COMPUTER PROCESSING OF TURKISH: MORPHOLOGICAL AND LEXICAL INVESTIGATION

by

TUNGA GÜNGÖR

B.S. in Computer Engineering, Boğaziçi University, 1987

M.S. in Computer Engineering, Boğaziçi University, 1989

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor
of
Philosophy

Boğaziçi University

1995

ACKNOWLEDGEMENTS

I would like to thank to Prof.Dr. Selahattin Kuru for his great help and guidance as the supervisor of this thesis. I also thank to Doç.Dr. H. Levent Akın, Doç.Dr. Taflan Gündem, Prof.Dr. Sumru Özsoy, and Prof.Dr. Nadir Yücel for both their helpful comments and serving on my thesis committee. In particular, I would like to express my gratitude to Prof.Dr. Sumru Özsoy for her invaluable guidance in reviewing many points and improving the linguistic content of the thesis.

I would like to thank to Boğaziçi University, TÜBİTAK, and Digital Equipment Corporation (DEC) for their support during the preparation of this thesis. Finally, I wish to thank to Kaan Arslanoğlu, Osman Ataker, Gülay Göktürk, and Yavuz Şanes who have contributed in forming the corpus used in this research; and to Mine Dilgimen Odok for her help.

ABSTRACT

The morphological analysis of Turkish is the subject of this thesis. Turkish belongs to the group of agglutinative languages. Because of its agglutinative nature, Turkish morphology is quite complex and includes many exceptional cases. Most recent research on Turkish morphology have limited themselves with a partial treatment of the language. The study has concentrated especially on the explanation and representation of the basic rules. The main objective of this thesis is to bring the full morphological structure of Turkish to light and to build its computer representation. Before this analysis is handled, the syntactic or semantic parsing of the language is quite impossible.

In this study, we divide the analysis of the morphology into two interrelated parts: morphophonemic analysis and morphotactic analysis. We investigate and define the morphological structure for both of these. Then we combine these in the Augmented Transition Network (ATN) formalism. This forms the formal representation of the Turkish morphological structure. This proposed morphological structure forms a basis for the language applications about Turkish. Among these applications, we design and implement a morphological parser and a spelling checker which incorporates a spelling corrector component.

We perform statistical analysis of Turkish based on this morphological representation and the implemented programs. This analysis is formed of two parts: lexical and morphological analysis, and corpus analysis. The first one uses the information about the structural parts of the language. The second one deals with the daily usage of the language. For this purpose, we form a corpus and run the spelling checker program on this corpus.

Key words : Computational linguistics, Natural language processing, Morphological analysis, Turkish, Augmented transition networks, Spelling checking, Corpus

ÖZET

Bu tezin konusu Türkçe'nin biçimbiliminin incelenmesidir. Türkçe bitişimli diller grubuna dahildir. Bu özelliğinden dolayı Türkçe'nin biçimbilimi oldukça karmaşıktır ve pek çok istisnai durumlar içerir. Son zamanlarda yapılan çalışmalarda dilin sadece bir kısmı incelenmiştir. Bu çalışmalar özellikle temel bazı kuralların açıklanması ve gösterilmesi üzerine kurulmuştur. Bu tezin başlıca amacı, Türkçe'nin tüm biçimbilimsel yapısını açığa çıkarmak ve bu yapının bilgisayardaki gösterimini oluşturmaktır. Bu işlem tamamlanmadan, dilin sözdizimsel ve anlambilimsel olarak incelenmesi hemen hemen olanaksızdır.

Bu çalışmada biçimbilim analizini iki kısma ayırıyoruz: Eklerin yapısının ve sıralanmasının incelenmesi. Dilin biçimbilimini bu iki kısmı göz önüne alarak tanımlayacağız. Daha sonra bu tanımlamaları Genişletilmiş Geçiş Ağı formasyonunda birleştireceğiz. Böylece, Türkçe'nin biçimbilimsel yapısının bilgisayardaki gösterimini elde edeceğiz. Bu önerilen yapı, Türkçe konusundaki dil uygulamaları için bir temel oluşturacaktır. Bu uygulamalar arasından, bir biçimbilimsel tarama programı ile yazım düzeltme elemanı da içeren bir yazım kontrol programı hazırlayacağız.

Bu biçimbilimsel gösterimi ve hazırlanan programları kullanarak Türkçe hakkında istatistik bilgi üreteceğiz. Bu üretim iki kısımdan oluşuyor: sözlük ve biçimbilimi analizi ile metin analizi. Bunlardan ilki dilin yapısal kısımları hakkındaki bilgilerden yararlanır. İkincisi ise dilin günlük kullanımıyla ilgilidir. Bu amaçla, bir metin oluşturacağız ve yazım kontrol programını bu metin üzerinde çalıştıracağız.

Anahtar sözcükler : Bilgisayarlı dilbilimi, Doğal dil işleme, Biçimbilimsel analiz, Türkçe, Genişletilmiş geçiş ağları, Yazım kontrolu, Metin

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Computational linguistics is a field of study that has developed from an intersection between the general discipline of linguistics and the subset of computer science called artificial intelligence. It aims to combine these two disciplines in order to solve the linguistic problems by the use of the computers. From the point of view of a computer scientist studying in the field of artificial intelligence, the motivation is clear: bringing computers closer to human activities. Linguistic issues are solved very easily and even without a notice by human beings. Thus this forms an important step that should be worked on in order to close the gap between us and the computers.

One important and more and more developing subfield of computational linguistics is referred to as *natural language processing*. When humans communicate with each other using language, they employ, almost effortlessly, extremely complex and still little understood processes. It has been very difficult to develop computer systems capable of generating and understanding even fragments of a natural language, such as English. The main source of the difficulty is the absence of the suitable structures (like the structures in the human brain) that are responsible of processing the natural language in a computer. Thus generating and understanding language forms an area of high complexity.

Natural language processing is a broad concept by itself. The problem that it deals with is defined more or less clearly: understanding natural language and performing various tasks based on this understanding. But the question immediately follows: How? How can a computer understand a natural language? The answer to this question is not a simple one and it conceals many theoretical and technical issues that must be tangled with.

In order to be able to cope with this problem, it ought to be brought into a more manageable level. For this reason, natural language processing has been divided into several parts. We can group these parts in the following way [1,2]:

Acoustic-phonetic: This is the process at the core level. Its function is to convert spoken language into written language. It takes the sounds (presumably represented as a plot of how much energy is coming at various sound frequencies) and translates the input into words.

Morphological: The input to this process is the words represented in written language. It takes these words, processes them with the use of several language elements (lexicon, morphological rules, etc.), and converts them into some kind of representation which will be utilized in the next step. This representation makes explicit the parts of each word (the root word, the category and the features of the word, the affixes, etc.).

Syntactic: The individual words decoded by the previous step are combined into larger language elements and processed in order to check if they conform with the grammatical rules of the language. The syntax of the language represented in a predefined formalism helps us to achieve this goal. The output of this step is the internal representation of these language elements [3,4].

Semantic-pragmatic: This final process is responsible of taking the syntactic representation and "understanding" it. This part is the most mysterious of all; it somehow uses the information in an appropriate way [5,6].

The whole process follows the sequence described above in the sense that one part requires that the previous one has been handled successfully and it must be handled before the next one. However, there is also a close relationship between them. For example, some of the morphological issues may only be solved by some amount of syntactic and even semantic knowledge.

The second one of the steps defined above, namely morphological analysis, forms the subject of the research in this thesis. We can define morphology as the study of the word structure [7]. It is clear that the words of a language are not totally separate items without any similarities between them or regularities within their structures. It has been traditional within linguistics to describe the regularities in the structure of words by postulating that a single word can be viewed as made up of one or more smaller units, called *morphemes*.

Morphological analysis aims to define the regularities in the word structure. It is of practical relevance to constructing a dictionary. If there is no need to store every form of a word because the various forms can be computed by some rules, the dictionary will be smaller. For a language like Turkish, this will be a large benefit.

1.1. Motivation

Turkish belongs to the group of agglutinative languages. A language in which the process of adding affixes (prefixes, infixes, and suffixes) to the root form is extremely productive is called an *agglutinative language*. Finnish, Hungarian, Quechua, and Swahili are other examples of agglutinative languages. In such a language, given a word in its root form, we can derive a new word by adding an affix to this root form, then derive another word by adding another affix to this new word, and so on. This iteration process may continue several levels. It is not unusual to find words obtained by affixing ten or fifteen suffixes to a word in its root form. A single word in an agglutinative language may correspond to a phrase made up of several words in a non-agglutinative language. The large number of suffixes and the combination of these suffixes in different orders lead to a large number of words. It is pointed out in [8,9] that it is possible to obtain over 10,000,000 words from a single noun in its root form in Turkish.

This distinctive characteristic of agglutinative languages can be exemplified by the following popular Turkish word:

Çekoslovakyalılaştıramadıklarımızdanmışsınız

whose equivalent in English is "(it is speculated that) you have been one of those whom we could not convert to a Czechoslovakian." In this example, one word in Turkish corresponds to a full sentence in English. The root word is *Çekoslovakya* (Czechoslovakia) and there are ten suffixes affixed to this root word.

Because of its agglutinative nature, Turkish morphology is quite complex and includes many exceptional cases. Most recent research on Turkish morphology have limited themselves with a partial treatment of the language. The study has concentrated especially on the explanation and representation of the basic rules. The exceptional cases have been omitted totally or, at best, different structures have been included in the analysis to handle some of them. For instance, [9] handles one of the frequently used rules of Turkish, the consonant harmony rule, by storing two different forms of the word in the lexicon.

The motivation of the research in this thesis is to bring the full morphological structure of Turkish to light and to build its computer representation. We can define morphology as the study of

the word structure. This type of study has a direct and an indirect consequence. The direct consequence allows us to understand the structure of the language at word level. In Turkish, word formation is a highly complex process. There are several rules that must be taken into account and, in addition, almost every rule comes with an exception. A complete study of this process is necessary in order to formalise the use of the language.

The indirect consequence is related with the further studies on the language. For all agglutinative languages in general and for Turkish in particular, morphological analysis forms the basic step of all language applications based on written language. As mentioned above, Turkish morphology is quite complex and the process of word formation is extremely productive. This productive nature forces us to have a thorough morphological analysis for the language. Before this morphological analysis is handled, the syntactic or semantic parsing of the language is quite impossible.

As an example, consider a natural language front end (NLFE) system which intends to cover most of the language. The input to the system will be Turkish text. As a generally accepted assumption, the system will own a lexicon (or dictionary) which stores information about the words and a parser (a sentence processing component). When a sentence is read, it must be parsed (analyzed) to see if it conforms with the syntax of the language and then it must be converted into some kind of internal representation provided that it is grammatically well-formed. The parsing and conversion processes are necessary since the following modules (whatever they are - a question-answering module, a language translation module, etc.) will operate on this well-formed representation.

While parsing the sentence, the parser must comprehend each word and consults the lexicon for this purpose. Thus the lexicon must be capable of giving information for each word. The necessity of a morphological component reveals itself at this step. Without such a component, the lexicon would be the unique source of information and the parser would expect to find in the lexicon every word that it encounters. However the number of Turkish words is almost infinitive and beyond the scope of any resource. The only solution is to support the lexicon with a morphological component in such a way to derive the missing words in the lexicon by the help of this component.

1.2. Subject of Thesis

The research in this thesis is about the morphology of the Turkish language. The main objective is to formalise the full Turkish morphology. We aim to achieve this objective in two parts. First we shall examine the Turkish morphology from a linguistic point of view. The linguistic literature owns a rich set of methods that we shall make use of from time to time. The result of our investigation will yield a definition of the Turkish morphology. Our main concern during this process will be to form a complete and uniform definition.

The second part will be about the representation of this definition of the Turkish morphology in the computer. We shall investigate several structures that are suitable for the representation of the Turkish language. These structures will be compared taking into account the agglutinative nature of Turkish and the special difficulties owned by the Turkish morphology. We shall then describe the representation formalism that we have chosen and explain in detail the reasons behind this choice. The morphological structure that will be proposed will then serve as the basis for computer applications about the language.

Both of the steps sketched above have their own difficulties which we shall attempt to solve. First of all, it is very difficult to find references on Turkish morphology in which the word formation rules are well-defined. We have referred to many grammar books to collect the rules [10,11,12,13]. In those books, after each rule is defined, usually it is reminded that there may occur some exceptions to that rule in some conditions, but mostly those conditions can not be well defined. For example, in all Turkish grammar books, it is said that "when a Turkish word ending with one of the consonants *p,ç,t,k* receives a suffix beginning with a consonant, that final consonant turns into a voiced consonant, but there are some such words whose final consonants do not change." However, none of the books says what the common property of those words which do not obey to that rule is, because most probably it is not known yet.

Some of the irregularities encountered in the Turkish language are even not mentioned in any of the grammar books. For example, although in some (but not all) of the grammar books we can see the rule "The verbal roots *de* (to say) and *ye* (to eat) change as *di* and *yi*, respectively, when they receive a suffix beginning with the consonant *y*," it is mentioned nowhere that the root *de* (to say) does not always obey this rule. For instance, it does not change when it receives the suffix *-yip*, i.e. the resulting word is not *diyip* as implied by the rule, but *deyip* (by saying).

The collection of the Turkish suffixes, especially the derivational ones, is much more difficult than the rules. Grammar books do not include the suffixes, or at best only some frequently used suffixes are shown during the description of the rules. We could find two references, [14,15], which give a detailed documentation of the suffixes. Another point that has been left unclear is the order of the suffixes during the affixation process, i.e. in which order the suffixes are affixed to a word and the interrelations between these suffixes.

The main cause of these difficulties is that the formalization process of Turkish morphology is relatively new. Until recently, an informal explanation of the grammar was enough for people to grasp the main ideas. In order to develop applications based on Turkish language, it is necessary first to define the morphology in a clear way. Even though it is claimed that Turkish word formation rules are well-defined and Turkish is a very regular language, as used today it shows many irregularities that cause the problem of parsing this language to become a hard and interesting problem.

Another part of the thesis is to design and implement a spelling checker program based on the defined morphological structure. A spelling checker is a program that parses a word and decides whether it is grammatically correct or not. The analysis is at the word level, i.e. each word is checked one by one and without taking into account the syntactic and semantic relations between consecutive words. A spelling checker program may have some additional features. For example, the same word written twice consecutively can be displayed to the user as a possible spelling error. The program that is presented in the thesis also includes a spelling corrector component. For a misspelled word, some alternatives are given according to a spelling corrector algorithm.

The statistics about the structure and the usage of a language is an important source of information. We shall analyze Turkish from a statistical point of view. This process is formed of two parts. The first one is an analysis of lexical and morphological elements. For this purpose, the morphological structure is used to obtain statistical data about the lexicon, the rules, and the suffixes. The second one is corpus analysis. This gives us results about how the language is used in daily life. For this, we shall use the spelling checker program and run it on a corpus formed of different topics.

1.3. Work Accomplished

Below we list the work accomplished in this research. Each of these items will be explained in the following chapters:

- An Augmented Transition Network (ATN) formalism is introduced for Turkish morphology, containing all of the categories and the suffixes. This includes 14 categories and about 200 suffixes.
- A root lexicon of about 21,500 words and a proper noun lexicon of about 11,500 words are formed in parallel to the ATN formalism.
- A parser and a spelling checker (including a spelling corrector) are implemented for Turkish to test the completeness (coverage) and the efficiency of the formalism.
- A test environment comprising of these elements is produced to study and test morphological properties of Turkish.
- The lexicon is analyzed to obtain statistics on the structural and usage patterns of the Turkish morphology.
- A corpus of about 2,200,000 words, which is currently the largest corpus on Turkish, is formed.
- This corpus is analyzed to obtain statistical properties of Turkish.

During this research, the following papers were produced:

- *Representation of Turkish Morphology in ATN*, Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks, İstanbul, 1993 [16]
- *A Spelling Checker and Corrector for Turkish*, Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks, İstanbul, 1993 [17]
- *Full Turkish Morphology Represented as an Augmented Transition Network*, submitted to Literary and Linguistic Computing [18]
- *Lexical and Morphological Statistics About Turkish Language*, in preparation

The research projects completed are:

- *Development of a Spelling Checker for Turkish*, Boğaziçi University Research Fund, Grant no. 92A0118

- *Development of a Spelling Checker for Turkish*, Scientific and Technical Research Council of Turkey (TÜBİTAK), Grant no. EEEAG-11
- *Development of a Spelling Checker for Turkish and Integration into ALL-IN-1*, Digital Equipment Corporation EERP program, Grant no. TK-001

1.4. Outline of the Thesis

The outline of the thesis is as follows: Chapter 2 gives a survey of the literature on the morphological analysis of agglutinative languages and especially of Turkish. Chapter 3 defines and discusses the formalisms that can be used for the representation of the morphology. Chapter 4 is devoted to the investigation of Turkish morphology from a linguistic point of view. In this chapter, we divide the analysis into two parts, namely morphophonemics and morphotactics. For each of these, the rules and the exceptional situations are derived. Chapter 5 is about the representation of the information of the previous chapter in ATN formalism. The full representation schema is given and the use of this mechanism is exemplified by some derivations. The following three chapters are based on the morphological structure presented in this chapter. In chapter 6, we analyze the lexical and morphological aspects of Turkish in order to obtain statistical data. The next chapter is about the corpus statistics. A large corpus is used to obtain the results of this chapter. Chapter 8 explains the design and implementation of the tools used in this study. The spelling checker and the spelling corrector programs are explained. The last chapter is a conclusion.

II. LITERATURE SURVEY

For a long time, the research on morphological analysis has concentrated on languages, mainly English, whose morphological structures were relatively simple. In these languages, there are a small number of affixes and these affixes do not undergo much change during the affixation process. As a result, words contain only a small number of affixes or none at all. This characteristic makes word derivation a trivial task for most cases. This simplicity of morphological structure has caused the researchers to develop and design simple algorithms and morphological components for these languages.

As the study on agglutinative morphology has begun, it became clear that a straightforward analysis was not enough to solve the newly-introduced problems of these languages. As opposed to non-agglutinative languages, there are a large number of suffixes and they change frequently according to the morphological rules. This indicates that words contain no direct indication of where the morpheme boundaries are, and furthermore morphemes take a shape dependent on the morphological environment.

This fact has forced the researchers to generate new techniques and to adopt the old techniques that had been widely used in other fields of natural language processing for the morphological analysis of agglutinative languages. Thus, a substantial increase in the research for agglutinative morphology has been observed.

The literature consists of several studies on the morphology of agglutinative languages in general and of Turkish in particular. One of the discussions facing agglutinative morphology is the direction of the analysis: whether it should be left-to-right (root matching) or right-to-left (suffix stripping). The former works by first determining a root word and then adding suffixes to this root, while the latter works by stripping off the suffixes at the end and arriving at a root word. Both approaches have been used from very early on in the history of morphological parsing.

Packard's [19] parser for ancient Greek proceeds by stripping affixes off the word, and then attempting to look up the remainder in a lexicon. Only if there is an entry in the lexicon matching the remainder and compatible with the stripped-off affixes is the parse deemed a success.

Brodda and Karlsson [20] apply a similar method to the analysis of Finnish, an agglutinative language, but without any lexicon of roots. Suffixes are stripped off from the end of the word until no more can be removed, and what is left is assumed to be a root.

Sagvall [21], on the other hand, devised a morphological analyzer for Russian which first looks in a lexicon for a root matching an initial substring of the word. It then uses grammatical information stored in the lexical entry to determine what possible suffixes may follow.

In the early 1980's, three different approaches to morphological analysis of agglutinative languages were developed independently: for Quechua [22,23], for Finnish [24], and for Turkish [25]. They all proceed from left to right, in the fashion of Sagvall's parser. Roots are sought in the lexicon that match initial substrings of the word, and the grammatical category of the root determines what class of suffixes may follow. When a suffix in the permitted class is found to match a further substring of the word, grammatical information in the lexical entry for that suffix determines once again what class of suffixes may follow. If the end of the word can be reached by the iteration of this process and if the last suffix analyzed is one which may end a word, the parse is successful.

Köksal [26] proposed a left-to-right parsing algorithm for Turkish in his Ph.D. thesis. This algorithm, called "Identified Maximum Match (IMM)" algorithm, tries to find the maximum length substring, which is present in a root dictionary. If a match is found, i.e. the root morpheme is identified, the remaining part of the word is considered as the search element for suffixes. This part is searched in a suffix morpheme forms dictionary and the morphemes are identified one by one. The process stops when nothing remains. However in some cases, although a solution is obtained, further consistency analysis proves that this solution is not the correct one. In such cases the previous pseudo-solution is reduced by one character and the search procedure is repeated.

Another discussion for agglutinative languages focuses on the question about what to include in the word lexicon. There are two approaches in general. At one extreme is to include all forms of all words in the lexicon without representation of internal structure and without representation of connections to morphologically related entries. This is referred to as *Full Listing Hypothesis (FLH)*. At the other extreme is to include only the basic (atomic) forms, i.e. root words and suffixes. The former approach excludes the need for morphological analysis, but is nearly impossible for agglutinative languages. There are several variations which mediate between these two extremes.

Taft and Forster [27] use a variation of FLH in which complex words have their own lexical entries, but the entries include a representation of morphological structure. For example, the word *unties* would have a lexical entry of its own, but that entry would contain a morphological analysis of the word, which is *un-tie-s*.

Bradley [28] and Lukatela et al. [29] devised another variation. In this algorithm, each word has its own entry but all entries for complex words are linked in the organization of the lexicon to a basic entry for the root word. For example, *unties* would have its own lexical entry in the lexicon, which would be linked to the basic entry *tie*. The basic entry is called the nucleus of the cluster of entries, and all the others are called satellites.

Laudanna and Burani [30] propose a full listing model in which there are entries for whole words and also entries for morphemes; a word is accessed via its full entry, and its morphological composition is given by pointers from the word entry to the component morpheme entries.

Other algorithms and discussions on these issues can be found in [8,9,31,32,33,34,35,36,37].

There are some well-known formalisms that can be used to represent the morphology of a language: Finite State Transition Network (FSTN), Recursive Transition Network (RTN), and Augmented Transition Network (ATN) [1,34,38,39]. In the next chapter, we shall give formal definitions of these formalisms and discuss the advantages and limitations of each one. In [32], Hankamer describes an FSTN representation of morphotactics and a treatment of morphophonemic alternations which employs generative-type rules operating cyclically (i.e. left-to-right) for Turkish morphology. Morphophonemic alternation is accounted for by listing roots and suffixes in a basic form, which is modified according to the surface environment when matching is done. The modifications correspond to the phonological rules of the language. The important point is that these rules do not have access to non-surface representations of previous or following morphemes, i.e. there are no dependencies between non-adjacent word segments. This is the basic fact that allows him to represent the morphotactics as an FSTN. The parser, named *keçi*, described in the paper, handles most of the Turkish morphology except some cases which require special treatment.

In [9,33], some points related to the Turkish morphology have been discussed. Among these points are the morphological characteristics of the language, the method that will be used, and the problems peculiar to Turkish. Then an algorithm based on these discussions and the proposed

solutions have been described. Being one of the first studies on Turkish morphology, some exceptions of the morphology were treated very simply in the analysis. This gives rise to some complications and to the expansion of the morphological structure of the language in a manner to include some grammatically wrong words. These shortcomings have been pointed out and discussed. The algorithm has also been evaluated in terms of performance and reliability.

A detailed analysis of Turkish morphology has been given by Solak [35]. In the paper, they discuss most of the morphophonemic rules accompanied with several examples and point out the exceptions of these rules. Root words are divided into two main groups: nominal and verbal. Based on this assumption, inflectional suffixes are also classified into two groups according to the root class that they can be affixed to. Then the order and the interrelation between the suffixes in each group are extracted.

The paper then describes the parser that has been designed. The parser utilizes a maximal match algorithm similar to that of Köksal [26]. In this algorithm, first the whole word is searched in the dictionary. If it is found then the word has no suffixes and therefore it does not need to be parsed. Otherwise, a letter is removed from the right and the resulting substring is searched. This process continues by removing letters from the right until a root is found. To reduce the parsing time, they make use of a preprocessing heuristic which analyzes the syllable structure of the word and accepts the word only if it conforms with the Turkish syllable structure rules. If not, the word does not need to be parsed and is marked as a wrong word.

In Stoop [40], the morphological analysis of Turkish forms the first step of a three step process which aims at a machine translation system for Dutch and Turkish. More concerned about syntactic structures and semantic analysis, the morphological part of the system limits itself with a subset of the morphophonemic rules and a small lexicon. The lexical categories used in the system are mainly derived from Lewis [41].

In his thesis, Darcan [36] explains the design of a portable natural language database interface system for Turkish. The system is complete in itself. It contains a morphological analyzer, a syntactic parser, a meaning representation and internal query generator module, and a translator which translates the internal query representation to a query in a declarative database query language. Being an interface for a limited domain, the morphological analyzer component is a simple one and has been

designed to take into account only the cases that could be encountered in this particular domain. The author pays particular attention to solve the problems peculiar to such a database interface system.

Two-level representation of the morphology has been proposed as an alternative description schema for agglutinative languages [24,42,43,44,45]. In [43,44], Koster and Willems make use of an AGFL (Affix Grammar over a Finite Lattice), which is a simple form of two-level grammars, in order to represent Turkish morphology. The morphemes are considered on two levels: the morphosyntactic level and the phonologic level, which correspond, roughly, to the morphotactics and the morphophonemics, respectively. Based on this description, the derivation of surface forms from abstract suffixes is defined as a two-step process: first there is a mapping between an abstract suffix (the form of the suffix as defined in the suffix lexicon) and an actual suffix (one of the etymologically different forms of the abstract suffix), second the abstract suffix gets its final surface shape based on the preceding morpheme according to morphophonemic rules.

In [42], an AGFL is used to describe the morphology and syntax of the Hungarian language. The authors explain some difficulties peculiar to the Hungarian language and discuss the usefulness of an affix grammar in handling these problems. In the paper, some aspects of the Hungarian morphology are examined and an experimental morphological analyzer is developed to handle a subset of the morphology. The authors leave open the question of how to represent the full morphology and propose the development of novel solutions.

In [24], Koskenniemi gives a two-level description of Finnish morphology. The model makes use of two-level rules which mediate between surface forms and lexical representations. Similarly, Oflazer [45] proposes a two-level description of Turkish morphology based on a lexical and a surface representation of word structures. The lexical level denotes the structure of the functional components of a word while the surface level denotes the standard orthographic realization of the word with the given lexical structure. The morphotactics is represented as an FSTN and the morphophonemics is compiled under 22 two-level rules.

III. INVESTIGATION OF FORMALISMS

In this chapter, we shall define and discuss some formalisms that can be used to represent the morphology of a language. For each of these formalisms, we shall first give its formal definition and then exemplify its use. The techniques that will be presented here are well-known in language theory. They are widely used for the development and analysis of programming (formal) languages. After the emergence of computational linguistics as a field, the researchers have begun to adapt these techniques for the processing of natural languages.

3.1. Finite State Transition Networks

A Finite State Automaton (FSA) is among the simplest computing machines that can be envisaged [34,38,46]. For a natural language application, we can refer to an automaton as a formalized definition of a process or mechanism for verifying the well-formedness of words or sentences. A given automaton normally produces only a pair of outputs: *True* (or *Yes*) if the word or the sentence is grammatical and *False* (or *No*) otherwise. Under these circumstances, it is customary to refer to the automaton as an *acceptor*. On the other hand, we might place additional requirements on the automaton such that it produce some non-trivial output; for example, translation from one language into another. Under these circumstances, we refer to the automaton as a *translator*.

A Finite State Transition Network (FSTN) can be regarded as a neutral description of a language (a set of sequences of symbols). It can also be interpreted, for instance, as a specification of an FSA to recognize elements of the language or as a specification of an FSA to generate elements of the language. We now give the formal definition of an FSTN:

An FSTN is a 5-tuple $M=(Q,A,K,q_0,F)$, where:

(1) Q is a finite set of states or nodes

(2) A is a finite set of acceptable input symbols

(3) K is a mapping from QXA to Q which dictates the behavior of the finite state control; K is called the state transition function

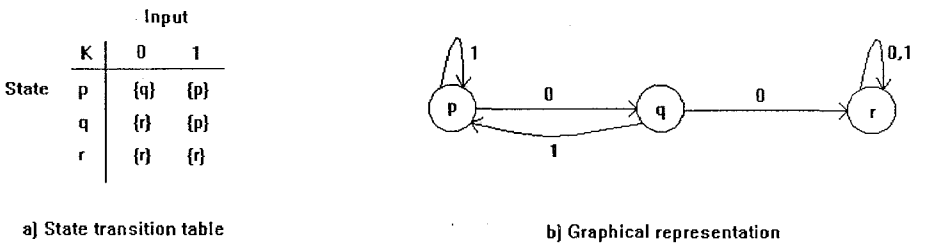
(4) q_0 in Q is the initial state in the network

(5) F in Q is the set of accepting or final states

The state transition function K is in the form $(q_i,a)=(q_{j1},q_{j2},...,q_{jn})$, where a is an element of A and $q_i,q_{j1},q_{j2},...,q_{jn}$ are elements of Q. When the system is in state q_i and the current input is a, then a transition can be made to either q_{j1} or q_{j2}, ..., or q_{jn}, as the next state. A given transition (q_i,a) is nondeterministic if there exists more than one option as the next state; it is deterministic as long as there is only a single possible transition to be made.

Note that the step-by-step behavior of these networks is defined as long as two items of information are known: the current state of the network and the input symbol being scanned. These two items of information, along with the remainder of the string to be scanned, completely determine the future behavior of the network. This is written as (q_i,w), where w is the sequence of symbols remaining to be read. (q_0,w) is called an *initial configuration* and (q_i,e), where q_i is an element of F and e denotes the empty string, is called a *final (or accepting) configuration.*

As a simple example, consider the FSTN M=($\{p,q,r\},\{0,1\},K,p,\{r\}$). Figure 3.1 shows the state transition table K and the graphical representation of M. M accepts all strings of *0*'s and *1*'s which have two consecutive *0*'s. That is, state p is the initial state and can be interpreted as "two consecutive *0*'s have not yet appeared, and the previous symbol was not a *0*". State q means "two consecutive *0*'s have not appeared, but the previous symbol was a *0*". State r means "two consecutive *0*'s have appeared." Note that once state r is entered, M remains in that state. Also note that M is a deterministic FSTN.



	K	Input 0	1
State	p	{q}	{p}
	q	{r}	{p}
	r	{r}	{r}

a) State transition table

b) Graphical representation

FIGURE 3.1. An example FSTN

On input *01001*, the only possible sequence of configurations, beginning with the initial configuration *(p,01001)* is: *(p,01001)* --> *(q,1001)* --> *(p,001)* --> *(q,01)* --> *(r,1)* --> *(r,e)*. Thus, the input *01001* is accepted by M.

3.2. Finite State Transducers

An FSA of the kind we have discussed in the previous section, used to analyze some existing input, is a recognizer, not a parser or a transducer, so all it can do is decide on the well-formedness of a string. If it can reach the end of the string in a final state, then the string is well-formed; if it can not reach the end of the string, or if can not reach the end of the string and simultaneously be in a final state, then the string is not well-formed. That is all the information it provides. To get more information, we need a parser or a transducer, not a recognizer.

Although we can interpret an FSTN as a machine that provides a simple *yes* or *no* output for any input, with some small extensions to the notation, we can interpret one as a Finite State Transducer (FST) [47,48]. An FST is simply a recognizer which emits an output string during each move made (the output may be *e*, however). The definition of an FST is as follows.

An FST is a 6-tuple M=(Q,A,O,K,q_0,F), where:

(1) Q is a finite set of states or nodes

(2) A is a finite set of acceptable input symbols

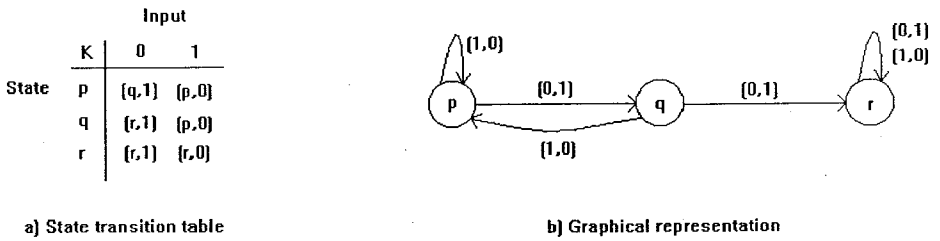
(3) O is a finite set of acceptable output symbols

(4) K is a mapping from QXA to QXO^*, where O^* represents n-times multiplication $OXOX...XO$, n>=0. K is called the state transition function

(5) q_0 in Q is the initial state

(6) F in Q is the set of accepting or final states

The operation of an FST is similar to that of an FSTN. The only difference is that, in addition to the next state, each move also emits an output.



a) State transition table

b) Graphical representation

FIGURE 3.2. An example FST

As an example, consider the FST M=({*p,q,r*},{*0,1*},{*0,1*},*K*,*p*,{*r*}). The state transition function *K* and the graphical representation are shown in Figure 3.2. This is a simple extension of the FSTN given previously in such a way that *0*'s and *1*'s in the input are replaced by *1*'s and *0*'s, respectively, in the output. For instance, the input *01001* results in the output *10110*. We can say that the output contains two consecutive *1*'s if and only if the input contains two consecutive *0*'s.

