DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ROOT-SUFFIX SEPERATION OF TURKISH WORDS

by

Çağdaş Can BİRANT

January, 2008
İZMİR

ROOT-SUFFIX SEPERATION OF TURKISH WORDS

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

by

Çağdaş Can BİRANT

January, 2008
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"ROOT-SUFFIX SEPERATION OF TURKISH WORDS"** completed by **ÇAĞDAŞ CAN BİRANT** under supervision of **ASSOCIATIVE PROFESSOR DR. YALÇIN ÇEBİ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Assoc. Prof. Dr. Yalçın ÇEBİ
Supervisor

.....

(Jury Member)

.....

(Jury Member)

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

ACKNOWLEDGEMENTS

I would like to thank my advisor Assoc. Prof. Dr. Yalçın ÇEBİ for offering me to study in Natural Language Processing and for his advises, support and endless help to complete my thesis.

I would also like to thank Özlem AKTAŞ, who shared her ideas about "Natural Language Processing" during the writing and developing phases of thesis, for her life coaching; and all my friends, especially Emrah GEZER, Medar ŞİMŞEK, Alpar BAYINDIRLI and Ozan FIRAT, who encouraged me during the writing of the thesis.

I have special thanks to my parents, especially my mother Gönül DENİZERİ and my brother Dr. Kökten Ulaş BİRANT, for their endless support, patience and making me encouraged.

Çağdaş Can BİRANT

ROOT-SUFFIX SEPERATION OF TURKISH WORDS

ABSTRACT

Defining structure of a language and getting inferences depending on this structure supply obtaining information about that language. The applications in real world prepare an infrastructure, in which new decisions can be developed, by using the obtained information as new automatic methods in a large sample space involving a huge word source. This infrastructure should be based on a trustworthy skeleton and an integration of rules which depends on this skeleton's accuracy.

In this study, it is aimed to get the most absolute structure as possible as it can be, to prepare this structure as a module that works in high performance by not making concessions in its correctness, to make this module work with the projects implemented before and to be the base to the projects which will be done in future.

The work called "DEVELOPMENT OF A METHOD TO DETERMINE ROOT AND SUFFIXES FOR TURKISH WORDS TO GENERATE LARGE SCALE TURKISH CORPUS", which was developed by Özlem AKTAŞ, is accepted as the base in this work. Some structures on that project, whose ratio of the correctness is approximately %99, were standardized by making some revisions. Standardization works were done by using the XML structures which were used to send information between the modules, and also used as the result of the program that was run for any morphological work and the source of the rules to separate the suffixes correctly. The suffixes are allowed to be tagged as readable and efficient for performance by being abbreviated and also named according to some predefined rules systematically.

The results have formed more explicit step by step. The duration of process will be decreased by developing new modules in the future.

Keywords: Natural language processing, morphological analysis, Turkish, determining root and suffixes.

TÜRKÇE KELİMELERDE KÖK-EK AYRIMI

ÖZ

Bir dilin yapısını belirlemek ve bu yapıya bağlı çıkarımlarda bulunmak, o dil hakkında bilgi edinilmesini sağlar. Edinilen bilgilerin otomatize edilmiş methodlarla geniş bir kelime kaynağında denenerek gerçek hayata uygulanması ise yeni fikirlere ve ürünlere sağlam bir alt yapı oluşturmaktadır. Bu altyapının günümüzde kurulması ve genelgeçerliliğe sahip olması için temelinde sağlam bir iskelet ve iskelete bağlı kurallar bütünü bulunması gerekmektedir.

Bu çalışmanın amacı, oluşturulabilecek mutlak yapıyı ortaya çıkarmak, bu yapıyı doğruluk payından ödün vermeden yüksek performansla çalışan bir modül olarak hazırlamak ve bu modülün önceki çalışmalarla iş birliği içinde olacak ve gelecekte yapılacak çalışmalara temel olacak şekilde çalışmasını sağlamaktır.

Bu çalışma esnasında, Özlem AKTAŞ tarafından geliştirilen cümle sonu bulma algoritması temel alınmış ve doğruluk oranı %99'u bulan bu çalışmanın üzerinde yapılan değişikliklerle kullanılan bazı yapılar standart bir hale getirilmiştir. Bu standardizasyon çalışmaları, öncelikle modüller arası veri geçişini sağlayan XML yapıları üzerinde başlamıştır. Hedeflenen biçimbirimsel çalışma programının sonucu olarak kullanıcıya sunulan ve biçimbirimsel ayrıştırma sırasında ekler arası ardalanma kurallarını içeren dosyalarda XML yapısı kullanılmaktadır. Bu XML yapılarında yer alacak ek çeşitlerinin ise isimleri, belirlenen bir takım kurallar çerçevesinde sistematik olarak kısaltılarak kök ve eklerin etiketlenmesine imkân sağlamıştır.

Elde edilen sonuçlar aşama aşama daha net bir şekil almaya başlamıştır. İşlem süresi ise ileride geliştirilecek diğer modüller sayesinde daha da azalacaktır.

Anahtar sözcükler: Doğal dil işleme, biçimbilimsel analiz, Türkçe, kök ve eklerin belirlenmesi.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

Since the prehistoric ages, people tried to find a tool which helps them to communicate with each other. The common tool is “Natural Language”. NLP is the subject which helps the human to make the life easier by different spectrums and to develop new ideas for other branches of science. In research works, computer technology is used to get more precise and effective results for academic and commercial spaces.

NLP can be defined as a system which computes the dialogs of real life or the printed materials according to some criteria which are closer to human brain’s “understanding”. The word “understand” in this definition can be defined as; the observable behavior of the system must make the people to assume that it is doing internally the same, or very similar, things that people do when they understand language (Güngördü, 1993).

Two types can be thought to carry out the necessary analyses on the textures: Morphological and Statistical analysis.

1. Morphological analysis is to seperate the words according to its root and suffixes and to get the types of these as verb, adjective, noun...

2. Statistical method is the analysis carried out on letters or words. In letter analysis, consonant and vowel placements, letter n-gram frequencies, relationship between letters such as letter positions in accordance with each other; these kinds of analyses can be carried out on the letters. Investigation of number of letters in a word, the order of the letters in a word, word n-gram frequencies, word orders in a sentence; these kinds of the analyses can be carried out on words, called Word Analysis.

Morphological analysis is based on the rules and boundaries which are defined by the structure of that natural language. Turkish is an ‘agglutinative language’ like Finnish, Hungarian, Quechua and Swahili, in which new words are formed by adding suffixes to the end of roots. In Turkish, new words are generated by using suffixes according to the defined rules about their order in two ways. A suffix adds a meaning to the given word or changes the exact meaning of the word. By using suffix chains, a word can change its part of speech whose meaning is closer to the its root. (Çekoslavakyalılaştıramadıklarmızdan mısınız?)

NLP is used in:

- Speech processing: The dialogs can be clarified by using this system. It makes some works in real life easier. For instance, a journalist can change a dialog record to text by using some software which uses a corpus-based structure.
- Natural language understanding: Treated as moving from isolated words (either written or determined via speech recognition) to 'meaning'. This may involve complete model systems or 'front-ends', driving other programs by NL commands.
- Natural language generation: Generating appropriate NL responses to unpredictable inputs.
- Machine translation (MT): Translating one NL into another. (Coxhead, 2002)
- Database applications: It helps the user by the familiarity and flexibility of the language while accessing the database. It is used in expert systems for explaning the generation by the knowledge of syntax and semantics of the fragment of natural language.
- Spelling correction: Spelling correctors are word-based, but nowadays there have been a lot of studies about syntax-based spelling correctors. There is a word-based spelling corrector for Turkish developed by Solak.

For these kinds of processes a corpus is needed. The word capacity of a corpus affects the analysis about the reliability and the usability of analyses in natural language processing areas. Reliable information which can be received from a large

corpus increases the use of NLP in scientific research as well as in daily life. As it is known, Turkish is always affected by the other languages. The real structure of Turkish can be defined and preserved by using a corpus. To generate a corpus, it is very important to determine root and suffixes of the words in a correct way.

The main goal of this study is to develop an algorithm to generate the ideal way while separating roots and suffixes. That may be accepted as the most critical according to its position in corpus because this module uses the sentence boundary detection as the input and the output of this section is used as a base of the word type detection.

This thesis is divided into 5 chapters. In Chapter 1, the aim of the thesis is introduced. In Chapter 2, there is included the Corpus is defined and the Corpora formed in English, Turkish and other languages are explained. In chapter 3, some of the previous research on morphological analysis of Turkish language is explained briefly. Turkish root determination algorithms in literature are introduced with their main specialities. A detailed explanation of the developed algorithm of morphological system is given in Chapter 4. Finally, last chapter presents conclusion.

CHAPTER TWO

CORPUS AND LARGE SCALE CORPUS

A corpus can may be defined as "a collection of naturally occurring language text, chosen to characterize a state or variety of a language." (Sinclair, 1991).

Crystal states "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*).*

Corpus can be defined as a special database, in which the roots, their properties, suffixes and their combinations with each other are all stored in tables. The source is all the texts obtained from web sites or scanned forms of printed materials like books, newspapers, technical manuals, conversations, etc.

The selection of sources for a corpus changes the structure and makes it available for use in different research areas. An optimum corpus may be be created by collecting large samples from all varieties of language. On the other hand, a corpus can be structured in two different ways, namely, "Balanced" and "Unbalanced". Most large corpora are "Unbalanced". A corpus may be structured as balanced by using similar size samples of all varieties of a language, i.e. written or spoken language, and different genres in these varieties, essay, newspapers, novels, technical manuals, scientifiec articles, etc. that makes corpus a "representative" of the language. But it is very difficult to take equal, small pieces of samples from different areas into a corpus. Instead of this, an unbalanced corpus may be generated because it will contain greater number of words from any area within a language. When working on letter analysis, small sized corpora are enough (Dalkılıç, 2001), but for word analysis large corpora are needed. And for some rarely used words, unbalanced corpus can be more powerful than a balanced corpus.

