T="fiil">
         </Word>
       - <Word Index="1" Value="bu">
           + <R I="0" V="bu" T="sıfat">
           + <R I="0" V="bu" T="zamir">
         </Word>
       - <Word Index="2" Value="akşam">
           + <R I="0" V="akşam" T="isim">
         </Word>
       - <Word Index="3" Value="yemeğe">
           + <R I="0" V="yemeğ" T="isim">
         </Word>
       - <Word Index="4" Value="gelecek">
           + <R I="0" V="gel" T="fiil">
           + <R I="1" V="gelecek" T="isim">
           + <R I="1" V="gelecek" T="sıfat">
         </Word>
        + <Word Index="1" Value=".">
          </Word>
        </S>
      </P>
  </File>
```

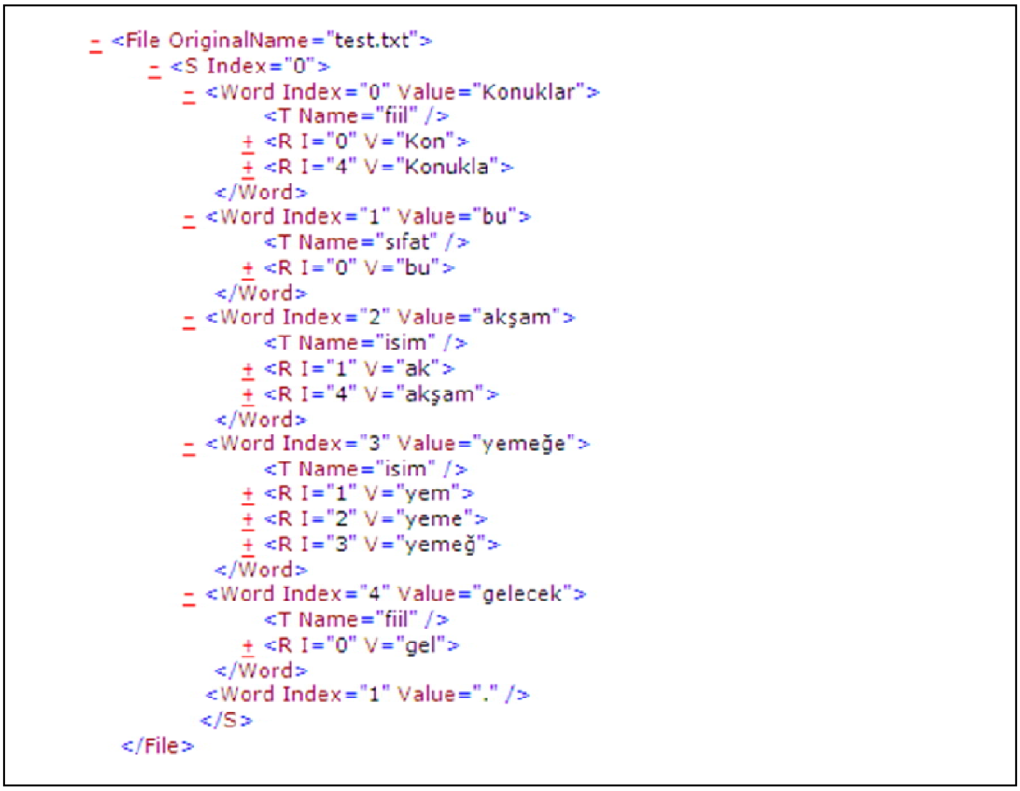
Output of “*Eliminating Roots and Suffixes according to the Type of Suffixes*” Module:



```
- <File OriginalName="test.txt">
- <P I="0">
    - <S Index="0">
        Konuklar bu akşam yemeğe gelecek .