It is important to note that FSTN and FST have the same power. That is, we can always construct an FSTN for a given FST, and vice versa. In evaluating such formalisms, there are two relevant notions of adequacy: mathematical adequacy and notational adequacy.

Mathematical adequacy is concerned with whether the formal objects characterized by the notation, under the intended semantics, have the properties manifested in the real-world objects that the notation and its interpretation are intended to model. Although FSTNs can recognize non-finite languages (i.e. languages that contain an infinite set of strings), there are many non-finite languages that they can not recognize. Thus, it is easy to build an FSTN to recognize the language of strings consisting of any number of *a*'s (known as the language of $a^{\mathbf{n}}$) or the language of strings consisting of any number *m* of *a*'s followed by any number *n* of *b*'s (the language $a^{\mathbf{m}}b^{\mathbf{n}}$), but impossible to build an FSTN to recognize the language of strings consisting of some number *n* of *a*'s followed by the same number of *b*'s (the language $a^{\mathbf{n}}b^{\mathbf{n}}$), unless an upper bound is put on the size of *n*, in which case we cease to have a non-finite language. This is a failure of mathematical adequacy. From a linguistic point of view, strings of the general form $a^{\mathbf{n}}b^{\mathbf{n}}$ arise in a language when the language permits one string to be embedded inside another and puts no limit on such embedding.

Notational adequacy is to do with how elegantly the notation describes the real-world objects. In general, a short description is preferable to a longer one; repetition increases the possibility of errors in the use of the notation. An ideal notation allows one to exploit the similarities between different structures and state general properties when they exist. The agglutinative morphology and natural language grammars are rich of such similar constructions.

3.3. Recursive Transition Networks

Basically, a Recursive Transition Network (RTN) is just like an FSTN except that it introduces the extra concept of a named subnetwork [46,48]. That is, it is possible for an arc to name a subnetwork to be traversed, instead of a specific input that is to appear. The idea is that if we have a commonly used bunch of arcs, we can express this abstraction by making it into a self-contained, named network. This network can then be referenced by its name in a network that needs it, rather than having to appear expanded out in every place.

Informally, in an RTN, to traverse an arc that is labelled with a subnetwork name, it is necessary to traverse the subnetwork named, but remembering where to resume when that has been done. This purpose is achieved by the use of a stack. For an RTN, network traversal is defined partially in terms of itself. This is the reason for the word "recursive" in recursive transition network.

An RTN is a 7-tuple $M=(Q,A,P,K,q_0,Z_0,F)$, where:

(1) Q is a finite set of states or nodes

(2) A is a finite set of acceptable input symbols

(3) P is a finite set of symbols that may appear on the stack

(4) K is a mapping from $QXAXP$ to QXP^*, where P^* represents n-times multiplication $PXPX...XP$, n>=0. K is called the state transition function

(5) q_0 in Q is the initial state

(6) Z_0 in P is the start symbol, i.e. the symbol that appears initially on the stack

(7) F in Q is the set of accepting or final states

A transition is written as (q_i,a,A) --> (q_j,B), whose interpretation is as follows: When in state q_i, if the input is a and the top-most symbol on the stack is A, then the next state becomes q_j and B is added to the stack. Reading the stack destroys the symbol being read (i.e. A is deleted from top of the stack before B is added). During a transition, the input symbol, the symbol read from the stack, or the symbol written to the stack may be empty.

As an example, consider the RTN M=($\{q_0,q_1,q_2\},\{0,1\},\{Z,0\},K,q_0,Z,\{q_0\}$). This is the RTN for the language 0^n1^n, where $n >= 0$. Figure 3.3 shows the state transition function K and the graphical representation of the network. For instance, with the input string *0011*, M makes the following sequence of moves: $(q_0,0011,Z)$ --> $(q_1,011,0Z)$ --> $(q_1,11,00Z)$ --> $(q_2,1,0Z)$ --> (q_2,e,Z) --> (q_0,e,e).

The use of the stack in an RTN allows us to deal naturally with some of the recursive structures in natural languages. In addition, there are some recursive language structures that can be recognized by RTNs but can not in principle be captured by FSTNs. The network shown above is such an example. In this respect, an RTN is more powerful than an FSTN.

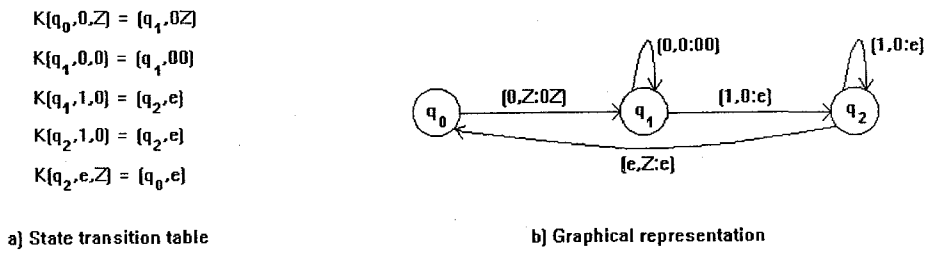


FIGURE 3.3. An example RTN

3.4. Augmented Transition Networks

An FSTN and an RTN have a common property: the order in which symbols appear in the output echoes the order in which corresponding symbols appear in the input. If we want to change the order, then we have to include a different set of arcs for each possible input-output pair. When the input and output languages have large vocabularies, this leads to large networks which can not be expressed concisely.

The main difference of an Augmented Transition Network (ATN) is the use of *registers* (variables) for storing information [1,48]. Values can be assigned to the registers and these values can be used for comparison. Each arc of the network may be annotated with instructions for how to shuffle information between these registers when it is traversed.

An ATN is an 8-tuple $M=(Q,A,P,K,O,R,q_0,Z_0)$, where:

(1) Q is a finite set of states or nodes
(2) A is a finite set of acceptable input symbols
(3) P is a finite set of symbols that may appear on the stack
(4) K is a mapping from $QXAXPXR_i$ to QXP^*XR_o, where R_i and R_o represent the contents of the registers before and after the transition is made, respectively.
(5) O is a finite set of operations for storing information in the registers
(6) R is a finite set of operations for reading information stored in the registers
(7) q_0 in Q is the initial state
(8) Z_0 in P is the start symbol, i.e. the symbol that appears initially on the stack

As a simple example, consider the ATN $M=(\{q_0,q_1\},\{0,1\},\{Z\},K,O,R,q_0,Z)$. We have a single register r_1. O consists of the operation *store(i)*, which stores the current input symbol i to the beginning of the register r_1. The set R is empty. This ATN accepts an input string of *0*'s and *1*'s, and stores the inverse of the string in the register r_1. Figure 3.4 shows the function K and the graphical representation.

FIGURE 3.4. An example ATN

The use of registers in an ATN enables what would be different paths through an RTN to be merged into one, and search to be avoided by storing information before its exact significance is known. The flavour of an ATN is different from other formalisms. Writing an ATN is much more like writing a computer program than writing either an FSTN or an RTN. For this purpose, ATNs are procedural formalisms whereas FSTNs and RTNs are essentially declarative.

IV. TURKISH MORPHOLOGY

In this chapter, we shall investigate Turkish morphology from a linguistic point of view. Before explaining the morphology, we must define the Turkish alphabet and the categorization of the letters in the Turkish alphabet:

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

Vowels = {a,e,ı,i,o,ö,u,ü}

Low vowels = {a,e,o,ö}

High vowels = {ı,i,u,ü}

Rounded vowels = {o,ö,u,ü}

Unrounded vowels = {a,e,ı,i}

Back vowels = {a,ı,o,u}

Front vowels = {e,i,ö,ü}

Consonants = {b,c,ç,d,f,g,ğ,h,j,k,l,m,n,p,r,s,ş,t,v,y,z}

Voiceless consonants = {ç,f,h,k,p,s,ş,t}

Voiced consonants = {b,c,d,g,ğ,j,l,m,n,r,v,y,z}

A remark about the definition of the alphabet is in order at this point. There are some words that have the same orthographical representation but different meanings. For example, the word *kar* means either "snow" or "profit". In order to differentiate the meanings, such words are pronounced differently and they own different phonetic representations.

Until recently, it was common among the linguistics to use some special phonemes in order to give different orthographical representations to such words. According to this system, the alphabet is enlarged with the three vowels *â*, *î*, and *û*, and different sounds of the phonemes are represented with these vowels. For instance, the word *kar* is separated into the forms *kar* (snow) and *kâr* (profit). These three vowels are the centralized forms of the back vowels *a, ı,* and *u*, respectively. The words that include these vowels are basically loanwords, originated mostly from the Arabic language.

Recent developments on linguistics and on Turkish tend to abandon this system for two reasons [13,49,50]. The first reason is the harmony between the eight vowels in the alphabet. These vowels and the categorization of these vowels form a sound mathematical model. The addition of new

vowels into this model means the disturbance of the harmony among them, which has negative effects on the formal analysis of the language. The second reason is related with the use of these three centralized vowels. It is a highly controversial subject among the language analysts when to use these vowels. There is not any common rule accepted by these people. Each new dictionary or spelling guide comes with its own rules.

In this thesis, as can be seen from the definition of the alphabet, we accept the modern linguistic approach explained above. We do not use these three centralized vowels. For the words that have the same orthographical shape but different phonetic representations, which of the possible meanings is used is deduced from the context.

We can examine the morphological analysis of Turkish (and of agglutinative languages in general) in two interrelated steps:

1. Morphophonemic analysis: There are some morphophonemic rules in Turkish used during the word formation process. These rules determine the form of the suffixes. According to some properties of a word, the form of a suffix that will be affixed to that word may change. For example, in the Turkish language, the first person singular possessive suffix may take one of the five forms *-ım, -im, -um, -üm*, or *-m* according to the last phoneme of the word. For the first four of these forms, the last phoneme is a consonant and the last vowel is in the set {a,ı}, {e,i}, {o,u}, and {ö,ü}, respectively. For the fifth form, the last phoneme is a vowel. So, with respect to this rule, the following derivations are valid (for the present, we represent the suffix as *-ım*):

kalem (pencil) + *-ım* ---> *kalemim* (my pencil) [1]
kol (arm) + *-ım* ---> *kolum* (my arm)
masa (table) + *-ım* ---> *masam* (my table)

It can be thought that these different forms of a suffix can be handled separately (as if they were different suffixes). In this case, the number of suffixes would be very large (a single suffix

[1] Throughout the thesis, we shall write Turkish words, phrases, and suffixes in italic; and the English translation of these words, phrases, and suffixes in parentheses. Word derivations will be shown in the following notation. The words and suffixes are written on the left-hand side of the arc separated by the plus sign, and the derived word is written on the right-hand side of the arc.

can have 24 different forms in Turkish - these different forms of a suffix are called *allomorphs*) and, worst of all, all of the morphotactic rules must be duplicated for each different form of a suffix.

2. Morphotactic analysis: The morphotactic analysis determines the order of the suffixes. That is, which suffixes can be affixed to a word in a predefined category (noun, verb, etc.) and in which order are these suffixes affixed. Words are grouped in different categories according to their functions and a suffix that can be affixed to a word in a particular category may or may not be affixed to a word in another category. Also, after a suffix is affixed to a word, some of the suffixes may be valid and the rest may not.

4.1. Morphophonemics

Turkish word formation uses a number of phonetic harmony rules. One of the most distinguishing characteristic of the Turkish language is its vowel and consonant harmony. While affixing a suffix to a word, the vowels and consonants change in certain ways such that these harmony constraints are not violated.

4.1.1. Vowel Harmony

The best known morphophonemic process in Turkish is the vowel harmony. Vowel harmony is a process in which the vowels in all syllables of a word except the first one assimilate to the preceding vowel with respect to certain phonetic features. Vowel harmony in Turkish is a left-to-right process operating sequentially from syllable to syllable. The rules are:

1. A non-initial vowel assimilates to the preceding vowel in frontness
2. A non-initial high vowel assimilates to the preceding vowel in rounding
3. A non-initial low vowel must be unrounded; that is, the vowels *o* and *ö* do not occur except in first syllables of the words.

The only exception to the third rule is the progressive tense suffix *-iyor*. In [51], it is pointed out that 96.74 per cent of the occurrences of the letter *ö* and 73.16 per cent of the occurrences of the

letter *o* correspond to the first or second characters of words. The remaining 3.26 per cent of the occurrences of *ö* all belong to foreign words, such as *akümülatör* (accumulator) or *aktör* (actor); and most of the remaining 26.84 per cent of the occurrences of *o* are due to the suffix *-iyor*.

The above rules imply that while any of the vowels may occur in the first syllable of a word, the vowel of the following syllable is restricted to a choice of two. That is, knowing the preceding vowel, the succeeding vowel must be one of the two possibilities. The features front/back and rounded/unrounded are entirely predictable, and only high/low remains distinctive. We can show this as follows:

1. If the preceding vowel is *a* or *ı*, then the succeeding vowel is either *a* or *ı*.
2. If the preceding vowel is *o* or *u*, then the succeeding vowel is either *a* or *u*.
3. If the preceding vowel is *e* or *i*, then the succeeding vowel is either *e* or *i*.
4. If the preceding vowel is *ö* or *ü*, then the succeeding vowel is either *e* or *ü*.

There are a large number of loanwords used in Turkish. Since most of these words do not obey to the vowel harmony rule, there are some roots that are not subject to vowel harmony internally. However, nearly all suffixes are in harmony with the vowel on their left.

As indicated above, there are no suffixes in which the low vowels *o* and *ö* appear except the suffix *-iyor*. Therefore, in citing suffixes, if we use the symbol *A* for a low vowel and *I* for a high vowel, their allophones will be as follows (an *allophone* is any of the variant forms of a phoneme as conditioned by position or adjoining sounds):

A = a | e

I = ı | i | u | ü

The symbol | indicates "or". The symbol *A*, for instance, means that the vowel it denotes is either *a* or *e*. Which one of these two must be used during affixation is decided by the vowel harmony rule. According to this definition, the negation suffix can be shown as *-mA* and the narrative past tense suffix as *-mIş*.

When a suffix is affixed to a root, the first vowel in the suffix changes according to the last vowel of the root. Succeeding vowels in the suffix change according to the vowel preceding them. If

we denote the preceding vowel (whether it is in the root or in the suffix) by *pv*, then the two classes of vowels are resolved as follows:

A	=	a	,	if *pv* is a | ı | o | u
	=	e	,	if *pv* is e | i | ö | ü

I	=	ı	,	if *pv* is a | ı
	=	i	,	if *pv* is e | i
	=	u	,	if *pv* is o | u
	=	ü	,	if *pv* is ö | ü

An allomorph is any of the variant forms of a morpheme. For example, the negation suffix *-mA* has two allomorphs, where narrative past tense suffix *-mIş* has four:

-mA	=	*-ma* | *-me*
-mIş	=	*-mış* | *-miş* | *-muş* | *-müş*

The allomorph of a suffix that is to be used is determined according to the phonemes of the root word that it is affixed to. For example, when the suffix *-mIş* is affixed to the root *gör* (to see), the allomorph *-müş* is used, because as the vowel preceding the vowel *I* is *ö* (*pv* = *ö*), *I* must resolve to an *ü* (i.e. *I* = *ü*):

gör (to see) + *-mIş* --> *görmüş* (he/she had seen)

There are also some non-harmonic suffixes, such as *-ken* and *-Iyor*, which are exceptions to harmonic conditioning from the vowel on their left. We shall distinguish the vowels that do not make allomorphs by not capitalizing them. That is, only the vowels written in capital form are subject to change according to the vowel harmony rule:

gel (to come) + *-Ir* + *-ken* --> *gelirken* (while coming)
gel (to come) + *-Iyor* --> *geliyor* (he/she is coming)

Similarly, the second vowel in compound verb suffixes (*-yAbil*, *-yAdur*, *-yAgel*, *-yAgör*, *-yAkal*, *-yAkoy*, *-yAyaz*, *-yIver*) does not change according to the preceding vowel either:

oku (to read) + *-yAbil* --> *okuyabil* (can read)

oku (to read) + *-yIver* --> *okuyuver* (just read)

giy (to dress) + *-In* + *-yAdur* --> *giyinedur* (go on dressing)

Such suffixes condition the vowel on their right normally:

okuyabil (can read) + *-Ir* --> *okuyabilir* (he/she can read)

okuyuver (just read) + *-dI* --> *okuyuverdi* (he/she just read)

Because of their different phonetic structures, some loanwords do not obey to the vowel harmony rule during agglutination. For example:

alkol (alcohol) + *-lI* --> *alkollü* (containing alcohol), not *alkollu*

saat (watch) + *-I* --> *saati* (the watch), not *saatı*

When certain suffixes beginning with a consonant are affixed to words ending with a consonant, a high vowel is inserted between them. We shall denote such vowels as *(I)*. The parentheses, (and), indicate that the phoneme inside is not affixed during the affixation process if the last phoneme of the word to which the suffix will be affixed is a vowel. This high vowel is also determined similarly as explained before. For example, the first person plural possessive suffix *(I)mIz* has eight different allomorphs:

(I)mIz = *-ımız* | *-imiz* | *-umuz* | *-ümüz*

= *-mız* | *-miz* | *-muz* | *-müz*

When this suffix is affixed to the root *kapı* (door), it takes the form *-mız*. The vowel *(I)* drops since the last phoneme of the word is a vowel (*ı*). But when it is affixed to the root *okul* (school), the allomorph *-umuz* is used. In this case, the vowel *(I)* is affixed since the last phoneme of the word is a consonant (*l*).

4.1.2. Consonant Harmony

Another basic concept of Turkish phonology is consonant harmony. The consonant harmony rules listed below are based on the classification of the consonants as voiceless and voiced:

1. In multi-syllabic words and in certain mono-syllabic roots, the final voiced consonants *b*, *c*, *d*, *g* (or *ğ*) generally (not always) become voiceless (i.e. it changes to *p*, *ç*, *t*, *k*, respectively) if the word is in root form or when a suffix beginning with a consonant is affixed:

akord (tune) --> *akort* (tune)
kitab (book) + *-lAr* --> *kitaplar* (books)
çiçeğ (flower) + *-lIk* --> *çiçeklik* (vase)
reng (color) + *-lI* --> *renkli* (with color)

but

ad (name) + *-lAr* --> *adlar* (names)

The change from *g* to *k*, as in the root *reng* (color) above, occurs in (some) words whose final consonant is *g* and the immediately preceding phoneme is *n*.

We represent the last phoneme of the words that are subject to the consonant harmony rule as *b*, *c*, *d*, *g* (or *ğ*), instead of the more traditional approach of representing them with *p*, *ç*, *t*, and *k* [13,50,52]. This approach of representing with voiced consonants is used by modern linguistic analysis of Turkish morphology. It eases the definition of the morphophonemic rules. Except some loanwords, there are no words in Turkish that end with the consonants *b*, *c*, *d*, or *g*. Thus, we can deduce that all the roots whose last phonemes are one of these consonants change these phonemes according to the above rules.

2. In some suffixes beginning with one of the consonants *c*, *d*, or *g*, this initial consonant might change according to the last phoneme of the word it follows. If we show these consonants as *C*, *D*, and *G*, their allophones will be:

C	=	*c*	|	*ç*
D	=	*d*	|	*t*
G	=	*g*	|	*k*

(Note that the phoneme *b* does not take place in this rule. Because, there are no suffixes in Turkish that begin with *b* such that this *b* can change). If the last phoneme of the word to which one of such suffixes is attached is a voiceless consonant, then the initial consonant of the suffix becomes voiceless (*ç*, *t*, or *k*, respectively), otherwise it remains as *c*, *d*, or *g*. Thus, the allomorphs of the definite past tense suffix *-DI* can be listed as:

-DI	= *-dı*	| *-di*	| *-du*	| *-dü*
	= *-tı*	| *-ti*	| *-tu*	| *-tü*

This can be exemplified as follows:

gel (to come) + *-DI* --> *geldi* (he/she came)
koş (to run) + *-DI* --> *koştu* (he/she ran)

In the case that both of the above rules apply to a word, the first rule takes precedence over the second one. In the following example, the final *b* changes into *p* since the suffix begins with a consonant, according to the first rule. Then, with respect to the second rule, the initial phoneme *C* of the suffix takes the form *ç* since the word ends with the voiceless consonant *p*:

kitab (book) + *-CI* --> *kitapçı* (book seller)

Furthermore, some suffixes beginning with a vowel are affixed to the words ending with a vowel with the insertion of one of the consonants *n*, *s*, *ş*, or *y*. We shall denote such consonants as *(n)*, *(s)*, *(ş)*, and *(y)*, respectively. Similar to their use for the vowels, the parentheses, (and), imply that the phoneme inside is not affixed if the last phoneme of the word is a consonant. For example, the genitive suffix can be shown as *-(n)In*, the third person singular possessive suffix as *-(s)I*, distributive numerical suffix as *-(ş)Ar*, and the acceleration suffix as *-(y)Iver*. As an example of the suffix *-(s)I*:

baş (head) + *-(s)I* --> *başı* (his/her head)
kapı (door) + *-(s)I* --> *kapısı* (his/her door)

The suffix *-(s)I* takes the form *-ı* in the first case. The phoneme *(s)* drops since the word ends with a consonant (*ş*). In the second case, since the last phoneme of the word is a vowel (*ı*), the suffix takes the form *-sı*.

There may be some exceptions to these morphophonemic rules. For instance, because of the former existence of an Arabic consonant not pronounced in Turkish, the consonant *s* is not inserted between some words ending with a vowel and the third person singular possessive suffix:

mevki (position) + *-(s)I* --> *mevkii* (his/her position), not *mevkisi*

For some such words both forms are valid:

cami (mosque) + *-(s)I* --> *camisi* (his/her mosque)
--> *camii* (his/her mosque)
sanayi (industry) + *-(s)I* --> *sanayisi* (his/her industry)
--> *sanayii* (his/her industry)

A similar case happens when a case suffix (*-DA*, *-DAn*, *-(y)A*, *-(y)I*) comes immediately after some pronouns such as *bu* (this), *şu* (that), *o* (it), *kendi* (self), after the relative suffix *-ki*, and after the third person possessive suffixes *-(s)I* and *-lArI*. In such cases an *n* is inserted in between:

bu (this) + *-(y)I* --> *bunu* (this one), not *buyu*
kendi (self) + *-DAn* --> *kendinden* (from himself/herself), not *kendiden*
seninki (yours) + *-(y)A* --> *seninkine* (to yours), not *seninkiye*
kapısı (his/her door) + *-DA* --> *kapısında* (at his/her door), not *kapısıda*

When all the rules above are considered, we reach the result that Turkish suffixes are highly changeable. As an example, the participle suffix *-DIĞ* [2] has 16 allomorphs:

---

[2] For some suffixes, the consonants *b*, *c*, *d*, *ğ* at the end of the suffix change when it is not followed by another suffix or it is followed by a suffix beginning with a consonant. The change is similar to the

-DIĞ = *-dığ* | *-diğ* | *-duğ* | *-düğ*
= *-tığ* | *-tiğ* | *-tuğ* | *-tüğ*
= *-dık* | *-dik* | *-duk* | *-dük*
= *-tık* | *-tik* | *-tuk* | *-tük*

Some examples are:

sat (to sell) + *-DIĞ* --> *sattık* (we sold)
sat (to sell) + *-DIĞ* + *-In* --> *sattığın* (the thing that you sold)

In the first word, the suffix takes the form *-tık*, because it follows the root *sat* (to sell) which ends with the voiceless consonant *t* (so, $D = t$), whose last vowel is *a* ($pv = a$, so $I = ı$), and the suffix is the last suffix affixed to the word (so, $\check{G} = k$). In the second word, there is another suffix whose first phoneme is a vowel (so, $\check{G} = ğ$).

4.1.3. Root Deformations

Normally Turkish roots are not flexed. However, there are some cases where some phonemes are changed by assimilation or various other deformations. An exceptional case related to the flexion of roots is observed in personal pronouns. When the first and second singular personal pronouns *ben* (I) and *sen* (you) take the dative suffix, they change as:

ben (I) + *-(y)A* --> *bana* (to me), not *bene*
sen (you) + *-(y)A* --> *sana* (to you), not *sene*

When these two roots take the plural suffix, their structures completely change:

ben (I) + *-lAr* --> *biz* (we), not *benler*
sen (you) + *-lAr* --> *siz* (you), not *senler*

---

first consonant harmony rule. Hence we treat these consonants like other phonemes that may have allophones. Their allophones are *p*, *ç*, *t*, and *k*, respectively.

A more systematic change occurs when the suffix *-(I)yor* comes after the verbs ending with the low vowel *A*. In such cases, the low vowel at the end of the word drops:

kapa (to close) + *-(I)yor* --> *kapıyor* (he/she is closing), not *kapayor*

As an exceptional case, when not only the suffix *(I)yor* but also any of the suffixes beginning with the consonant *y* is affixed to the roots *de* (to say) or *ye* (to eat), they change as *di* and *yi*, respectively. This exception is valid only for these two roots:

de (to say) + *-(I)yor* --> *diyor* (he/she is saying), not *deyor*
de (to say) + *-(y)An* --> *diyen* (the one who says), not *deyen*
ye (to eat) + *-(y)Ip* --> *yiyip* (by eating), not *yeyip*

However, the root *de* (to say) does not always obey to this exception, as in the following case:

de (to say) + *-(y)Ip* --> *deyip* (by saying), not *diyip*

One of the most important deformations in words occurs as the result of the first consonant harmony rule. This rule says that when some words ending with one of the voiced consonants *b*, *c*, *d*, *g* (or *ğ*) take a suffix beginning with a consonant, that voiced consonant changes into *p*, *ç*, *t*, or *k*, respectively:

dörd (four) + *-gen* --> *dörtgen* (quadrangle), not *dördgen*
tabağ (plate) + *-lAr* --> *tabaklar* (plates), not *tabağlar*
reng (color) + *-sIz* --> *renksiz* (without color), not *rengsiz*

A similar change occurs when a suffix beginning with a consonant is affixed to a word ending with *loğ*. In such a case, the final *ğ* changes into *g*:

psikoloğ (psychologist) + *-lAr* --> *psikologlar* (psychologists), not *psikoloğlar*

Another root deformation occurs as a vowel insertion. When a suffix beginning with a consonant comes after some nouns, generally designating parts of the human body, the vowel *I* is inserted before the last consonant. We represent the root form of these nouns without this vowel:

ağz (mouth) + *-DA* --> *ağızda* (in the mouth), not *ağzda*

Similarly, when some suffixes are affixed to verbs whose last vowels are omitted, the vowel *I* is inserted before the last consonant:

ayr (to separate) + *-Dl* --> *ayırdı* (he/she separated), not *ayrdı*

When a suffix beginning with a vowel is affixed to some originally Arabic roots ending with a consonant, the final consonant of the root is duplicated:

hak (right) + *-(I)m* --> *hakkım* (my right), not *hakım*
zan (opinion) + *-et* --> *zannet* (to think), not *zanet*

A compound word is a word that is formed of two other words. The meaning of the compound word may be a combination of the meanings of the underlying words or may be totally different. For example, *alınyazısı* (destiny) is formed of *alın* (forehead) and *yazı* (writing). When the plural suffix *-lAr* is affixed to compound words, a deformation occurs. This suffix, coming before the possessive suffix at the end of the word, forms a mid-fixing:

gözyaşı (tear) + *-lAr* --> *gözyaşları* (tears), not *gözyaşılar*

Sometimes, more than one deformation may happen on the same root:

rengeyiği (reindeer) + *-lAr* --> *rengeyikleri* (reindeers), not *rengeyiğiler* or *rengeyiğleri*
ademoğlu (mankind) + *-lAr* --> *ademoğulları* (mankinds), not *ademoğlular* or *ademoğlları*
kayd (registration) + *-lAr* --> *kayıtlar* (registrations), not *kaydlar* or *kayıdlar*

As can be noticed, the last phoneme of the words that are subject to the consonant harmony are shown with voiced consonants (e.g., *kitab* (book)), and the last vowel of words subject to the vowel insertion are dropped (e.g., *ağz* (mouth)). A similar approach is used by Hankamer [32]. However, when we look at the literature on Turkish morphology, we see that this representation schema is not used by most of the researchers [9,33,35,45] and is relatively new. They utilize the voiceless consonants for the consonant harmony and replace vowel insertion with the vowel ellipsis.

4.2. Morphotactics

We can classify the suffixes in two groups: inflectional (or conjugational) suffixes and derivational suffixes [9,10,14,17,33,35]. An inflectional suffix does not change the meaning of the word that it is affixed; it only adds something to the functional properties (such as the possession or the tense) of the word. An inflectional suffix that is defined for a word category can be affixed to all of the words in that category. A derivational suffix, on the other hand, changes the meaning of the word that it is affixed, i.e. it forms a new word. It can also change the category of the word; for example, a noun may be a verb after a derivational suffix is affixed. Also, the productivity of the derivational suffixes is highly changeable. Some of these suffixes are productive (i.e. can be affixed to nearly all of the words in the related category) while the others are not (i.e. can be affixed to a few words only).

In this section, we shall investigate the morphotactics of Turkish language mainly for the inflectional suffixes. At the end of the section, we shall give a brief explanation for the derivational suffixes. An inflectional suffix can be attached to a noun or a verb. The suffixes that can be received by either of these groups are different, i.e. a suffix which can be affixed to a noun can not be affixed to a verb with the same semantic function.

4.2.1. Nouns

The inflectional suffixes that can be affixed to nouns and the order of these suffixes are shown in Figure 4.1. All of these suffixes are optional.

The plural suffix *-lAr* is added directly to the root before any other suffix. In the plural forms of the pronouns *bu* (this), *şu* (that), and *o* (he/she/it), an *n* is inserted between the word and the suffix. For example:

bu (this) + *-lAr* --> *bunlar* (these), not *bular*

Possessive pronouns (in English: my, your, his/her/its, our, your, their) are represented by suffixes in Turkish. For example:

1. Plural suffix	:	*-lAr*	
2. Possessive suffixes	:	*-(I)m*	*-(I)mIz*
		-(I)n	*-(I)nIz*
		-(s)I	*-lArI*
3. Case suffixes	:	*-DA*	*-CA*
		-DAn	*-lI*
		-(y)A	*-sIz*
		-(y)I	*-(y)lA*
		-(y)InAn	*-(y)IlAn*
Genitive suffix	:	*-(n)In*	
4. Relative suffix	:	*-ki*	

FIGURE 4.1. Inflectional suffixes for nouns

ev (house) + *-(I)m* --> *evim* (my house)

araba (car) + *-(I)n* --> *araban* (your car)

If the possessed noun is plural, possessive suffixes come after the plural suffix:

ev (house) + *-lAr* + *-(I)m* --> *evlerim* (my houses)

araba (car) + *-lAr* + *-(I)n* --> *arabaların* (your cars)

When the third person plural suffix *-lArI* comes after a plural noun, two *lAr*'s combine and one of them drops:

ev (house) + *-lAr* + *-lArI* --> *evleri* (their houses), not *evlerleri*

Compound words have the third person singular possessive suffix already in their structure. For example, the last phoneme (*i*) in *ateşböceği* (firefly), the last two phonemes (*si*) in *safrakesesi* (gall bladder). Such words receive the possessive suffixes after removing the possessive suffix which is already in their structures:

ateşböceği (firefly) + *-(s)I* --> *ateşböceği* (the firefly), not *ateşböceğisi*

safrakesesi (gall bladder) + *-(I)nIz* --> *safrakeseniz* (your gall bladder), not *safrakesesiniz*

The root words *su* (water) and *ne* (what) create some irregular cases when they receive possessive suffixes:

su (water) + *-(I)m* --> *suyum* (my water), not *sum*
su (water) + *-(I)n* --> *suyun* (your water), not *sun*
su (water) + *-(s)I* --> *suyu* (his/her water), not *susu*
su (water) + *-(I)mIz* --> *suyumuz* (our water), not *sumuz*
su (water) + *-(I)nIz* --> *suyunuz* (your water), not *sunuz*
su (water) + *-lArI* --> *suları* (their water)

ne (what) + *-(I)m* --> *neyim* (what ... of me)
ne (what) + *-(I)n* --> *neyin* (what ... of you)
ne (what) + *-(s)I* --> *neyi* (what ... of he/she)
ne (what) + *-(I)mIz* --> *neyimiz* (what ... of us)
ne (what) + *-(I)nIz* --> *neyiniz* (what ... of you)
ne (what) + *-lArI* --> *neleri* (what ... of them)

For the root *ne* (what), the regular forms (*nem*, *nen*, *nesi*, *nemiz*, *neniz*, and *neleri*) are also valid, but mostly the irregular ones listed above are used.

Case suffixes can be grouped in two classes as internal and external case suffixes. Internal case suffixes are more frequently used than the external ones. They are named as follows: *-DA* (locative), *-DAn* (ablative), *-(y)A* (dative), and *-(y)I* (accusative).