The main properties of a corpus can be defined by using some methods like Different Word Usage Ratio (DWUR). DWUR, n-gram analysis, Number of Different Words (NODW) are also types of analyses which may be used to inquire

corpora data. For example, an n-grams algorithm can estimate the order of words and this can be used in speech recognition systems (Nadas, 1984). The results of n-gram analysis can be evaluated to correct uncompleted or wrong word forms in many applications. By matching patterns and using a certain method, printed materials can be transferred to digital environmrent. This method is also useful for OCR (Optical Character Recognition) (Kukich, 1992). Results of this method can also show themselves in language-oriented encryption algorithms and data compression algorithms succesfully.

2.1 Corpus

The necessity of a corpus is valid for each language naturally. Different states conduct a great deal of research to build corpora of their languages. Some of them are representative and some are large (Church & Mercer, 1993). As it was mentioned above, by using the corpus, different types of analyses can be performed, such as different word usage statistics and n-gram analyses for letters (Shannon, 1951) and words (Jurafsky & Martin, 2000), etc. Also, character recognition operations, cryptanalytical procedures, spelling correction (Church & Gale, 1991) processes can be carried out by using corpora in NLP applications.

Some examples of the corpora in different languages are given in the following sections.

2.1.1 English Corpora

2.1.1.1 Brown Corpus

This corpus was first assembled in 1963-1964 at Brown University. In 1964, it had 1 million words with 61,805 different words and in a later edition in 1992, the new Brown corpus had 583 million words with 293,181 different words (Jurafsky & Martin, 2000).

2.1.1.2 British National Corpus (BNC)

The British National Corpus is a very large (over 100 million words) corpus of

modern English, both spoken and written. Non-British English and foreign language words also occur in the corpus (BNC: What is the BNC, (n.d.)). 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. A project of Oxford University Press also including some other members, the corpus was completed in 1994 and it was released in February 1995 (British National Corpus (BNC), (n.d.)). An unannotated example of a raw BNC text is given below:

```
<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 2.1 An unannotated example of a raw BNC text.

2.1.1.3 The Bank of English

The Bank of English is a collection of samples of modern English language held on computer for analysis of words, meanings, grammar and usage. Such a collection is also termed as a corpus in linguistics and lexicography (The Bank of English - Terms & Conditions, (n.d.)).

The Bank of English was launched in 1991 by COBUILD (a division of HarperCollins Publishers) and The University of Birmingham. It had 450 million words with over half million different words and it continues to grow with the constant addition of new material. The corpus includes spoken and written language. 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 (The Bank of English, (n.d.)).

2.1.1.4 English Gigaword

It is an English corpus having 1,756,504,000 words obtained from 4,111,240 documents. It is a product of Linguistic Data Consortium (English Gigaword, (n.d.)).

2.1.1.5 American National Corpus

The ANC will include 100 million words, including both written and spoken (transcripts) data. In addition to the core 100 million words, the ANC will include an additional component of potentially several hundreds of millions of words, chosen to provide both the broadest and largest selection of data possible (The ANCProject, (n.d.))

2.1.2 Turkish Corpora

Some Turkish corpora are listed below:

- Koltuksuz Corpus
- YTÜ Corpus
- Dalkilic Corpora
- METU Turkish Corpus
- TurCo Turkish Corpus

There are also other corpora for Turkish approximately ~2.2M Words (Güngör, 1995).

2.1.2.1 Koltuksuz Corpus

Koltuksuz Corpus has 6,095,457 characters and formed of 24 novels and stories from 22 different authors. It is used for letter statistics and finding out some of the characteristics of Turkish language (Koltuksuz, 1995).

2.1.2.2 YTÜ Corpus

This corpus has 4,263,847 characters from 14 different documents: 3 Novels, 1 PhD Thesis, 1 Transcription and 9 Articles (Diri, 2000).

2.1.2.3 Dalkilic Corpora

Dalkilic Corpus was created to study letter statistics and to define the characteristics of Turkish language. It has 1,473,738 characters from Hurriyet newspaper web archive (Dalkılıç, 2001).

Another *Dalkilic Corpus* 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).

2.1.2.4 METU Turkish Corpus

It is a collection of Turkish samples written after 1990, comprising approximately 2 million words (METU Corpus, (n.d.)).

2.1.2.5 TurCo Turkish Corpus

TurCo includes 50,111,828 words. It consists of text data taken from 11 different websites, and novels and stories in Turkish that belong to more than 100 authors.

In TurCo, Number of Words (NOW), number of different words (NODW) and Different Word Usage Ratio (DWUR) are calculated and shown in Table 2.2. NODW in all sites are 1,235,056, but some words are repeated in different sites.

2.1. 3 Corpora of Other Languages

2.1.3.1 The Czech National Corpus (CNC)

The Czech National Corpus (CNC) is an academic project aimed to build a large computer-based corpus, containing written Czech language.

The CNC was started to build in 1991 by 8 representatives: Faculty of Philosophy Charles University, Faculty of Mathematics and Physics, Charles University, Masaryk University, Palack=EC university and the Institute of Czech Language, Academy of Sciences (Klimova, 1996, *CZECH NATIONAL CORPUS (CNC)*).

2.1.3.2 Croatian National Corpus

This corpus includes 30 million words from older and contemporary texts. It is available through Croatian Academic Research Network (*Croatian National Corpus,* (n.d.)).

2.1.3.3 PAROLE

PAROLE is a multilingual corpus, in which the languages involved: Belgian French, Catalan, Danish, Dutch, English, French, Finnish, German, Greek, Irish, Italian, Norwegian, Portuguese and Swedish. It has 20,000 entries per language. All the data are chosen from the texts produced after 1970.

2.1.3.4 French Corpus

French Corpus has 20,093,099 words from written materials, such as books, newspapers, periodicals, etc. (French Corpus, (n.d.)).

2.1.3.5 COSMAS (Corpus Search Management Analysis System)

COSMAS is a German corpus that includes 1,903,000,000 running words. It is a product of "Institut für Deutsche Sprache, Mannheim" (COSMAS, German Corpus, (n.d.)).

2.2 Large Scale Corpus

Large corpora are needed if word analysis is carried out; but there is no need for such a large corpus to be unbalanced. Combining large and representative (balanced and unbalanced) corpora is required to use the corpora in natural language processing research areas such as speech recognition, spell checking etc. in an effective way.

There isn't any corpus for Turkish large enough to make efficient analyses on it. In case there is such a corpus, some statistical properties of the words in Turkish language can be investigated easily and can be used in various areas of research.

CHAPTER THREE
PREVIOUS WORK

In morphological analysis, finding the real root, which is meant with the word by the user, is really important. Due to some difficulties in Turkish, as ambiguities in homonyms, all researchers try to find the nearest root to the intended one instead of finding the real root. This process is called Morpholocial Parsing. Parsing algorithms may be divided into two classes as “root-driven” and “affix stripping” analysis methods (Solak and Oflazer, 1993). Both methods have been used in terms of morphological parsing.

The root driven approach, firstly, finds the stem of the word in a lexicon before starting the morphological analysis. Most morphological analyzers such as PC_Kimmo (Antworth, 1990) and Ample (Weber, 1988) use the root driven approach and confirm the method’s success for different languages. Root driven methods are also widely used in the studies conducted for Turkish.

Some affix stripping methods have been developed and successful results were achieved for agglutinative languages. The cost of the search process required to find the stem in root driven approach is very high. The examining of each subpart is a time consuming process especially for the languages where the words can appear in very long and complex forms, such as Turkish. But the searching process is very fast as the search is only performed for affixes in the affix stripping approach.

3.1 Morphological Parsing in Other Languages

For Greek, Packard's parser runs by getting affixes off the word, and then looking up the remainder in a lexicon. Generally, the parse is successful. But when there is a word in the lexicon matches the remainder and this remainder is like affixes.

Brodda and Karlsson use a similar method for Finnish, an agglutinative language, without any lexicon of roots. Suffixes are taken off from the end of the word until no more can be removed, and what is left is assumed to be the root.

Three different approaches to morphological parsing of agglutinative languages were developed in the early 1980s: for Quechua (R. Kasper, 1982), for Finnish (Koskenniemi, 1983) and for Turkish (Hankamer, 1984). These three approaches are all left to right parsers.

3.2 Stem and Root Finding Algorithms for Turkish

Some of the methods that determine root or stem of words in Turkish are investigated as below:

1. AF Algorithm
2. Longest-Match (L-M) Algorithm
3. Identified Maximum Match (IMM) Algorithm
4. FindStem Algorithm
5. Solak and Oflazer's Approach
6. Root Reaching Method without Dictionary
7. Extended Finite State Approach

3.2.1 AF Algorithm

AF algorithm, developed by Solak and Can in 1994, works by a lexicon that keeps actively used stems for Turkish in which each record is explained with 64 tags. The word searched is iteratively looked up in the lexicon from right to left by pruning a letter at each step. 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 matched root found, add the word into root-word list.
4. If the word remained as a single letter, the root-word list
is empty then go to step 6, if root-word 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 searched word into unfounded record and exit.
7. Get the root word from the root-word 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 stem-list.
10. If there are any elements in root-word list then go to step
7.
11. Choose all stems in the stem-list as a word stem.
```

This algorithm finds all possible stems of the word; the number of stems that are found may be too high, e.g., the root '*göz*' (eye) is a source of derivation to nearly 150 stems which have totally different meanings.