      - <Word Index="0" Value="Konuklar">
          + <R I="2" V="Konuk" T="isim">
          + <R I="3" V="Konukla" T="fiil">
          </Word>
      - <Word Index="1" Value="bu">
          + <R I="0" V="bu" T="sıfat">
          + <R I="0" V="bu" T="zamir">
          </Word>
      - <Word Index="2" Value="akşam">
          + <R I="2" V="akşam" T="isim">
          </Word>
      - <Word Index="3" Value="yemeğe">
          + <R I="3" V="yemeğ" T="isim">
          </Word>
      - <Word Index="4" Value="gelecek">
          + <R I="0" V="gel" T="fiil">
          + <R I="2" V="gelecek" T="isim">
          + <R I="2" V="gelecek" T="sıfat">
          </Word>
      + <Word Index="1" Value=".">
    </S>
  </P>
</File>
```

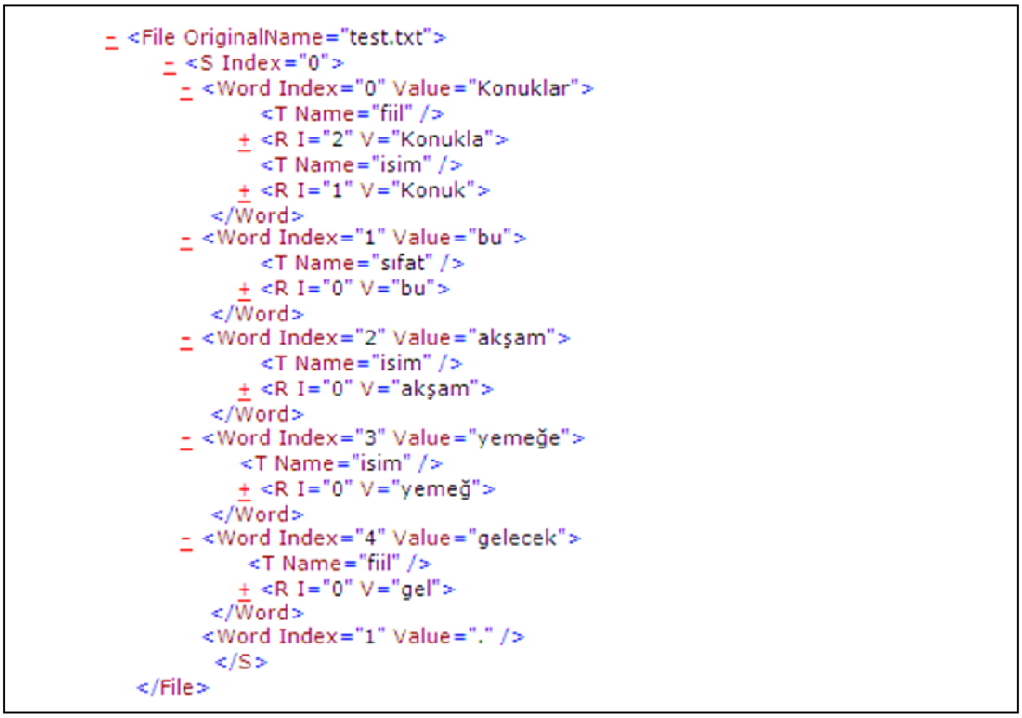
APPENDIX C.4.5 (Cont'd.)

Output of "*POS Tagging*" Module (input all roots, stems and suffixes):



```
- <File OriginalName="test.txt">
  - <S Index="0">
    - <Word Index="0" Value="Konuklar">
        <T Name="fiil" />
      + <R I="0" V="Kon">
      + <R I="4" V="Konukla">
      </Word>
    - <Word Index="1" Value="bu">
        <T Name="sıfat" />
      + <R I="0" V="bu">
      </Word>
    - <Word Index="2" Value="akşam">
        <T Name="isim" />
      + <R I="1" V="ak">
      + <R I="4" V="akşam">
      </Word>
    - <Word Index="3" Value="yemeğe">
        <T Name="isim" />
      + <R I="1" V="yem">
      + <R I="2" V="yeme">
      + <R I="3" V="yemeğ">
      </Word>
    - <Word Index="4" Value="gelecek">
        <T Name="fiil" />
      + <R I="0" V="gel">
      </Word>
      <Word Index="1" Value="." />
    </S>
  </File>
```

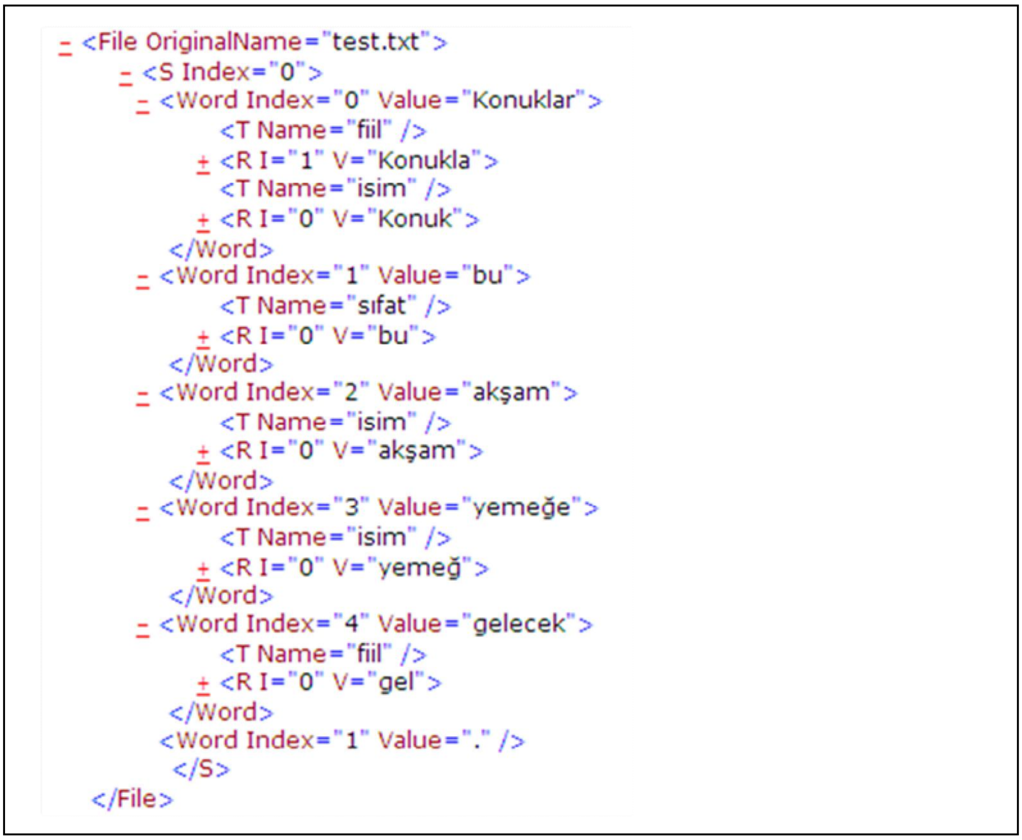
Output of "*POS Tagging*" Module (input all stems and suffixes):



```
- <File OriginalName="test.txt">
  - <S Index="0">
    - <Word Index="0" Value="Konuklar">
        <T Name="fiil" />
      + <R I="2" V="Konukla">
        <T Name="isim" />
      + <R I="1" V="Konuk">
      </Word>
    - <Word Index="1" Value="bu">
        <T Name="sıfat" />
      + <R I="0" V="bu">
      </Word>