Declensions of pronouns have some irregular forms. In the dative cases of *ben* (I) and *sen* (you), the front vowels become back (see section 4.1.3). In the genitive of *ben* (I) and *biz* (we), *-im* is used instead of the regular form *-in*:

ben (I) + *-(n)In* --> *benim* (my), not *benin*
biz (we) + *-(n)In* --> *bizim* (our), not *bizin*

Additionally, when a case suffix is attached to certain nouns, an *n* is put in before the case suffix. Among such nouns, we should add the compound words having the characteristics mentioned above:

ateşböceği (firefly) + *-(y)A* --> *ateşböceğine* (to the firefly), not *ateşböceğiye*

safrakesesi (gall bladder) + *-DA* --> *safrakesesinde* (in the gall bladder), not *safrakeseside*

The roots *su* and *ne* show exceptions for the genitive suffix, as for the possessive suffixes. In their genitive cases, a *y* is inserted instead of an *n*:

su (water) + *-(n)In* --> *suyun* (of the water), not *sunun*

ne (what) + *-(n)In* --> *neyin* (of what), not *nenin*

The relative suffix *-ki* may be added only to the genitive suffix or to the locative case suffix. For example:

kapı (door) + *-(n)In* + *-ki* --> *kapınınki* (the door's)

biz (we) + *-DA* + *-ki* --> *bizdeki* (the one which is in our hand)

It is possible to affix the relative suffix directly to a temporal adverb or a noun adverbially used, e.g.

demin (a while ago) + *-ki* --> *deminki* (of a while ago)

yarın (tomorrow) + *-ki* --> *yarınki* (tomorrow's)

or to a directional adverb or an adverb of place, e.g.

karşı (opposite side) + *-ki* --> *karşıki* (the one on the opposite side)

aşağı (lower part) + *-ki* --> *aşağıki* (the lower one)

The number of such roots is quite limited. A noun word that received the relative suffix may take the plural suffix and any case ending:

buradaki (the one who is here) + *-lAr* --> *buradakiler* (those who are here)

buradakiler (those who are here) + *-(y)lA* --> *buradakilerle* (with those who are here)

In its singular form, an *n* is inserted between *-ki* and the case ending:

buradaki (the one who is here) + *-DAn* --> *buradakinden* (from the one who is here)

In general, the relative suffix is not subject to vowel harmony. However, in the following cases, *-ki* changes into *-kü*:

dün (yesterday) + *-ki* --> *dünkü* (yesterday's)
bugün (today) + *-ki* --> *bugünkü* (today's)
gün (day) + *-ki* --> *günkü* (that day's)
öbür (other) + *-ki* --> *öbürkü* (the other one)

4.2.2. Verbs

The affixation process for verbs is more complex than for nouns. The suffixes and the order of these suffixes are shown in Figure 4.2.

There are four voices of verbs in Turkish: reflexive, reciprocal, causative, and passive. Combination of these suffixes are possible, with the following restrictions: The reflexive voice suffix *-(I)n*, the reciprocal voice suffix *-(I)ş*, and the causative voice suffixes *-Ar*, *-Ir*, and *-(I)t* are mutually exclusive, i.e. only one of these can be affixed to a verb. After these suffixes, the other causative voice suffixes *-DIr* and *-t*, and one of the passive voice suffixes *-(I)l* and *-(I)n* can be affixed, in this order. The following sequence of derivations is an example:

gör (to see) + *-(I)ş* --> *görüş* (to see each other)
görüş (to see each other) + *-DIr* --> *görüştür* (to cause to see each other)
görüştür (to cause to see each other) + *-(I)l* --> *görüştürül* (to be caused to see each other)

Neither the reflexive nor the reciprocal can be affixed to all verb roots. The following derivations are valid:

döv (to beat) + *-(I)n* --> *dövün* (to beat oneself)
anla (to understand) + *-(I)ş* --> *anlaş* (to understand each other)

1. Voice suffixes			
Reflexive voice suffix	:	*-(I)n*	
Reciprocal voice suffix	:	*-(I)ş*	
Causative voice suffixes	:	*-Ar*	
		-Ir	
		-(I)t	
		-DIr	
		-t	
Passive voice suffixes	:	*-(I)l*	
		-(I)n	
2. Negation suffixes	:	*-mA*	
		-(y)AmA	
3. Compound verb suffixes	:	*-(y)Adur*	*-(y)Akoy*
		-(y)Agel	*-(y)Ayaz*
		-(y)Agör	*-(y)Iver*
		-(y)Akal	
4. Mood suffix	:	*-(y)Abil*	
5. Main tense suffixes	:	*-(A)r*	*-(y)A*
		-DI	*-(y)AcAĞ*
		-(I)r	*-(y)AlIm*
		-(I)yor	*-(y)In*
		-mAktA	
		-mAlI	
		-mIş	
		-sA	
6. Question suffix	:	*-mI*	
7. Second tense suffixes	:	*-(y)DI*	
		-(y)mIş	
		-(y)sA	
8. Person suffixes	:	*-m*	*-(y)Im*
		-n	*-sIn*
		-k	*-(y)Iz*
		-nIz	*-sInIz*
		-lAr	
9. Predicate suffix	:	*-DIr*	

FIGURE 4.2. Inflectional suffixes for verbs

while the following are not:

koş (to run) + *-(I)n* --> *koşun*

oku (to read) + *-(I)ş* --> *okuş*

The causative voice of verbs takes various forms. The causative verb suffixes can be used repeatedly in the indicated order. The following sequence is an example of three causative suffixes used consecutively:

kapa (to close) + *-(I)t* --> *kapat* (to close)
kapat (to close) + *-DIr* --> *kapattır* (to have someone to close)
kapattır (to have someone to close) + *-t* --> *kapattırt* (to have someone to have someone to close)

The passive voice verb suffix also takes different forms. The rule that determines the form to be used is as follows: If the word ends with a vowel or the consonant *l* then the suffix *(I)n*, otherwise the suffix *-(I)l* is used:

öde (to pay) + *-(I)n* --> *öden* (to be paid)
bul (to find) + *-(I)n* --> *bulun* (to be found)
sev (to love) + *-(I)l* --> *sevil* (to be loved)

The passive and reflexive forms of some verbs have the same structure, but they differ in their meanings. For example, the verb *yıkan* is in passive voice in the sentence *bulaşık yıkandı* (the dishes were washed), where it is in reflexive voice in the sentence *Ali yıkandı* (Ali washed himself).

There are two suffixes which give a verb negative meaning: *-mA* (not) and *-(y)AmA* (can not). The suffix *-(y)AmA* is used to express impossibility:

söyle (to say) + *-mA* + *-m* --> *söylemem* (I don't say)
söyle (to say) + *-(y)AmA* + *-m* --> *söyleyemem* (I can't say)

Compound verb suffixes can be affixed to verbs to add them some extra meanings. Among them, the acceleration suffix *-(y)Iver* is the most frequently used one. The mood suffix *-(y)Abil* gives a verb possibility meaning. In some cases, a compound verb suffix can be followed by the mood suffix:

söyle (to say) + *-(y)Iver* --> *söyleyiver* (just say)
söyleyiver (just say) + *-(y)Abil* + *-Ir* --> *söyleyiverebilir* (he/she can say)

None of the compound verb suffixes can be used after the impossibility suffix *-(y)AmA*. Similarly, the suffixes *-(y)Akal*, *-(y)Akoy*, and *-(y)Ayaz* are not used after negation suffixes.

As causative and passive voice suffixes, the aorist suffix also changes according to some specific rules. In the negative form of a verb which is in present tense, the aorist suffix is not used. The first singular and plural person suffixes are directly affixed to the negation suffix, while the other person suffixes are affixed with the insertion of a *z* in between, as in the following example:

ver (to give) + *-mA* + *-m* --> *vermem* (I don't give)

but

ver (to give) + *-mA* + *-sIn* --> *vermezsin* (you don't give), not *vermesin*

The progressive tense suffix *-(I)yor* causes a deformation on some roots it is affixed to (see section 4.1.3). The same deformation occurs in the negation suffix when it is followed by the suffix *-(I)yor*:

sev (to love) + *-mA* + *-(I)yor* --> *sevmiyor* (he/she doesn't love), not *sevmeyor*

The suffix *-mAktA* can also be considered as a progressive tense suffix since it is used to indicate that an action continues in the present time.

There is no special suffix for imperative in Turkish. Whether a verb is in imperative form is understood through its person suffix. Every verb root can be considered as in the second person singular imperative form (for positive orders positive roots, for negative orders negative ones): e.g. *gel* (come), *kapama* (don't close).

The question suffix *-mI* is written separate from the word it follows; but it is subject to vowel harmony. Its place within the verb is not consistent; it may appear after the main tense suffix or after the person suffix, depending on the tense of the verb. It comes after the person suffix if the tense suffix is definite past (*-DI*), conditional (*-sA*), or optative (*-(y)A*):

gel (to come) + *-DI* + *-n* + *-mI* --> *geldin mi ?* (did you come?)

gel (to come) + *-sA* + *-m* + *-mI* --> *gelsem mi ?* (should I come?)

For the remaining tenses, the place of the question suffix is between the main tense suffix and the person suffix:

gel (to come) + *-(I)r* + *-mI* + *-(y)Iz* --> *gelir miyiz ?* (do we come?)

gel (to come) + *-(y)AcAĞ* + *-mI* + *-sIn* --> *gelecek misin ?* (will you come?)

No matter in which tense the verb is, the question suffix comes after the third person plural suffix:

gel (to come) + *-mAlI* + *-lAr* + *-mI* --> *gelmeliler mi ?* (must they come?)

gel (to come) + *-(I)yor* + *-lAr* + *-mI* --> *geliyorlar mı ?* (are they coming?)

In addition to the time concept coming from the main tense suffix, a second time may be added to a verb through the second tense suffixes. These suffixes are formed by removing the *i* from the definite past, narrative past, and conditional forms of the verb *imek*, i.e. *idi*, *imiş*, and *ise*, respectively:

gel (to come) + *-(I)yor* + *-(y)DI* + *-m* --> *geliyordum* (I was coming)

gel (to come) + *-(I)r* + *-(y)mIş* + *-sIn* --> *gelirmişsin* (I am told that you come)

gel (to come) + *-(y)AcAĞ* + *-(y)sA* + *-k* --> *geleceksek* (if we shall come)

When these forms are used as independent words, without being subject to the vowel harmony, they play the same role as the second tense suffixes:

gel (to come) + *-(I)yor* and *idi* + *-m* --> *geliyor idim* (I was coming)

gel (to come) + *-(I)r* and *imiş* + *-sIn* --> *gelir imişsin* (I am told that you come)

gel (to come) + *-(y)AcAĞ* and *ise* + *-k* --> *gelecek isek* (if we shall come)

The second tense suffixes are affixed to verbs ending with a vowel with the insertion of a *y* in between:

gel (to come) + *-sA* + *-(y)DI* --> *gelseydi* (if he/she came)
gel (to come) + *-(y)A* + *-(y)mIş* --> *geleymiş* (I wish he/she had come)
gel (to come) + *-mAlI* + *-(y)sA* --> *gelmeliyse* (if he/she must come)

The compound imperfect and conditional forms of the definite past tense can be used in two ways; the second tense suffix may come after or before the person suffix:

gel (to come) + *-DI* + *-n* + *-(y)DI* --> *geldindi* (you had come) or
gel (to come) + *-DI* + *-(y)DI* + *-n* --> *geldiydin* (you had come)

gel (to come) + *-DI* + *-k* + *-(y)sA* --> *geldikse* (if we came) or
gel (to come) + *-DI* + *-(y)sA* + *-k* --> *geldiysek* (if we came)

In the third person plural, the first form is more frequently used than the second:

gel (to come) + *-DI* + *-lAr* + *-(y)DI* --> *geldilerdi* (they had come)
gel (to come) + *-DI* + *-lAr* + *-(y)sA* --> *geldilerse* (if they came)

None of the second tense suffixes can be used without a main tense suffix. Additionally, the narrative second tense suffix can not be used with definite past tense suffix (*-DI*), and the conditional second tense suffix can not come after the optative (*-(y)A*) and the conditional (*-sA*) tense suffixes. That is, the following derivations are not valid:

oku (to read) + *-(y)DI* --> *okuydu*
oku (to read) + *-DI* + *-(y)mIş* --> *okuduymuş*
oku (to read) + *-sA* + *-(y)sA* --> *okusaysa*

For the person suffix, different suffixes are used to represent the first, second, and third singular, and plural persons. They also show differences depending on the main or second tense suffix they are affixed to.

Different person suffixes may have the same form. For example, the suffix *-sIn* may be the second person singular suffix or the third person singular suffix:

gel (to come) + *-mAlI* + *-sIn* --> *gelmelisin* (you must come)
gel (to come) + *-sIn* --> *gelsin* (let him/her come)

No suffix is used for the third singular person; if no person suffix exists in the verb its person is accepted as the third singular person:

gel (to come) + *-DI* --> *geldi* (he/she came)
gel (to come) + *-(I)r* + *-(y)sA* --> *gelirse* (if he/she comes)

The imperative form shows an exception in this rule. With this form, no suffix is used for the second singular person, while the suffix *-sIn* is used for the third singular person:

gel ([you] come)
gel (to come) + *-sIn* --> *gelsin* (let him/her come)

Additionally, the imperative forms of the first singular and plural persons do not exist.

4.2.3. Verbal Nouns

In Turkish, sentences can be classified as verb sentences and noun sentences. In verb sentences, there is an action, and this action is represented by a verb within the sentence: e.g. *okula gittim* (I went to the school). On the other hand, in a noun sentence there is no explicit verb: e.g. *öğrenciyim* (I am a student). The noun sentences of Turkish correspond to the sentences formed by the verb *to be* in English. In Turkish, instead of using an extra verb in such sentences, some suffixes which play the role of the verb *to be* in English are added to the subject of the sentence. These suffixes can be shown as in Figure 4.3.

Negation concept shows differences in noun and verb sentences. In a verb sentence, it is obtained by adding a negation suffix (*-mA* or *-(y)AmA*) to the verb of the sentence (see section 4.2.2): e.g. *okula gitmedim* (I didn't go to the school), where *gid* (to go) is the verb. There is no such a suffix

1. Question suffix	:	*-mI*	
2. Tense suffixes	:	*-(y)DI*	
		-(y)mIş	
		-(y)sA	
3. Person suffixes	:	*-m*	*-(y)Im*
		-n	*-sIn*
		-k	*-(y)Iz*
		-nIz	*-sInIz*
		-lAr	
4. Predicate suffix	:	*-DIr*	

FIGURE 4.3. Inflectional suffixes for verbal nouns

for the verbal noun of a noun sentence. Instead, the word *değil* (not) is used for this purpose: *öğrenci değilim* (I am not a student).

As for the verb sentences, interrogative noun sentences are formed by adding the question suffix:

gid (to go) + *-DI* + *-n* + *-mI* --> *gittin mi ?* (did you go?)
öğrenci (student) + *-mI* + *-sIn* --> *öğrenci misin ?* (are you a student?)

Time concept is given with the help of the tense suffixes in a noun sentence. As seen in Figure 4.3, there are three tense suffixes that can be added to a noun. They correspond to the second tense suffixes of verb sentences. Thus, they are the definite past, narrative past, and conditional forms of the verb *imek* (see section 4.2.2), and they may also be used as independent words, i.e. *idi*, *imiş*, and *ise*. That is, both of the following are valid:

öğrenci (student) + *-(y)DI* + *-m* --> *öğrenciydim* (I was a student)
öğrenci (student) and *idi* + *-m* --> *öğrenci idim* (I was a student)

To express the remaining tenses and modes apart from these three tenses in noun sentences, the infinitive *ol* (to become) is used:

öğrenci (student) and *ol* + *-(y)AcAĞ* + *-(y)Im* --> *öğrenci olacağım*

(I will be a student)

öğrenci (student) and *ol* + *-mAlI* + *-(y)Im* --> *öğrenci olmalıyım*

(I must be a student)

The suffix *-DIr* is not an obligatory suffix. It is usually not used in spoken language. In fact, it changes the meaning of the sentence a bit; it adds a probability or sometimes a definiteness concept. For example, the sentence *arkadaşınız burada* (your friend is here) means "I am sure that he/she is here", but the sentence *arkadaşınız buradadır* (your friend is here) means "he/she must be here (perhaps)". However, it is certainly used in statements which express permanent validities: e.g. *kedi bir hayvandır* (cat is an animal). *-DIr* can also be used after the verbs in narrative past, progressive, or future tense, in necessitative mode, or in narrative form of one of these tenses:

gel (to come) + *-(I)yor* + *-DIr* --> *geliyordur* (I am sure that he/she is coming)

gel (to come) + *-mAlI* + *-(y)mIş* + *-DIr* --> *gelmeliymiştir* (probably he/she must have come)

4.2.4. Participles

In Turkish, verb sentences can be transformed into a noun, adjective, or adverb clause by adding certain derivational suffixes to the verb of the sentence. Some of the mostly used ones of these suffixes are listed in Figure 4.4 in two groups. The first group shows the suffixes that make a noun or an adjective, and the second group is for the suffixes that result in an adverb. Among the participles, only *-mAdAn* and *-mAksIzIn* can not be used with the negation suffix since they include negation within their structures.

The suffix *-mAĞ* forms the infinitive form of the Turkish verbs. The infinitive can be used as a noun, and may take any of the case suffixes. It never takes possessive suffixes. The followings are valid:

oku (to read) + *-mAĞ* + *-(y)A* --> *okumağa* (to the reading)

oku (to read) + *-mAĞ* + *-DAn* --> *okumaktan* (of the reading)

1. Noun and adjective	:	*-DIĞ*	*-(y)AcAĞ*
		-mA	*-(y)An*
		-mAĞ	*-(y)AsI*
		-mIş	*-(y)Iş*
2. Adverb	:	*-cAsInA*	*-(y)ArAĞ*
		-mAdAn	*-(y)IncA*
		-mAksIzIn	*-(y)Ip*
		-(y)AlI	*-(y)ken*

FIGURE 4.4. Some derivational suffixes affixed to verbs

while the followings are not:

oku (to read) + *-mAĞ* + *-(n)In* --> *okumağın*
oku (to read) + *-mAĞ* + *-lArI* --> *okumakları*

Similarly, all the participles that transform the verb into a noun or an adjective may be used as a noun root, i.e. they may take all the suffixes that a noun root can take. For example:

gel (to come) + *-(y)Iş* + *-(I)nIz* + *-(y)A* --> *gelişinize* (to your coming)
ver (to give) + *-DIĞ* + *-lAr* + *-(s)I* + *-n* + *-DAn* + *-(y)DI* -->
verdiklerindendi (it was one of those that you/they gave)

The participles that make adverbs from verbs usually do not take any suffixes. Some of them can take only certain suffixes. For example, *-(y)ArAĞ* participle can take the suffix *-DAn*, which adds nothing to its meaning:

yap (to do) + *-(y)ArAĞ* --> *yaparak* (by doing something)
yap (to do) + *-(y)ArAĞ* + *-DAn* --> *yaparaktan* (by doing something)

-(y)ken has a somewhat different usage than the other participles. Originally it is the *-(y)An* relative participle of the verb *imek*. Like the other forms of this verb, it may be used as a suffix or as an independent word, i.e. *iken*. It is an invariable suffix, that is, it is not subject to the vowel harmony. It is affixed to a verb in the necessitative mode or in any tense except the definite past. It is not used with person suffixes, but it can follow the third person plural suffix *-lAr*. Second tense suffixes are not

used with *-(y)ken*. It can also be affixed to a noun causing a noun sentence to transform into a noun clause:

öğrenci (student) + *-(y)ken* --> *öğrenciyken* (when one was a student)
ev (house) + *-DA* + *-lAr* + *-(y)ken* --> *evdelerken* (when they are at home)

-cAsInA shows some similarities with *-(y)ken*. It is affixed to certain tense bases, namely, present, narrative past, and narrative of progressive and future:

uç (to fly) + *-(A)r* + *-cAsInA* --> *uçarcasına* (as if flying)
uç (to fly) + *-(I)yor* + *-(y)mIş* + *-cAsInA* --> *uçuyormuşcasına* (as if he/she was flying)

and it can also be affixed to nouns and adjectives:

çocuğ (child) + *-cAsInA* --> *çocukcasına* (like a child)
çılgın (crazy) + *-cAsInA* --> *çılgıncasına* (like crazy).

V. MORPHOLOGICAL REPRESENTATION OF TURKISH

In the previous chapter, the morphological structure of Turkish has been investigated. We have divided the analysis into two interrelated parts: morphophonemic analysis and morphotactic analysis. For the first one, we have defined the morphophonemic rules and explained the irregularities of these rules. For the morphotactics, we have displayed the order of the suffixes for different root types.

In this chapter, we shall attempt to formalize this morphological structure in the form of an Augmented Transition Network (ATN). The morphotactics, that is the order of the suffixes, will be defined as transitions on the network. The morphophonemic rules will accompany the network as rules that are activated when transitions between the states occur. Combination of the state transitions in the network and the rules will form the structure of a morphological parser for the Turkish language.

5.1. Root Categories

Before discussing the morphology of Turkish words, we must categorize the roots. This is necessary because all the suffixes are not affixed to all of the words. For example, it is syntactically incorrect to affix the first person possessive suffix *-(I)m* (my), which is affixed to nouns, to a verb. Table 5.1 shows the word categories in alphabetical order used in this research. Also some example root words are displayed in the table to clarify the meaning of the categories. For each category, we shall make use of a single-letter abbreviation. These abbreviations will ease the definition of the morphotactic structure.

As can be seen from the table, we divide the words into 14 categories. As the number of the categories increases, it becomes easier to define the morphotactics of the language. The suffixes that are affixed to words in different categories can be distinguished more easily. Each category owns a different structure which represents its morphotactics and the suffixes can be placed in the structures that they should be.

An expanded form of this idea is referred to as *semantic categorization*. In this process, the words are not only categorized with respect to their basic properties (such as noun, verb, adjective), but also categorized taking into account their semantics. For example, a subcategory of nouns

TABLE 5.1. Word categories

Category	Category name	Examples
A	Adjective	*adil, güzel, sarı*
B	Chemical symbol	*Au, Br, Eu*
C	Conjunction	*ama, ancak, eğer*
D	Adverb	*kıskıvrağ, kolaylıkla, lıkır lıkır*
E	Postposition	*aid, dair, değin*
I	Interjection	*agu, ah, aman, eh, elveda*
K	Acronym	*AET, Cad., Doç., kg., TBMM*
L	Letter	*a, b, c, ... ,z*
N	Noun	*adam, kitab, masa*
P	Pronoun	*bazısı, ben, biz, bu, kendi, kim, o, siz, şu*
R	Proper noun	*Ankara, Atatürk, Kutupyıldızı*
S	Number	*bin, bir, iki, milyar, milyon, on, yüz*
V	Verb	*ağla, gözet, oku*
W	Unknown category	*adfül, gapar, isfilt*

comprises the words that define the professions, a subcategory of proper nouns can be used for the country names, and so on. The aim of this subcategorization is to affix a suffix only to the relevant words. For instance, a suffix that can be affixed to a country name may not be affixed to other proper nouns. In section 5.4, we shall utilize a similar method for verbs. The verbs will be divided into subcategories based on their syntactic properties.

Note that we have reserved a category, W, for words whose categories are not known. We have collected these words from [53] which is the main spelling guide and which does not include word explanations. We could not find these words in the dictionaries [54,55,56]. Thus it was not possible to guess their categories. But we have included these words in our lexicon for completeness and grouped them in a separate category. Since the original categories of these words are not known, no suffixes can be attached to the words in the category W.

5.2. Morphophonemics

The morphophonemic rules are used, in general, to determine the form of a suffix that will be affixed to a word. In addition to the suffix formation, some of the rules may operate on the word itself instead of the suffix; i.e. the rules change the form of the word. This situation is rare in Turkish, but to arrive at a complete morphological structure, we must consider these exceptional situations.

In what follows, we have derived all the rules that are used in our morphological structure. These rules include some well-known rules such as the vowel harmony rule, and some rules which are used for a very limited number of cases such as the vowel insertion rule 1. In fact, rules of this second kind are not considered as morphophonemic rules in grammar books on Turkish morphology, instead they are treated as exceptional cases [10,11,12,53,57]. Hence they are not given a name as a rule; the names for some of the following rules are due to the authors. In order to be able to build a uniform morphophonemic component, we have derived all the rules that modify the suffixes and/or the words.

Below we list the morphophonemic rules. Some of the rules (rules 1, 2, 4, 8, 9, and 23) apply to each of the suffixes that are affixed to a word, while the rest of the rules apply only to the first suffix that is affixed to the word. These rules make use of the category (vowel, low vowel, etc.) and the order of the phonemes in the word and/or the suffix. To make the rules easy to read, we have used the following abbreviations:

fps : first phoneme of the suffix
fvs : first vowel of the suffix
lpw : last phoneme of the current word
lvw : last vowel of the current word
lcw : last consonant of the current word
ltpw : last two phonemes of the current word

The affixation process goes in this way: given a word in its root form, we derive a new word by adding an affix to this root form, then derive another word by adding another affix to this new word, and so on. The phrase *current word* denotes the word parsed up to that time, i.e. it signifies either the root word (for the first suffix) or the word derived from the root word by the affixation of the previous suffixes (for the succeeding suffixes).

Rule 1 Vowel harmony rule : Depending on the last vowel of the current word and the vowel of the suffix, the latter one changes. Nearly all of the Turkish words obey the vowel harmony rule. But some loanwords do not obey this rule due to their different phonetic structures (in fact, they obey another rule which we can name as *inverse vowel harmony rule*). So, we classify the words in two categories: words that obey the vowel harmony rule, and words that obey the inverse vowel harmony rule. Tables 5.2 and 5.3 show the rules for each of these groups.

Examples: *kalem* (pencil) + *-DA* ---> *kalemde* (at the pencil)

kuş (bird) + *-(y)I* ---> *kuşu* (the bird)

gel (to come) + *-(I)yor* ---> *geliyor* (he/she is coming)

Example: *saat* (watch) + *-(I)m* ---> *saatim* (my watch)

Note that in the derivation of the word *geliyor* (he/she is coming), the vowel *o* in the suffix *-(I)yor* does not change. This vowel is not subject to the vowel harmony rule since it is not capitalized.

The first part of this rule (the rule for words that obey the vowel harmony) is applied to each vowel (that is subject to the rule) in each suffix. The second part (the rule for words that obey the inverse vowel harmony), on the other hand, is applied only to the first vowel (if it is subject to the rule) in the first suffix. In other words, for the first vowel in the first suffix, either the first or the second part of the rule applies; for other vowels in all the suffixes, the first part of the rule applies. As an example, consider the following derivation, where *saat* (watch) is a word that obeys the inverse vowel harmony rule:

saat (watch) + *-(I)mIz* + *-DA* ---> *saatimizde* (at our watch)

We can represent this derivation in the following sequence:

saat (watch) + *-(I)m* ---> *saatim*

saatim + *-Iz* ---> *saatimiz* (our watch)

saatimiz (our watch) + *-DA* ---> *saatimizde* (at our watch)

In the first derivation, the second part of the rule applies. In the next two derivations, the first part of the rule applies.

Rule 2 Consonant harmony rule 1 : For some current words whose last phoneme is one of *b,c,d,g* or *ğ*, this last phoneme changes when no more suffixes are affixed to the current word or when a suffix beginning with a consonant is affixed. Table 5.4 shows the rule.

TABLE 5.2. Vowel harmony rule

fvs	lvw	Rule
a	back vowel	fvs does not change
a	front vowel	fvs is replaced by *e*
ı	back and unrounded vowel	fvs does not change
ı	back and rounded vowel	fvs is replaced by *u*
ı	front and unrounded vowel	fvs is replaced by *i*
ı	front and rounded vowel	fvs is replaced by *ü*

TABLE 5.3. Inverse vowel harmony rule

fvs	lvw	Rule
a	back vowel	fvs is replaced by *e*
a	front vowel	fvs does not change
ı	back and unrounded vowel	fvs is replaced by *i*
ı	back and rounded vowel	fvs is replaced by *ü*
ı	front and unrounded vowel	fvs does not change
ı	front and rounded vowel	fvs is replaced by *u*

Examples: *kitab* (book) ---> *kitap* (book)

çiçeğ (flower) + *-lAr* ---> *çiçekler* (flowers) (also rule 1 applies)

emred (to order) + *-DI* --> *emretti* (he/she has ordered) (also rule 1 applies)

In this rule, there is a distinction between the first suffix and the succeeding suffixes. For the first suffix affixed to the word, the word determines whether the rule will be applied or not. For example, the word *kitab* (book) obeys this rule, while the word *ad* (name) does not. For the succeeding suffixes, the last suffix that has already been attached to the word determines whether the current word obeys the rule or not. The word obeys the rule if the last phoneme of the suffix is capitalized. For example:

ağla (to cry) + *-mAĞ* + *-lI* --> *ağlamaklı* (crying)

TABLE 5.4. Consonant harmony rule 1

Suffix	lpw	Rule
no suffix or fps is consonant	*b*	lpw is replaced by *p*
no suffix or fps is consonant	*c*	lpw is replaced by *ç*
no suffix or fps is consonant	*d*	lpw is replaced by *t*
no suffix or fps is consonant	*g,ğ*	lpw is replaced by *k*

Rule 3 Consonant harmony rule 2 : *If lpw is ğ and either there is no suffix or fps is a consonant, then lpw is replaced by g.*

Example: *psikoloğ* (psychologist) + *-lAr* ---> *psikologlar* (psychologists)

Rule 4 Consonant harmony rule 3 : For some suffixes that begin with *b,c,d,* or *g*, the first phoneme of the suffix changes if the last phoneme of the current word is a voiceless consonant. Table 5.5 shows the rule.

Example: *kitab* (book) + *-CI* ---> *kitapçı* (book seller)

Note that rule 2 (consonant harmony rule 1) is applied before this rule, which changes *b* into *p*. Also note that we represent the consonants that can change according to this rule in capitalized form. For example, the following suffix does not obey to this rule although the word *kuş* (bird) ends with a voiceless consonant:

kuş (bird) + *-baz* --> *kuşbaz* (bird catcher)

TABLE 5.5. Consonant harmony rule 3

lpw	fps	Rule
voiceless consonant	*b*	fps is replaced by *p*
voiceless consonant	*c*	fps is replaced by *ç*
voiceless consonant	*d*	fps is replaced by *t*
voiceless consonant	*g*	fps is replaced by *k*

Rule 5 Vowel insertion rule 1 : *If there is no suffix or fps is a consonant, then I is inserted before the last consonant of the current word.* This rule applies to nouns only.

Example: *ağz* (mouth) + *-lAr* ---> *ağızlar* (mouths)

Rule 6 Vowel insertion rule 2 : *If there is no suffix or the suffix is not one of -I,-(I)Ğ,-(I)l,-(I)m,-(I)ntI,-(I)t, then I is inserted before the last consonant of the current word.* This rule applies to verbs only.

Examples: *ayr* (to separate) + *-DI* ---> *ayırdı* (he/she separated)
çağr (to call) + *-mAz* ---> *çağırmaz* (he/she does not call)

Rule 7 Double consonant rule : *If the suffix is one of -eD,-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(n)In,-ol,-(s)I,-(y)A,-(y)I,-(y)IlAn,-(y)InAn, then lpw doubles.* This rule applies to nouns only.

Example: *tıb* (medicine) + *-(y)I* ---> *tıbbı* (the medicine)

Rule 8 Phoneme deletion rule : *If fps and lpw are either both vowels or both consonants, then fps drops.*

Examples: *sev* (to love) + *-(y)AcAĞ* ---> *sevecek* (he/she will love)
(also rule 1 applies for the two vowels in the suffix)
masa (table) + *-(I)m* ---> *masam* (my table)

Rule 9 Phoneme deletion rule for the suffix *-(I)yor* : *If the suffix is -(I)yor and lpw is a vowel, then lpw drops.*

Example: *ağla* (to cry) + *-(I)yor* ---> *ağlıyor* (he/she is crying)

Rule 10 Phoneme deletion rule for verbs : *If lpw is a vowel and fps is y, then lpw is replaced by I.* This rule applies to verbs only.

Examples: *de* (to say) + *-(y)An* ---> *diyen* (the one who says) (also rule 1 applies)
de (to say) + *-(y)AcAĞ* ---> *diyecek* (he/she will say) (also rule 1 applies)

This rule has an exception for the word *de* (to say). If the word is *de* (to say) and the suffix is one of *-(y)Ip* or *-(y)IncA*, then the rule is not applied:

de (to say) + *-(y)Ip* ---> *deyip* (by saying)
de (to say) + *-(y)IncA* ---> *deyince* (when one says)

Rule 11 Possessive suffix rule 1 : *If the suffix is -(s)I, then fps (which is s) drops.*

Example: *mevki* (position) + *-(s)I* ---> *mevkii* (his/her position) (also rule 1 applies)

Rule 12 Possessive suffix rule 2 : *If the suffix is -(s)I, then fps (which is s) may or may not drop.*

Example: *sanayi* (industry) + *-(s)I* ---> *sanayisi* (his/her industry) (also rule 1 applies)
sanayi (industry) + *-(s)I* ---> *sanayii* (his/her industry) (also rule 1 applies)

Rule 13 Rule for compound words 1 : A compound word is a word that is formed of two other words. The meaning of the compound word may be a combination of the meanings of the underlying words or may be totally different. For example, *alınyazısı* (destiny) is formed of *alın* (forehead) and *yazı* (writing). This rule applies to compound words that end in *sI* or *suyu*. Table 5.6 shows the rule.