3.2.2 LM Algorithm

Longest-Match (L-M) is developed by Kut et al. in 1995. It is based on the word search over a lexicon that includes Turkish word stems and their possible variants. 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 matched root is found, go to step 5.
4. If the word remains as a single letter, go to step 6.
Otherwise, remove the last letter from the word and go to step
2.
5. Choose the root that is as a stem and go to step 7.
6. Add the searched word into the unfounded records.
7. Exit.
```

This algorithm finds the first match stem by beginning with the last letter of the word and removing the rest one by one.

3.2.3 Identified Maximum Match (IMM) Algorithm

IMM algorithm is developed by Köksal in 1975. It is a left-to-right parsing algorithm and tries to find the maximal substring which exists in a root lexicon. If a match is found, the remaining part of the word is considered as the suffixes; this part searched in a suffix morpheme forms dictionary and morphemes are identified one by one until there is no element left.

3.2.4 Solak and Oflazer's Approach

Solak and Oflazer used a dictionary of 23,000 words, which was based on the Turkish Writing Guide (Solak and Oflazer, 1993). The words are placed in a sorted order. Each entry of the dictionary contains a root word in Turkish and a series of flags showing certain properties of that word. 64 different flags are reserved for each entry, but only 41 flags have been used. Some of the flags are given in the Table 3.1.

Table 3.1 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 algorithm searches the root in the dictionary using a maximal match algorithm. Firstly, the whole word is searched in the dictionary, if it is found then the word is accepted as I, which means that the word doesn't have any suffixes. If not, then a letter from the right is removed and the substring is searched. This step is repeated until the root is found. If no root is found although the last letter of the word is reached, the word's structure is accepted as incorrect.

3.2.5 Root Reaching Method without Dictionary

This method is developed by Cebiroğlu and Adalı. It is claimed that the analysis of a Turkish word to its root and suffixes can be formulated. The suffixes 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 group-specific machines. In the morphological analysis performed using the main machine, the root is obtained by extracting the suffixes from the end towards the beginning. Here are the abbreviations used in suffixes:

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

Example: "-cU" can be -cı, -ci, -cu, -cü

The morphological rules can be determined by finite state machines. To reach the root of the word, these rules may be interpreted from right to left and from the end to the beginning. For all sets, different modules are developed, dependent to each other.

The following table shows the verbal affixes in Turkish. This is determined as a set of the verbal affixes:

Table 3.2 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 following figure is a finite state machine which is the implementation of this table:

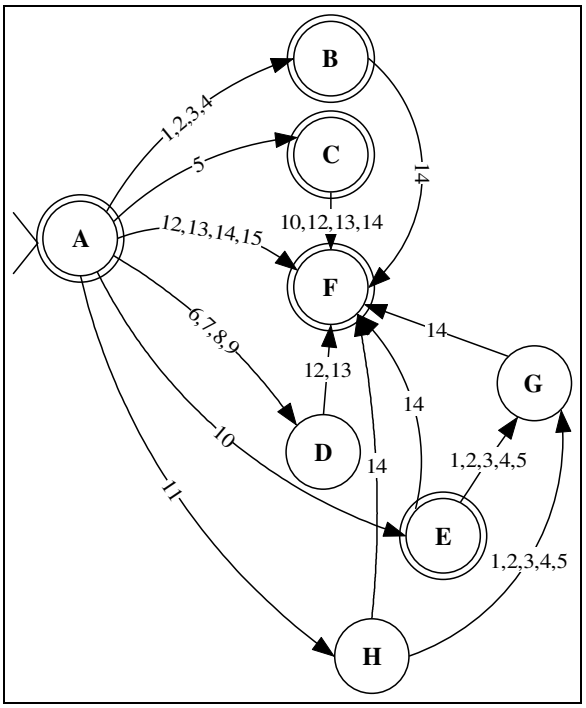


Figure 3.1 Finite state machine of Table 3.2.

For all sets like the affixes that are used for nouns and verbs, new finite state machines are implemented. All of them are all combined in one finite state machine at the end and the roots are found. The following figure shows the main finite state machine.

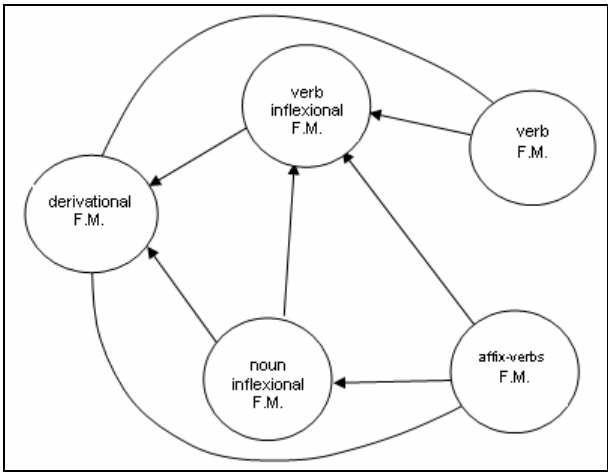


Figure 3.2 The main finite state machine

3.2.6 Extended Finite State Approach

In this algorithm developed by Oflazer, 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 $Infl_n$ denote relevant inflectional features including the part-of-speech for the root, or any of the derived forms. For instance, the derived determiner "sağlamlaştırdığımızdaki" ((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+Plsg
5. +Noun+Zero+A3sg +Pnon+Loc
6. +Det

In this sense a sentence may be represented as a sequence of the IGs constituting the words. 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), as exemplified in the following figure:

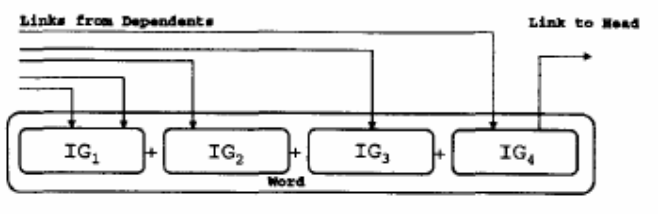
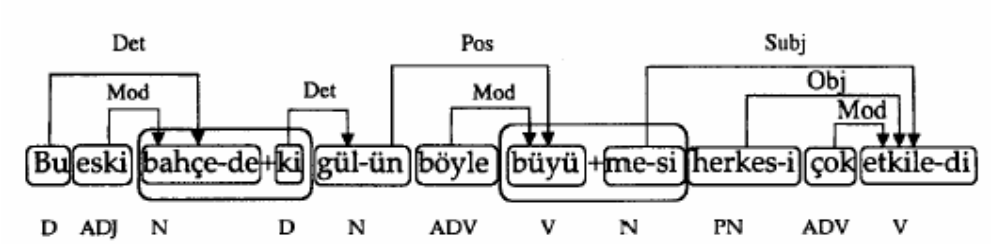


Figure 3.3 Links and inflectional groups

The following figure shows a dependency tree for a sentence laid on top of the words segmented along IG boundaries.



Last line shows the final POS for each word.

Figure 3.4 Dependency links in an example Turkish sentence

This approach relies on augmenting the input with "channels" that (logically) reside above the IG sequence and "laying" links representing dependency relations in these channels. The parser works in a number of iterations; a new empty channel at each iteration is stacked on top of the input, and any possible links are established using these channels, until no new links can be added. If a link starts from an IG, then a start symbol denoting the syntactic relation is used on the right side of the IG. The syntactic relations 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).

3.2.7 FindStem Algorithm

FindStem**,** developed by Sever and Bitirim (Sever & Bitirim, 2003), 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 an examined word are found by starting with the first character of the examined word and searching the lexicon for this item. Roots that are found and the production rules are used to derive the examining word. In the lexicon, the type information for every root word and possible root changes (when a root word combines with suffix) is coded for morphological analysis. During the root and the suffix combination in Turkish, two alterations on a root word structure may be observed: last vowel alteration (e.g. ar**a**-arıyor) or consonant alteration (e.g. kita**p**-kitabı) of the root word and ellipsis of middle vowel (e.g. oğ**u**l-oğlum).

A morphological analyzer is used in”Morphological Analysis” component. In Turkish, there are a number of rules to determine the form and order of suffixation. Addition of the derivational suffixes to the end of a word is determined by word category. If this procedure is applied, all possible stems can be found. Consider the word ”edebilecek” as word in examination. The longest possible roots retrieved from the lexicon are ”edebi” and ”edep”. According to the algorithms LM and Identified Maximum Match (IMM) that assigns a stem by matching the examined word with the longest root words, “edebi” will be selected as output. But it is not possible to produce the examined word,”edebilecek”, by using this root.

In the final component, “Choose the Stem”, the word stem is chosen by a selection of a derivation among derivation list.

Here is the algorithm:

```
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 word-list.
3. If root word-list is empty, add the word into the unfounded
records and exit.
4. Get the root word from root-word list.
5. Apply morphological analysis to the root word.
6. After morphological analysis, add derivations that are
formed into derivation-list.
7. If there are any elements in root-word list then go to step
4.
8. Choose the word stem by a selection among derivations in
the derivation-list.
```

3.3 Sentence Boundary Detection

Determining sentence boundary is one of the most important prerequisites in generating corpus. Most natural language processing tools do not perform a reliable detection of sentence boundaries.

Employing a list of end-of-sentence markers (punctuation marks, e.g. “.”, “!”) is useful to find the end of the sentence. But a marker can be used in different functions for exmple “.” (dot) can be used in an abbreviation, as a decimal point, in e-mail addresses etc. Some examples are shown below:

- She comes here by 5 p.m. on Saturday evening.
- At 5 p.m. I have to go to the hospital.

These kinds of situations are called ambiguities. Such ambiguities are the main problem of sentence boundary detection, and there are not any application for Turkish or for any other language that solves these kind of ambiguities encoutered in the language.

CHAPTER FOUR

PROPOSED SYSTEM

Word analysis for Turkish is carried out according to the root of the word and the suffixes added to that root. After the analysis, the roots and suffixes are determined. Depending on the analysis, the determination of the type of word can be carried out easily.

For word type determination, at first, the sentences and then words should be determined. The Sentence Boundary Determination (SBD) method developed by Aktaş (2006) was used in this study. The general work of the analysis module is given below:

1. Parsing the XML files which are received from SBD method as its result.
2. Process the source text word by word.
3. Find all root possibilities.
4. Find the suffix possibilities according to each root possibility.
5. Eliminate impossible combinations in space.
6. Save all the combinations in a file to make the word analysis easier.
7. Present the result to the user in an XML file.

This flow is also drawed in Figure 4.1.

In the first and second steps, the algorithm developed by Aktaş is used. The results obtained from these procedures can be saved in a database to create a corpus.

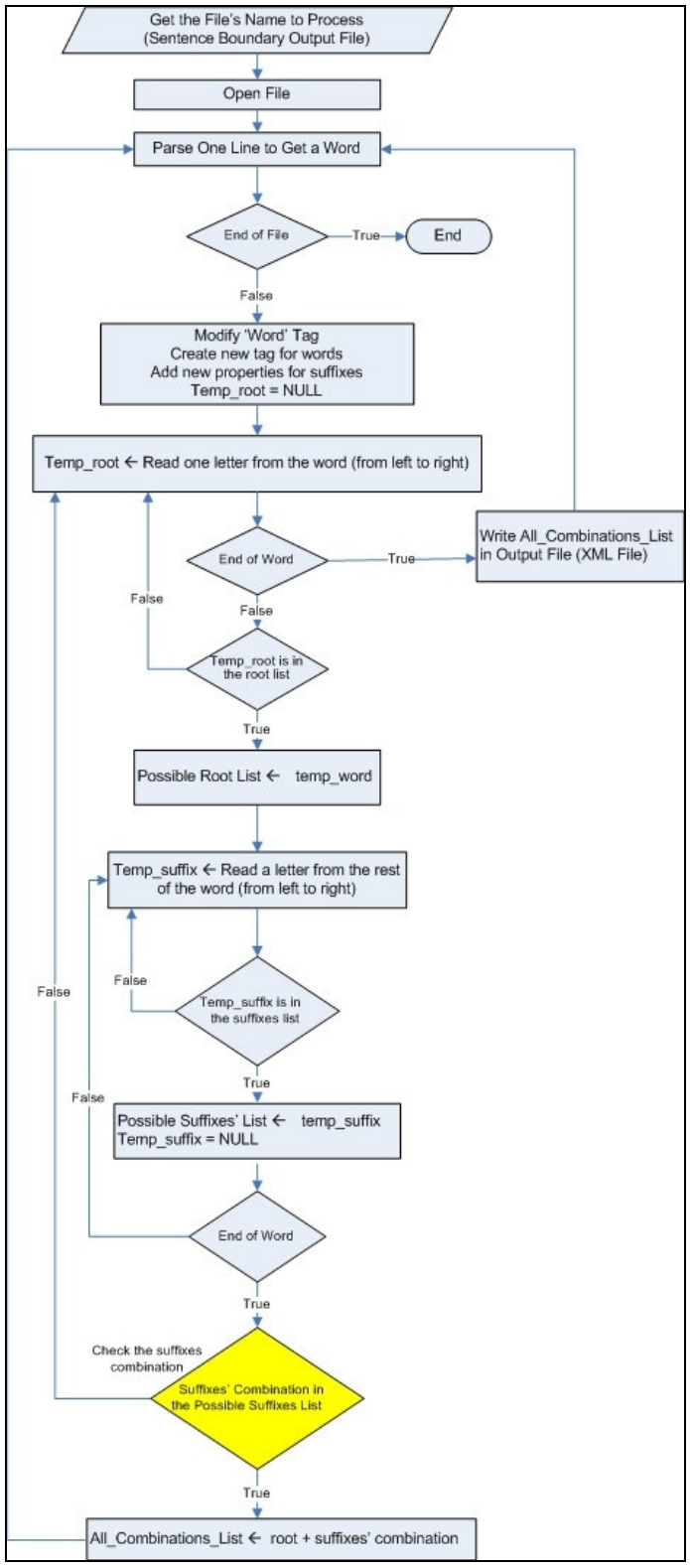


Figure 4.1 Block diagram of algorithm for generating corpus

4.1 Algorithm

The result file of SBD method presents a XML file to morphological analysis module as input. Words can come to this module one by one owing to this system.

At firstly, the word to be analyzed is taken from the data prepared by SBD process. As a nature of agglutinative languages, a word in Turkish can be separated into many different alternatives. Each alternative is formed by different root and suffix combinations. The first step in the morpohological analysis is to find possible root and suffix combinations.

In Turkish, words are written left to right as known. So that, root of a word starts with the first character from left. Each character generates a new character group and this process continues until this character group is also a meaningful root. The root space is taken from Turkish Linguistic Association (Türk Dİl Kurumu, TDK) that is shown in Appendix A and this list is modified according to all vowel phenomena. A part of the modified list is shown in Appendix B.

After finding a meaningful root, the rest of the word is accepted as a suffix or suffix combination. With this process, the system starts to find all possible suffix combinations.

Finding all possibilities the process is repeated, similar to the one to find the root. Each character in the rest of the word is checked, whether it is a suffix or not. In this control mechanism, a suffix list is used. The base of this list, which is given in Appendix C, is created by Linguists from The Department of Linguistics. Then the list of suffixes is generated according to all character-change possibilities. The result list is given in Appendix D. Each suffix possibility is controlled from the list. The possible suffixes for the possible root are controlled according to the rules of suffix orderings. The valid combinations are saved for the result file.

The steps given above is repeated until the whole word is controlled whether it is a root or not. At the end of analysis, an XML file is given to the user which includes

all the sentences, words and the morphological analysis of words.

As an example, the sentence “Dümenin terbiye edemediğini kayalar terbiye eder.” (Those who will not be ruled by the rudder mist he ruled by the rock.) is analysed as follows:

Step 1: SBD system separates the sentence to words (Aktaş, 2006). The result file is shown in Figure 4.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <File OriginalName="ornek.txt">
  - <Paragraph Index="0">
    - <Sentence Index="0">
        <Word Index="0">Dümenin</Word>
        <Word Index="1">terbiye</Word>
        <Word Index="2">edemediğini</Word>
        <Word Index="3">kayalar</Word>
        <Word Index="4">terbiye</Word>
        <Word Index="5">eder</Word>
      </Sentence>
  </Paragraph>
</File>
```

Figure 4.2 The result file from SBD system

Step 2: Each word is taken to analysis process to find the root

For first word “Dümenin”;

- “D” checked from root list and it’s not a valid root.
- “Dü” checked from root list and it’s not a valid root.
- “Düm” checked from root list and it’s not a valid root.
- “Düme” checked from root list and it’s not a valid root.
- “Dümen” checked from root list and it’s a valid root. Then go to the STEP 3.

Step 3: Find all possible suffixes

- "i" can be used as Suffix that derives Noun From Verb (TBEAi)
  - "n" can be used as Suffix that derives Noun From Verb (TBEAn)
  - "n" can be used as Suffix that derives Noun From Noun (TBAAn)
  - "n" can be used as Letter that combines root and suffix (KaySes)
- "i" can be used as Suffix that derives Verb From Verb (TBEEi)
  - "n" can be used as Suffix that derives Noun From Verb (TBEAn)
  - "n" can be used as Suffix that derives Noun From Noun (TBAAn)
  - "n" can be used as Letter that combines root and suffix (KaySes)
- "in" can be used as Suffix that inflects Verb in 3.Person Group as 2. Person Plural (DuEKGr3C2)
- "in" can be used as Possesive Suffix as 2. Person Singular (DuAUyKT2)
- "in" can be used as Possesive Suffix as 3. Person Singular (DuAUyKT3)
- "in" can be used as Genitival Suffix (DuADurTam)

- "Dümeni" checked from root list and it's not a valid root.
- "Dümenin" checked from root list and it's not a valid root.
- First word ends.Then, go to STEP 1 for the next word.

On the other hand, the second word "terbiye" is analysed according to its two possible roots. Second word is taken to analysis process to find the root:

Step 2: Take each character to analyse and try to find the root

- "T" checked from root list and it's not a valid root.
- "Te" checked from root list and it's not a valid root.
- "Ter" checked from root list and it's a valid root.

  Step 3: Find Suffix Combinations for root "ter", if possible.

  - "bi" is not a valid suffix
  - "biy" is not a valid suffix
  - "biye" is not a valid suffix

  So, there can't be any valid suffix combination for the root "ter". Therefore "ter" is not accepted as a valid root for the word "terbiye".

- "Terb" checked from root list and it's not a valid root.
- "Terbi" checked from root list and it's a valid root.

  Step 3: Find Suffix Combinations for root "terbi", if possible.

  - "y" can be used as Suffix that derives Noun From Verb (TBEAy)
    - "e" (DuADurYon)
    - "e" (DuEKipA)
    - "e" (TBEAa)
    - "e" (TBAAa)
  - "y" can be used as Suffix that derives Noun From Noun (TBEEy)
    - "e" (DuADurYon)
    - "e" (DuEKipA)
    - "e" (DuEKipYet)
    - "e" (TBEAa)
  - "y" can be used as Suffix that derives Noun From Noun (TBEAy)
    - "e" (DuEKipA)
    - "e" (TBAEa)
    - "e" (TBEAa)
  - "y" can be used as Letter that combines root and suffix (KaySes)

- "e" (DuEKipA)
- "e" (DuEKipYet)

- "Terbiy" checked from root list and it's not a valid root.
- "Terbiye" checked from root list and it's a valid root. It has no suffixes.

These suffix possibilities are existed according to the rules which are defined by linguists. The result for this word is given to the user in the result file as in Figure 4.3.

```
- <Word Index="1">
  <ValueOfWord>terbiye</ValueOfWord>
    - <Root Index="0">
      <Value>terbi</Value>
        - <Suffixes>
          - <SuffixCombination Index="0">
              <TBEAy>y</TBEAy>
              <DuADurYon>e</DuADurYon>
            </SuffixCombination>
          - <SuffixCombination Index="1">
              <TBEEy>y</TBEEy>
              <DuADurYon>e</DuADurYon>
           </SuffixCombination>
          - <SuffixCombination Index="2">
              <TBEAy>y</TBEAy>
              <DuEKipA>e</DuEKipA>
            </SuffixCombination>
          - <SuffixCombination Index="3">
              <TBEEy>y</TBEEy>
              <DuEKipA>e</DuEKipA>
          </SuffixCombination>
```

Figure 4.3 The part of result file about the word "terbiye"

```
- <SuffixCombination Index="4">
    <TBEAy>y</TBEAy>
    <TBAEa>e</TBAEa>
  </SuffixCombination>
- <SuffixCombination Index="5">
    <TBEEy>y</TBEEy>
    <TBAEa>e</TBAEa>
  </SuffixCombination>
- <SuffixCombination Index="6">
    <TBEAy>y</TBEAy>
    <TBEAa>e</TBEAa>
  </SuffixCombination>
- <SuffixCombination Index="7">
    <TBEEy>y</TBEEy>
    <TBEAa>e</TBEAa>
  </SuffixCombination>
- <SuffixCombination Index="8">
    <TBEAy>y</TBEAy>
    <DuEKipYet>e</DuEKipYet>
  </SuffixCombination>
- <SuffixCombination Index="9">
    <TBEEy>y</TBEEy>
    <DuEKipYet>e</DuEKipYet>
  </SuffixCombination>
- <SuffixCombination Index="10">
    <KaySes>y</KaySes>
    <DuEKipYet>e</DuEKipYet>
  </SuffixCombination>
</Suffixes>
</Root>
<Root Index="1">
<Value>terbiye</Value>
<Suffixes />
</Root>
</Word>
```

Figure 4.3 (continued)

This process is repeated for each word in the sentence. A part of the result file is given in Appendix E.