    - <Word Index="2" Value="akşam">
        <T Name="isim" />
      + <R I="0" V="akşam">
      </Word>
    - <Word Index="3" Value="yemeğe">
        <T Name="isim" />
      + <R I="0" V="yemeğ">
      </Word>
    - <Word Index="4" Value="gelecek">
        <T Name="fiil" />
      + <R I="0" V="gel">
      </Word>
      <Word Index="1" Value="." />
    </S>
  </File>
```

APPENDIX C.4.5 (Cont’d.)

Output of “*POS Tagging*” Module (input all roots, stems and suffixes- eliminated suffixes types):



```
- <File OriginalName="test.txt">
   - <S Index="0">
     - <Word Index="0" Value="Konuklar">
          <T Name="fiil" />
        + <R I="1" V="Konukla">
          <T Name="isim" />
        + <R I="0" V="Konuk">
       </Word>
     - <Word Index="1" Value="bu">
          <T Name="sıfat" />
        + <R I="0" V="bu">
       </Word>
     - <Word Index="2" Value="akşam">
          <T Name="isim" />
        + <R I="0" V="akşam">
       </Word>
     - <Word Index="3" Value="yemeğe">
          <T Name="isim" />
        + <R I="0" V="yemeğ">
       </Word>
     - <Word Index="4" Value="gelecek">
          <T Name="fiil" />
        + <R I="0" V="gel">
       </Word>
       <Word Index="1" Value="." />
     </S>
  </File>
```

APPENDIX D Metadata of Documents

Metadata	Expression	Document Type
Document Variety	Variety of the stored document: • Ekonomi • Siyaset	Newspaper Book Report Magazine Parliamentary Report Official Gazette
Header	The header of the document	
Publication Date	The date that the document is published	
Writing Date	The date that the document is written	
Publisher Name		
Resource	The URL address of the document	
Size	Size of the document in bytes	
Number of Pages		
File Address	Document address on disk	
Number of Wordforms		
Publication Frequency		Newspaper Report Magazine
Authors	The name of authors	Book
ISBN	ISBN number of the book	
Publishing Number		
Session	The session number of report	Parliamentary Report
Time	The time of session	
Period	The time period of report	
Report Year	The year of report	
Meeting	The number of meeting	