Examples: *alınyazısı* (destiny) + *-lAr* ---> *alınyazıları* (destinies)
alınyazısı (destiny) + *-(s)I* ---> *alınyazısı* (his/her destiny)
alınyazısı (destiny) + *-(I)m* ---> *alınyazım* (my destiny) (also rule 8 applies)
alınyazısı (destiny) + *-CA* ---> *alınyazısınca* (according to the destiny)

Note that for compound words that end with *suyu*, in the cases where the last two phonemes of the word drop, the word takes a form which ends with *su* and thus rule 18 (rule for the morpheme *su*) is applied.

Rule 14 Rule for compound words 2 : This rule applies to compound words that end in a high vowel and lcw is one of *b,c,d,g*, or *ğ*. Table 5.7 shows the rule.

TABLE 5.6. Rule for compound words 1

Suffix	**Rule**
-lAr	ltpw drop, *-lAr* is inserted, *-I* is inserted
-(s)I	suffix drops
-(I)m,-(I)mIz,-(I)n,-(I)nIz	ltpw drop before the suffix
-CA,-DA,-DAn,-(y)A,-(y)I	*n* is inserted before the suffix
-(y)lA	no change
other suffixes	ltpw drop before the suffix

TABLE 5.7. Rule for compound words 2

Suffix	**Rule**
-lAr	lvw drops, lcw is replaced by *p,ç,t,k* respectively, *-lAr* is inserted, *-I* is inserted
-(s)I	suffix drops
-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(y)lA	no change
-CA,-DA,-DAn,-(y)A,-(y)I	*n* is inserted before the suffix
other suffixes	lvw drops, lcw is replaced by *p,ç,t,k* respectively before the suffix

Examples: *ayakucu* (foot) + *-lAr* ---> *ayakuçları* (feet)

ayakucu (foot) + *-(s)I* ---> *ayakucu* (his/her foot)

ayakucu (foot) + *-(I)m* ---> *ayakucum* (my foot) (also rule 8 applies)

ayakucu (foot) + *-CA* ---> *ayakucunca* (according to the foot)

Rule 15 Rule for compound words 3 : This rule applies to compound words that end in a high vowel. Table 5.8 shows the rule.

Examples: *aslanağzı* (snapdragon) + *-lAr* ---> *aslanağızları* (snapdragons)

aslanağzı (snapdragon) + *-(s)I* ---> *aslanağzı* (his/her snapdragon)

aslanağzı (snapdragon) + *-(I)m* ---> *aslanağzım* (my snapdragon) (also rule 8 applies)

aslanağzı (snapdragon) + *-CA* ---> *aslanağzınca* (according to the snapdragon)

TABLE 5.8. Rule for compound words 3

Suffix	Rule
-lAr	ltpw are interchanged, *-lAr* is inserted, *-I* is inserted
-(s)I	suffix drops
-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(y)lA	no change
-CA,-DA,-DAn,-(y)A,-(y)I	*n* is inserted before the suffix
other suffixes	ltpw are interchanged before the suffix

TABLE 5.9. Rule for compound words 4

Suffix	Rule
-lAr	lvw drops, *-lAr* is inserted, *-I* is inserted
-(s)I	suffix drops
-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(y)lA	no change
-CA,-DA,-DAn,-(y)A,-(y)I	*n* is inserted before the suffix
other suffixes	lvw drops before the suffix

Rule 16 Rule for compound words 4 : This rule applies to compound words to whom the rules 13, 14, and 15 are not applicable. Table 5.9 shows the rule.

Examples: *adaçayı* (garden sage) + *-lAr* ---> *adaçayları* (garden sages)

adaçayı (garden sage) + *-(s)I* ---> *adaçayı* (his/her garden sage)

adaçayı (garden sage) + *-(I)m* ---> *adaçayım* (my garden sage) (also rule 8 applies)

adaçayı (garden sage) + *-CA* ---> *adaçayınca* (according to the garden sage)

Rule 17 Aorist suffix rule : There are two forms of the aorist suffix in Turkish: *-(A)r* and *-(I)r*. Some of the verbs take the first one, while the rest take the second. There is no specific rule to decide whether *-(A)r* or *-(I)r* is affixed to a verb. So, we accept the form *-(A)r* as the default aorist suffix for verbs and handle the verbs that use the form *-(I)r* by the following rule: *If the suffix is -(A)r, then the suffix is replaced by -(I)r.*

Example: *gel* (to come) + *-(A)r* ---> *gelir* (he/she comes) (also rule 1 applies)

Rule 18 Rule for the morpheme *su* : Words that end in *su* show irregularities when a possessive suffix (*-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(s)I*) or the genitive suffix (*-(n)In*) is affixed. *If the suffix is one of -(I)m,-(I)mIz,-(I)n,-(I)nIz,-(n)In,-(s)I, then y is inserted before the suffix.*

Examples: *su* (water) + *-(I)m* ---> *suyum* (my water) (also rule 1 applies)

su (water) + *-(s)I* ---> *suyu* (his/her water) (also rule 1 and rule 8 apply)

akarsu (running water) + *-(n)In* ---> *akarsuyun* (of the running water) (also rule 1 and rule 8 apply)

Rule 19 Rule for proper nouns : This rule applies to proper nouns only. *If the suffix is an inflectional suffix, then the apostrophe character (') is inserted before the suffix. If the suffix is -lAr, then the apostrophe character (') may or may not be inserted before the suffix* [53].

Examples: *Atatürk* + *-(I)n* ---> *Atatürk'ün* (also rule 1 applies)

Atatürk + *-lAr* ---> *Atatürk'ler* (also rule 1 applies)

Atatürk + *-lAr* ---> *Atatürkler* (also rule 1 applies)

Rule 20 Rule for acronyms 1 : This rule applies to acronyms only. When a suffix is affixed to an acronym, other rules (vowel harmony rule, consonant harmony rules, etc.) apply to the *reading* of the acronym; not to the *reading* of the expanded form of the acronym. In other words, we consider the reading of the acronym as if it were the word and then apply the relevant rules. This rule tries to extract the reading of the acronym by considering the last vowel and the last consonant in the word. But the rule does not always yield true derivations since we can not extract the reading by only syntactic knowledge. Hence, the rule is designed as to accept the suffix attachments for all the possible reading forms of the word. This means that some invalid attachments are also included in the rule. Table 5.10 shows the rule.

Examples:

1. *TBMM* + *-DA* ---> *TBMM'de* (also rule 21 applies)
2. *TBMM* + *-(n)In* ---> *TBMM'nin* (also rule 21 applies)
3. *AET* + *-(n)In* ---> *AET'nin* (also rule 21 applies)
4. *AET* + *-(n)In* ---> *AET'in* (also rule 21 applies)
5. *ASELSAN* + *-(y)I* ---> *ASELSAN'ı* (also rule 21 applies)
6. *ASELSAN* + *-(y)I* ---> *ASELSAN'yi* (also rule 21 applies)
7. *UNSC* + *-(y)I* ---> *UNSC'yi* (also rule 21 applies)
8. *UNSC* + *-(y)I* ---> *UNSC'u* (also rule 21 applies)
9. *ÖSS.* + *-(y)I* ---> *ÖSS.yi*
10. *p.m.* + *-DA* --> *p.m.de*
11. *İÖ* + *-(n)In* ---> *İÖ'nün* (also rule 21 applies)
12. *CENTO* + *-(y)I* ---> *CENTO'yu* (also rule 21 applies)

TABLE 5.10. Rule for acronyms 1

lpw	lvw	Rule
consonant	front vowel	other rules apply as if lvw is a front vowel and lpw is either a vowel or a consonant
consonant	back vowel	other rules apply as if either lvw is a front vowel and lpw is a vowel or lvw is a back vowel and lpw is a consonant
consonant	word has no vowels	other rules apply as if lvw is a front vowel and lpw is a vowel
vowel	front vowel	other rules apply as if lvw is a front vowel and lpw is a vowel
vowel	back vowel	other rules apply as if lvw is a back vowel and lpw is a vowel

The examples 4, 6, and 8 above are not correct; the valid derivations for these examples are 3, 5, and 7, respectively. As indicated above, the reason is that acronyms with the same syntactic form have different *readings*. For instance, *ASELSAN* and *UNSC* both end in a consonant and their last vowels are back vowels. However, they are pronounced differently; the first one is pronounced as a single word, while the second one is pronounced letter by letter. Hence, the form of the suffix which is true for one of them is wrong for the other. In order to correct this situation, an internal classification of the acronyms, which divides them into two groups according to the pronunciation, should be done in the lexicon.

Rule 21 Rule for acronyms 2 : This rule applies to acronyms only. *If lpw is not '.', then the apostrophe character (') is inserted before the suffix* [53].

Example: *TBMM + -DA ---> TBMM'de* (also rule 20 applies)

Rule 22 Rule for numbers : This rule applies to the numbers that are written as a sequence of digits. When a suffix is affixed to a number, other rules (vowel harmony rule, consonant harmony rules, etc.) apply as if the number is written explicitly (i.e. as a sequence of words corresponding to the digits). Therefore, we first convert the number into its *written* form.

This conversion procedure is straightforward if the last digit of the number is not zero. For example, if the number is *34*, it is enough to convert *4* into *dörd* (four). However, if the last digit is zero, then we must know how many zeros there are at the end of the number. If there is one zero,

then the written form ends in *on* (ten), *yirmi* (twenty), and so on, depending on the digit preceding the last one. If there are two zeros, then the written form ends in *yüz* (hundred). If there are three zeros, then the written form ends in *bin* (thousand). The conversion process continues in this way. So, the rule is: *The number is converted into its written form, the apostrophe character (') is inserted before the suffix, and the suffix is affixed as if the number were written in this form* [53].

Examples: *8 + -(I)ncI ---> 8'inci* (also rule 1 applies to *sekiz* (eight))
4 + -(ş)Ar ---> 4'er (also rule 1 and rule 8 apply to *dörd* (four))
130 + -(ş)Ar ---> 130'ar (also rule 8 applies to *yüz otuz* (one hundred and thirty))

Rule 23 Rule for particles : The particles *-dA* (the emphatic particle, which has a meaning like *even*), *-ki* (complementizer), and *-mI* (question particle) have a special feature: they are written separate from the current word they follow. [3] We generally refer to these as *particles* because of this property and to distinguish them from the suffixes *-DA* (locative case suffix) and *-ki* (relative suffix). But these particles are subject to all the rules we have defined as other suffixes. *If the suffix is one of the particles -dA,-ki, or -mI, then a space character is inserted before the suffix.*

Examples: *çocuğ* (child) + *-dA* --> *çocuk da* (even this child) (also rule 2 applies)
kitab (book) + *-ki* --> *kitap ki* (such a book) (also rule 2 applies)
geldi (he/she came) + *-mI* --> *geldi mi ?* (did he/she come?) (also rule 1 applies)

Note that the consonant *d* in the particle *-dA* and the vowel *i* in the particle *-ki* do not obey the harmony rules. Hence they are not capitalized.

Most of the rules listed above have some exceptions; i.e. they are not valid for all of the words. Only the rules 9, 19, 20, 21, 22, and 23 are valid all the time. The rules 4 and 8 are handled by the use of the symbols *capital letter* and *( )*; i.e. if the phoneme is represented with these symbols, then it can change or drop, respectively. For rule 1, we have handled the exceptional cases (i.e. words that do not obey the vowel harmony rule - or, words that obey the inverse vowel harmony rule) with a separate rule within rule 1; the reason is that the exceptional cases also obey a different rule. For the rest of the rules that have exceptions, we did not mention those exceptional cases; the exceptional

[3] The transitions of the ATN in which these particles are written separate from the current word can be seen in appendix A.

cases will be treated as if there is no such rule. For example, consider rule 5 for the following derivations:

ağz (mouth) + *-lAr* ---> *ağızlar* (mouths)
tank (tank) + *-lAr* ---> *tanklar* (tanks)

The first word obeys the rule, while the second one does not. The second word takes the suffix *-(I)m* without considering rule 5 and the word *tanklar* (tanks) derives.

Some of the rules that have exceptions are valid for most of the words (e.g. vowel harmony rule) and some of them are valid for only a very limited number of cases. For example, rule 10 is valid for only two words in Turkish, which are *de* (to say) and *ye* (to eat). In section 5.4, we shall see how we can handle these exceptional cases formally.

Some of the studies on Turkish morphology treat some of the rules in a simple way. Usually the acronyms and the numbers shown in digits are not counted as language elements despite the fact that they occur frequently inside texts. Hence, no rules are designed for these elements. As another example, in [25] the aorist suffix rule is not implemented and the parser does not reject a word affixed with the wrong aorist suffix.

As a final remark, Hankamer [32] divides the morphophonemic rules into two groups: rules for roots and rules for suffixes. The relevant rule set is applied depending on whether the root or a suffix is to be modified. In our approach, we did not make a distinction between these two. The rules have the capability of operating on the root and the suffixes as determined by the functions of the ATN.

5.3. Morphotactics

Turkish morphology is quite rich in the number of suffixes, especially the derivational ones. This forced us to make a detailed research in compiling the suffixes that are used in Turkish [9,10,11,14,15,35]. Among these references, [14,15] are the ones that we have consulted most. In these references most of the suffixes are listed and they are accompanied with brief explanations on their use. The type of the suffixes (inflectional or derivational) are also mentioned. Other than these

references, we have also consulted the word dictionaries [54,55,56], in order to extract the suffixes that are rarely used and hence not mentioned in grammar books.

Derivational suffixes are the suffixes which produce a new word having a different meaning than the word they are affixed to. Some of these suffixes also change the category of the word; for example, a noun may be a verb after a derivational suffix is affixed.

Some of the derivational suffixes are highly productive while the others are not. The participles are among the suffixes that may be received by all of the words in the category that they belong to. Another group of the derivational suffixes can be attached to a great number, but not all, of the words in their categories. For example, *-CI*, *-lAş*, *-lIĞ* are such suffixes. There are also some derivational suffixes that can be affixed to a few words only. As an extreme example, the suffix *-kek* can only be affixed to the noun *er* (male) to form the noun *erkek* (man).

There is a large number of derivational suffixes used in Turkish. Table 5.11 lists the derivational suffixes that are used in this research. *Originating category* denotes the category of the words that the suffix can be affixed; *destination category* denotes the category of the new word after the suffix is affixed. The interrelationships between these suffixes will be defined formally in the following section within the ATN representation.

5.4. Turkish Morphology in ATN

In this section, we shall explain the morphological structure of Turkish language by using the morphotactics and the morphophonemic rules. The morphotactics will be shown in a list format and also a small part as a transition network. The morphophonemic rules will be defined in a pseudo language. When defining these rules, we shall make use of some functions. The explanation of these functions will also be given.

For each word category shown in Table 5.1, there is a transition network that is used to parse a word that is originated from that category, i.e. a word whose root form belongs to that category. For example, the root form of the word *kitaplarımız* (our books) is *kitab* (book) which belongs to the noun category. So, we begin parsing from the noun network. Also, in addition to the main network for a

TABLE 5.11. Derivational suffixes

Originating category	Suffixes	Destination category
A (adjective)	*-bIz, -CA, -(I)msAr, -(I)msI, -(I)mtıraĞ, -lIĞ, -şIn*	A (adjective)
	-CA, -DAn, -en, -(I)nA, -sInA, -sIz	D (adverb)
	-(A)z	N (noun)
	-(A)l	V (verb)
D (adverb)	*-(y)sA*	C (conjunction)
	-CAcIĞ, -CAnA, -DAn, -lIklA, -mAsInA	D (adverb)
I (interjection)	*-lA*	V (verb)
N (noun)	*-(A)l, -CA, -CI, -CIl, -(i)Ğ, -kar, -lI, -lIĞ, -perver,*	A (adjective)
	-rAĞ, -sAl, -sever, -sI, -sIl, -sIz, -(v)i, -(y)ane	A (adjective)
	-CA, -leyin, -lI, -sIzIn, -(y)ane, -(y)IlAn, -(y)In,	D (adverb)
	-(y)InAn, -(y)lA	D (adverb)
	-(A)C, -(A)ğI, -(a)j, -(A)Ğ, -(A)lAĞ, -(A)t, -başı, -baz,	N (noun)
	-CA, -cAğIz, -CAĞ, -CI, -CIĞ, -dA, -dAĞ, -dAlIĞ,	N (noun)
	-DAm, -DAn, -Dar, -DAş, -dIrIĞ, -dIz, -GA, -GAn,	N (noun)
	-giller, -hane, -(I)l, -(I)t, -(i)st, -(i)zm, -keĞ, -keş, -ki,	N (noun)
	-lA, -lAĞ, -lAm, -lIĞ, -mAn, -mAr, -name, -ölçer,	N (noun)
	-sAĞ, -sIz, -tay, -zade	N (noun)
	-A, -(A)n, -(A)r, -(A)ş, -DAn, -DAş, -DAt, -eD, -I,	V (verb)
	-(I)Ğ, -(I)msA, -lA, -lAn, -lAş, -lAt, -ol, -sA, -sI	V (verb)
P (pronoun)	*-cIlAyIn*	P (pronoun)
R (proper noun)	*-giller, -lAr, -lIĞ,*	N (noun)
	-cAğIz, -CI, -CIĞ, -(i)st, -(i)zm, -lI, -sIz	R (proper noun)
	-lA, -lAş	V (verb)
S (number)	*-gen, -(I)ncI, -(I)z, -(ş)Ar*	A (adjective)
	-dAlIĞ	N (noun)
	-altı, -altmış, -beş, -bin, -bir, -doksan, -dokuz, -dörD,	S (number)
	-elli, -iki, -kırk, -milyar, -milyon, -on, -otuz, -sekiz,	S (number)
	-seksen, -trilyon, -üç, -yedi, -yetmiş, -yirmi, -yüz	S (number)
V (verb)	*-(A)cAn, -(A)ğAn, -GIn, -(I)Ğ, -(I)msAr, -mIş, -pAĞ,*	A (adjective)
	-sAĞ, -sAl, -vAn, -(y)AsIcA, -(y)IcI	A (adjective)
	-CA, -CAsInA, -C, -DAn, -(I)n, -(I)na, -sA, -sIzIn,	D (adverb)
	-(y)A, -(y)AlI, -(y)An, -(y)ArAĞ, -(y)AsIyA, -(y)IncA,	D (adverb)
	-(y)Ip	D (adverb)
	-lA	E (postposition)
	-(A)C, -(A)Ğ, -(A)lgA, -(A)m, -(A)mAĞ, -(A)mık,	N (noun)
	-(A)nAk, -(A)r, -(A)v, -(A)y, -CA, -CAĞ, -CAmA,	N (noun)
	-GA, -GAC, -GAn, -GI, -GIC, -GIt, -I, -(I)cIĞ, -(I)m,	N (noun)
	-(I)n, -(I)ntI, -(I)t, -lIĞ, -mAC, -mAn, -mAz, -mIĞ,	N (noun)
	-mur, -nAĞ, -sI, -tI, -(y)AcAĞ, -(y)An, -(y)AsI	N (noun)
	-Ar, -dA, -DAn, -DIĞ, -dIr, -(I)l, -(I)n, -Ir, -(I)ş, -(I)t,	V (verb)
	-ki, -mA, -mAz, -t, -(y)Iş	V (verb)

category, there may be subnetworks for that category. These subnetworks are called recursively from other networks. In the transition networks, in addition to the arcs for recursive call, we also include a jump arc. A jump arc indicates that the parsing continues from another network. This process is different from a recursive call; when a jump to another network is performed, the process does not return to the originating network. The use of the jump arcs simplifies the design of the network; it can be eliminated by introducing additional nodes. Also, if we think of all the morphotactics as a single network, this jump operation can be seen just as a transition from one node to another.

In the list and the network notation, we have used the following conventions:

- A node represents a state of the network. An arc represents a transition from one node to another.

- A circular node represents a node of the network. A rectangular node represents a node of another network, i.e. after the transition, the process continues from the indicated node in the other network. As indicated above, this is referred to as a jump (not a recursive call) to the other network.

- A node labelled * indicates that parsing ends after the transition occurs.

- The end node of a network is labelled by *n/e* in the graph and by *n/end* in the list, where *n* is the name of the network.

- The label on an arc can be one of three types:

    - A suffix

    - A recursive call to another network, indicated as *(parse n)*, where *n* represents a node in another network. The process continues from node *n* and when the end of the network is reached, the process returns to the node following the arc labelled *(parse n)*. While processing the network that contains the node *n*, if a jump (not a recursive call) to another network is encountered, then the process continues from this new network and it does not return to the calling network.

- An empty suffix, indicated as ~. This means that the process may jump from the node preceding the arc to the node following the arc without affixing a suffix or performing a recursive call.

- If there is more than one transition between two nodes, then these alternatives are written as different labels on the same arc and are separated by commas.

- *(x)*, where *x* is a phoneme, indicates that the phoneme deletion rule (rule 8) can be applied to the phoneme *x*.

- *X*, where *x* is a phoneme, indicates that the vowel harmony rule (rule 1) (if *x* is a vowel) or the consonant harmony rule 3 (rule 4) (if *x* is a consonant) can be applied to the phoneme *x*.

- *(X)*, where *x* is a phoneme, indicates that both *(x)* and *X* are valid.

The full set of morphotactics is listed in appendix A. Figure 5.1 shows the ATN representation of a part of the noun network in graphical form. In the following paragraphs, we shall summarize the parsing of some of the networks.

Parsing of the noun network begins from node N. Then two subnetworks are called; network ND (network that consists of the derivational suffixes for nouns) followed by the network NC (network that consists of the inflectional suffixes for nouns). The NC network is further divided into two parts; the first part contains the plural suffix (*-lAr*), the possessive suffixes (*-(I)m, -(I)n*, etc.), and some other suffixes that are used in this state; the second part calls the network NE which contains the case suffixes (*-DA*:locative, *-DAn*:ablative, *-(y)A*:dative, *-(y)I*:accusative), the genitive suffix *-(n)In*, and the relative suffix *-ki*. The network NC ends with the particles (the emphatic particle *-dA* and the complementizer *-ki*) which are written separate from the word.

The verb network is the most complex one in the morphological structure. One important point that must be noted is that, in addition to the categorization of the words as noun, verb, adjective, etc., we have also categorized the verbs internally. The reason of this subcategorization is that the affixation of the voice suffixes (*-(I)ş*:reciprocal, *-(I)n*:reflexive, *-Ar,-DIr,-Ir,-(I)t,-t*:causative, *-(I)l*:passive)

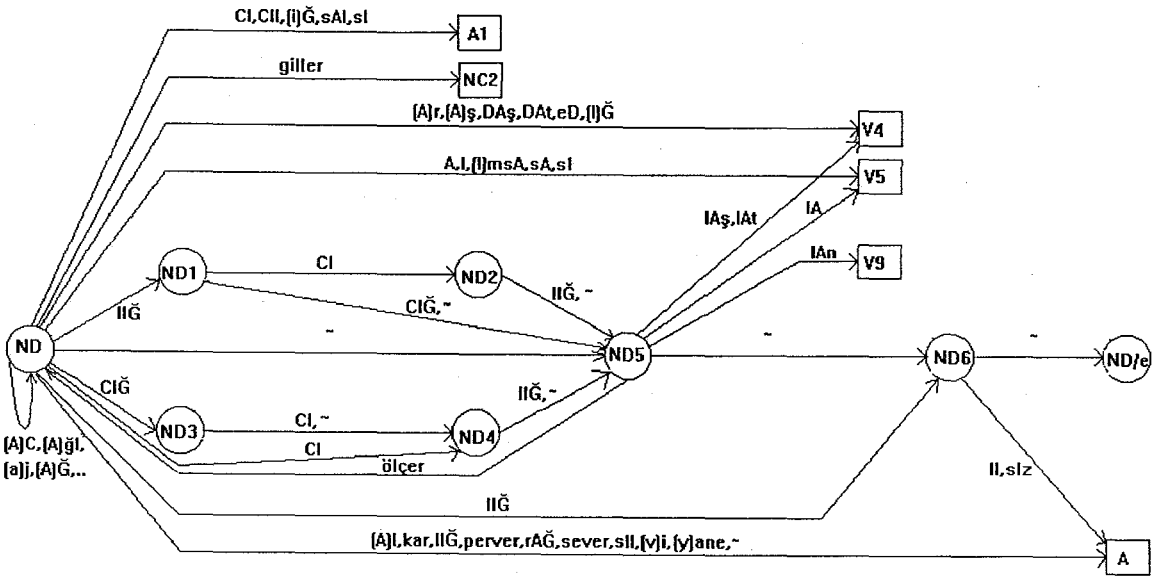


FIGURE 5.1. ATN representation of the ND network

to the verbs are different. These suffixes are affixed to the root form of a verb. The subcategorization process simplifies the morphotactic rules that control the use of the voice suffixes. The categorization of the verbs is listed in Table 5.12, with an explanation of each subcategory. There are nine subcategories. Although the starting node of the verb network is labelled V, none of the verbs start parsing from this node; instead the nodes V1 through V9 are used as the starting nodes for the verbs.

The V network handles the affixation of some of the voice suffixes, and then calls the networks VA and VB. The VA network contains the passive suffix *-(I)l* and some other derivational suffixes that cause a jump to noun, adjective, or adverb networks. The VB network includes the compound verb suffixes (*-(y)Abil, -(y)Adur*, etc.) and the negation suffix (*-mA*). It then parses the main tense suffix (*-DI, -mIş, -mAlI*, etc.) and performs a jump to the network corresponding to this main tense suffix. The VC network contains the derivational suffixes that can be affixed to a verb in this state. The networks VD, VE, VF, and VG include the question suffix (*-mI*), second tense suffixes (*-(y)DI, -(y)mIş, -(y)sA*), and the person suffixes (*-m, -n, -k, -nIz, -lAr*) that can follow each of the main tense suffixes.

TABLE 5.12. Subcategories of verbs

Subcategory	Explanation
V1	Verbs that can take the causative suffix *-Ar, e.g. kop* (to break)
V2	Verbs that can take the causative suffix *-Ir*, e.g. *bat* (to sink), *doy* (to be satiated), *piş* (to be cooked)
V3	Verbs that can take the causative suffix *-t*, e.g. *ayr* (to separate), *aşır* (to pass over), *bağır* (to shout)
V4	Verbs that do not belong to other verb subcategories, e.g. *abart* (to exaggerate), *as* (to hang)
V5	Verbs that end with a vowel, e.g. *ağla* (to cry), *sıçra* (to leap)
V6	Verbs that can take the causative suffix *-(I)t*, e.g. *kok* (to smell), *kork* (to be afraid)
V7	Monosyllabic verbs that end with the consonant *l*, e.g. *al* (to take), *gel* (to come)
V8	Polysyllabic verbs that end with the consonant *l*, e.g. *alçal* (to stoop), *küçül* (to become small), *yüksel* (to rise)
V9	Verbs that end with the consonant *n*, e.g. *dayan* (to endure), *dön* (to spin)

Similar to the verb network, the network for acronyms (category K) is categorized internally in the following way: K1 (acronyms whose letters must be in upper case, e.g. *TBMM*), K2 (acronyms with the first letter in upper case and the rest in lower case, e.g. *Fr.*), and K3 (acronyms whose letters must be in lower case, e.g. *kg.*). The node K is not used as the starting node; instead the nodes K1 through K3 are used as the starting nodes for the acronyms.

The category of a derived word is the category in which the parsing ends. For example, the root form of the word *güzelleşti* (he/she became beautiful) is the word *güzel* (beautiful), which is an adjective. We begin parsing this word from the adjective network. The parsing is completed in the verb network. Hence the category of the word *güzelleşti* (he/she became beautiful) is verb.

Now we define the morphophonemic rules in a pseudo language. The rules make use of some variables and functions. The variables used by the rules are described below:

word : the current word
suffix : the current suffix
fps : the first phoneme of the current suffix
fvs : the first vowel of the current suffix

The functions can be divided into two groups: general purpose functions and morphophonemic rules functions. The second group is categorized into two subgroups. Morphophonemic character check

functions are used in the vowel harmony rules, consonant harmony rule 3, and the phoneme deletion rule. Morphophonemic rule check function denotes whether the word obeys a particular morphophonemic rule or not. We assume that, when the parse of a word in the root form begins, the value of the morphophonemic rule check function for all the rules for this word are available. Since most of the rules have exceptions, the use of this function is necessary. For example, most of the words, but not all, obey the vowel harmony rule; we must know which ones obey and which ones do not in order to arrive at a correct parsing. The implementation details of the morphophonemic rule check function will be explained in section 8.5.

Appendix B contains the morphophonemic rules given in section 5.2 in terms of these variables and functions.

5.5. Examples

In this section, we shall try to explain the proposed morphological structure by the use of some examples. The transitions during parsing are written under the columns *originating state*, *label*, and *destination state*. The column *current word* holds the word parsed up to that time. The column *rules applied* indicates the rules applied at that state. To reserve room on the paper, we cease the examples as soon as the word is parsed; we do not continue until the end of the network is reached.

Example 1:

word : *kutuplardaki* (at the poles)
root : *kutb* (pole)
root category : noun

Table 5.13 illustrates the parsing of the word.

Example 2:

word : *güzelleşti* (he/she became beautiful)
root : *güzel* (beautiful)
root category : adjective

Table 5.14 illustrates the parsing of the word.

TABLE 5.13. Parsing of the word *kutuplardaki* (at the poles)

Originating state	Label	Destination state	Current word	Rules applied
N	(parse ND)	N1	*kutb*	
ND	~	ND5	*kutb*	
ND5	~	ND6	*kutb*	
ND6	~	ND/end	*kutb*	
N1	(parse NC)	N/end	*kutb*	
NC	~	NC1	*kutb*	
NC1	*-lAr*	NC2	*kutuplar*	rule 5 (*u* inserted)
				rule 2 (*b* --> *p*)
				rule 1 (*A* --> *a*)
NC2	~	NC4	*kutuplar*	
NC4	~	NC7	*kutuplar*	
NC7	(parse NE)	NC9	*kutuplar*	
NE	*-DA*	NE5	*kutuplarda*	rule 4 (*D* --> *d*)
				rule 1 (*A* --> *a*)
NE5	*-ki*	NE4	*kutuplardaki*	

TABLE 5.14. Parsing of the word *güzelleşti* (he/she became beautiful)

Originating state	Label	Destination state	Current word	Rules applied
A	~	A1	*güzel*	
A1	~	N	*güzel*	
N	(parse ND)	N1	*güzel*	
ND	~	ND5	*güzel*	
ND5	*-lAş*	V4	*güzelleş*	rule 1 (*A* --> *e*)
V4	(parse VA)	V19	*güzelleş*	
VA	~	VA2	*güzelleş*	
VA2	~	VA3	*güzelleş*	
VA3	~	VA5	*güzelleş*	
VA5	~	VA/end	*güzelleş*	
V19	(parse VB)	V20	*güzelleş*	
VB	~	VB2	*güzelleş*	
VB2	~	VB6	*güzelleş*	
VB6	~	VB9	*güzelleş*	
VB9	*-DI*	VD	*güzelleşti*	rule 4 (*D* --> *t*)
				rule 1 (*I* --> *i*)

TABLE 5.15. Parsing of the word *ayrılmıyordu* (he/she was not departing)

Originating state	Label	Destination state	Current word	Rules applied
V3	(parse VA2)	V19	*ayr*	
VA2	~	VA4	*ayr*	
VA4	*-(I)l*	VA5	*ayrıl*	rule 1 (*I* --> *ı*)
VA5	~	VA/end	*ayrıl*	
V19	(parse VB)	V20	*ayrıl*	
VB	~	VB2	*ayrıl*	
VB2	~	VB3	*ayrıl*	
VB3	*-mI*	VB11	*ayrılmı*	rule 1 (*I* --> *ı*)
VB11	*-(I)yor*	VF	*ayrılmıyor*	rule 9 (*ı* drops)
				rule 1 (*I* --> *ı*)
VF	~	VF1	*ayrılmıyor*	
VF1	~	VF2	*ayrılmıyor*	
VF2	*-(y)DI*	VF3	*ayrılmıyordu*	rule 8 (*(y)* drops)
				rule 4 (*D* --> *d*)
				rule 1 (*I* --> *u*)

Example 3:

word : *ayrılmıyordu* (he/she was not departing)

root : *ayr* (to separate)

root category : verb (V3)

Table 5.15 illustrates the parsing of the word.

5.6. Reasons for Choosing ATN Formalism for the Representation of Turkish Morphology

The formalisms FSTN, RTN, and ATN [46,47,58] have been evaluated for Turkish and we have chosen ATN as the representation schema in this research. There are several reasons underlying this decision. The first one is the efficiency in terms of speed and space. The peculiarity and the complexity of Turkish morphophonemic rules make it difficult to use FSTN or RTN formalisms. When affixing a suffix to a word, it is not enough just to affix the suffix as it is written on the network. Meanwhile, several processes need to be handled. For example, a vowel of the suffix may change due to the vowel harmony rule, the last consonant of the word may change due to the consonant harmony rule 1, or a vowel may be inserted to the word before the last consonant. These modifications (change, deletion,

and insertion of letters) cannot be handled by an FSTN or RTN elegantly; we need a more powerful formalism.