4.2 Refinements of the Developed Method

The developed system works correctly but during the programming stage, some performance problems were faced. Finding all possible root and suffixes took a lot of time and this caused inacceptable tilues. In order to increase overall system performance, some improvements and modifications were realized.

First refinement for getting better performance is about character checking process to find whether it is a valid root or suffix. At each control, algorithm tried to connect to database and controlled the given value from the related table. Each connection took 18 ms in average. The analysis were done for the sample sentence given below.

Dümenin terbiye edemediğini kayalar terbiye eder. (1)
Those who will not be ruled by the rudder mist he ruled by the rock.

Algorithm should connect over 80.000 times to analyse this sentence. Therefore, analysis of this sentence takes 1440 seconds, 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, keeping root and suffix information in the memory was realized. To do this, developed software connects to database only one time at the beginning. All root and suffix information is saved into two lists, one for roots and the other one for suffixes. Software checks the character combinations from these lists. By using this approach, the process time decreases radically from 24 min to 24 sec for the sample sentence, which causes a system speed increase by 60 times.

Although this new process time is better than the old one, a column can be analysed in 2 hours by using this value of time. This performance is also not acceptable and makes the system unusable. Therefore, the suffix combination control part of the algorithm was changed. This refinement is done after the determination of each suffix combination. This developed the efficiency in the rate of % 25. The sentence (1) started to be analysed in 6 sec. This value is achieved according to the given rules. By increasing number of rules, the suffix possibilities and the time of checking processes will decrease. This statement causes decreasing the time. The graphic for this statement can be shown in Figure 4.4.

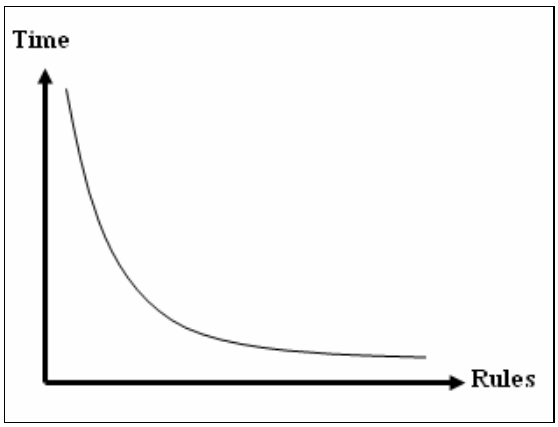


Figure 4.4 The Relationship Between The Number of Rules and The Process Time

A refinement was made by saving the found roots and suffixes of the analysed word to a database, in order to avoid re-analysis of the same word. This approach is realistic until a semantic analysis module will be developed, because each word is analysed by comparing with itself.

In this algorithm all known roots are arranged in ascending order in a list. There can be a special dictionary structure to get easier response while checking the root possibilities. In the method of Yiğit (2007), reaching to the searched word is easier and this structure takes less space in memory. It can be explained as bunch of grabes. Words are saved letter by letter and these letters are linked to each other if they have a meaningful combination. A part of dictionary is given in Figure 4.5.

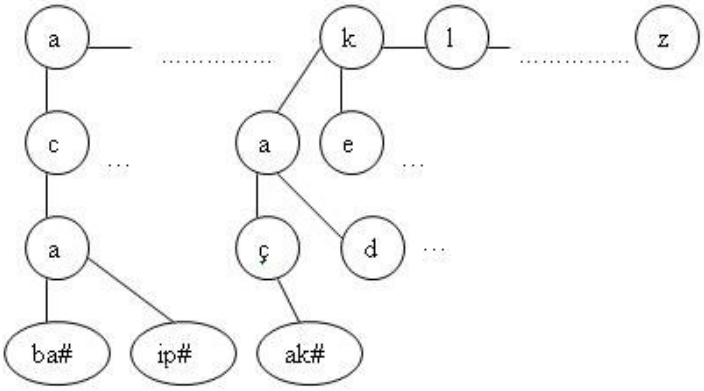


Figure 4.5 The Bunch Structure For Roots Like A Dictionary

4.3 The Effects and Reasons of Selected Technologies

The algorithm developed in this study was developed in C# by using Visual Studio .NET. Since this algorithm can be used in many applications, in which C# is used as one of the most-preferred languages, this programming language was selected.

On the other hand, C# is adaptable to Object Oriented Programming. Object Oriented Programming is a critical point for the usability of this system. The technological structure of a Corpus must be updatable. It is important for its maintenance and development periods. The project is developed module by module by using object oriented mentality.

The program uses Microsoft (MS) SQL Server for the database. It is preferred because MS SQL Server is the one of the most powerful relational database management systems. Its main properties are easiness to use and integrate old versions with the new ones. User should not do anything after changing the version. It is critical for this project because this system is proposed to be used in the future. The database technology also must be adaptable according to future's technologies.

CHAPTER FIVE

CONCLUSION

The main property of Turkish is being agglutinative. This property adds new meanings to the roots or derives new words from any root, which is also related to the root, by adding suffixes. Because of this, anyone, who wants to learn Turkish, should learn all the grammer rules about it. This is the first step. Then this person should examine the place of a word in a sentence. These two steps shows the exact meaning of a word in a dialog or texture.

This project works correctly according to its aim as explained above. Its structure is adaptable to fix it with the other projects. There is only needed an input XML file, which contains a text, seperated to its words before. The result file is also designed to be able to be used by another applications.

This Morphologic Analyser has also enough good performance to do its processes. A text which contains 300 sentences can be analysed in 5 seconds. This performance will be developed after using this system many times.

This system will work as a module with the other projects done in the Department of Computer Engineering in Dokuz Eylul University to create a General Large-Scaled Turkish Corpus.

In the future, a corpus for Turkish will be used in many applications. There will be needs to get more efficient results about Turkish and its use. This work will be the base of a system about a CORPUS.

REFERENCES

Aktaş Ö. (2006). Türkçe için Verimli bir Cümle Sonu Belirleme Yöntemi. *Akademik Bilisim 2006 - Bilgi Teknolojileri Kongresi IV*, Pamukkale University, Denizli, Türkiye.

Antworth E. (1990). *PC-KIMMO: A two-level processor for morphological analysis*. TX: Summer Institute of Linguistics, Dallas.

BNC: What is BNC. (n.d.). Retrieved March 3, 2005, from http://www.natcorp.ox.ac.uk

British National Corpus (BNC). (n.d.). Retrieved March 3, 2005, from http://www.hcu.ox.ac.uk/ BNC/what/index.html

Brodda, B., & Karlsson, F. (1980). *An experiment with morphological analysis of Finnish*. Papers from the Institute of Linguistics, University of Stockholm, Publication 40, Stockholm.

Cebiroğlu, G. & Adalı, E. (2002). *Root reaching method without dictionary*. Istanbul Technical University Computer Engineering Department, Istanbul, Turkey.

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

Church, K., & Mercer, R. (1993). Introduction to the Special Issue on Computational Linguistics Using Large Corpora. *Computational Linguistics, 19*(1), 1-24.

COSMAS, German Corpus. (n.d.). Retrieved January 5, 2005, from http://corpora.ids-mannheim.de /~ cosmas/.

Coxhead, P. (2002). An Introduction to Natural Language Processing (NLP), Retrieved June 26, 2005, from www.cs.bham.ac.uk.

Croatian National Corpus. (n.d.). Retrieved January 15, 2005, from http://www.hnk.ffzg.hr/corpus. htm.

Crystal,D. (1991). *A Dictionary of Linguistics and Phonetics*, Blackwell, 3rd Edition.

Çebi, Y. & Dalkılıç, G. (2004). *Turkish Word N-gram Analyzing Algorithms for a Large Scale Turkish Corpus - TurCo,* ITCC 2004, IEEE International Conference on Information Technology, Vol:2, pp. 236-240.

Dalkılıç, G. (2001). *Some Statistical Properties of Contemporary Printed Turkish and A Text Compression Application*. MSc Thesis. International Computing Institute, Ege University.

Dalkılıç, M.E., & Dalkılıç, G. (2001). Some Measurable Language Characteristics of Printed Turkish. *Proc. of the XVI. International Symposium on Computer and Information Sciences*, 217-224.

Diri, B. (2000). A Text Compression System Based on the Morphology of Turkish Language. *Proc. of the XV International. Symposium on Computer and Information Sciences*, 12-23.

English Gigaword. (n.d.). Retrieved January 5, 2005, from http://www.ldc.upenn.edu/Catalog/ Catalog Entry. jsp?catalogId=LDC200 3T05

French Corpus. (n.d.). Retrieved January 5, 2005, from http://www.elda.fr/cata/text/W0020.html, 02/01/2003.

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

Hankamer, J. (1984). Turkish generative morphology and morphological parsing, *Second International Conference on Turkish Linguistics*, Istanbul.

Hankamer, J. (1989). *Morphological parsing and lexicon.* Lexical Representation and Process, MIT Press.

Jurafsky, D., & Martin, J.H. (2000). *Speech and Language Processing*, Prentice Hall, 193-199.

Kasper, R. & Weber, D. (1982). *User's reference manual for the C's Quechua adaptation program.* Occasional Publications in Academic Computing, (8,9), Summer Institute of Linguistic, Inc.

Klimova, J. (1996). *CZECH NATIONAL CORPUS (CNC)*. Retrieved June 28, 2005, from http://www.ling.ohio-state.edu/~dm/events/EastWest96/cnc.html

Korkmaz, Z. (2003). Türkiye Türkçesi Grameri, TDK, Ankara.

Koskenniemi, K. (1983). *Two-level morphology*. University of Helsinki, Department of General Linguistics, Publication No. 11, Helsinki, Finland.

Köksal, A. (1975). *Automatic Morphological Analysis of Turkish.* Hacettepe University, Ankara, Turkey.

Kukich K. (1992). Technique for automatically correcting words in text. *Periodical Issue Article of ACM Press*, 377-439

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

METU Corpus. (n.d.). Retrieved October 20, 2004, from http://www.ii.metu.edu.tr/~corpus/corpus.html

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

Oflazer, K. (1999). *Dependency Parsing with an Extended Finite State Approach*. Department of Computer Engineering, Bilkent University Ankara, Turkey.

Packard, D. (1973). Computer-assisted morphological analysis of Ancient Greek. *Computational and Mathematical Linguistics: Proceedings of the International Conference on Computational Linguistics*, Pisa Leo S. Olschki, Firenze, 343-355.

Sagvall,A. (1973). *A system for automatic inflectional analysis implemented for Russian*, Data Linguistica 8, Almquist and Wiksell, Stockholm.

Sinclair,J. (1991). *Corpus Concordance, Collocation*. OUP.

Sever H., & Bitirim Y. (2003). *FindStem: Analysis and evaluation of A Turkish stemming algorithm.* Department of Computer Engineering Başkent University Ankara, 06530 Turkey, Department of Computer Engineering Eastern Mediterranean University Famagusta, T.R.N.C.

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

Solak, A. (1991). *Design and Implementation of a spelling checker for Turkish*, Department of Comp. Eng. And Information Sciences, Bilkent Unv., Ankara, Türkiye.

Solak, A., & Oflazer, K. (1993). *Design and Implementation of a Spelling Checker for Turkish*, Department of Computer Engineering and Information Science, Bilkent Unv. Ankara, Türkiye.

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

The ANCProject. (n.d.). Retrieved January 5, 2005, from http://americannationalcorpus.org

The Bank of English. (n.d.). Retrieved March 10, 2005, from http://www.cobuild.collins.co.uk/ boe_info.html

The Bank of English - Terms & Conditions. (n.d.). Retrieved March 10, 2005, from http://www.titania.bham.ac.uk/docs/svenguide.html

The Czech National Corpus (CNC). (n.d.). Retrieved January 15, 2005, from http://ucnk.ff.cuni.cz/ english/index.html

Weber, D. J., Black H. A., & McConnel, S. R. (1988). *AMPLE: a tool for exploring morphology*, TX: Summer Institute of Linguistics, Dallas.

Yiğit, F., (2007) *Dictionary Algorithms for Turkish.* MSc Thesis. Dokuz Eylul University, Institute of Science, Department of Computer Engineering, Izmir, Türkiye.

ABBREVIATIONS

Following acronyms have been used in this thesis:

MT	**M**achine **T**ranslation
NL	**N**atural **L**anguage
NLP	**N**atural **L**anguage **P**rocessing

APPENDICES

A. A Sample List of Turkish Roots

ab
aba
abadî
abajur
abaküs
aban
abandone
abanî
abanoz
abart
abaşo
abat
Abaza
abazan
Abbasî
abd
abdal
aberasyon
abes
abide
abis
abiye
abla
ablak
ablâtif
ablatya
abli
abluka
abone
abonman
aborda
abra
abraş
abril
abstraksiyonizm
abstre
absürt
abu
abuli
abullabut
abus
aç
acaba
açalya
acar
Acara
acayip
accelerando
acele
açelya
acem
acemi
acente
acep
aceze
acı
açı
acibe
acil
acırga

B. A Sample List of Alternative Turkish Roots

ab
aba
abadî
abajur
abaküs
aban
abandone
abanî
abanoz
abart
abaşo
abat
abad
Abaza
abazan
Abbasî
abd
abdal
aberasyon
abes
abide
abis
abiye
abla
ablak
ablâtif
ablatya
abli
abluka
abone
abonman
aborda
abra
abraş
abril
abstraksiyonizm
abstre
absürt
abu
abuli
abullabut
abullabud
abus
aç
ac
acaba
açalya
acar
Acara
acayip
accelerando
acele
açelya
acem
acemi
acente
acep
aceb

C. The Suffixes List with Its Tags

Ulamlar	**Etiket**	**Açıklama**
Ad	**A**	**Ad**
Özel ad	AOz	Özel ad
Genel ad	AG	Cins ad
Ad Öbeği	AO	Ad takımı
Eylem	**E**	**Eylem**
Geçişsiz Eylem	EGs	Geçişsiz eylem
Geçişli Eylem	EGl	Geçişli eylem
Eylem Öbeği	EO	
Sıfat	**S**	
Sıfat Öbeği	SO	Sıfat takımı
Niteleme sıfatı	SNit	
Belirtme Sıfatı	SBel	
- Gösterme	SBelGos	
- Sayı	SBelSay	
- Belgisiz	SBelBlg	
- Soru	SBelSor	
Unvan sıfatı	SUnv	
Pekiştirme sıfatı	SPek	
Küçültme	SKuc	
Belirteç	**B**	**Belirteç**
Zaman Belirteci	BZam	
Yer Belirteci	BYer	
Durum Belirteci	BDur	
Nicelik Belirteci	BNic	
Soru Belirteci	BSor	
Belirteç Öbeği	BO	Belirteç takımı
Bağlaç	**Bag**	
Ekleme	BagEk	
Genişletme	BagGen	
Seçenek	BagSec	

Karşıtlık	BagKar	
Neden Sonuç	BagNs	
İlgeç	**I**	**İlgeç**
İlgeç Öbeği	IO	
Adıl	**Adil**	**Adıl**
Kişi	AdilK	
Gösterme	AdilGos	
Belgisiz	AdilBlg	
Soru	AdilSor	
İlgi	AdilIlgi	
İyelik	AdilIye	
DİLBİLGİSEL ULAMLAR - (Çekim Ekleri)		
Ad Çekim Biçimbirimleri – (AD ULAMLARI)		
Sayı	**DuASay**	
{-lAr}	DuASayC	Adlara eklenen çoğul eki
{-Ø}	DuASayØ	Tekil adlar
Uyum	**DuAUy**	
{-(I)m}	DuAUyKT1	1. Tekil kişi iyelik eki
{-(I)n} / {-(I)nIz}	DuAUyKT2	2. Tekil kişi iyelik eki
{-(s)I(n)}	DuAUyKT3	3. Tekil kişi iyelik eki
{-(I)mIz}	DuAUyKC1	1. Çoğul kişi iyelik eki
{-(I)nIz}	DuAUyKC2	2. Çoğul kişi iyelik eki
{-lArI(n)}	DuAUyKC3	3. Çoğul kişi iyelik eki
Durum	**DuADur**	
{-Ø}	DuADurYal	Yalın durum (Özne Durumu)
{-(y)I}	DuADurBel	Belirtme durumu
{-(y)A}	DuADurYon	Yönelme durumu
{-DA}	DuADurBul	Bulunma durumu
{-Dan}	DuADurCik	Çıkma durumu
{-(n)In/-Im}	DuADurTam	Tamlayan durumu

Cinsiyet	**DuACins**	Sözcük Düzeyinde farklılık var
		tavuk-horoz, koç-koyun, kadın-erkek
Ad ulamlarının sıralanışı: Kök(Gövde)>DuASay>DuAUy>DuADur		
Eylem Çekim Biçimbirimleri – (EYLEM ULAMLARI)		
Zaman	**DuEZ**	
{-DI}	DuEZGD	Di'li Geçmiş Zaman
{-mIş}	DuEZGM	Miş'li Geçmiş Zaman
{-(A, I)r)}	DuEZGen	Geniş Zaman
{-(i)yor}	DuEZSim	Şimdiki Zaman
{-EcEK}	DuEZGel	Gelecek Zaman
Görünüş	**DuEG**	
{-DI}	DuEGBitD	Bitmişlik Görünüşü Di'li
{-mIş}	DuEGBitM	Bitmişlik Görünüşü Miş'li
{-(A, I)r)}	DuEGBtmsR	Bitmemişlik Görünüşü
{-(i)yor}	DuEGSur	Sürerlik Görünüşü
{-EcEK}	DuEGBtmsE	Bitmemişlik Görünüşü
Kiplik	**DuEKip**	
{-sA}	DuEKipSa	Dilek kipi
{-A}	DuEKipA	İstek Kipi
{-mAlI}	DuEKipMali	Gereklilik Kipi
{-Ø}	DuEKipEmir	Emir Kipi
Çatı	**DuEC**	
Ø	DuECEt	Etken Çatı
{-Il}, {-(I)n}	DuECEdil	Edilgen Çatı
{-(I)n}, ***{-Il}***	DuECDonus	Dönüşlü Çatı
{-(I)ş}	DuECIstes	İşteş Çatı
{(A, I) -r / -t / -rt}, {-DIr}	DuECEttir	Ettirgen Çatı