FSTN and RTN result in a huge number of transition networks while defining the morphology. This is due to the fact that the suffixes have different allomorphs and we must place each allomorph into a different network. It is common for a single suffix to have several allomorphs and the number of possible combinations of these allomorphs is very large. Which allomorph is used for a particular case is determined by the morphophonemic rules.

The ATN and the FSTN representations can be compared by an example. Figure 5.2 illustrates one of the ND networks (in the form of an FSTN) that corresponds to the ATN in Figure 5.1 and Table 5.16 is a list of the relevant rules for this FSTN. In other words, only the words that own the properties shown by these rules are parsed using this network. For example, *adam* (man) is such a word - it obeys the vowel harmony rule and the last vowel is a back and unrounded vowel (rule 1), it does not end in *b,c,d,g,* or *ğ* (rules 2 and 3), and so on. Other words make use of other copies of the ND network during parsing. Our analysis has showed us that the number of the ND networks that must be built under this representation is 48 (that is, a single network in ATN representation corresponds to 48 networks in FSTN representation). Considering the fact that other networks also have several copies and the networks have to be merged in different combinations, it is obvious that the result of the FSTN approach is a huge number of networks. This increase in the number of networks means more space to store the network and more time to traverse it.

Another reason is the uniformity of the structure. It is very hard, if not impossible, for some of the Turkish morphophonemic rules to be expressed on an FSTN. The rules that necessitate a change in the root form of the word constitute this group. The vowel insertion rule 1 (see section 5.2) is an example: for some words, a vowel is inserted before the last consonant in the word when a suffix beginning with a consonant is affixed to the word. This modification is outside the power of FSTNs.

One way to handle this is to include two different entries for the word in the lexicon. We can illustrate the case by considering the word *ağz* (mouth), which obeys the vowel insertion rule 1, and the following derivations:

ağz (mouth) + *-(I)m* ---> *ağzım* (my mouth)
ağz (mouth) + *-lAr* ---> *ağızlar* (mouths)

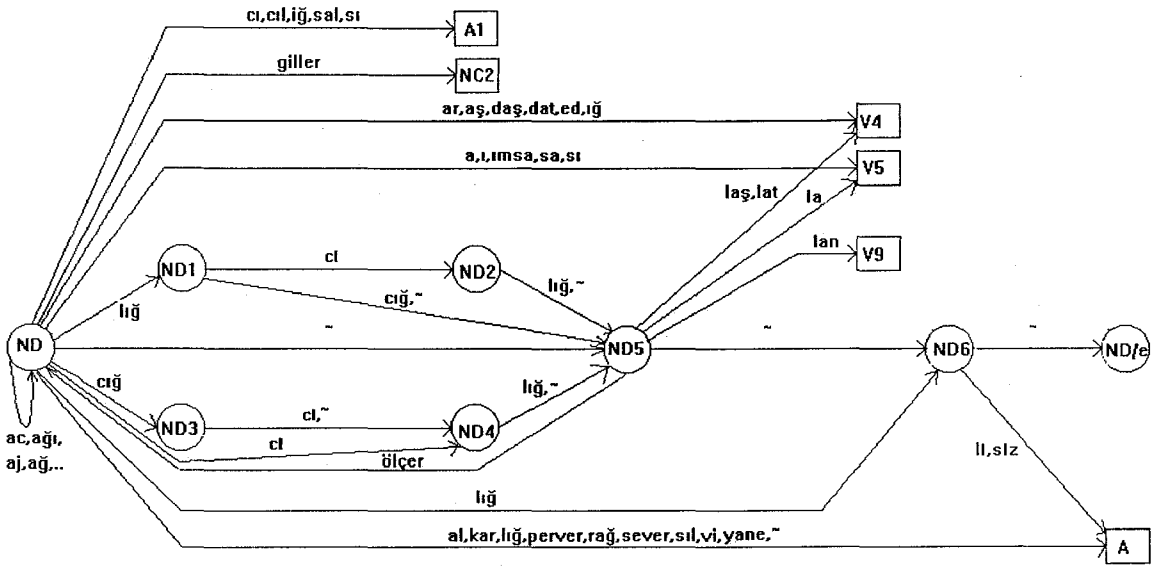


FIGURE 5.2. FSTN representation of the ND network

TABLE 5.16. Rules corresponding to the ND network in Figure 5.2

Rule	Explanation
1	either obeys the rule and lvw is back and unrounded or does not obey the rule and lvw is front and unrounded
2	lpw is not one of *b,c,d,g,ğ*
3	lpw is not *ğ*
4	lpw is not a voiceless consonant
5	does not obey the rule
7	does not obey the rule
8	lpw is a consonant
13	does not obey the rule
14	does not obey the rule
15	does not obey the rule
16	does not obey the rule
18	does not obey the rule

The first entry that must be stored in the lexicon is the base form of the word (*ağz*) and the second one is the form with the last vowel inserted (*ağız*). Then we must create different networks and each form traverses the appropriate network. For example, the network for the first form does not contain any suffix beginning with a consonant as the first suffix (so, *ağz* + *-lAr* ---> *ağzlar*, which is grammatically incorrect, does not produce), and the network for the second form does not contain any suffix beginning with a vowel as the first suffix (so, *ağız* (mouth) + *-(I)m* ---> *ağızım*, which is grammatically incorrect, does not produce). This solution further increases the number of networks.

We can think of using an FST (Finite State Transducer) for these cases [38]. An FST is another kind of finite state automata that allows a string of output symbols to be produced as an input string is recognized (see section 3.2). In the case of the word *ağz* (mouth), the transducer takes the word as input and outputs the form *ağız*. The rest of the affixation process is similar to the process in FSTNs. But the use of FSTs does not free us from creating too many structures; we still need different transducers for traversing.

Another important criteria which is a direct consequence of the uniformity is the clarity of the representation. In this research, we have tried to handle all the cases using the same approach. This is important for a clear understanding and for the contribution of this study in future research.

It is worthwhile to state that the approach proposed in this thesis has some similarities with the two-level description of the morphology. The basic characteristic of the two-level formalism is that the morphotactics is applied to the root word at the first level to obtain the lexical representation, then this lexical representation is modified by the morphophonemic rules at the second level to arrive at the surface representation. In the ATN formalism, these two steps are mixed. Instead of considering the morphotactics as an FSTN and the morphophonemic rules as mediating structures, the full morphology is compiled under a uniform representation environment. While a suffix, determined by the morphotactics, is affixed to a word, the morphophonemic rules work on the word and/or the suffix simultaneously to obtain the surface form. This means that, after each affixation process, the result is the surface form of the word parsed up to that point. It is an easy matter to change the description from one formalism to the other.

In [32,35], the morphotactics is represented as an FSTN and the morphophonemic rules are assumed to take effect during affixation. That is, recognition of suffixes is mediated by a routine which allows for influence of phonological and grammatical context on suffix shape. This approach divides

the process into two. First the network is traversed and a suffix is determined. Then a routine is called which gives its proper form to this suffix according to the current phonological environment.

In the ATN formalism proposed here, however, it is the functions augmenting the network that is responsible from the environment. Each function is triggered automatically when its conditions, which depend on the current context, are satisfied. In this respect, the network serves as the sole mechanism in which the morphology is represented.

VI. LEXICAL AND MORPHOLOGICAL STATISTICS

In this and the next chapter, we shall present statistical information about the morphological structure and the usage of the Turkish language in daily life. The results of these two chapters are based on the morphological structure that has been proposed in the previous sections.

We can group the statistical data analysis in two main categories: lexical and morphological analysis, and corpus analysis. The former one takes into account all the parts of the structure of the language (the words, the affixes, the grammatical rules of the language, etc.). Among the results obtained from this analysis are: the number of roots in the language, the distribution of these roots to the word categories (e.g. noun, verb), the average length of the suffixes, the number of words that obey and do not obey to the rules of the language, and so on. On the other hand, the latter one concerns with the daily usage of the language [59]. For example, what percentage of the language is utilized by people, which are the most frequently used words and suffixes, how many suffixes are affixed to a word on the average (which is the basic point that distinguishes agglutinative and non-agglutinative languages), and so on. The main function of this both type of data is the following:

1. It serves to know the statistics about Turkish. It is not possible to find this kind of data in grammar books or in other references. These references describe the structure and the rules of the language. On the other hand, the statistics give us information on how the language is used in daily life and how these structures and rules are utilized. For example, it may be useful to know the longest word in the dictionary, to know the average length of root words, or to know how many times the plural suffix occurs inside a text.

2. It acts as a base for the researchers who intend to develop language applications, e.g. spelling checker programs or electronic dictionaries. The data contains many useful information for this purpose. Before beginning the application, the data can be analyzed in order to get information about the content of the application and to set it up based on this information. For example, by looking at the statistics about the lexicon (the number of words in the lexicon, the length of the longest word, etc.), one can design different data structures to store the lexicon and then analyze the feasibility of these data structures with respect to the lexical statistics in order to obtain the one with the desired properties. As another example,

for a spelling checker program, it is possible to speed up the parsing of words by designing special data structures and/or algorithms for the frequently used words and suffixes.

3. It can be used for the compression of texts for storage and especially for transmission. Text compression is a challenging and developing area of research. The amount of data that are used by people and that must be distributed among people increases from day to day. The storage and the transmission of this data are expensive operations. The methods that can reduce this cost is of great benefit for this field. The usage patterns of Turkish texts can be analyzed in order to decide how to compress it. For example, a method similar to Huffman coding can be used based on the usage frequencies of letters or words. Alternatively, the words can be split into (morphological) parts and these parts are encoded. The storage and transmission of these parts may consume less space than the individual words.

This chapter deals with the lexical and morphological statistics. As mentioned above, this type of analysis concerns with the structure of the language. The structure is formed from the words of the language, the affixes that are used in building new words, the morphophonemic rules, the rules for the syllabification process, and so on. These can be regarded as static language elements since they do not change from day to day. Of course, a language is an evolving concept and the structural parts may change in time. New words may be included or a rule may be modified to conform with its usage by the people. But from the point of an analyst, they can be regarded as unchanging parts during a fixed period of time.

We divide lexical and morphological statistical data analysis into three groups: lexicon statistics, rule statistics, and suffix statistics. Lexicon statistics refers to the data that are obtained from the root lexicon. Rule statistics is based on the Turkish word formation rules. Finally, suffix statistics corresponds to the results about the suffixes in the suffix lexicon.

Throughout this and the following chapter, we shall consider the root lexicon and the proper noun lexicon as a single lexicon. That is, the statistical results will be obtained as if we have a single lexicon including all these words. However, most of the proper nouns are foreign words and some of the results are more meaningful for Turkish words only. Hence, wherever appropriate, we shall also show the results obtained by excluding the proper nouns.

The results will be presented in the form of tables. Because of lack of space, we shall display here the important parts of these tables. Other parts of the tables, especially the list of the words which correspond to the given statistical data, will be displayed in appendix C.

6.1. Lexicon Statistics

Lexicon statistics refers to the statistical data collected solely from the root lexicon. A root word is defined as a word that is stripped off all the suffixes. The root word lexicon contains, for each root word, the following information: the word and a list of the categories that the word possesses. Regardless of the number of categories, each word occupies a single entry. That is, there does not exist different entries for different categories of a word.

We have made use of the references [53,54,55,56] in forming the lexicon. [53] is the main spelling guide and the others are Turkish dictionaries. We accepted the spelling guide as our basic reference. This means that for words that have different spellings in the references, we have accepted the one in [53] as the correct spelling. However, since we did not use the centralized vowels *â, î,* and *û* in this research because of the reasons sketched in chapter 4, we represented these vowels that take place in [53] with *a, i,* and *ü*, respectively, in our lexicon. The other references have contributed with the words that were absent in the spelling guide. In this way, the lexicon has been built as a combination of these references. Note that these references do not include only the root words; they also include most of the words that were affixed with derivational suffixes. Therefore it was necessary to exclude the words that are not in root form in building the lexicon. The statistical data that will be presented below is obtained from the root lexicon.

The number of root words in the root lexicon is 31,255. The number of root words, excluding the proper nouns, is 21,727.

Table 6.1 shows the number and the percentage of roots in each category in descending order. Note that the total number of roots in this table is greater than the number of roots in the lexicon (which is 31,255). This is due to the fact that some words belong to more than one category (see the next table), hence they appear in all the categories that they own in the table. For example, a word that is both a noun and an adjective increases the word numbers in both of these categories. We see from the table that nearly half of the Turkish root words serve as noun and almost 90 per cent of the

TABLE 6.1. Distribution of words to categories

Category	Number of words	Percentage
n (noun)	16,759	47.68 %
r (proper noun)	11,632	33.09 %
a (adjective)	3,669	10.44 %
v (verb)	1,184	3.37 %
d (adverb)	858	2.44 %
k (acronym)	381	1.08 %
i (interjection)	200	0.57 %
w (unknown)	144	0.41 %
b (chemical symbol)	105	0.30 %
c (conjunction)	76	0.22 %
e (postposition)	50	0.14 %
p (pronoun)	38	0.11 %
l (letter)	32	0.09 %
s (number)	23	0.07 %
Average : 2,511		

root words belong to the three categories noun, proper noun, and adjective. The average number of words per category (obtained by dividing the number of words by the number of categories) is 2,511.

Table 6.2 shows the distribution of the words to the number of categories they own. The left part of the table is obtained by considering all the roots while the right part is obtained after the proper nouns are excluded. The table indicates that a root may belong to as much as six categories. Most of the words, nearly 90 per cent have a single category, nearly 99 per cent have either one or two categories, and there is only one root word in the lexicon that has six categories. We see that, on the average, each root word owns 1.12 categories when the proper nouns are included and 1.11 categories when they are excluded. Table C.1 lists the roots that own more than three categories.

Table 6.3 shows the statistics about the initial letters of root words. Note that the table contains three extra letters (*q,w*, and *x*) that do not belong to the Turkish alphabet. This is due to the categories proper noun (e.g. *Washington*), chemical symbols (e.g. *xe*), and letters (*q,w,x*). In order to arrive at a complete lexicon that includes all the root words that are used in daily usage of the language, we did not exclude these words despite the fact that they contain foreign letters. As a result, the table is formed of 32 rows. We see that nearly 10 per cent of the root words begin with the letter

TABLE 6.2. Distribution of words to number of categories

Number of categories	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
1	27,833	89.05 %	19,737	90.84 %
2	2,995	9.58 %	1,674	7.70 %
3	388	1.24 %	297	1.37 %
4	32	0.10 %	18	0.08 %
5	6	0.02 %	0	0.00 %
6	1	0.00 %	1	0.00 %
	Weighted average : 1.12		Weighted average : 1.11	

k. There is only one word which begins with the letter *ğ*, and this is the letter itself (the category is *l*). This is a consequence of a rule of Turkish: no word begins with *ğ*.

Table 6.4 shows the statistics about the final letters of root words. The layout of the table is similar to the preceding one. One difference is that the number of rows is more than that of Table 6.3. In addition to the three foreign letters *q,w*, and *x*, it includes the character '.', which comes from the acronyms (category *k*). For example, *vs.* is a Turkish acronym (which corresponds to *etc.* in English) whose last character is '.'. The acronyms were collected from [53].

Another statistical figure is the statistics about the initial two letters of root words. Since this is a long table, it will be given in appendix C as Table C.2. It shows the number and the percentage of words that begin with each of the initial two-letter combinations.

Table 6.5 shows the distribution of the root words to the length of the words. The table indicates that the maximum word length is 20 and there are two root words of this length in the lexicon. We see that the mostly occurring length is five which possesses more than 20 per cent of the words. An important result that we get from the table is that the average length of root words is 6.60 for the whole lexicon and is 6.66 when the proper nouns are not taken into account. The words whose lengths are greater than or equal to 17 are shown in Table C.3.

Table 6.6 shows the distribution of all the letters in the lexicon. The lexicon contains 206,258 letters in the first case and 144,746 letters in the second. We notice two extra characters, ' and -. These characters arise from the proper nouns, acronyms, and words whose categories are unknown. The mostly occurring letter is *a* and the mostly occurring three letters are unrounded vowels.

TABLE 6.3. Distribution of words to initial letters

Initial letter	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
k	3,076	9.84 %	2,382	10.96 %
a	2,532	8.10 %	1,524	7.02 %
m	2,486	7.95 %	2,038	9.38 %
t	2,348	7.51 %	1,656	7.62 %
s	2,336	7.47 %	1,568	7.22 %
b	2,049	6.56 %	1,321	6.08 %
d	1,511	4.84 %	1,122	5.16 %
g	1,434	4.59 %	777	3.58 %
h	1,263	4.04 %	988	4.55 %
e	1,237	3.96 %	750	3.45 %
i	1,161	3.71 %	936	4.31 %
p	1,134	3.63 %	936	4.31 %
y	1,016	3.25 %	709	3.26 %
f	850	2.72 %	666	3.07 %
ç	811	2.59 %	595	2.74 %
n	744	2.38 %	404	1.86 %
c	627	2.01 %	395	1.82 %
ş	624	2.00 %	418	1.92 %
o	611	1.95 %	359	1.65 %
r	592	1.89 %	408	1.88 %
ö	568	1.82 %	244	1.12 %
l	462	1.48 %	363	1.67 %
v	433	1.39 %	316	1.45 %
u	405	1.30 %	201	0.93 %
z	398	1.27 %	294	1.35 %
ü	243	0.78 %	128	0.59 %
ı	193	0.62 %	156	0.72 %
j	87	0.28 %	64	0.29 %
w	14	0.04 %	6	0.03 %
q	6	0.02 %	0	0.00 %
x	3	0.01 %	2	0.01 %
ğ	1	0.00 %	1	0.00 %

TABLE 6.4. Distribution of words to final letters

Final letter	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
n	4,130	13.22 %	2,108	9.70 %
r	2,878	9.21 %	1,829	8.42 %
a	2,861	9.15 %	2,094	9.64 %
e	2,604	8.33 %	1,864	8.58 %
t	2,538	8.12 %	2,107	9.70 %
k	2,418	7.74 %	1,843	8.48 %
i	2,313	7.40 %	1,833	8.44 %
l	1,756	5.62 %	1,090	5.02 %
m	1,410	4.51 %	1,191	5.48 %
ı	1,318	4.22 %	1,135	5.22 %
z	881	2.82 %	631	2.90 %
y	780	2.50 %	158	0.73 %
u	780	2.50 %	551	2.54 %
s	685	2.19 %	533	2.45 %
ş	624	2.00 %	400	1.84 %
p	620	1.98 %	407	1.87 %
ç	461	1.47 %	305	1.40 %
o	364	1.16 %	280	1.29 %
f	344	1.10 %	321	1.48 %
h	343	1.10 %	221	1.02 %
ü	334	1.07 %	232	1.06 %
.	186	0.60 %	186	0.86 %
ğ	149	0.48 %	63	0.29 %
g	132	0.42 %	96	0.44 %
v	127	0.41 %	77	0.35 %
j	83	0.27 %	82	0.38 %
d	64	0.20 %	34	0.16 %
b	31	0.10 %	26	0.12 %
c	19	0.06 %	17	0.08 %
ö	10	0.03 %	8	0.04 %
w	7	0.02 %	4	0.02 %
x	3	0.01 %	1	0.00 %
q	2	0.00 %	0	0.00 %

TABLE 6.5. Distribution of words to lengths

Word length	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
1	35	0.10 %	30	0.14 %
2	230	0.74 %	227	1.04 %
3	1,050	3.36 %	983	4.52 %
4	2,653	8.49 %	2,072	9.54 %
5	6,756	21.62 %	4,548	20.93 %
6	6,336	20.29 %	3,827	17.61 %
7	5,025	16.08 %	3,122	14.37 %
8	4,011	12.83 %	2,669	12.28 %
9	2,064	6.60 %	1,540	7.09 %
10	1,458	4.66 %	1,201	5.53 %
11	776	2.48 %	697	3.21 %
12	401	1.28 %	374	1.72 %
13	242	0.77 %	232	1.07 %
14	107	0.34 %	101	0.46 %
15	57	0.18 %	52	0.24 %
16	30	0.10 %	28	0.13 %
17	6	0.02 %	6	0.03 %
18	12	0.04 %	12	0.06 %
19	4	0.01 %	4	0.02 %
20	2	0.01 %	2	0.01 %
	Weighted average : 6.60		Weighted average : 6.66	

TABLE 6.6. Distribution of occurrences to letters

Letter	[with proper nouns] Number of occurrences	Percentage	[without proper nouns] Number of occurrences	Percentage
a	27,149	13.16 %	18,586	12.84 %
e	17,960	8.71 %	12,472	8.62 %
i	14,278	6.92 %	10,767	7.44 %
r	13,721	6.65 %	9,426	6.51 %
n	12,165	5.90 %	7,354	5.08 %
k	11,427	5.54 %	8,366	5.78 %
t	11,119	5.39 %	8,503	5.87 %
l	10,492	5.09 %	7,167	4.95 %
m	8,228	3.99 %	6,418	4.43 %
s	7,948	3.85 %	5,804	4.01 %
o	6,612	3.21 %	5,065	3.50 %
u	6,575	3.19 %	4,134	2.86 %
y	5,850	2.84 %	3,729	2.58 %
ı	5,774	2.80 %	4,646	3.21 %
d	5,216	2.53 %	3,593	2.48 %
b	5,076	2.46 %	3,396	2.35 %
ü	4,631	2.24 %	2,942	2.03 %
z	4,109	1.99 %	2,858	1.97 %
h	3,731	1.81 %	2,490	1.72 %
g	3,513	1.70 %	2,030	1.40 %
p	3,361	1.63 %	2,677	1.85 %
ş	3,105	1.51 %	2,232	1.54 %
v	2,652	1.29 %	1,917	1.32 %
f	2,540	1.23 %	2,060	1.42 %
ç	2,414	1.17 %	1,637	1.13 %
c	2,320	1.12 %	1,521	1.05 %
ö	1,850	0.90 %	1,124	0.78 %
ğ	1,669	0.81 %	1,179	0.81 %
j	446	0.22 %	392	0.27 %
.	241	0.12 %	241	0.17 %
w	39	0.02 %	10	0.01 %
-	23	0.01 %	8	0.01 %
x	11	0.01 %	2	0.00 %
q	10	0.00 %	0	0.00 %
’	3	0.00 %	0	0.00 %
	Total number of letters : 206,258		Total number of letters : 144,746	

6.2. Usage of Lexicon Statistics

The statistics about the initial letters, final letters, and the initial two letters of the words imply several useful informations from the point of view of a language application. For instance, the interpretation of the data for a spelling checker program suggests two important implications. The first one is that we can develop some heuristics about the language. We list below some heuristics that can be noticed at a first glance:

a) Since there are no words beginning with *aö* (see Table C.2), we can stop parsing a word and mark it as a "wrong word" if the initial two letters are *aö*. In other words, we do not need to continue the parsing with all the possible root words and the suffixes. For this heuristic to have a correct effect, we also have to analyze the suffixes to be sure that no derived words in the language can begin with these letters. This analysis can be done beforehand, all the letter combinations that cannot form the initial two characters of the words are extracted, and this can be incorporated into the spelling checker program as a precheck before the parsing of the word.

b) There can be no words in Turkish beginning with the letter *ğ*, except the one that is the letter itself (Table 6.3).

c) Except the categories *letter* and *chemical symbol*, all the words that begin with the foreign letters *q*,*w*, and *x* are proper nouns (this can be seen from the lexicon), and thus the first character must be capitalized. If not, such a word can be marked as misspelled.

The second implication of the data is that it helps one to build a data structure for the organization of the lexicon. We can design different candidate data structures and then compare them in terms of speed and storage space. Here we wish to exemplify this idea with an index mechanism. One possibility is to store the lexicon in alphabetical order and index the words with respect to the initial letters. The search algorithm first identifies the word group corresponding to the initial letter of the word (i.e. the first and last record numbers of the words beginning with this letter) and then performs binary search within this group. Table 6.3 indicates that the word group with the maximum number of entries contains 3,076 entries, hence the binary search needs 12 accesses ($2^{11} < 3{,}076 < 2^{12}$) in the worst case. If the index is with respect to the initial two letters (see Table C.2), then the binary search necessitates 11 accesses ($2^{10} < 1{,}110 < 2^{11}$) in the worst case (after identifying the word

group). Obviously this second data structure consumes more storage space compared to the first one, since the number of the entries increases as the depth of the index mechanism increases. A similar analysis can be performed for other storage structures.

6.3. Rule Statistics

Rule statistics refers to the statistical information about the rules of the Turkish language. Because of the complexity of its morphological structure, there are a large number of rules that are used in Turkish. We can separate the rules into two groups according to their functions:

a) Rules for the root words: This group consists of the rules that are related to the (internal) structure of the root words in the root word lexicon. We shall investigate three rules in this category: primary vowel harmony rule, secondary vowel harmony rule, and last phoneme rule.

b) Rules for the derived words: These are the rules that are used while deriving new words by the suffix affixation process.

6.3.1. Rules for the Root Words

Normally, nearly all of the Turkish words obey the rules in this group. But the language consists of a large number of loanwords; i.e. words that have originated from foreign languages. The principal languages that had an influence on Turkish are Arabic, Persian, English, and French. Some of the words that originated from these languages have been modified so as to comply with the rules of Turkish, but the rest, which constitute the dominant part, have been accepted with little modifications without the concern of the obeyance to the rules. This process has led to a large number of loanwords in the dictionary.

In this section, we shall present statistical data for three well-known rules. We have consulted the root word lexicon in order to obtain the results of this section. The table for each rule displays the number of root words that obey and do not obey the rule. These numbers are classified with respect to the categories. We do not include the category *proper noun* in these tables since the proper noun lexicon contains many foreign words which we can not expect to obey the rules for Turkish roots.

The acronyms are not included either for a similar reason. The percentage in the tables indicates what percentage each category contributes to the total number of words that obey or do not obey the rule. Note that the totals at the end of the tables are less than the sum of the corresponding columns. The reason is the same as that of Table 6.1. In addition to these figures, the tables C.4 through C.6 in appendix C also include a list of some of the words that do not obey to each of the rules.

Primary vowel harmony rule: *All the vowels of a word are either back vowels or front vowels.* See Table 6.7.

Secondary vowel harmony rule: *If the first vowel of a word is unrounded, then the rest of the vowels are unrounded. If the first vowel is rounded, then the rest of the vowels are either high and rounded {u,ü} or low and unrounded {a,e}.* A consequence of this rule is that, except the first vowel, the words do not include low and rounded vowels (*o* and *ö*). See Table 6.8.

Last phoneme rule: *No word ends in the consonants b,c,d, or g.* In this rule, we do not consider the roots that are represented with one of these consonants in the lexicon but that change according to the consonant harmony. For instance, the root *kitab* (book) takes the form *kitap* (book) in its word form, thus does not form an exception to this rule. See Table 6.9.

TABLE 6.7. Primary vowel harmony rule

Category	No of words that obey	Percentage	No of words that do not obey	Percentage
a	2,216	16.00 %	1,453	15.64 %
b	104	0.75 %	1	0.01 %
c	55	0.40 %	21	0.23 %
d	612	4.42 %	246	2.65 %
e	43	0.31 %	7	0.08 %
i	145	1.05 %	55	0.59 %
l	32	0.23 %	0	0.00 %
n	9,308	67.23 %	7,451	80.19 %
p	35	0.25 %	3	0.03 %
s	20	0.14 %	3	0.03 %
v	1,158	8.36 %	26	0.28 %
w	118	0.85 %	26	0.28 %
Total :	12,565		8,807	

TABLE 6.8. Secondary vowel harmony rule

Category	No of words that obey	Percentage	No of words that do not obey	Percentage
a	2,651	15.67 %	1,018	16.36 %
b	103	0.61 %	2	0.03 %
c	70	0.41 %	6	0.10 %
d	729	4.31 %	129	2.07 %
e	48	0.28 %	2	0.03 %
i	171	1.01 %	29	0.47 %
l	32	0.19 %	0	0.00 %
n	11,789	69.70 %	4,970	79.85 %
p	36	0.21 %	2	0.03 %
s	21	0.12 %	2	0.03 %
v	1,147	6.78 %	37	0.59 %
w	117	0.69 %	27	0.43 %
Total :	15,429		5,943	

TABLE 6.9. Last phoneme rule

Category	No of words that obey	Percentage	No of words that do not obey	Percentage
a	3,665	15.94 %	4	2.82 %
b	85	0.37 %	20	14.08 %
c	76	0.33 %	0	0.00 %
d	858	3.73 %	0	0.00 %
e	50	0.22 %	0	0.00 %
i	200	0.87 %	0	0.00 %
l	28	0.12 %	4	2.82 %
n	16,649	72.40 %	110	77.46 %
p	38	0.17 %	0	0.00 %
s	23	0.10 %	0	0.00 %
v	1,182	5.14 %	2	1.41 %
w	142	0.62 %	2	1.41 %
Total :	21,232		140	

6.3.2. Rules for the Derived Words

We shall present here statistical figures about the morphophonemic rules of section 5.2. We shall consider only the rules that have exceptions. We are not interested in the rules that are valid for all of the words since our aim is to collect statistical information about the validity of the rules.

We classify these rules in two groups. The first group consists of the vowel harmony rule, the consonant harmony rule 1, and the aorist suffix rule. For these three rules, we consider the words that do not obey the rule. The second group includes the rest of the rules. For the rules in this group, we consider the words that obey the rule. This classification is due to the generality of the rule. Being a general rule of the language (e.g. vowel harmony rule) means that all the words obey the rule unless stated otherwise. Being an exceptional rule of the language (e.g. double consonant rule) means that none of the words obey the rule unless stated otherwise.

Table 6.10 shows the number of words that do not obey the rules in the first group. Table 6.11 shows the number of words that obey the rules in the second group. It must be noted that, for each rule, only the words that are related to that rule are taken into account. For example, vowel insertion rule 1 applies to nouns only, therefore only the nouns are considered. The list of the words corresponding to these rules can be seen in Tables C.7 through C.21.

6.4. Suffix Statistics

Suffix statistics refers to the statistical data collected solely from the suffix lexicon. The suffix lexicon contains, for each suffix, the following information: the suffix, the source category of the suffix, the destination category of the suffix, and the type of the suffix. The source category indicates the category of the words that the suffix can be affixed to. The destination category indicates the category of the word after the suffix is affixed to. The type of a suffix is either inflectional or derivational. As can be seen, a suffix has as many occurrences in the lexicon as the number of its source and destination category combinations. The lexicon includes only one allomorph of each suffix.

The number of suffixes in the suffix lexicon is 199. Note that this is the number of distinct suffixes. That is to say, regardless of the number of occurrences in the lexicon (because of different source and destination categories for the same suffix), each suffix is counted as a single suffix in this figure.

TABLE 6.10. General rules for the derived words

Rule	Number of words that do not obey the rule
Vowel harmony rule	280
Consonant harmony rule 1	185
Aorist suffix rule	526

TABLE 6.11. Exceptional rules for the derived words

Rule	Number of words that obey the rule
Consonant harmony rule 2	24
Vowel insertion rule 1	171
Vowel insertion rule 2	14
Double consonant rule	43
Phoneme deletion rule for verbs	2
Possessive suffix rule 1	81
Possessive suffix rule 2	5
Rule for compound words 1	233
Rule for compound words 2	428
Rule for compound words 3	25
Rule for compound words 4	840
Rule for the morpheme *su*	19

There are 57 inflectional suffixes and 158 derivational suffixes. Note that the total of these two figures is greater than the number of the suffixes since some of the suffixes function both as an inflectional suffix and as a derivational suffix depending on the source and destination categories.

Table 6.12 shows the distribution of the suffixes to the source categories. For the same reason, the total number of suffixes in this table is greater than the number of suffixes in the lexicon (which is 199), since the suffixes that can originate from more than one category contribute to the numbers in each of these categories. We see that more than 40 per cent of the suffixes are affixed to verbs. Also noun and verb serve as the source categories for nearly 80 per cent of the suffixes. We must note that these figures do not correspond to the usage of the suffixes; they are based on the number of the suffixes, not to the use of them inside a text. The statistics of this second type will be given in chapter 7.

TABLE 6.12. Distribution of suffixes to source categories

Source category	No of suffixes	Percentage
v	112	42.11 %
n	92	34.59 %
s	28	10.53 %
a	14	5.26 %
r	12	4.51 %
d	6	2.26 %
i	1	0.38 %
p	1	0.38 %

Table 6.13 shows the distribution of the suffixes to the destination categories. The layout of the table is similar to the preceding one. The character * as the category implies the ending category, i.e. no more suffixes can be affixed to the words that arrive to this state.

Table 6.14 is an extended form of the previous two tables. It shows the suffix distribution for both the source and the destination categories. For example, the number of suffixes that are affixed to adjectives and result in adjectives is seven. We see that the total number of suffixes for the category adjective is 15, which differs from Table 6.12. This is because of the same reason.

TABLE 6.13. Distribution of suffixes to destination categories

Destination category	No of suffixes	Percentage
n	84	32.56 %
v	69	26.74 %
a	37	14.34 %
d	30	11.63 %
s	23	8.91 %
r	7	2.71 %
*	5	1.94 %
c	1	0.39 %
e	1	0.39 %
p	1	0.39 %

TABLE 6.14. Distribution of suffixes to source and destination categories

Source category	Destination category	No of suffixes
a	a	7
	d	6
	n	1
	v	1
d	c	1
	d	5
i	v	1
n	a	17
	d	9
	n	56
	v	18
	*	1
p	p	1
r	n	3
	r	7
	v	2
s	a	4
	n	1
	s	23
v	a	12
	d	15
	e	1
	n	37
	v	54
	*	2

Table 6.15 shows the distribution of the suffixes to the length of the suffixes. The maximum suffix length is seven and there are three suffixes of this length in the lexicon. We see that the mostly occurring length is three which possesses more than 30 per cent of the suffixes. An important result that we get from the table is that the average length of the suffixes is 3.56.

Table 6.16 shows the statistics about the initial letters of the suffixes. In this table, all the possible initial phonemes of the suffixes are taken into account. For example, the initial phoneme of the suffix *-(I)m* may be *ı, i, u, ü,* or *m*, depending on the word that it is affixed to. Thus this suffix increases by one the number on the column *number of suffixes* for each of these five phonemes. This is better than just counting the first phoneme of the suffixes as they are shown in the suffix lexicon. In such a case, for example, the phoneme *e* would not take part in the table (although it is used frequently during affixation) since *e* is represented by *A* in the suffix lexicon.

TABLE 6.15. Distribution of suffixes to suffix length

Length	Number of suffixes	Percentage
1	7	3.52 %
2	41	20.60 %
3	61	30.65 %
4	36	18.09 %
5	35	17.59 %
6	16	8.04 %
7	3	1.51 %

Weighted average : 3.56

TABLE 6.16. Distribution of suffixes to initial letters

Initial letter	Number of suffixes	Percentage
y	49	11.09 %
a	41	9.28 %
e	40	9.05 %
i	38	8.60 %
ı	35	7.92 %
ü	34	7.69 %
u	33	7.47 %
s	25	5.66 %
m	22	4.98 %
d	15	3.39 %
k	15	3.39 %
t	15	3.39 %
l	14	3.17 %
n	12	2.71 %
c	11	2.49 %
ç	10	2.26 %
g	8	1.81 %
b	5	1.13 %
z	4	0.90 %
o	3	0.68 %
ş	3	0.68 %
r	2	0.45 %
v	2	0.45 %
ğ	2	0.45 %
h	1	0.23 %
j	1	0.23 %
p	1	0.23 %
ö	1	0.23 %

There are 28 rows in the table. We notice that there are no suffixes that begin with the letter *f* and with the foreign letters (as expected). The letter *y* is the leading letter serving as the beginning letter for more than 11 per cent of the suffixes.

Table 6.17 shows the statistics about the final letters of the suffixes in a similar layout with that of the preceding table. As in the preceding table, all the allophones of the final letters of the suffixes are considered. One difference from Table 6.16 is the smaller number of rows in the table. There are only 22 phonemes that can occur as the final phoneme of suffixes.

TABLE 6.17. Distribution of suffixes to final letters

Final letter	**Number of suffixes**	**Percentage**
e	30	9.80 %
n	30	9.80 %
a	26	8.50 %
k	25	8.17 %
i	24	7.84 %
r	22	7.19 %
ğ	21	6.86 %
ı	20	6.54 %
u	18	5.88 %
z	18	5.88 %
ü	18	5.88 %
m	10	3.27 %
l	9	2.94 %
ş	9	2.94 %
t	8	2.61 %
ç	6	1.96 %
c	4	1.31 %
y	3	0.98 %
d	2	0.65 %
j	1	0.33 %
p	1	0.33 %
v	1	0.33 %

VII. CORPUS STATISTICS

The statistical data obtained in the previous chapter are based on lexical and morphological structure of the language. The root word lexicon, the suffix lexicon, and the rules of the language form the basis of the analysis. Another important issue concerning natural language processing is the use of the language in daily life. The data obtained in this way can be regarded as dynamic data. This second kind of data is of vital importance for the language analysts. It shows explicitly how the language is used by people. The direction in which the language is evolving, the acceptance percentage of the new words by people, the effects of foreign languages on the language are among the conclusions that the analyst can acquire from the analysis.

This chapter is devoted to the presentation of statistical information about the usage of Turkish language. The method that we employ is to run a spelling checker program on a corpus and record the output of the program. The spelling checker program that we utilize is the one that has been developed as a part of this research. The program will be explained in detail in chapter 8.

The program has been run on a corpus of 2,203,787 words. The content of this corpus was formed from the following:

- Text from a daily newspaper, *Sabah* (1,837,451 words). The content includes all type of news.
- Text from Bilkent University (190,224 words). The text is formed of different domains.
- A novel (66,347 words). The novel is about the life of a family [60].
- Text from a weekly periodical, *Aktüel* (42,180 words). This periodical contains news from daily life.
- A novel (38,105 words). The author defines his novel as *a psychological and philosophical investigation of the nature of the human being* [61].
- A novel (29,480 words).

It is obvious that the statistics reflects the real usage of the language more clearly as the number of the input data increases. This requires the necessity of developing comprehensive corpora which will serve as a data bank for any natural language processing in Turkish.

Now we present the tables that contain the results about the corpus. Table 7.1 includes general statistical figures about the corpus. The numbers under the left column are obtained by considering all the words. The numbers under the right column are obtained after the proper nouns are excluded from the lexicon. For this second case, the lexicon does not contain any proper noun, hence the proper nouns in the corpus are accepted as misspelled words. The meaning of these figures is as follows:

a) **Number of words:** This figure shows the number of words in the corpus.

b) **Number of distinct words:** This figure shows the number of distinct words in the corpus. This means that all the occurrences of a word are regarded as a single occurrence.

c) **Average word usage:** This figure indicates, on the average, how many times each word is used in the corpus. It is obtained by the formula "a / b".

d) **Number of successful parses:** This figure shows the number of words that the spelling checker program had been able to parse. In other words, these words comply with the rules of the Turkish syntax. They either are root words that take place in the root word lexicon or can be derived from the root words with the application of the morphotactics and morphophonemic rules. The number of successful parses decreases when we exclude proper nouns from the lexicon.

e) **Number of unsuccessful parses:** This figure shows the number of words that the spelling checker program was unable to parse and marked as *grammatically wrong*. In other words, these words do not comply with the rules of the Turkish syntax. It is obtained by the formula "a - d". Depending on the program used, these may either be grammatically wrong words as said by the program or be grammatically correct words but were outside the capacity of the program. The major reason of this second kind of unsuccessful parses, as also encountered by the program used in this research, is the proper nouns that are not included in the lexicon. The number of proper nouns is huge and beyond the capacity of any lexicon.

We see that 8.88 per cent of the words in the corpus are counted as grammatically wrong words. For the case in which the proper nouns are not taken into account, this percentage increases to 13.24. This indicates that the proper noun lexicon used by this spelling checker program can handle 4.36 per cent of the words for this particular corpus.

TABLE 7.1. General figures about the corpus

	[with proper nouns]	[without proper nouns]
a) Number of words	: 2,203,787	
b) Number of distinct words	: 200,120	
c) Average word usage	: 11.01	
d) Number of successful parses	: 2,008,145	1,912,061
e) Number of unsuccessful parses	: 195,642	291,726
f) Number of distinct roots	: 11,806	9,491
g) Average root usage	: 170.10	201.46
h) Percentage of lexicon usage	: 37.77	43.68
i) Number of affixed words	: 1,026,095	982,152
j) Number of unaffixed words	: 982,050	929,909
k) Number of words that do not change category	: 1,568,741	1,501,110
l) Number of words that change category	: 439,404	410,951
m) Minimum word length	: 1	
n) Maximum word length	: 25	
o) Average word length	: 6.13	
p) Minimum root length	: 1	1
q) Maximum root length	: 16	16
r) Average root length	: 4.03	4.01
s) Minimum number of suffixes	: 0	0
t) Maximum number of suffixes	: 8	8
u) Average number of suffixes for all words	: 0.94	0.97
v) Average number of suffixes for affixed words	: 1.85	1.90
w) Minimum suffix length	: 1	1
x) Maximum suffix length	: 7	7
y) Average suffix length	: 2.44	2.44

f) Number of distinct roots: This figure shows the number of distinct roots in the corpus.

g) Average root usage: This figure indicates, on the average, how many times each root word is used in the corpus. It is obtained by the formula "d / f".

h) Percentage of lexicon usage: This figure shows what percentage of the root word lexicon is utilized by the corpus. It is obtained by the formula "f / number of root words in the lexicon * 100". The number of root words is 31,255 including the proper nouns and 21,727 without the proper nouns (see section 6.1). We must note that since the contents of the lexicons differ slightly, this figure yields different numbers for different spelling checker programs. However, we include it here in order to give a general idea about the proportion of the roots used.

i) Number of affixed words: This figure shows the number of words in the corpus that are affixed with at least one suffix, i.e. the words that are not in root form.

More than half of the words in the corpus are in affixed form.

j) Number of unaffixed words: This figure shows the number of words in the corpus that are not affixed with any suffix, i.e. the words that are in root form. It is obtained by the formula "d - i".

k) Number of words that do not change category: This figure shows the number of words whose initial category (the category of the root word of which the word has derived) and final category (the ending category of the word) are the same. This number is always greater than or equal to the number shown in part j, since the root category acts as both the initial and the final category for unaffixed words.

About 78 per cent of the successfully parsed words in the corpus do not change their categories.

l) Number of words that change category : This figure shows the number of words whose initial and final categories differ. It is obtained by the formula "d - k". In a similar way, this number is always less than or equal to the one shown in part i.

m) Minimum word length: This figure shows the length of the shortest word in the corpus. It is obvious that for almost every corpus this number evaluates to one.

n) Maximum word length: This figure shows the length of the longest word in the corpus. The words of this maximum length can be seen in Table C.22.

o) Average word length: This figure shows the average length of the words contained in the corpus. This is an important figure as it is an indication of the word lengths used in daily life.

p) Minimum root length: This figure shows the length of the shortest root word in the corpus.

q) Maximum root length: This figure shows the length of the longest root word in the corpus. The list of the longest roots is given in Table C.23.

We see that the roots in the lexicon with length greater than 16 are not used in this corpus.

r) Average root length: This figure shows the average length of the root words contained in the corpus.

The average root length for this corpus is 4.03 (or, 4.01 without proper nouns). We see from Table 6.5 that the same figure for the lexicon is 6.60 (or, 6.66). The difference between these two implies that people prefer shorter root words during typing.

s) Minimum number of suffixes: This figure shows the least number of suffixes that are affixed to a word in the corpus. Obviously, it evaluates to zero for almost every corpus.

t) Maximum number of suffixes: This figure shows at most how many suffixes are affixed to a word in the corpus. For agglutinative languages, theoretically there is no upper limit in the number of affixations. And it is not unusual to find words formed of several suffixes in texts. This is the basic point that distinguishes agglutinative and non-agglutinative languages.

u) Average number of suffixes for all words: This figure shows the number of suffixes that are affixed to a word on the average. It is obtained by considering all the successfully parsed words (part d). It is calculated as: "total number of suffixes used in the corpus / d".

In the corpus, on the average, each word is affixed with nearly one suffix.

v) Average number of suffixes for affixed words: This figure shows the number of suffixes that are affixed to a word on the average. It is obtained by considering only the affixed words (part i). It is calculated as: "total number of suffixes used in the corpus / i". This number is always greater than or equal to the one shown in part u.

w) Minimum suffix length: This figure shows the length of the shortest suffix that is used in the corpus. It evaluates to one for almost every corpus since there are several suffixes of length one in Turkish.

x) Maximum suffix length: This figure shows the length of the longest suffix that is used in the corpus. This number is less than or equal to the maximum suffix length shown in Table 6.15. Being less than this number implies that the longer suffixes are not used in the corpus.

y) Average suffix length: This figure shows the average length of the suffixes in the corpus. An interesting result that can be obtained is the following: The average root word length plus the average number of suffixes multiplied by the average suffix length yields more or less the average word length. Stated in another way, (r + u * y) is more or less equal to o.

In the suffix lexicon, the average suffix length is 3.56 (Table 6.15). In the corpus it is 2.44. This means that shorter suffixes are used more frequently in the corpus.

Table 7.2 and Table 7.3 list some of the most frequently used words and roots, respectively, in the corpus. For each word and root, its number and percentage of occurrence in the corpus are given. The tables contain words and roots whose percentages of occurrence are greater than 0.10 per cent. In these tables, we notice some words like *yüzde* (percentage), a proper noun *Rusya* (Russia), and the number *10*. We think that these words do not appear frequently (at least, with the frequencies shown in the tables) in Turkish texts. In our opinion, this is due to the heavy periodical content of the corpus. Especially the word *yüzde* (percentage) bears this property. The reason for the word *Rusya* (Russia) is that the text contains an investigation about Russia.

The next two tables, Table 7.4 and Table 7.5, display the distribution of words to the categories. The first one lists the number of root words originating from each category. The second one lists the number of words that end in each category. Table 7.6 shows the number of occurrences of the suffixes used in the corpus.

In appendix C, two more tables, Table C.22 and Table C.23, are given. These tables contain a list of some of the longest words and roots, respectively, used in the corpus.

TABLE 7.2. Most frequently used words

Word	Number of usage	Percentage
bir	49,438	2.24 %
ve	42,236	1.92 %
da	25,233	1.14 %
bu	24,376	1.11 %
için	12,000	0.54 %
ile	9,246	0.42 %
çok	8,759	0.40 %
o	8,277	0.38 %
daha	7,415	0.34 %
olarak	6,752	0.31 %
olan	6,701	0.30 %
sonra	6,697	0.30 %
ama	6,314	0.29 %
gibi	5,530	0.25 %
en	5,449	0.25 %
kadar	5,375	0.24 %
büyük	4,516	0.20 %
her	4,440	0.20 %
ise	4,203	0.19 %
konu	3,802	0.17 %
ne	3,527	0.16 %
bin	3,398	0.15 %
iki	3,234	0.15 %
yeni	3,230	0.15 %
yıl	2,837	0.13 %
yüzde	2,617	0.12 %
ancak	2,539	0.12 %
10	2,381	0.11 %
ilk	2,264	0.10 %

TABLE 7.3. Most frequently used roots

Root	Number of usage	Percentage
bir	52,951	2.40 %
ve	42,235	1.92 %
ol	39,883	1.81 %
da	25,233	1.14 %
bu	24,472	1.11 %
ed	17,866	0.81 %
için	12,066	0.55 %
yap	11,894	0.54 %
yıl	11,209	0.51 %
de	11,009	0.50 %
ile	8,801	0.40 %
baş	8,490	0.39 %
al	8,426	0.38 %
çoğ	8,417	0.38 %
o	7,667	0.35 %
gör	7,355	0.33 %
ara	7,201	0.33 %
daha	7,120	0.32 %
ver	6,935	0.31 %
sonra	6,601	0.30 %
bul	6,002	0.27 %
kendi	5,891	0.27 %
geç	5,454	0.25 %
ama	5,379	0.24 %
var	5,329	0.24 %
konu	5,102	0.23 %
ben	4,747	0.22 %
on	4,349	0.20 %
kadar	4,262	0.19 %
iste	4,256	0.19 %
bun	4,220	0.19 %
gibi	4,154	0.19 %
iki	4,138	0.19 %
en	4,096	0.19 %
büyüğ	3,369	0.15 %
karşı	3,352	0.15 %
bil	3,304	0.15 %
gel	3,221	0.15 %
zaman	3,189	0.14 %
ise	3,130	0.14 %
söyle	3,084	0.14 %
bin	2,807	0.13 %
ne	2,794	0.13 %
çalış	2,774	0.13 %

TABLE 7.3. Most frequently used roots (continued)

Root	Number of usage	Percentage
son	2,772	0.13 %
yüzde	2,673	0.12 %
yer	2,629	0.12 %
kadın	2,292	0.10 %
aç	2,261	0.10 %
rusya	2,225	0.10 %

TABLE 7.4. Distribution of words to initial categories

Category	Number of words	Percentage
n (noun)	1,038,323	47.12 %
v (verb)	459,457	20.85 %
a (adjective)	212,459	9.64 %
s (number)	150,366	6.82 %
c (conjunction)	96,822	4.39 %
r (proper noun)	96,084	4.36 %
d (adverb)	89,507	4.06 %
e (postposition)	28,918	1.31 %
p (pronoun)	22,695	1.03 %
i (interjection)	5,527	0.25 %
l (letter)	2,346	0.11 %
w (unknown)	710	0.03 %
b (chemical symbol)	387	0.02 %
k (acronym)	186	0.01 %

TABLE 7.5. Distribution of words to final categories

Category	Number of words	Percentage
n (noun)	1,100,980	49.96 %
v (verb)	333,613	15.14 %
a (adjective)	266,311	12.08 %
d (adverb)	151,859	6.89 %
s (number)	134,115	6.09 %
c (conjunction)	101,496	4.61 %
r (proper noun)	67,631	3.07 %
e (postposition)	26,119	1.19 %
p (pronoun)	14,013	0.64 %
i (interjection)	4,032	0.18 %
l (letter)	2,346	0.11 %
w (unknown)	710	0.03 %
b (chemical symbol)	387	0.02 %
k (acronym)	175	0.01 %

TABLE 7.6. Distribution of suffixes

Suffix	Number of words	Percentage
-ın	224,645	11.89 %
-sı	220,804	11.69 %
-lar	168,180	8.90 %
-ya	111,828	5.92 %
-da	100,585	5.32 %
-ma	73,843	3.91 %
-yı	60,142	3.18 %
-dığ	47,322	2.50 %
-dan	45,082	2.39 %
-dır	44,825	2.37 %
-yan	44,083	2.33 %
-nın	41,836	2.21 %
-lı	41,398	2.19 %
-ıl	39,001	2.06 %
-ım	37,786	2.00 %
-ıyor	33,470	1.77 %
-n	32,882	1.74 %
-lığ	32,727	1.73 %
-dı	32,449	1.72 %
-ır	27,673	1.46 %
-mış	24,517	1.30 %
-yla	22,898	1.21 %
-yacağ	22,583	1.20 %
-la	22,521	1.19 %
-ydı	22,045	1.17 %
-mağ	16,949	0.90 %
-lan	16,107	0.85 %
-yarağ	15,905	0.84 %
-ı	15,070	0.80 %
-iğ	14,788	0.78 %
-cı	13,944	0.74 %
-ki	13,506	0.71 %
-al	13,219	0.70 %
-t	11,975	0.63 %
-ca	9,326	0.49 %
-yabil	9,227	0.49 %
-ımız	8,555	0.45 %
-sız	7,688	0.41 %
-yıp	7,265	0.38 %
-yım	7,196	0.38 %
-gı	7,163	0.38 %
-m	7,154	0.38 %
-laş	7,070	0.37 %
-ığ	7,068	0.37 %

TABLE 7.6. Distribution of suffixes (continued)

Suffix	Number of words	Percentage
-tı	6,702	0.35 %
-makta	5,882	0.31 %
-ış	5,364	0.28 %
-maz	5,237	0.28 %
-yın	4,842	0.26 %
-yız	4,828	0.26 %
-ağ	4,794	0.25 %
-yken	4,508	0.24 %
-ınız	3,916	0.21 %
-sın	3,741	0.20 %
-ysa	3,691	0.20 %
-ed	3,678	0.19 %
-yıcı	3,518	0.19 %
-mı	3,336	0.18 %
-ymış	3,108	0.16 %
-gın	2,960	0.16 %
-sal	2,387	0.13 %
-malı	2,351	0.12 %
-sa	2,227	0.12 %
-ıncı	2,147	0.11 %
-yınan	1,805	0.10 %
-sınız	1,767	0.09 %
-ına	1,596	0.08 %
-lat	1,428	0.08 %
-k	1,221	0.06 %
-yınca	1,215	0.06 %
-nız	971	0.05 %
-yılan	910	0.05 %
-ist	854	0.05 %
-yamı	798	0.04 %
-ıt	778	0.04 %
-en	766	0.04 %
-izm	675	0.04 %
-şar	501	0.03 %
-hanē	457	0.02 %
-ıntı	348	0.02 %
-mac	270	0.01 %
-name	251	0.01 %
-yıver	230	0.01 %
-cıl	204	0.01 %
-cığ	192	0.01 %
-sınlar	190	0.01 %
-ar	142	0.01 %
-lıkla	131	0.01 %

TABLE 7.6. Distribution of suffixes (continued)

Suffix	Number of words	Percentage
-ız	96	0.01 %
-bin	86	0.00 %
-yınız	78	0.00 %
-bir	71	0.00 %
-yagel	69	0.00 %
-yasıya	69	0.00 %
-gan	65	0.00 %
-gen	55	0.00 %
-yalı	52	0.00 %
-yası	45	0.00 %
-ac	36	0.00 %
-beş	29	0.00 %
-na	29	0.00 %
-kca	25	0.00 %
-casına	24	0.00 %
-ımsı	23	0.00 %
-yakal	21	0.00 %
-yüz	18	0.00 %
-casi	16	0.00 %
-milyon	13	0.00 %
-nıza	12	0.00 %
-mamaz	10	0.00 %
-yadur	9	0.00 %
-dörd	9	0.00 %
-üç	9	0.00 %
-iki	8	0.00 %
-larsa	8	0.00 %
-si	8	0.00 %
-altı	7	0.00 %
-yedi	7	0.00 %
-sekiz	5	0.00 %
-giller	5	0.00 %
-on	4	0.00 %
-elli	3	0.00 %
-yabilin	2	0.00 %
-yaca	2	0.00 %
-y	2	0.00 %
-d	2	0.00 %
-yagör	2	0.00 %
-otuz	2	0.00 %
-dokuz	1	0.00 %
-nı	1	0.00 %
-s	1	0.00 %
-yakoy	1	0.00 %

VIII. DESIGN AND IMPLEMENTATION OF TOOLS

The morphological structure of Turkish proposed in this research has been used for the design of a spelling checker program [17,62]. Spelling checker is one of the major application areas for agglutinative languages. Until recently spelling checking was done by human beings. However, as the number of written material that must be processed increases it becomes impossible to spell check all these documents in a reasonable time and at a reasonable cost, thus the process must be automated. For non-agglutinative languages like English and French, spelling checking is almost completely solved. On the other hand, due to the complexity of the word formation process, spelling checking in agglutinative languages is a difficult problem.

The spelling checker program implemented as a part of this research also incorporates a spelling corrector component. As far as we know, this is the first spelling corrector implemented for Turkish.

The source code of the program was written in standard Pascal. It was first developed on a PC and later ported to a VAX 4000/200 running VMS 5.5 machine. The executable code of the program occupies 37 KB of space. In the following sections, we shall describe the components of the program.

8.1. Root Lexicon

The root lexicon holds the words in their root forms, i.e. stripped off the suffixes. It contains approximately 21,500 roots. We built the lexicon by the use of the references Aksoy [53] and Eren [55,56]. Aksoy [53] is the main spelling guide; Eren [55,56] are the main Turkish dictionaries published by the Turkish Language Institution. Between the spelling guide and the dictionaries there are some controversial points about the spelling of the words. We have accepted the spelling guide as our basic reference (except the use of the centralized vowels *â, î,* and *û*). This means that for words that have different spellings in the references, we have accepted the one in [53] as the correct spelling. The other two references have contributed with the words that were absent in the first one.

The spelling guide does not include the categories of the words. Hence we have found the categories from the dictionaries. However, we have encountered with words that exist in the spelling guide but not in the dictionaries. In order to find their categories, we consulted another dictionary, [54], which is a publication of the Redhouse press. Finally, we have compiled the words that do not exist in this dictionary either in the lexicon under a special category (the category for unknown words, W - see section 5.1).

Note that the references listed above do not include only the root words; they also include most of the words that were affixed with derivational suffixes. Therefore it was necessary to exclude the words that were not in root form in building the lexicon.

Some of the derivational suffixes are not productive. As will be explained in section 8.5, if a suffix is not affixed to all the words in the related category, then a flag is reserved in the lexicon in order to determine to which words it can be affixed. Each flag consumes some storage. Therefore, in order to minimize storage requirements and to decrease search time we collected the suffixes that can be affixed to less than 20 words and inserted these words in their affixed form directly into the lexicon. For example, the suffix *-perver* is used for the following roots only: *cumhuriyet* (republic), *hayr* (philanthropy), *menfaat* (advantage), *milliyet* (nationality), *misafir* (guest), *sulh* (peace), *terakki* (progress), and *vatan* (motherland). It adds a meaning like "a person who is fond of ..." as in the following example:

misafir (guest) + *-perver* --> *misafirperver* (hospitable)

The lexicon contains both of the words *misafir* (guest) and *misafirperver* (hospitable).

The implemented program is a general purpose spelling checker. That is, it was not designed for a particular field. Hence the root lexicon does not include technical terms. However, the program has the feature of using other lexicons together with the standard root lexicon. In this way, the user can build his/her own lexicons and add these to the program.

8.2. Data Structure and Storage of the Root Lexicon

For each root word, the following information is stored in the lexicon:

- The word
- The categories of the word
- The flags for the word (see section 8.5).

Regardless of the number of categories, each root word occupies a single entry. For instance, the root *tad*, which is both a noun (meaning "taste") and a verb (meaning "to taste"), is stored as: *tad , n v.*

From the point of view of minimizing the storage space and reducing the search time, that are two design choices that must be decided. These two usually conflict with each other. If a compression scheme is used to save space, then the number of applicable search techniques is limited due to the nature of the compression. There is also the additional overhead for decompression. This overhead may be a very important computational burden in the case of spelling correction where dozens of alternatives for a given incorrectly spelled word must be considered. For these reasons, the root lexicon was stored in an incompressed way.

For the data structure that is to be used to store the root lexicon, we investigated some alternatives. In accordance with the root matching algorithm, the following access method was selected: The root lexicon is sorted alphabetically and stored in an array. While searching a root, an indexing mechanism is used. There is another array which holds the first and last positions of the roots in the root lexicon array for each initial two letters of the roots. For example, to search the root *kalem* (pencil), we take the initial two letters, which is *ka*, consult the indexing array to determine the position of the first root word in the root lexicon array that begins with *ka* (say, 11000) and the position of the last root word in the root lexicon array that begins with *ka* (say, 11050). Then we sequentially search all the roots in the root lexicon array between these two limits (i.e. 11000 and 11050).

Note that in consulting the indexing array we make use of a unique hashing function. Each character has an ordinal value. For example, the ordinal value of *k* is 14 and that of *a* is 1. The

hashing function is as follows: *ord(k)*33+ord(a)*. Thus, we do not perform any search on the indexing array; we access it only once for each word.

8.3. Proper Noun Lexicon

The proper nouns usually bring additional difficulties to spelling checking. One source of difficulty originates from the foreign words. These words may contain some letters that do not exist in the native alphabet. For example, the letter *w* in the word *Washington* is not an element of the Turkish alphabet.

The number of proper nouns in a language is much greater than the number of words in all other categories. Even the people names constitute a large group by itself [63,64,65,66]. As the number of entries in the lexicon increases, the search time also increases. Keeping the number of entries in the root lexicon at a minimum has the benefit of decreasing the average parsing time.

To handle these problems, the proper nouns were separated and put into a special lexicon. From a computational point of view, the probability of a given word in a document to be a proper noun is much less than the probability for all other categories. Given a word, the program first searches the root lexicon and only if the word is not found the proper noun lexicon is accessed. In this way, for most of the words, the proper noun lexicon is not searched at all.

The lexicon contains approximately 11,500 proper nouns. We have included three types of words:

- Turkish people names from [66]
- Geographical names like countries, cities, mountains, lakes, etc. from [67]
- All the proper nouns in the spelling guide [53]. These form a large spectrum. They include geographical names, names of nations, names of institutions, religious terms, astronomic terms, and so on.

For each proper noun, the following information is stored in the lexicon:

- The word
- The flags for the word (see section 8.5).

Since the lexicon is dedicated to the proper nouns only, it is no longer necessary to have a field that indicates word category.

The data structure used for the proper noun lexicon is similar to that of the root lexicon. The only difference is that we designed the indexing array according to the initial three letters of words instead of the initial two letters.

8.4. Suffix Lexicon

The suffix lexicon contains approximately 200 suffixes. In Turkish, most of the suffixes have several allomorphs (a single suffix can have 24 allomorphs). The suffix lexicon contains only one form of each suffix; the allomorph that must be used in a particular case is determined by the rules.

For each suffix, the following information is stored:

- The suffix
- Type of the suffix: The type is either inflectional or derivational. This information is utilized by the rule for proper nouns (see section 5.2).
- Affixation flag index (see section 8.5)
- Termination flag : If this flag is set, then the suffix can be the last suffix of a word. In other words, after this suffix is affixed to the word, the word is in a grammatically correct state. Otherwise the word formed after this suffix is attached is not a correct Turkish word. For example, the termination flag for the suffix *-mAz* is not set. Hence the following is not a correct word:

  gel (to come) + *-mA* + *-mAz* --> *gelmemez*

  while the following is a grammatically correct word since the termination flag of the suffix *-lIĞ* is set:

  gel (to come) + *-mA* + *-mAz* + *-lIĞ* --> *gelmemezlik* (not coming)

- Destination category : This field indicates the category of the word after the suffix is affixed.

■ Particle flag: This flag is used to denote whether the suffix must be written separate from the word it follows or not. If it is set, then the suffix is a particle and must be written separate from the word it is affixed to. In the suffix lexicon, this flag is set for the particles *-dA* (the emphatic particle), *-ki* (complementizer), and *-mI* (the question particle).

■ For each letter in the suffix, a deformation flag and a drop flag : These flags correspond to the symbols *capital letter* and *( )*, respectively, that we have been using throughout the thesis. For example, the suffix *-(I)mIz* has the deformation flags (1,0,1,0) and the drop flags (1,0,0,0), where 1 indicates that the flag is set and 0 indicates that the flag is not set.

The data structure utilized for the suffix lexicon is a bucket structure. The suffixes that can be affixed in a particular category are grouped together and form a bucket. This is the source category of these suffixes. The suffixes within a bucket are sorted alphabetically. Given the source category, the suffixes in this bucket are searched sequentially.

Before the suffix lexicon was formed, we made a detailed analysis for the suffixes used in Turkish. There are a large number of derivational suffixes. We collected these suffixes and extracted the words that use these suffixes. If a suffix can not be affixed to all the words in the related category, we must reserve a flag for the suffix in the lexicon. Each additional flag implies more storage. For this reason, we decided not to include rarely used suffixes in the suffix lexicon. Instead, the words that can receive these suffixes were put into the lexicon in their affixed forms. We give the listing of the suffixes that were not inserted into the suffix lexicon in appendix D. The listing also includes the words to which these suffixes can be affixed.

8.5. Flags

Most of the morphophonemic rules defined in section 5.2 are valid only for a subset of the words, not for all of the words. In order to obtain the correct spelling of a word, we must know if a particular rule is applicable to this word or not. For example, consider the vowel harmony rule in the following derivation:

kalem (pencil) + *-(I)m* --> *kalemim* (my pencil) or

kalem (pencil) + *-(I)m* --> *kalemım*

Only one of the above derivations is valid. If the root *kalem* (pencil) obeys the vowel harmony then the first one, otherwise the second one is valid. In order to be able to decide which one is the correct word, we must know if the root obeys to the rule or not.

As discussed in section 5.3, the productivity of a derivational suffix highly differs from one suffix to another. At one extreme it can be affixed to all the words in the related category; at the other extreme it can be affixed to a single word. An example of this second type of suffixes is the suffix *-kek*, which can only be affixed to the noun *er* (male) to form the noun *erkek* (man).

During the parsing of a word, we must know whether a suffix can be affixed to a word or not. Consider the following derivations:

çirkin (ugly) + *-lAş* --> *çirkinleş* (to become ugly)
solgun (pale) + *-lAş* --> *solgunlaş*

Both of the words *çirkin* (ugly) and *solgun* (pale) are adjectives. *-lAş* is a suffix that is attached to adjectives or nouns, forming a verb. It adds a meaning like "to become ...". But the second derivation above does not yield a meaningful Turkish word. Hence in order to prevent such derivations, we must know to which adjectives and nouns the suffix *-lAş* can be affixed.

We use the *flag* concept in order to handle both of these issues. Each word in the root lexicon and the proper noun lexicon has a number of associated flags. The value of each flag is either 1 (the flag is set) or 0 (the flag is not set). There are two types of flags: rule flags and suffix flags. A rule flag indicates a property of the word (more precisely, whether the word obeys the property indicated by the flag or not). The flag that shows the obeyance to the vowel harmony rule is a rule flag. A suffix flag indicates whether a suffix can be affixed to the word (if the flag is set) or not (if the flag is not set). The flag for the suffix *-lAş* is of this type.

The words in different categories are subject to different rules and suffixes. For example, the aorist suffix rule is applicable to verbs only. Therefore the categories own different flag sets. We have four flag sets:

1. Flags for categories other than verbs and adverbs
2. Flags for verbs

3. Flags for adverbs

4. Flags for proper nouns

Note that the first three correspond to the root lexicon while the last one corresponds to the proper noun lexicon. The first three groups are mutually exclusive with respect to the flags used. That is, a verb in its root form cannot be treated as a noun or adjective; a suffix must be affixed in order to convert it to a noun. A noun can function as a verb (see section 4.2.3), but at the portion of the verb network where the noun (in its root form) jumps to, none of the flags used for verbs are applicable.