Olumsuzluk	**DuEOlz**	
{-mA/-m}	DuEOlz	Olumsuzluk
Kişi 1. Grup	**DuEKGr1**	**(DuEZGD ve DuEKipSa'dan sonra)**
{-m}	DuEKGr1T1	1. Tekil Kişi
{-n}	DuEKGr1T2	2. Tekil Kişi
{-Ø}	DuEKGr1T3	3. Tekil Kişi
{-k}	DuEKGr1C1	1. Çoğul Kişi
{-n-Iz}	DuEKGr1C2	2. Çoğul Kişi
{-lAr}	DuEKGr1C3	3. Çoğul Kişi
Kişi 2. Grup	**DuEKGr2**	**DuEZGen / DuEZSim / DuEZGel / DuEKipMali'dan sonra**
{-(y)Im}	DuEKGr2T1	1. Tekil Kişi
{-sIn}	DuEKGr2T2	2. Tekil Kişi
{-Ø}	DuEKGr2T3	3. Tekil Kişi
{-(y)Iz}	DuEKGr2C1	1. Çoğul Kişi
{-sIn-Iz}	DuEKGr2C2	2. Çoğul Kişi
{-lAr}	DuEKGr2C3	3. Çoğul Kişi
Kişi 3. Grup	**DuEKGr3**	**DuEKipA'dan sonra**
{-(y)Im}	DuEKGr3T1	1. Tekil Kişi
{-sIn}	DuEKGr3T2	2. Tekil Kişi
{-Ø}	DuEKGr3T3	3. Tekil Kişi
{-lIm}	DuEKGr3C1	1. Çoğul Kişi
{-sIn-Iz}	DuEKGr3C2	2. Çoğul Kişi
{-lAr}	DuEKGr3C3	3. Çoğul Kişi
Kişi 4. Grup	**DuEKGr4**	**DuEKipEmir'den sonra**
—	DuEKGr3T1	1. Tekil Kişi
{-Ø}	DuEKGr3T2	2. Tekil Kişi
{-sIn}	DuEKGr3T3	3. Tekil Kişi
—	DuEKGr3C1	1. Çoğul Kişi
{-In}, {-In-Iz}	DuEKGr3C2	2. Çoğul Kişi
{-sIn-lAr}	DuEKGr3C3	3. Çoğul Kişi
{-EcEK}	DuEKGr3T1	Gelecek Zaman

Eylem Ulamlarının sıralanışı:

Kök(Gövde)>DuEC>DuEOlz>DuEZ/DuEG/DuEKip>Koşaç>DuEK-***DIr***

(Özsoy ve Kerslake 2005:74)

Çekim Ekleri		
Eylemlere Eklenen Diğer Çekim Ekleri		
{-mAktA}	DuEGBtms	Bitmemişlik görünüşü
{-DIr}	DuEKipGnl	Genelleme Kiplik belirticisi
Birleşik Eylem	BE	
{-Abil}	DuEKipYet	Yeterlik
{-Iver}	DuEBETez	Tezlik
{-Agel / -Adur / -Akal /}	DuEBESur	Sürerlik
{-Ayaz}	DuEBEYak	Yaklaşma
{-Abil}	DuEKipOl	Olasılık
Koşaçlar		
Koşaç	K	
{-(y)DI}	KDi	Geçmiş Zaman Koşacı
{-(y)mIş}	KMis	Tanıtsallık Koşacı
{-(y)sA	KSa	Koşul Koşacı
Yantümceleme Biçimbirimleri (Çekimsiz Eylemlere Eklenen Çekim Ekleri)		
Tümce	Tu	
Yantümce	Ytu	
{-DIK}	YtuAdlDik	Adlaştırma Belirtcisi
{-(y)AcAK}	YtuAdlAcak	Adlaştırma Belirtcisi
{-mA}	YtuAdlMa	Adlaştırma Belirtcisi
{-mAK}	YtuAdlMak	Adlaştırma Belirtcisi
{-DIK}	YtuOrDik	Sıfat Yantümcesi Belirticisi (Ortaç)
{-(y)An}	YtuOrAn	Sıfat Yantümcesi Belirticisi (Ortaç)
{-(y)AcAK}	YtuOrAcak	Sıfat Yantümcesi Belirticisi (Ortaç)
{-(y)Iş}	YtuOrIs	Sıfat Yantümcesi Belirticisi (Ortaç)
{-DIK}	YtuUDik	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)AcAK}	YtuUAcak	Belirteç Yantümcesi Belirticisi (Ulaç)
{-mA}	YtuUMa	Belirteç Yantümcesi Belirticisi (Ulaç)

{-mAK}	YtuUMak	Belirteç Yantümcesi Belirticisi (Ulaç)
{-mAdAn}	YtuUMdn	Belirteç Yantümcesi Belirticisi (Ulaç)
{-mAzdAn}	YtuUMzdn	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)IncA}	YtuUInca	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)Arak	YtuUArak	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)AlI}	YtuUAli	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(A/I)rcAsIna / mIşcAsInA}	YtuURcsn	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)Ip}	YtuUIp	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)ken}	YtuUKen	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(y)A … -(y)A}	YtuU	Belirteç Yantümcesi Belirticisi (Ulaç)
{-DI … -(y)AlI	YtuU	Belirteç Yantümcesi Belirticisi (Ulaç)
{-(A/I)r … -mAz}	YtuU	Belirteç Yantümcesi Belirticisi (Ulaç)
Türetim Biçimbirimleri *(Yapım Ekleri)*		
Türetim sonucu bağımsız sözlüksel biçimbirimler oluşturan türetim biçimleri		
-A/E	TBAE:a	Addan Eylem boş boşa-
-A/E	TBEA:a	Eylemden Ad yar- yara
-AcAk	TBEA:acak	Eylemden Ad iç- içecek
-AcAn	TBAS:acan	Addan Sıfat baba babacan
-AcAn	TBES:acan	Eylem Sıfat sev- sevecen
-AğAn	TBES:agan	Eylemden Sıfat dur- durağan
-AlA	TBEE:ala	Eylemden Eylem serp- serpele-
-AlgA	TBEA:alga	Eylemden Ad çiz- çizelge
-AmAk	TBEA:amak	Eylemden Ad bas- basamak
-An	TBEA:an	Eylemden Ad çarp- çarpan
-AnAk	TBEA:anak	Eylemden Ad yet- yetenek
-CA	TBAA:ca	Addan Ad çekme çekmece
-CA	TBEA:ca	Eylemden Ad düşün- düşünce
-CA	TBSS:ca	Sıfattan Sıfat yavaş yavaşça
-CAk	TBAA:cak	Addan Ad yavru yavrucak
-CI	TBAA:ci	Addan Ad emek emekçi

-CIk	TBAA:cik	Addan Ad ada adacık
-CIl	TBAA:cil	Addan Ad balık balıkçıl
-(A)Ç	TBAA:ac	Addan Ad ana anaç
-(A)Ç	TBEA:ac	Eylemden Ad bağla bağlaç
-DA	TBBE:da	Yansıma şapır şapırda
-DA	TBAA:da	Addan Ad göz gözde
-DAm	TBAA:dam	Addan Ad yön yöntem
-Dan	TBAS:dan	Addan Sıfat iç içten
-DAş	TBAA:das	Addan Sıfat çağ çağdaş
-DI	TBEA:di	Eylemden Ad uy- uydu
-DIk	TBEA:dik	Eylemden Ad tanı- tanıdık
-GA	TBEA:ga	Eylemden Ad diz- dizge
-GAç	TBEA:gac	Eylemden Ad süz- süzgeç
-GAç	TBES:gac	Eylemden Sıfat utan- utangaç
-GAn	TBES:gan	Eylemden Sıfat atıl- atılgan
-GI	TBEA:gi	Eylemden Ad sil- silgi
-GIç	TBEA:gic	Eylemden Ad dal- dalgıç
-GIç	TBES:gic	Eylemden Sıfat bil- bilgiç
-GIn	TBES:gin	Eylemden Sıfat sar- sargın
-I	TBEA:i	Eylemden Ad yap- yapı
-I	TBEE:i	Eylemden Eylem kaz- kaz-ı-
-ICI	TBEA:ici	Eylemden Ad koş- koşucu
-ICI	TBES:ici	Eylemden Sıfat üz- üzücü
-IlI	TBES:ili	Eylemden Sıfat as- asılı
-(y)Iş	TBEA:is	Eylemden Ad yağ- yağış
-Iz	TBAA:iz	Addan Ad yavrucak yavrucağız
-(A)K	TBAE:k	Addan Eylem göz gözük-
-(A)K	TBAA:k	Addan Ad sol solak
-(A)K	TBEA:k	Eylemden Ad otla- otlak
-Kır		fış fışkır-
-(A)l	TBAE:l	Addan eylem koca kocal-
-(A)l	TBEA:l	Eylemden Ad oku- okul
-(A)l	TBAA:l	Addan Ad kum kumul
-lA	TBAE:la	Addan Eylem ak akla-
-leyin	TBAB:leyin	Addan Belirteç sabah sabahleyin
-LI	TBAS:li	Addan Sıfat ün ünlü

-sIz	TBAS:siz	Addan Sıfat ün ünsüz
-lIK	TBAA:lik	Addan Ad taş taşlık
-(A/I)M	TBEA:am	Eylemden Ad düzle- düzlem
-mA	TBEA:ma	Eylemden Ad yaz- yazma
-mAcA	TBEA:maca	Eylemden Ad düz- düzmece
-mAç	TBEA:mac	Eylemden Ad de- demeç
-mAk	TBEA:ma	Eylemden Ad ye- yemek
-mAn	TBEA:man	Eylemden Ad az- azman
-mAn	TBAS:man	Addan Ad uz uzman
-mAz	TBEA:maz	Eylemden Ad aç- açmaz
-mIk	TBEA:mik	Eylemden Ad kıy- kıymık
-mIş	TBEA:mis	Eylemden Ad er- ermiş
-msA	TBAE:msa	Addan Eylem ben benimse
-msA	TBSE:msa	Sıfattan Eylem az azımsa
-msA	TBEE:msa	Eylemden Eylem gül- gülümse-
-msI	TBSS:msi	Sıfattan Sıfat sarı sarımsı
-msI	TBAS:msi	Addan Sıfat hamur hamurumsu
-mtrak	TBSS:mtrak	Sıfattan Sıfat acı acımtrak
-(A/I)n	TBEA:in	Eylemden Ad tüt tütün
-(A/I)n	TBAA:in	Addan Ad kök köken
-(I)ncI	TBAS:inci	Addan sıfat bir birinci
-(I)nç	TBEA:nc	Eylemden Ad bas- basınç
-(I)ntI	TBEA:nti	Eylemden Ad yay- yayıntı
-(A/I)r	TBAE:r	Addan Eylem deli delir-
-(A/I)r	TBEA:r	Eylemden Ad dön- döner
-rA	TBAB:ra	Addan belirteç son sonra
-rAk	TBAS:rak	Addan Sıfat küçük küçürek
-sA	TBAE:sa	Addan Eylem su susa
-sAk	TBEA:sak	Eylemden Ad tut- tutsak
-sAk	TBAS:sak	Addan Sıfat ırak ıraksak
-sAl	TBEA:sal	Eylemden Ad uy- uysal
-sAl	TBAS:sal	Addan Sıfat bölge bölgesel
-sI	TBEA:si	Eylemden Ad tüt- tütsü
-sI	TBAS:si	Addan Sıfat diken dikensi
-sI	TBEE:si	Eylemden Eylem yan- yansı-
-(A/I)t	TBEA:t	Eylemden Ad um- umut