We considered the categories like postposition and interjection inside the first group (which is basically the group of nouns and adjectives). Although there are no special rules and suffixes for the words in these categories, it is possible that a word in one of these categories jumps to the noun network without receiving any suffix. After it jumps to the noun network, it is treated as a noun and the rules and the suffixes in that part of the noun network are applicable to this word also. Hence it is necessary to know the values of the flags corresponding to these rules and suffixes for this word.

There are a total of 56 flags that should be stored in the lexicons. But, as mentioned above, the flag sets are mutually exclusive. Thus to save storage space, instead of storing all the flags for each word, only the relevant flags are stored. For instance, the entry in the lexicon for a noun is formed of the root word, its category, and a set of 38 flags. Since the flag set of adverbs is formed of only two flags and the number of adverbs in the language is small, we decided to store the flags of verbs and adverbs together in order to improve the processing time at the expense of increasing the storage space a little. So, the entry in the lexicon of a verb is formed of the root word, its category, and a set of 18 flags, where the last two of these flags are not used. The situation for an adverb is similar, except that the first 16 flags remain unused.

As mentioned in section 8.4, each suffix in the suffix lexicon has a field called the affixation flag index. This field holds the index of the flag for the suffix in the flag array. During parsing of a word, in order to decide whether a suffix can be affixed to a word or not, the affixation flag index is found from the suffix lexicon and the value of the corresponding flag for this word in the root (or proper noun) lexicon is extracted. The suffix can be affixed provided that the value of the flag is 1. For suffixes that can be affixed to all the words in the related category, the affixation flag index is zero, which means that there is no flag reserved for this suffix.

This type of semantic information is used by some researchers. The flag concept presented here is also used by Solak and Oflazer [35]. On the other hand, Köksal does not make a distinction between the suffixes that may and may not be affixed to the words [26]. It is only the grammatical category of the word which determines the suffixes that may follow. Thus, both of the forms given above *çirkinleş* (to become ugly) and *solgunlaş* are accepted by his parser, since the roots *çirkin* (ugly) and *solgun* (pale) belong to the same category.

8.6. The Parser

The parser performs the following operations on each word:

- Consulting hash table
- Syllabification check
- Root search
- Suffix search

8.6.1. Consulting Hash Table

The parser first checks whether the input word has already been examined by consulting a hash table. The hash table is stored as an array of 1193 (a prime number) buckets with external chaining to handle overflows [17]. If the word has already been checked for spelling, then the result stored in the hash table is used; the following routines are not called, i.e. the parsing of this word is completed.

8.6.2. Syllabification Check

If the word is encountered for the first time then the parser checks its syllable structure. Syllabification check is used as a preprocessing heuristic in the spelling checker. The heuristic is as follows: if a word does not have the proper syllable structure of Turkish, then it is misspelled. The syllabification checking algorithm used in the program was borrowed from [68]. This is a regular expression and a corresponding finite state automaton for validating if a word matches the syllable structure rules of Turkish. The word whose spelling is to be checked is first processed with the regular

expression. It is reported as misspelled if its syllable structure can not be matched with this expression, i.e. the phonemes of the word do not form valid sequences according to Turkish syllable structure. On the other hand, if it can be matched, its morphological structure is analyzed as it may still be a non-Turkish or a misspelled word.

8.6.3. Root Search

If the word passes the syllabification check, then the parser begins spelling checking. First it tries to match an initial substring of its input to one of the roots in the root lexicon. Using another hashing function which considers the first two letters of the word to be checked, all the words starting with those two letters are examined as possible candidate roots (see section 8.2). After the root search, the roots that do not match the characters of the input word other than the first two are eliminated and a list of candidate roots is formed.

There are some variants of the root search algorithm in the literature. Kibaroğlu [9] parses the word from left to right and accepts the first substring of the word that matches an entry in the lexicon as a possible root. On the other hand, Köksal [26] and Solak [35] try to find the root with the maximum length. They parse the word from right to'left during root search. All of these algorithms include a backtracking component, i.e. if the root found does not lead to a successful parse, they attempt to find another root and continue the analysis.

8.6.4. Suffix Search

The category of each candidate root determines a class of suffixes that are permitted in the next position. The parser searches the suffix lexicon for a suffix that is in the permitted class and matches the surface string at the current point. If one is found, a pointer is advanced to the new current point in the word and the parser jumps to the new state, corresponding to the derived stem category, determined by the suffix. This is a recursive process. If the end of the input string is reached and the parser is in a designated final state, then the string is said to be successfully parsed and the input word is a well-formed Turkish word.

An issue that can be taken into account during suffix search is the iteration of the suffixes. The design of the morphotactics prevents the consecutive usage of the same suffix twice where it does not result to a meaningful word. But we do not have a mechanism for the control of the iterative usage of a sequence of suffixes. Consider the following words:

göz (eye)
gözlük (glasses)
gözlükçü (optician)
gözlükçülük (the occupation of opticians)
gözlükçülükçü
gözlükçülükçülük

Though it is arguable [8], we think that the last two words above are not meaningful. However, they are considered as well-formed words by our parser. The situation is the same for most of the parsers on Turkish morphology [26,35]. Some semantic control mechanisms should be included within the parser to avoid accepting such words.

After the processes described above, the spelling checking of the word is completed. The result of the spelling checking is inserted into the hash table together with the word. If the word is misspelled then the spelling corrector routine is called and a list of alternative words is displayed (see section 8.7). In some cases, a simple look-in of the root lexicon is not enough to find the root word. Due to some morphophonemic rules, the root part of the input word may be deformed, and the parser considers such cases by inserting and deleting letters into the input during root search.

The morphophonemic alternation is accounted for by a phonological component that mediates between the lexical and surface forms of morphemes. This process is a part of the matching process briefly expressed above.

The morphophonology is encoded in the functions that determine whether a given surface string matches a root or suffix entry. There are two different procedures, one for roots and one for suffixes. What these functions do is to modify the basic form of the morpheme to make it compatible with its surface environment using the rules.

8.7. Spelling Corrector

In producing a document using a word processor two basic kinds of spelling errors occur:

- Typing errors
- Errors made due to lack of grammatical knowledge

In this study, only the typing errors are considered. This is due to the fact that most of the spelling errors that can be made while writing a text in Turkish language are typing errors, such as omitting a letter, interchanging two letters, writing an extra letter, or writing an incorrect letter. The following typing errors are considered:

- Transposition of two letters
- One extra letter
- One letter missing
- One letter wrong

It is also assumed that in each misspelled word there is only one type of error.

Spelling correction is performed by first detecting a misspelled word, then checking for the above mentioned typing errors and forming a list of candidate words based on each error, spelling checking each candidate word to determine whether it is a well-formed word, and finally listing the well-formed words as suggestion. In this respect spelling correction is a very costly operation. If the misspelled word contains w characters, then from considering transposition **w-1**, from considering one extra letter **w**, from considering one letter missing **(w+1)*32** (we use an alphabet formed of 32 characters), and from considering one letter wrong **w*31**, that is a total of **65*w+31** alternative words have to be formed and spell checked.

To reduce the time complexity of spelling correction, we use the syllabification check heuristic explained in section 8.6.2. The syllable structure of each candidate word is checked before displaying to the user as a suggestion. The word is displayed only if it passes the check. Although the syllabification check reduces the time considerably, the time complexity of spelling correction remains in general an order of magnitude greater than spelling checking. To further speed up the process, new heuristics can be designed based on the statistical results about the language (see section 6.2).

8.8. Performance Results

The spelling checker and corrector were tested on VAX environment using two books and performance results were obtained [17]. Tables 8.1 and 8.2 show the results for the spelling checker and the spelling corrector, respectively.

Table 8.1 indicates that the average spelling checking rate is 1,750 - 2,200 words/min. when there are errors in the text, and 3,300 words/min. when there are no spelling errors. The analysis of the list of misspelled words has shown that approximately 60 per cent of them are proper nouns and the remaining consists of technical terms not included in the root lexicon and really misspelled words. The performance therefore can be improved by using additional specialized lexicons for different domains.

Table 8.2 indicates that spelling corrector is a costly operation. For each misspelled word, it takes 7-8 seconds to generate the candidate correct words.

TABLE 8.1. Performance results for the spelling checker

Number of words	Number of misspelled words	Total CPU time (sec)	Total CPU time for correct words (sec)
65,155	5,204	1,758.4	1,062.7
40,166	5,468	1,378.4	627.4

TABLE 8.2. Performance results for the spelling corrector

Number of misspelled words	Average number of alternatives per word	Average CPU time per word (sec)
5,204	6.5	7.73
5,468	7.1	8.14

IX. CONCLUSION

As the conclusion of this thesis, we would like to mention the work done and state some recommendations for future work on this field.

9.1. Work Accomplished

In this thesis we have analyzed the morphological structure of Turkish and formed its computer representation. Our main objective was to formalise the full Turkish morphology. To achieve this goal, we first investigated the morphology from a linguistic point of view. We collected all the word formation rules, and defined the rules and the irregularities among them. Our main concern during this process was to form a complete and uniform definition.

This definition of the morphology has been represented in the form of an ATN. In this representation, the morphotactics, that is the order of the suffixes, are defined as transitions on the network. The morphophonemic rules accompany the network as rules that are activated when transitions between the states occur. Combination of the state transitions in the network and the rules forms the structure of a morphological parser for the Turkish language.

An important part of the research was the investigation of the formalisms FSTN, RTN, and ATN as the possible representation schema for the morphology. This analysis showed us that it was not feasible to use an FSTN or RTN for this purpose. It is very hard, if not impossible, for some of the Turkish morphophonemic rules to be expressed on an FSTN. The rules that necessitate a modification in the root form of the word constitute this group. Such modifications are outside the power of FSTNs.

The morphological structure proposed in this thesis forms a basis for the language applications about Turkish. There are several areas that make use of morphological information. Some of these are: parsing [3,36,69], text generation [70], machine translation [40,71], dictionary tools, text-to-speech systems, speech recognition, spelling checker, document retrieval. Among these applications, we designed and implemented a morphological parser and a spelling checker. The spelling checker also

incorporates a spelling corrector component. As far as we know, this is the first spelling corrector implemented for Turkish.

Lexical and morphological investigation of Turkish from a statistical point of view was another part of this thesis. This investigation was also based on our morphological structure. The corpus analysis is an important process that reveals many useful results about the language. An important issue here is the formation of a corpus which serves as a data bank for such an analysis. It should be large enough and comprise text on different topics.

In summary, the work done in this thesis consists of the following:

- An Augmented Transition Network (ATN) formalism was introduced for Turkish morphology, containing all of the categories and the suffixes.
- A root lexicon of about 21,500 words and a proper noun lexicon of about 11,500 words were formed in parallel to the ATN formalism.
- A parser and a spelling checker (including a spelling corrector) were implemented for Turkish to test the completeness (coverage) and the efficiency of the formalism.
- A test environment comprising of these elements was produced to study and test morphological properties of Turkish.
- The lexicon was analyzed to obtain statistics on the structural and usage patterns of the Turkish morphology.
- A corpus of about 2,200,000 words, which is currently the largest corpus on Turkish, was formed.
- This corpus was analyzed to obtain statistical properties of Turkish.

In parallel to this research, the following papers were produced:

- *Representation of Turkish Morphology in ATN*, Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks, İstanbul, 1993 [16]
- *A Spelling Checker and Corrector for Turkish*, Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks, İstanbul, 1993 [17]
- *Full Turkish Morphology Represented as an Augmented Transition Network*, submitted to Literary and Linguistic Computing [18]
- *Lexical and Morphological Statistics About Turkish Language*, in preparation

9.2. Recommendations for Future Research

One of the challenging areas for future research is the semantic categorization of the lexicon. Currently we use syntactic categories such as noun, adjective, verb, etc. In addition to this, the words can be grouped taking into account their meanings. This necessitates the division of words into subcategories. For example, some subcategories for the nouns may be: abstract nouns, nouns that define professions, nouns that refer to animals, and so on. The advantage of this process is that it allows a simpler and more modular morphological structure to be built. Because, in general, it is the semantics of the words that denote whether a suffix can be affixed to a word or not. If semantically close entities are grouped together, then it becomes easier to define the morphotactics. In such a case, a suffix that can be received by a word in a group will be received by all other words in that group. Currently, we use flags to decide whether a suffix can be affixed to a word or not. In the case of semantic categorization, the use of the flags will reduce to a minimum or will not be used at all. However, this is a hard topic and necessitates a great effort and too much time. All of the words in the lexicon must be examined carefully with the help of linguistically oriented people.

Another possible feature extension of this research may be in the use of acronyms. As discussed in section 5.2, the rule that we use for acronyms (rule 20) does not always give the true derivation. In order to correct the rule, some form of semantic information is needed, which indicates the vowel and consonant categories in the last morpheme in the reading form of the acronym. This is an easy matter and can be obtained by the an analysis of the acronyms. For the spelling checker program, this information can be added to the lexicon.

In this study, we have started the statistical investigation of Turkish. We formed a quite large corpus and obtained some statistical figures about the language. The development of this analysis is an attractive research area. The corpus can be enlarged with respect to both the number of words it contains and the topics it comprises. In addition to the results about the language as a whole, this also allows us to obtain results on individual topics. For example, we can get information about the usage of the language in a particular domain and compare it with its general usage.

As a final remark, natural language processing is a broad concept and morphological analysis is only one part of it. It must be followed by syntactic and semantic analysis in order to be able to fully computerize the process.

APPENDIX A. MORPHOTACTICS OF TURKISH LANGUAGE IN THE FORM OF STATE TRANSITION TABLE

In this appendix, we list the morphotactics of Turkish language. The transitions are grouped into different networks (N, ND, etc.). *Name of network* stands for the name of the network, *originating state* and *destination state* are the beginning and ending states of the transition, and *suffixes for transition* shows the suffixes used in this transition. If the destination state is *, it indicates that the parsing ends after this transition occurs. The symbol *(#)* on the right of a suffix indicates that the suffix is a particle and is written separate from the word it follows. Table A.1 lists the full set of morphotactic rules for Turkish language.

TABLE A.1. Morphotactics in state transition table form

Name of Network	Originating state	Destination state	Suffixes for transition
A	A	A	şIn
		A1	~, bIz, CA, (I)msAr, (I)msI, (I)mtıraĞ
		D1	CA, DAn, en, sInA, sIz
		N	(A)z
		V8	(A)l
	A1	A/end	~
		A2	lIĞ
		N	~
	A2	D1	(I)nA
		N	~
B	B	B/end	~
C	C	C/end	~
D	D	C	(y)sA
		D/end	~, CAcIĞ, CAnA, DAn, lIklA, mAsInA
		D1	~
	D1	VF1	~
E	E	E/end	~
		NC2	~
		VF	~
I	I	I/end	~
		NC	~
		V5	lA
K	K	K1	~
		K2	~
		K3	~
	K1	K/end	~, (parse NC)
	K2	K/end	~, (parse NC)
	K3	K/end	~, (parse NC)
L	L	L/end	~
N	N	N1	(parse ND)
	N1	N/end	(parse NC)
NC	NC	NC1	~
		NC2	~
	NC1	NC2	lAr
	NC2	A1	CA
		D1	CA
		NC3	(I)m, (I)mIz, (I)n, (I)nIz
		NC4	~
		NC5	~, CA
	NC3	*	sIz
		D1	CA
		NC4	~
		ND6	CA

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
NC	NC4	NC7	~
		NC10	~
	NC5	NC6	(s)I
	NC6	NC7	n
		NC9	~
		NC10	~
		VF1	~
	NC7	NC9	(parse NE)
	NC9	NC/end	~, dA (#), ki (#)
	NC10	D1	(y)IlAn, (y)InAn, (y)lA
ND	ND	A	~, (A)l, kar, lIĞ, perver, rAĞ, sever, sIl, (v)i, (y)ane
		A1	CI, CIl, (i)Ğ, sAl, sI
		D1	leyin, lI, sIzIn, (y)ane, (y)In
		NC2	giller
		ND	(A)C, (A)ğI, (a)j, (A)Ğ, (A)lAĞ, (A)t, başı, baz, cAğIz, CAĞ, dAĞ, dAlIĞ, DAm, DAn, Dar, DAş, dIrIĞ, dIz, GA, GAn, hane, (I)l, (I)t, (i)st, (i)zm, keĞ, keş, lA, lAĞ, lAm, mAn, mAr, name, sAĞ, tay, zade
		ND1	lIĞ
		ND3	CIĞ
		ND4	CI
		ND5	~
		ND6	lIĞ
		V4	(A)r, (A)ş, DAş, DAt, eD, (I)Ğ
		V5	A, I, (I)msA, sA, sI
		V7	ol
		V9	(A)n, DAn
	ND1	ND2	CI
		ND5	~, CIĞ
	ND2	ND5	~, lIĞ
	ND3	ND4	~, CI
	ND4	ND5	~, lIĞ
	ND5	ND	ölçer
		ND6	~
		V4	lAş, lAt
		V5	lA
		V9	lAn
	ND6	A	lI, sIz
		ND/end	~

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
NE	NE	NE/end	(y)I
		NE1	(y)A
		NE2	~, DAn
		NE3	~, (n)In
		NE5	DA
	NE1	NE/end	~
		VF6	~
	NE2	NE/end	~
		VF1	~
	NE3	NE2	~
		NE4	ki
	NE4	NC1	~
		NC6	~
		NE2	~
	NE5	NE2	~
		NE4	ki
		VF	~
P	P	NC	~
		P	cIlAyIn
		P/end	~
R	R	NC2	giller, lAr
		R1	~, CI, CIĞ, (i)st, (i)zm, lI, sIz
		R2	cAğIz
		V4	lAş
		V5	lA
	R1	NC2	~, lIĞ
		R/end	~
	R2	NC2	~, lAr
S	S	S1	yüz
		S3	~
	S1	S2	~, altmış, doksan, elli, kırk, on, otuz, seksen, yetmiş, yirmi
	S2	N	dAlIĞ
		S3	~, altı, beş, bir, dokuz, dörD, iki, sekiz, üç, yedi
	S3	S4	~
		S20	bin
	S4	S5	trilyon
		S10	milyar
		S15	milyon
		S24	~
	S5	S6	~, altı, beş, dokuz, dörD, iki, sekiz, üç, yedi
		S7	~
	S6	S7	yüz
		S9	~

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
S	S7	S8	~, altmış, doksan, elli, kırk, on, otuz, seksen, yetmiş, yirmi
	S8	S9	~, altı, beş, bir, dokuz, dörD, iki, sekiz, üç, yedi
	S9	S10	milyar
		S15	milyon
		S20	bin
		S24	~
	S10	S11	~, altı, beş, dokuz, dörD, iki, sekiz, üç, yedi
		S12	~
	S11	S12	yüz
		S14	~
	S12	S13	~, altmış, doksan, elli, kırk, on, otuz, seksen, yetmiş, yirmi
	S13	S14	~, altı, beş, bir, dokuz, dörD, iki, sekiz, üç, yedi
	S14	S15	milyon
		S20	bin
		S24	~
	S15	S16	~, altı, beş, dokuz, dörD, iki, sekiz, üç, yedi
		S17	~
	S16	S17	yüz
		S19	~
	S17	S18	~, altmış, doksan, elli, kırk, on, otuz, seksen, yetmiş, yirmi
	S18	S19	~, altı, beş, bir, dokuz, dörD, iki, sekiz, üç, yedi
	S19	S20	bin
		S24	~
	S20	S21	~, altı, beş, dokuz, dörD, iki, sekiz, üç, yedi
		S22	~
	S21	S22	yüz
		S24	~
	S22	S23	~, altmış, doksan, elli, kırk, on, otuz, seksen, yetmiş, yirmi
	S23	S24	~, altı, beş, bir, dokuz, dörD, iki, sekiz, üç, yedi
	S24	A1	~, gen, (I)ncI, (I)z, (ş)Ar
		S/end	~
V	V	V1	~
		V2	~
		V3	~
		V4	~
		V5	~
		V6	~
		V7	~
		V8	~
		V9	~
	V1	V3	Ar
		V19	(parse VA3)
	V2	V3	Ir

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
V	V2	V19	(parse VA3)
	V3	V10	~
		V19	(parse VA2)
	V4	V11	~
		V19	(parse VA)
	V5	V10	~
		V11	~
	V6	V11	~
		V12	(I)t
		V19	(parse VA3)
	V7	V13	(I)n, (I)§
		V14	DIr
		V19	(parse VA3)
	V8	V10	~
		V15	(I)n, (I)§
	V9	V16	~, (I)§
	V10	V17	t
		V19	(parse VA3)
	V11	V18	(I)n, (I)§
	V12	V19	(parse VA)
	V13	V19	(parse VA)
	V14	V19	(parse VA1)
	V15	V19	(parse VA)
	V16	V19	(parse VA)
	V17	V19	(parse VA)
	V18	V19	(parse VA)
	V19	V20	(parse VB)
	V20	V/end	dA (#), ki (#)
VA	VA	D1	C, (y)AlI
		VA1	DIr
		VA2	~
	VA1	VA2	~, t
	VA2	VA3	~
		VA4	~
	VA3	A	(A)cAn, ,(A)ğAn, GIn, (I)msAr, pAĞ, sAĞ, sAl, vAn, (y)AsIcA
		A1	(y)IcI
		N	(A)C, (A)Ğ, (A)lgA, (A)m, (A)mAĞ, (A)mIĞ, (A)nAĞ, (A)v, (A)y, CA, CAĞ, CAmA, GA, GAC, GAn, GI, GIC, GIt, I, (I)cIĞ, (I)n, (I)t, mAn, mAz, mIĞ, mur, nAĞ, sI, tI
		V5	~

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
VA	VA3	VA5	~
	VA4	VA5	~, (I)l
	VA5	A	(I)Ğ
		D1	(I)n, (y)An
		N	(I)m, (I)ntI
		VA/end	~
VB	VB	VB1	~, mA
		VB2	~
	VB1	VB2	(y)Adur, (y)Agel, (y)Agör, (y)Akal, (y)Akoy, (y)Ayaz, (y)Iver
		VB12	(y)Iş
	VB2	VB3	~, (y)A
		VB4	(y)Abil
		VB6	~
	VB3	N	mAC
		VB5	mA
		VB7	mAz
		VB11	~, mI
	VB4	VB3	(y)A
		VB11	~
	VB5	D1	DAn
		V20	m, (y)Iz
		VB6	~
	VB6	D1	(y)A, (y)AlI, (y)AsIyA
		VB8	mAz
		VB9	~, (y)Abil
	VB7	VB8	~
		VF	~
	VB8	NC	lIĞ
	VB9	VB10	(A)r
		VC	~
		VD	DI
		VE	mIş
		VF	mAktA, mAlI, mIş, (y)AcAĞ
		VG	sA
		VG4	(y)A
	VB10	D1	CAsInA
		N	~
		VF	~
	VB11	VF	(I)yor
	VB12	D1	DAn
		N	~, GAn
VC	VC	*	sAnA, sAnIzA

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
VC	VC	A	mIş
		D1	(y)ArAĞ, (y)IncA, (y)Ip
		N	(A)r
		ND6	(y)AcAĞ, (y)An, (y)AsI
		V20	~, (y)In, (y)InIz
		VC1	(y)A
		VC3	sIn, sInIAr
		VC5	DIĞ
		VC6	mA
		VC7	mAĞ
	VC1	*	lAr
		V20	sIn, sInIz
		VC2	lIm, (y)Im
	VC2	*	mI (#)
		V20	~
	VC3	V20	~
		VC4	~, mI (#)
	VC4	*	(y)DI
	VC5	D1	(I)nA
		NC	~
	VC6	D1	~
		N	~, CA
	VC7	D1	sIzIn
		E	lA
		N	~
		VC8	DAn
	VC8	D1	~, sA
VD	VD	V20	lArsA
		VD1	~, mI (#)
		VD3	~, lAr, m, n, nIz
		VD5	k
		VD6	(y)sA
	VD1	VD2	(y)DI
	VD2	V20	~, k, lAr, m, n, nIz
	VD3	V20	~
		VD4	mI (#)
	VD4	V20	~, (y)DI
	VD5	D1	CA
		VD3	~
	VD6	V20	~, k, lAr, m, n, nIz
VE	VE	D1	CAsInA
		VE1	~, mI (#)
		VE4	mI (#)

TABLE A.1. Morphotactics in state transition table form (continued)

Name of network	Originating state	Destination state	Suffixes for transition
VE	VE	VE6	lAr
	VE1	VE2	~, sIn, sInIz, (y)Im, (y)Iz
		VE3	(y)DI
	VE2	V20	~, DIr
	VE3	V20	~, k, m, n, nIz
	VE4	VE5	(y)DI
	VE5	*	lAr
	VE6	VE7	~, mI (#)
	VE7	V20	~, DIr, (y)DI
VF	VF	VF1	~
		VF6	lAr
	VF1	VF2	~, mI (#)
		VF6	~
		VF10	mI (#)
		VF12	(y)sA
	VF2	VF3	(y)DI
		VF4	(y)mIş
		VF5	~, sIn, sInIz, (y)Im, (y)Iz
	VF3	V20	~, k, m, n, nIz
	VF4	V20	~, (I)m, (I)z, lAr, sIn, sInIz
	VF5	V20	~, DIr
	VF6	V20	(y)sA
		VF7	~, mI (#)
		VF8	(y)ken
	VF7	V20	~, DIr, (y)DI, (y)mIş
	VF8	VF9	~, mI (#)
	VF9	V20	~, DIr, (y)DI, (y)mIş
	VF10	VF11	(y)DI
	VF11	V20	~, lAr
	VF12	V20	k, lAr, m, n, nIz
VG	VG	VG1	lAr
		VG2	~, k, m, n, nIz
		VG4	~, mI (#)
	VG1	VG2	~
		VG3	~
	VG2	V20	~
		VG3	mI (#)
	VG3	V20	(y)DI, (y)mIş
	VG4	VG5	(y)DI
		VG6	(y)mIş
	VG5	V20	~, k, lAr, m, n, nIz
	VG6	V20	~, (I)m, (I)z, lAr, sIn, sInIz
W	W	W/end	~

APPENDIX B. MORPHOPHONEMIC RULES

In this appendix, we list the morphophonemic rules in a pseudo language utilizing the variables defined in section 5.4. The functions are defined in Table B.1 (in the descriptions, *str1* and *str2* denote character strings; *ch1* and *ch2* denote letters; *n1* denotes a number).

Figures B.1 through B.13 show the morphophonemic rules.

TABLE B.1. List of functions

a) General purpose functions:

consonant (ch1)	: returns true if ch1 is a consonant, returns false otherwise.
delete (str1, str2)	: deletes the string str2 from the string str1.
concat (str1, str2)	: inserts the string str2 to the end of the string str1.
insert (str1, ch1, n1)	: inserts the character ch1 as the $n1^{th}$ last character in the string str1.
interchange (str1, ch1, ch2)	: interchanges the characters ch1 and ch2 in the string str1.
last_consonant (str1)	: returns the last consonant in the string str1.
last_phoneme (str1)	: returns the last phoneme in the string str1.
last_phoneme_p (str1, n1)	: returns the $n1^{th}$ last phoneme in the string str1, where n1 is greater than 1.
last_vowel (str1)	: returns the last vowel in the string str1.
no_suffix	: returns true if there are no more suffixes affixed to the word, returns false otherwise.
vowel (ch1)	: returns true if ch1 is a vowel, returns false otherwise.
vowel_exists (str1)	: returns true if at least one vowel exists in the string str1, returns false otherwise.

b) Morphophonemic character check functions:

can_change (ch1)	: returns true if the character ch1 can change; i.e. if ch1 is written in capitalized form in the network. It returns false otherwise.
can_delete (ch1)	: returns true if the character ch1 can drop; i.e. if ch1 is written in the form (ch1) in the network. It returns false otherwise.

c) Morphophonemic rule check function:

obey (rule, str1)	: returns true if the string str1 obeys the rule; returns false otherwise. For the rule names, we shall use an abbreviation which is formed from the first letter of each word in the name of the rule.

```
Rule 1 : Vowel harmony rule
    if can_change (fvs) then
      if obey (vhr, word) then
        case fvs of
          'a' : case last_vowel (word) of
                  {a,ı,o,u} : fvs := 'a'
                  {e,i,ö,ü} : fvs := 'e'
                endcase
          'ı' : case last_vowel (word) of
                  {a,ı} : fvs := 'ı'
                  {e,i} : fvs := 'i'
                  {o,u} : fvs := 'u'
                  {ö,ü} : fvs := 'ü'
                endcase
        endcase
      else
        case fvs of
          'a' : case last_vowel (word) of
                  {a,ı,o,u} : fvs := 'e'
                  {e,i,ö,ü} : fvs := 'a'
                endcase
          'ı' : case last_vowel (word) of
                  {a,ı} : fvs := 'i'
                  {e,i} : fvs := 'ı'
                  {o,u} : fvs := 'ü'
                  {ö,ü} : fvs := 'u'
                endcase
        endcase
      endif
    endif
```

Figure B.1. Vowel harmony rule

a) Rule 2 : Consonant harmony rule 1

```
if obey (chr1, word) then
  if no_suffix or consonant (fps) then
    case last_phoneme (word) of
      'b'     : last_phoneme (word) := 'p'
      'c'     : last_phoneme (word) := 'ç'
      'd'     : last_phoneme (word) := 't'
      'g','ğ' : last_phoneme (word) := 'k'
    endcase
  endif
endif
```

b) Rule 3 : Consonant harmony rule 2

```
if obey (chr2, word) then
  if no_suffix or consonant (fps) then
    last_phoneme (word) := 'g'
  endif
endif
```

c) Rule 4 : Consonant harmony rule 3

```
if can_change (fps) then
  if last_phoneme (word) is in {ç,f,h,k,p,s,ş,t} then
    case fps of
      'b' : fps := 'p'
      'c' : fps := 'ç'
      'd' : fps := 't'
      'g' : fps := 'k'
    endcase
  endif
endif
```

Figure B.2. Consonant harmony rules

a) Rule 5 : Vowel insertion rule 1

```
if obey (vir1, word) then
   if no_suffix or consonant (fps) then
      insert (word, I, 2)
   endif
endif
```

b) Rule 6 : Vowel insertion rule 2

```
if obey (vir2, word) then
   if no_suffix or suffix is not in {-I,-(I)Ğ,-(I)l,-(I)m,-(I)ntI,-(I)t} then
      insert (word, I, 2)
   endif
endif
```

Figure B.3. Vowel insertion rules

Rule 7 : Double consonant rule

```
if obey (dcr, word) then
   if suffix is in {-eD,-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(n)In,-ol,-(s)I,-(y)A,-(y)I,-(y)IlAn,-(y)InAn} then
      concat (word, last_phoneme (word))
   endif
endif
```

Figure B.4. Double consonant rule

a) Rule 8 : Phoneme deletion rule

```
if can_delete (fps) then
   if (vowel (fps) and vowel (last_phoneme (word))) or
      (consonant (fps) and consonant (last_phoneme (word))) then
      delete (suffix, fps)
   endif
endif
```

b) Rule 9 : Phoneme deletion rule for the suffix *-(I)yor*

```
if suffix = -(I)yor then
   if vowel (last_phoneme (word)) then
      delete (word, last_phoneme (word))
   endif
endif
```

c) Rule 10 : Phoneme deletion rule for verbs

```
if obey (pdrfv, word) then
   if fps = 'y' then
      last_phoneme (word) := I
   endif
endif
```

Figure B.5. Phoneme deletion rules

a) Rule 11 : Possessive suffix rule 1

```
if obey (psr1, word) then
   if suffix = -(s)I then
      delete (suffix, fps)
   endif
endif
```

b) Rule 12 : Possessive suffix rule 2

```
if obey (psr2, word) then
   if suffix = -(s)I then
      {do nothing and continue parsing}
      if parse fails then
         delete (suffix, fps)
      endif
   endif
endif
```

Figure B.6. Possessive suffix rules

a) Rule 13 : Rule for compound words 1

```
if obey (rfpw1, word) then
  if suffix = -lAr then
    delete (word, ltpw)
    concat (word, -lAr)
    concat (word, -I)
  else
    if suffix = -(s)I then
      delete (suffix, -(s)I)
    else
      if suffix is in {-(I)m,-(I)mIz,-(I)n,-(I)nIz} then
        delete (word, ltpw)
      else
        if suffix is in {-CA,-DA,-DAn,-(y)A,-(y)I} then
          concat (word, 'n')
        else
          if suffix <> -(y)lA then
            delete (word, ltpw)
          endif
        endif
      endif
    endif
  endif
endif
```