-(A/I)t	TBAS:t	Addan Sıfat yaş yaşıt
-tAy	TBEA:tay	Eylemden Ad danış- Danıştay
-tAy	TBAA:tay	Addan Ad kurul kurultay
-tI	TBEA:ti	Eylemden Ad doğrul- doğrultu
-(A)v	TBEA:av	Eylemden Ad işle- işlev
-(A)y	TBEA:ay	Eylemden Ad dene- deney
-(A)y	TBAA:ay	Addan Ad yüz yüzey
-z	TBAA:z	Addan Ad iki ikiz

D. A Sample of Entire List of Suffixes

i	TBEAi	TB	e	DuEKipYet	DuEKipYet	l	TBEAl	TB
i	TBEEi	TB	m	DuAUyKT1	DuAUy	l	TBAAl	TB
i	DuADurBel	DuADur	m	DuEkGr1T1	DuEKGr1	l	TBAEl	TB
i	DuAUyKT3	DuAUy	m	TBEAm	TB	r	TBAEr	TB
i	KaySes	KS	m	DuEOlz	DuEOlz	r	TBEAr	TB
i	TBEAi	TB	di	TBEAdi	TB	r	DuECEttir	DuEC
i	TBEEi	TB	di	KDi	K	r	DuEZGen	DuEZ
i	DuADurBel	DuADur	di	DuEGBitD	DuEG	a	DuADurYon	DuADur
i	DuAUyKT3	DuAUy	di	DuEZGD	DuEZ	a	DuEKipA	DuEKipA
i	KaySes	KS	ğ>	DuEKGr1C1	DuEKGr1	a	TBAEa	TB
n	TBEAn	TB	ğ>	TBAEk	TB	a	TBEAa	TB
n	TBAAn	TB	ğ>	TBEAk	TB	a	DuEKipYet	DuEKipYet
n	KaySes	KS	ğ>	TBAAk	TB	in	DuEKGr3C2	DuEKGr3
y	TBEAy	TB	me	DuEOlz	DuEOlz	in	DuAUyKT2	DuAUy
y	TBEEy	TB	me	YtuAdlMa	Ytu	in	DuAUyKT3	DuAUy
y	KaySes	KS	me	YtuUMa	Ytu	diğ>	YtuAdlDik	Ytu
e	DuADurYon	DuADur	me	TBEAme	TB	diğ>	YtuOrDik	Ytu
e	DuEKipA	DuEKipA	dik	YtuAdlDik	Ytu	diğ>	YtuUDik	Ytu
e	TBAEa	TB	dik	YtuOrDik	Ytu	diğ>	TBEAdik	TB
e	TBEAa	TB	dik	YtuUDik	Ytu			
			dik	TBEAdik	TB			

E. A Sample of Proposed System's Result File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <File OriginalName="ornek.txt">
  - <Paragraph Index="0">
  - <Sentence Index="0">
    - <Word Index="0">
      <ValueOfWord>Dümenin</ValueOfWord>
      - <Root Index="0">
        <Value>Dümen</Value>
      - <Suffixes>
        - <SuffixCombination Index="0">
          <TBEAi>i</TBEAi>
          <TBEAn>n</TBEAn>
        </SuffixCombination>
        - <SuffixCombination Index="1">
          <TBEEi>i</TBEEi>
          <TBEAn>n</TBEAn>
        </SuffixCombination>
        - <SuffixCombination Index="2">
          <TBEAi>i</TBEAi>
          <TBAAn>n</TBAAn>
        </SuffixCombination>
        - <SuffixCombination Index="3">
          <TBEEi>i</TBEEi>
          <TBAAn>n</TBAAn>
        </SuffixCombination>
        - <SuffixCombination Index="4">
          <TBEAi>i</TBEAi>
          <KaySes>n</KaySes>
        </SuffixCombination>
        - <SuffixCombination Index="5">
          <TBEEi>i</TBEEi>
          <KaySes>n</KaySes>
        </SuffixCombination>
        - <SuffixCombination Index="6">
          <KaySes>i</KaySes>
```

```
      <KaySes>n</KaySes>
    </SuffixCombination>
    - <SuffixCombination Index="7">
      <DuEKGr3C2>in</DuEKGr3C2>
    </SuffixCombination>
    - <SuffixCombination Index="8">
      <DuAUyKT2>in</DuAUyKT2>
    </SuffixCombination>
    - <SuffixCombination Index="9">
      <DuAUyKT3>in</DuAUyKT3>
    </SuffixCombination>
    </Suffixes>
  </Root>
 </Word>
- <Word Index="1">
  <ValueOfWord>terbiye</ValueOfWord>
   - <Root Index="0">
  <Value>terbi</Value>
   - <Suffixes>
     - <SuffixCombination Index="0">
       <TBEAy>y</TBEAy>
       <DuADurYon>e</DuADurYon>
         </SuffixCombination>
     - <SuffixCombination Index="1">
       <TBEEy>y</TBEEy>
       <DuADurYon>e</DuADurYon>
         </SuffixCombination>
     - <SuffixCombination Index="2">
       <TBEAy>y</TBEAy>
       <DuEKipA>e</DuEKipA>
         </SuffixCombination>
     - <SuffixCombination Index="3">
       <TBEEy>y</TBEEy>
       <DuEKipA>e</DuEKipA>
         </SuffixCombination>
     - <SuffixCombination Index="4">
       <KaySes>y</KaySes>
```

```
        <DuEKipA>e</DuEKipA>
          </SuffixCombination>
      - <SuffixCombination Index="5">
        <TBEAy>y</TBEAy>
        <TBAEa>e</TBAEa>
          </SuffixCombination>
      - <SuffixCombination Index="6">
        <TBEEy>y</TBEEy>
        <TBAEa>e</TBAEa>
          </SuffixCombination>
      - <SuffixCombination Index="7">
        <TBEAy>y</TBEAy>
        <TBEAa>e</TBEAa>
          </SuffixCombination>
      - <SuffixCombination Index="8">
        <TBEEy>y</TBEEy>
        <TBEAa>e</TBEAa>
          </SuffixCombination>
      - <SuffixCombination Index="9">
        <TBEAy>y</TBEAy>
        <DuEKipYet>e</DuEKipYet>
          </SuffixCombination>
      - <SuffixCombination Index="10">
        <TBEEy>y</TBEEy>
        <DuEKipYet>e</DuEKipYet>
          </SuffixCombination>
      - <SuffixCombination Index="11">
        <KaySes>y</KaySes>
        <DuEKipYet>e</DuEKipYet>
       </SuffixCombination>
      </Suffixes>
    </Root>
    - <Root Index="1">
      <Value>terbiye</Value>
      <Suffixes />
    </Root>
</Word>
```

```
- <Word Index="2">
<ValueOfWord>edemediğini</ValueOfWord>
- <Root Index="0">
  <Value>ed</Value>
   - <Suffixes>
     - <SuffixCombination Index="0">
        <TBAEa>e</TBAEa>
        <TBEAm>m</TBEAm>
        <TBAEa>e</TBAEa>
        <TBEAdi>di</TBEAdi>
        <TBAEk>ğ</TBAEk>
        <TBEAi>i</TBEAi>
        <TBEAn>n</TBEAn>
        <TBEAi>i</TBEAi>
     </SuffixCombination>
     - <SuffixCombination Index="1">
       <TBEAa>e</TBEAa>
       <TBEAm>m</TBEAm>
      ...
- <Word Index="5">
   <ValueOfWord>eder</ValueOfWord>
   - <Root Index="0">
     <Value>ed</Value>
     - <Suffixes>
        - <SuffixCombination Index="0">
            <TBAEa>e</TBAEa>
            <TBAEr>r</TBAEr>
        </SuffixCombination>
        - <SuffixCombination Index="1">
            <TBEAa>e</TBEAa>
            <TBAEr>r</TBAEr>
        </SuffixCombination>
        - <SuffixCombination Index="2">
            <TBAEa>e</TBAEa>
            <TBEAr>r</TBEAr>
        </SuffixCombination>
        - <SuffixCombination Index="3">
```

```
            <TBEAa>e</TBEAa>
            <TBEAr>r</TBEAr>
        </SuffixCombination>
        - <SuffixCombination Index="4">
            <TBAEa>e</TBAEa>
            <DuECEttir>r</DuECEttir>
        </SuffixCombination>
        - <SuffixCombination Index="5">
            <TBEAa>e</TBEAa>
            <DuECEttir>r</DuECEttir>
        </SuffixCombination>
        - <SuffixCombination Index="6">
                <DuEKipA>e</DuEKipA>
                <DuEZGen>r</DuEZGen>
        </SuffixCombination>
        - <SuffixCombination Index="7">
                    <TBAEa>e</TBAEa>
                    <DuEZGen>r</DuEZGen>
        </SuffixCombination>
        - <SuffixCombination Index="8">
                    <TBEAa>e</TBEAa>
                    <DuEZGen>r</DuEZGen>
        </SuffixCombination>
        - <SuffixCombination Index="9">
                <DuEKipYet>e</DuEKipYet>
                <DuEZGen>r</DuEZGen>
        </SuffixCombination>
    </Suffixes>
</Root>
- <Root Index="1">
  <Value>ede</Value>
  - <Suffixes>
      - <SuffixCombination Index="0">
            <TBAEr>r</TBAEr>
        </SuffixCombination>
        - <SuffixCombination Index="1">
          <TBEAr>r</TBEAr>
```

```
                </SuffixCombination>
                - <SuffixCombination Index="2">
                  <DuECEttir>r</DuECEttir>
                </SuffixCombination>
                - <SuffixCombination Index="3">
                  <DuEZGen>r</DuEZGen>
                </SuffixCombination>
              </Suffixes>
            </Root>
          </Word>
        </Sentence>
      </Paragraph>
    </File>
```