Figure B.7. Rules for compound words

b) Rule 14 : Rule for compound words 2

```
if obey (rfpw2, word) then
  if suffix = -lAr then
    delete (word, last_phoneme (word))
    case last_consonant (word) of
      'b'    : last_consonant (word) := 'p'
      'c'    : last_consonant (word) := 'ç'
      'd'    : last_consonant (word) := 't'
      'g','ğ': last_consonant (word) := 'k'
    endcase
    concat (word, -lAr)
    concat (word, -I)
  else
    if suffix = -(s)I then
      delete (suffix, -(s)I)
    else
      if suffix is in {-CA,-DA,-DAn,-(y)A,-(y)I} then
        concat (word, 'n')
      else
        if suffix is not in {-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(y)lA} then
          delete (word, last_phoneme (word))
          case last_consonant (word) of
            'b'    : last_consonant (word) := 'p'
            'c'    : last_consonant (word) := 'ç'
            'd'    : last_consonant (word) := 't'
            'g','ğ': last_consonant (word) := 'k'
          endcase
        endif
      endif
    endif
  endif
endif
```

Figure B.7. Rules for compound words (continued)

c) Rule 15 : Rule for compound words 3

```
if obey (rfpw3, word) then
  if suffix = -lAr then
    interchange (word, ltpw))
    concat (word, -lAr)
    concat (word, -I)
  else
    if suffix = -(s)I then
      delete (suffix, -(s)I)
    else
      if suffix is in {-CA,-DA,-DAn,-(y)A,-(y)I} then
        concat (word, 'n')
      else
        if suffix is not in {-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(y)lA} then
          interchange (word, ltpw)
        endif
      endif
    endif
  endif
```

d) Rule 16 : Rule for compound words 4

```
if obey (rfpw4, word) then
  if suffix = -lAr then
    delete (word, last_vowel (word))
    concat (word, -lAr)
    concat (word, -I)
  else
    if suffix = -(s)I then
      delete (suffix, -(s)I)
    else
      if suffix is in {-CA,-DA,-DAn,-(y)A,-(y)I} then
        concat (word, 'n')
      else
        if suffix is not in {-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(y)lA} then
          delete (word, last_vowel (word))
        endif
      endif
    endif
  endif
endif
```

Figure B.7. Rules for compound words (continued)

Rule 17 : Aorist suffix rule

```
if obey (asr, word) then
  suffix := -(I)r
endif
```

Figure B.8. Aorist suffix rule

Rule 18 : Rule for the morpheme *su*

```
if obey (rftms, word) then
  if suffix is in {-(I)m,-(I)mIz,-(I)n,-(I)nIz,-(n)In,-(s)I} then
    concat (word, 'y')
  endif
endif
```

Figure B.9. Rule for the morpheme *su*

Rule 19 : Rule for proper nouns

```
if suffix = -lAr then
  {do nothing and continue parsing}
  if parse fails then
    concat (word, "')
  endif
else
  if suffix is inflectional then
    concat (word, "')
  endif
endif
```

Figure B.10. Rule for proper nouns

a) Rule 20 : Rule for acronyms 1

```
if consonant (last_phoneme (word)) then
  if vowel_exists (word) then
    if front_vowel (last_vowel (word)) then
      {assume that last_vowel (word) is a front vowel and
                last_phoneme (word) is a vowel or a consonant}
    else
      {assume that
       (last_vowel (word) is a front vowel and last_phoneme (word) is a vowel) or
       (last_vowel (word) is a back vowel and last_phoneme (word) is a consonant)}
    endif
  else
    {assume that last_vowel (word) is a front vowel and
              last_phoneme (word) is a vowel}
  endif
else
  if front_vowel (last_vowel (word)) then
    {assume that last_vowel (word) is a front vowel and
              last_phoneme (word) is a vowel}
  else
    {assume that last_vowel (word) is a back vowel and
              last_phoneme (word) is a vowel}
  endif
endif
```

b) Rule 21 : Rule for acronyms 2

```
if last_phoneme (word) <> '.' then
  concat (word, "")
endif
```

Figure B.11. Rules for acronyms

Rule 22 : Rule for numbers

```
if last_phoneme (word) is in {1,..,9} then
  case last_phoneme (word) of
    '1' : w_form := 'bir'
    '2' : w_form := 'iki'
    '3' : w_form := 'üç'
    '4' : w_form := 'dörd'
    '5' : w_form := 'beş'
    '6' : w_form := 'altı'
    '7' : w_form := 'yedi'
    '8' : w_form := 'sekiz'
    '9' : w_form := 'dokuz'
  endcase
else
  if last_phoneme_p (word, 2) is in {1,..,9} then
    case last_phoneme_p (word, 2) of
      '1' : w_form := 'on'
      '2' : w_form := 'yirmi'
      '3' : w_form := 'otuz'
      '4' : w_form := 'kırk'
      '5' : w_form := 'elli'
      '6' : w_form := 'altmış'
      '7' : w_form := 'yetmiş'
      '8' : w_form := 'seksen'
      '9' : w_form := 'doksan'
    endcase
  else
    if last_phoneme_p (word, 3) is in {1,..,9} then
      w_form := 'yüz'
    else
      if (last_phoneme_p (word, 4) is in {1,..,9}) or
        (last_phoneme_p (word, 5) is in {1,..,9}) or
        (last_phoneme_p (word, 6) is in {1,..,9}) then
        w_form := 'bin'
      else
        if (last_phoneme_p (word, 7) is in {1,..,9}) or
          (last_phoneme_p (word, 8) is in {1,..,9}) or
          (last_phoneme_p (word, 9) is in {1,..,9}) then
          w_form := 'milyon'
        else
          if (last_phoneme_p (word, 10) is in {1,..,9}) or
            (last_phoneme_p (word, 11) is in {1,..,9}) or
            (last_phoneme_p (word, 12) is in {1,..,9}) then
            w_form := 'milyar'
          else
            w_form := 'yon'
          endif
```

Figure B.12. Rule for numbers

```
            endif
          endif
        endif
      endif
    endif
    {assume that word is written in w_form}
```

Figure B.12. Rule for numbers (continued)

Rule 23 : Rule for particles

```
if suffix is in the set of particles {-dA,-ki,-mI} then
  concat (word, ' ')
endif
```

Figure B.13. Rule for particles

APPENDIX C. REST OF STATISTICS

In chapter 6 and chapter 7, statistical data about the language have been given. We did not place some details of the tables in those chapters because of lack of space. In this appendix, we display the remaining results.

TABLE C.1. Words that have more than three categories

Number of categories	Words	Categories
4	aç	a d n v
	ak	a n r v
	as	a b n v
	aşağı	a d e n
	ay	a i r v
	az	a b d v
	baylan	a d r v
	beri	a e n r
	dal	a n r v
	diril	a n r v
	efendi	a i n r
	eşkin	a d n r
	geri	a d i n
	gönen	a n r v
	güc	a d n r
	güzel	a d n r
	kara	a n r v
	karığ	a n r v
	karşı	a d e n
	katı	a d n r
	koca	a n r v
	o	a i l p
	öğür	a n r v
	öz	a n p r
	salt	a d k r
	sek	a d k v
	tamam	a d i n
	tez	a d n r
	yakın	a d n v
	yalnız	a c d n
	zahir	a d n r
	zor	a d i n
5	al	a b n r v
	çek	a k n r v
	doğru	a d e n r
	er	b d n r v
	ileri	a d i n r
	tek	a d k n r
6	ne	a b c d n p

TABLE C.2. Distribution of words to initial two letters

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
ka	1,110	3.55 %	892	4.10 %
ba	734	2.35 %	478	2.20 %
te	730	2.34 %	597	2.75 %
ta	722	2.31 %	478	2.20 %
sa	694	2.23 %	438	2.01 %
ma	616	1.97 %	510	2.35 %
ha	598	1.92 %	461	2.12 %
me	589	1.88 %	448	2.06 %
gü	551	1.76 %	141	0.65 %
se	525	1.68 %	293	1.35 %
mü	512	1.64 %	452	2.08 %
ko	488	1.56 %	368	1.69 %
ya	481	1.54 %	354	1.63 %
de	435	1.39 %	329	1.51 %
be	425	1.36 %	242	1.11 %
mu	380	1.22 %	312	1.44 %
al	373	1.20 %	209	0.96 %
ku	364	1.16 %	224	1.03 %
pa	351	1.12 %	293	1.35 %
ak	348	1.11 %	149	0.68 %
er	347	1.11 %	67	0.31 %
di	315	1.01 %	224	1.03 %
bi	306	0.98 %	190	0.87 %
ke	292	0.93 %	240	1.10 %
öz	286	0.92 %	58	0.27 %
gö	281	0.90 %	150	0.69 %
kı	281	0.90 %	206	0.95 %
ça	278	0.89 %	184	0.85 %
ar	275	0.88 %	167	0.77 %
ay	257	0.82 %	98	0.45 %
pe	253	0.81 %	177	0.81 %
şa	237	0.76 %	152	0.69 %
da	236	0.76 %	178	0.82 %
na	230	0.74 %	137	0.63 %
to	229	0.73 %	138	0.64 %
si	224	0.72 %	178	0.82 %
ne	219	0.70 %	127	0.58 %
mi	209	0.67 %	170	0.78 %
şe	209	0.67 %	116	0.53 %
ye	202	0.65 %	136	0.63 %
an	200	0.64 %	166	0.76 %
tu	196	0.62 %	100	0.46 %
sü	195	0.62 %	140	0.64 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
ge	193	0.62 %	144	0.66 %
re	192	0.61 %	141	0.65 %
is	191	0.61 %	155	0.71 %
fe	190	0.61 %	107	0.49 %
bo	189	0.60 %	120	0.55 %
ga	189	0.60 %	166	0.76 %
ra	189	0.60 %	123	0.56 %
so	188	0.60 %	125	0.58 %
ca	185	0.59 %	77	0.35 %
fa	182	0.58 %	134	0.62 %
su	182	0.58 %	120	0.55 %
bu	174	0.56 %	123	0.57 %
la	173	0.56 %	143	0.66 %
in	171	0.55 %	146	0.67 %
va	171	0.55 %	123	0.56 %
il	168	0.54 %	86	0.40 %
ce	164	0.52 %	111	0.51 %
hi	163	0.52 %	140	0.64 %
çe	162	0.52 %	107	0.49 %
tü	160	0.51 %	76	0.35 %
do	157	0.50 %	110	0.51 %
fi	153	0.49 %	129	0.59 %
ki	150	0.48 %	121	0.56 %
he	142	0.45 %	125	0.58 %
kü	141	0.45 %	116	0.53 %
po	140	0.45 %	118	0.54 %
sı	131	0.42 %	110	0.51 %
ve	131	0.42 %	82	0.38 %
or	130	0.42 %	92	0.42 %
çi	129	0.41 %	103	0.47 %
el	127	0.41 %	99	0.46 %
as	126	0.40 %	95	0.44 %
ab	123	0.39 %	49	0.23 %
at	117	0.37 %	62	0.29 %
kö	116	0.37 %	89	0.41 %
nu	116	0.37 %	8	0.04 %
mo	114	0.36 %	85	0.39 %
dü	110	0.35 %	93	0.43 %
du	109	0.35 %	55	0.25 %
en	109	0.35 %	82	0.38 %
pi	109	0.35 %	97	0.45 %
ci	106	0.34 %	75	0.35 %
fo	105	0.34 %	102	0.47 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
za	105	0.34 %	79	0.36 %
ho	103	0.33 %	78	0.36 %
ze	100	0.32 %	67	0.31 %
yu	96	0.31 %	55	0.25 %
ni	95	0.30 %	64	0.29 %
es	93	0.30 %	59	0.27 %
ok	93	0.30 %	31	0.14 %
pr	93	0.30 %	84	0.39 %
ad	92	0.30 %	51	0.23 %
tr	91	0.29 %	83	0.38 %
le	90	0.29 %	64	0.29 %
ço	89	0.28 %	71	0.33 %
ti	88	0.28 %	68	0.31 %
hü	87	0.28 %	47	0.22 %
li	84	0.27 %	68	0.31 %
ek	83	0.27 %	71	0.33 %
çı	82	0.26 %	72	0.33 %
ön	82	0.26 %	53	0.24 %
şi	80	0.26 %	61	0.28 %
kr	75	0.24 %	70	0.32 %
st	74	0.24 %	68	0.31 %
bü	73	0.23 %	52	0.24 %
hu	71	0.23 %	54	0.25 %
ac	70	0.22 %	43	0.20 %
ağ	70	0.22 %	52	0.24 %
em	70	0.22 %	47	0.22 %
fı	70	0.22 %	67	0.31 %
hı	70	0.22 %	56	0.26 %
iç	70	0.22 %	67	0.31 %
ul	70	0.22 %	17	0.08 %
yü	70	0.22 %	47	0.22 %
ik	68	0.22 %	63	0.29 %
on	68	0.22 %	28	0.13 %
yı	67	0.21 %	42	0.19 %
zi	67	0.21 %	49	0.23 %
vi	66	0.21 %	50	0.23 %
ru	65	0.21 %	37	0.17 %
tı	65	0.21 %	63	0.29 %
ev	64	0.20 %	50	0.23 %
ot	60	0.19 %	50	0.23 %
iş	59	0.19 %	51	0.23 %
dö	58	0.19 %	44	0.20 %
cı	57	0.18 %	55	0.25 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
ıs	57	0.18 %	55	0.25 %
im	57	0.18 %	48	0.22 %
iz	56	0.18 %	39	0.18 %
gi	55	0.18 %	40	0.18 %
lo	55	0.18 %	48	0.22 %
ün	53	0.17 %	10	0.05 %
yo	53	0.17 %	41	0.19 %
ah	52	0.17 %	39	0.17 %
eş	52	0.17 %	46	0.21 %
ir	52	0.17 %	42	0.19 %
sö	52	0.17 %	38	0.17 %
uz	52	0.17 %	33	0.15 %
az	51	0.16 %	32	0.15 %
it	51	0.16 %	49	0.23 %
ol	51	0.16 %	33	0.15 %
af	50	0.16 %	38	0.17 %
ör	50	0.16 %	34	0.16 %
ap	49	0.16 %	46	0.21 %
av	49	0.16 %	37	0.17 %
dı	49	0.16 %	48	0.22 %
aş	48	0.15 %	38	0.18 %
bö	48	0.15 %	37	0.17 %
ış	48	0.15 %	25	0.12 %
zı	48	0.15 %	48	0.22 %
üs	47	0.15 %	35	0.16 %
br	46	0.15 %	30	0.14 %
gr	46	0.15 %	40	0.18 %
ih	46	0.15 %	43	0.20 %
pl	45	0.14 %	42	0.19 %
fr	44	0.14 %	35	0.16 %
pu	44	0.14 %	38	0.17 %
kl	43	0.14 %	41	0.19 %
uy	43	0.14 %	32	0.15 %
et	42	0.13 %	37	0.17 %
cu	41	0.13 %	29	0.13 %
ro	41	0.13 %	32	0.15 %
mı	40	0.13 %	40	0.18 %
us	39	0.12 %	23	0.11 %
zü	39	0.12 %	19	0.09 %
gu	38	0.12 %	27	0.12 %
ed	37	0.12 %	21	0.10 %
fl	37	0.12 %	35	0.16 %
no	37	0.12 %	29	0.13 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
ri	37	0.12 %	24	0.11 %
ur	37	0.12 %	9	0.04 %
eğ	35	0.11 %	29	0.13 %
şı	34	0.11 %	33	0.15 %
çö	33	0.11 %	28	0.13 %
ip	33	0.11 %	29	0.13 %
uğ	33	0.11 %	10	0.05 %
oğ	32	0.10 %	7	0.03 %
pı	32	0.10 %	29	0.13 %
rü	32	0.10 %	22	0.10 %
uç	32	0.10 %	15	0.07 %
fu	31	0.10 %	27	0.12 %
gı	31	0.10 %	28	0.13 %
oy	30	0.10 %	18	0.08 %
sp	30	0.10 %	27	0.12 %
ül	30	0.10 %	11	0.05 %
ür	30	0.10 %	17	0.08 %
ef	29	0.09 %	19	0.09 %
je	29	0.09 %	24	0.11 %
pü	29	0.09 %	24	0.11 %
ib	28	0.09 %	20	0.09 %
ök	28	0.09 %	15	0.07 %
tö	28	0.09 %	16	0.07 %
fü	27	0.09 %	19	0.09 %
aç	26	0.08 %	17	0.08 %
ag	26	0.08 %	24	0.11 %
co	26	0.08 %	9	0.04 %
çu	26	0.08 %	21	0.10 %
id	26	0.08 %	24	0.11 %
bı	25	0.08 %	25	0.12 %
eb	25	0.08 %	21	0.10 %
um	25	0.08 %	10	0.05 %
zo	25	0.08 %	19	0.09 %
cü	24	0.08 %	23	0.11 %
ıl	24	0.08 %	17	0.08 %
ja	24	0.08 %	15	0.07 %
lü	24	0.08 %	15	0.07 %
ep	23	0.07 %	22	0.10 %
go	23	0.07 %	16	0.07 %
nü	23	0.07 %	19	0.09 %
öğ	23	0.07 %	18	0.08 %
şu	23	0.07 %	20	0.09 %
vo	23	0.07 %	21	0.10 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
if	22	0.07 %	21	0.10 %
ec	21	0.07 %	11	0.05 %
op	21	0.07 %	19	0.09 %
eg	20	0.06 %	17	0.08 %
os	20	0.06 %	7	0.03 %
yi	20	0.06 %	13	0.06 %
dr	19	0.06 %	17	0.08 %
od	19	0.06 %	11	0.05 %
ır	19	0.06 %	17	0.08 %
üç	19	0.06 %	11	0.05 %
gl	18	0.06 %	17	0.08 %
iy	18	0.06 %	10	0.05 %
rö	18	0.06 %	17	0.08 %
eh	17	0.05 %	15	0.07 %
ez	17	0.05 %	14	0.06 %
lu	16	0.05 %	6	0.03 %
om	16	0.05 %	13	0.06 %
vı	16	0.05 %	16	0.07 %
yö	16	0.05 %	10	0.05 %
bl	15	0.05 %	11	0.05 %
ey	15	0.05 %	13	0.06 %
iğ	15	0.05 %	15	0.07 %
öd	15	0.05 %	14	0.06 %
öt	15	0.05 %	12	0.06 %
şo	15	0.05 %	14	0.06 %
üz	15	0.05 %	11	0.05 %
vu	15	0.05 %	14	0.06 %
ob	14	0.04 %	13	0.06 %
öl	14	0.04 %	9	0.04 %
uf	14	0.04 %	10	0.05 %
öv	13	0.04 %	5	0.02 %
ic	12	0.04 %	11	0.05 %
ji	12	0.04 %	9	0.04 %
of	12	0.04 %	10	0.05 %
oz	12	0.04 %	6	0.03 %
ps	12	0.04 %	12	0.06 %
rı	12	0.04 %	6	0.03 %
sk	12	0.04 %	10	0.05 %
un	12	0.04 %	9	0.04 %
hö	11	0.04 %	10	0.05 %
şö	11	0.04 %	11	0.05 %
üm	11	0.04 %	6	0.03 %
öy	10	0.03 %	9	0.04 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
pö	10	0.03 %	9	0.04 %
şü	10	0.03 %	7	0.03 %
ut	10	0.03 %	7	0.03 %
zu	10	0.03 %	9	0.04 %
ıt	9	0.03 %	9	0.04 %
iv	9	0.03 %	8	0.04 %
lö	9	0.03 %	9	0.04 %
nö	9	0.03 %	9	0.04 %
og	9	0.03 %	0	0.00 %
çü	8	0.03 %	6	0.03 %
nı	8	0.03 %	8	0.04 %
üt	8	0.03 %	6	0.03 %
ai	7	0.02 %	6	0.03 %
ov	7	0.02 %	6	0.03 %
uc	7	0.02 %	5	0.02 %
uk	7	0.02 %	5	0.02 %
üş	7	0.02 %	7	0.03 %
ch	6	0.02 %	0	0.00 %
ık	6	0.02 %	6	0.03 %
jü	6	0.02 %	4	0.02 %
ög	6	0.02 %	1	0.00 %
öm	6	0.02 %	1	0.00 %
th	6	0.02 %	4	0.02 %
vü	6	0.02 %	6	0.03 %
aj	5	0.02 %	4	0.02 %
au	5	0.02 %	2	0.01 %
f.	5	0.02 %	5	0.02 %
hr	5	0.02 %	4	0.02 %
ıh	5	0.02 %	5	0.02 %
jo	5	0.02 %	2	0.01 %
ju	5	0.02 %	4	0.02 %
qu	5	0.02 %	0	0.00 %
sl	5	0.02 %	3	0.01 %
tb	5	0.02 %	5	0.02 %
up	5	0.02 %	5	0.02 %
üv	5	0.02 %	4	0.02 %
dd	4	0.01 %	4	0.02 %
dz	4	0.01 %	4	0.02 %
ej	4	0.01 %	3	0.01 %
ıp	4	0.01 %	4	0.02 %
ız	4	0.01 %	4	0.02 %
ia	4	0.01 %	4	0.02 %
mö	4	0.01 %	4	0.02 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
p.	4	0.01 %	4	0.02 %
tc	4	0.01 %	4	0.02 %
tm	4	0.01 %	4	0.02 %
ub	4	0.01 %	1	0.00 %
uş	4	0.01 %	4	0.02 %
üb	4	0.01 %	0	0.00 %
a.	3	0.01 %	3	0.01 %
ae	3	0.01 %	3	0.01 %
ao	3	0.01 %	3	0.01 %
b.	3	0.01 %	3	0.01 %
bş	3	0.01 %	3	0.01 %
c.	3	0.01 %	3	0.01 %
cr	3	0.01 %	1	0.00 %
dm	3	0.01 %	3	0.01 %
eo	3	0.01 %	3	0.01 %
hs	3	0.01 %	3	0.01 %
ıd	3	0.01 %	0	0.00 %
ığ	3	0.01 %	3	0.01 %
ın	3	0.01 %	3	0.01 %
jö	3	0.01 %	3	0.01 %
ks	3	0.01 %	3	0.01 %
lı	3	0.01 %	3	0.01 %
oc	3	0.01 %	2	0.01 %
oe	3	0.01 %	3	0.01 %
oh	3	0.01 %	3	0.01 %
oo	3	0.01 %	3	0.01 %
öb	3	0.01 %	3	0.01 %
öc	3	0.01 %	1	0.00 %
ös	3	0.01 %	3	0.01 %
ph	3	0.01 %	2	0.01 %
pt	3	0.01 %	3	0.01 %
sb	3	0.01 %	3	0.01 %
sf	3	0.01 %	2	0.01 %
sm	3	0.01 %	3	0.01 %
sn	3	0.01 %	3	0.01 %
sr	3	0.01 %	2	0.01 %
ts	3	0.01 %	3	0.01 %
ug	3	0.01 %	0	0.00 %
uh	3	0.01 %	3	0.01 %
üf	3	0.01 %	3	0.01 %
we	3	0.01 %	0	0.00 %
bn	2	0.01 %	2	0.01 %
cb	2	0.01 %	2	0.01 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
cö	2	0.01 %	2	0.01 %
fk	2	0.01 %	2	0.01 %
g.	2	0.01 %	2	0.01 %
gn	2	0.01 %	2	0.01 %
h.	2	0.01 %	2	0.01 %
hv	2	0.01 %	2	0.01 %
ıc	2	0.01 %	2	0.01 %
ım	2	0.01 %	2	0.01 %
k.	2	0.01 %	2	0.01 %
kd	2	0.01 %	2	0.01 %
kg	2	0.01 %	2	0.01 %
kh	2	0.01 %	1	0.00 %
lt	2	0.01 %	2	0.01 %
lv	2	0.01 %	2	0.01 %
mb	2	0.01 %	0	0.00 %
mg	2	0.01 %	2	0.01 %
mk	2	0.01 %	2	0.01 %
mn	2	0.01 %	2	0.01 %
mp	2	0.01 %	2	0.01 %
ms	2	0.01 %	2	0.01 %
mt	2	0.01 %	2	0.01 %
nd	2	0.01 %	1	0.00 %
nh	2	0.01 %	0	0.00 %
oj	2	0.01 %	2	0.01 %
öç	2	0.01 %	1	0.00 %
öf	2	0.01 %	2	0.01 %
öp	2	0.01 %	2	0.01 %
öş	2	0.01 %	1	0.00 %
ss	2	0.01 %	2	0.01 %
tk	2	0.01 %	2	0.01 %
tl	2	0.01 %	2	0.01 %
tp	2	0.01 %	2	0.01 %
tt	2	0.01 %	2	0.01 %
tv	2	0.01 %	2	0.01 %
ty	2	0.01 %	2	0.01 %
uv	2	0.01 %	2	0.01 %
üc	2	0.01 %	2	0.01 %
üg	2	0.01 %	0	0.00 %
ük	2	0.01 %	0	0.00 %
wa	2	0.01 %	0	0.00 %
wi	2	0.01 %	0	0.00 %
y.	2	0.01 %	2	0.01 %
yb	2	0.01 %	2	0.01 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
yd	2	0.01 %	2	0.01 %
a	1	0.00 %	1	0.00 %
aa	1	0.00 %	1	0.00 %
aı	1	0.00 %	1	0.00 %
aü	1	0.00 %	1	0.00 %
b	1	0.00 %	1	0.00 %
bb	1	0.00 %	1	0.00 %
bç	1	0.00 %	1	0.00 %
bh	1	0.00 %	0	0.00 %
bk	1	0.00 %	1	0.00 %
bp	1	0.00 %	1	0.00 %
c	1	0.00 %	1	0.00 %
cc	1	0.00 %	1	0.00 %
cd	1	0.00 %	1	0.00 %
cf	1	0.00 %	1	0.00 %
cg	1	0.00 %	1	0.00 %
cl	1	0.00 %	1	0.00 %
cm	1	0.00 %	1	0.00 %
cs	1	0.00 %	1	0.00 %
ç	1	0.00 %	1	0.00 %
çn	1	0.00 %	0	0.00 %
çs	1	0.00 %	1	0.00 %
çv	1	0.00 %	1	0.00 %
d	1	0.00 %	1	0.00 %
d.	1	0.00 %	1	0.00 %
dc	1	0.00 %	1	0.00 %
dg	1	0.00 %	1	0.00 %
dh	1	0.00 %	1	0.00 %
dk	1	0.00 %	1	0.00 %
dl	1	0.00 %	1	0.00 %
dn	1	0.00 %	1	0.00 %
dp	1	0.00 %	1	0.00 %
ds	1	0.00 %	1	0.00 %
dt	1	0.00 %	1	0.00 %
dv	1	0.00 %	1	0.00 %
dy	1	0.00 %	1	0.00 %
e	1	0.00 %	1	0.00 %
eç	1	0.00 %	1	0.00 %
ei	1	0.00 %	1	0.00 %
eu	1	0.00 %	1	0.00 %
f	1	0.00 %	1	0.00 %
fb	1	0.00 %	1	0.00 %
fö	1	0.00 %	1	0.00 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
ft	1	0.00 %	1	0.00 %
g	1	0.00 %	1	0.00 %
gd	1	0.00 %	1	0.00 %
gm	1	0.00 %	1	0.00 %
gs	1	0.00 %	1	0.00 %
gy	1	0.00 %	0	0.00 %
h	1	0.00 %	1	0.00 %
hf	1	0.00 %	1	0.00 %
hg	1	0.00 %	1	0.00 %
hm	1	0.00 %	1	0.00 %
hp	1	0.00 %	1	0.00 %
hz	1	0.00 %	1	0.00 %
ı	1	0.00 %	1	0.00 %
ıb	1	0.00 %	1	0.00 %
ıo	1	0.00 %	1	0.00 %
ıv	1	0.00 %	1	0.00 %
i	1	0.00 %	1	0.00 %
ie	1	0.00 %	1	0.00 %
ig	1	0.00 %	1	0.00 %
iö	1	0.00 %	1	0.00 %
iü	1	0.00 %	1	0.00 %
j	1	0.00 %	1	0.00 %
j.	1	0.00 %	1	0.00 %
jg	1	0.00 %	1	0.00 %
k	1	0.00 %	1	0.00 %
kk	1	0.00 %	1	0.00 %
km	1	0.00 %	1	0.00 %
kv	1	0.00 %	1	0.00 %
kw	1	0.00 %	1	0.00 %
ky	1	0.00 %	1	0.00 %
l	1	0.00 %	1	0.00 %
lr	1	0.00 %	1	0.00 %
lw	1	0.00 %	1	0.00 %
ly	1	0.00 %	0	0.00 %
m	1	0.00 %	1	0.00 %
m.	1	0.00 %	1	0.00 %
md	1	0.00 %	1	0.00 %
mm	1	0.00 %	1	0.00 %
mv	1	0.00 %	1	0.00 %
mw	1	0.00 %	0	0.00 %
my	1	0.00 %	0	0.00 %
mz	1	0.00 %	0	0.00 %
n	1	0.00 %	1	0.00 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
n'	1	0.00 %	0	0.00 %
nb	1	0.00 %	1	0.00 %
o	1	0.00 %	1	0.00 %
oş	1	0.00 %	1	0.00 %
ou	1	0.00 %	0	0.00 %
ö	1	0.00 %	1	0.00 %
öa	1	0.00 %	0	0.00 %
öh	1	0.00 %	1	0.00 %
p	1	0.00 %	1	0.00 %
pb	1	0.00 %	1	0.00 %
pd	1	0.00 %	1	0.00 %
pk	1	0.00 %	0	0.00 %
pm	1	0.00 %	1	0.00 %
py	1	0.00 %	0	0.00 %
r	1	0.00 %	1	0.00 %
rb	1	0.00 %	1	0.00 %
rc	1	0.00 %	1	0.00 %
rd	1	0.00 %	1	0.00 %
rh	1	0.00 %	1	0.00 %
rn	1	0.00 %	1	0.00 %
s	1	0.00 %	1	0.00 %
s.	1	0.00 %	1	0.00 %
sc	1	0.00 %	1	0.00 %
sh	1	0.00 %	0	0.00 %
sv	1	0.00 %	0	0.00 %
sy	1	0.00 %	0	0.00 %
sz	1	0.00 %	0	0.00 %
ş	1	0.00 %	1	0.00 %
ş.	1	0.00 %	1	0.00 %
şb	1	0.00 %	1	0.00 %
şr	1	0.00 %	0	0.00 %
şv	1	0.00 %	1	0.00 %
t	1	0.00 %	1	0.00 %
td	1	0.00 %	1	0.00 %
tf	1	0.00 %	1	0.00 %
tg	1	0.00 %	1	0.00 %
tğ	1	0.00 %	1	0.00 %
u	1	0.00 %	1	0.00 %
ua	1	0.00 %	0	0.00 %
ue	1	0.00 %	0	0.00 %
ü	1	0.00 %	1	0.00 %
ü.	1	0.00 %	1	0.00 %
üd	1	0.00 %	1	0.00 %

TABLE C.2. Distribution of words to initial two letters (continued)

Initial letters	[with proper nouns] Number of words	Percentage	[without proper nouns] Number of words	Percentage
üğ	1	0.00 %	1	0.00 %
üy	1	0.00 %	1	0.00 %
v	1	0.00 %	1	0.00 %
vb	1	0.00 %	1	0.00 %
vd	1	0.00 %	1	0.00 %
vl	1	0.00 %	0	0.00 %
vs	1	0.00 %	1	0.00 %
w	1	0.00 %	1	0.00 %
w.	1	0.00 %	1	0.00 %
wb	1	0.00 %	1	0.00 %
wc	1	0.00 %	1	0.00 %
wh	1	0.00 %	1	0.00 %
wm	1	0.00 %	1	0.00 %
wo	1	0.00 %	0	0.00 %
x	1	0.00 %	1	0.00 %
xa	1	0.00 %	0	0.00 %
xe	1	0.00 %	1	0.00 %
y	1	0.00 %	1	0.00 %
yr	1	0.00 %	1	0.00 %
ys	1	0.00 %	1	0.00 %
yy	1	0.00 %	1	0.00 %
yz	1	0.00 %	1	0.00 %
z	1	0.00 %	1	0.00 %
zf	1	0.00 %	1	0.00 %
zn	1	0.00 %	1	0.00 %
zr	1	0.00 %	1	0.00 %

TABLE C.3. Longest words in the lexicon

Length	Word	Categories
17	elektromanyetizma	n
	kafadanbacaklılar	n
	meryemanaeldiveni	n
	püskülkuyruklular	n
	vaybabacığımcılığ	n
	yuvarlakağızlılar	n
18	dikenlibalıkgiller	n
	dikenliyüzgeçliler	n
	elektrodinamometre	n
	gevişgetirmeyenler	n
	güneşhayvancıkları	n
	karındanbacaklılar	n
	kırlangıçfırtınası	n
	mikrosinematografi	n
	özkedibalığıgiller	n
	sürmemantarıgiller	n
	üniversitelerarası	a
	yassısolungaçlılar	n
19	denizkaplumbağaları	n
	elektroansefalogram	n
	elektrokardiyografi	n
	geridöndürülemezliğ	n
20	çitarsarmısağıgiller	n
	elektroansefalografi	n

TABLE C.4. Root words that do not obey the primary vowel harmony rule

Category	Words that do not obey the rule
a	abani, abes, abidevi, abstre, acayip, acele, acemi, acil, adali, adapte, adet, adi, adil, adli, aerodinamiğ, afaki
b	eu
c	bari, bazen, dahi, hakeza, halbuki, keza, kezaliğ, lakin, madem, mademki, mamafih, oysaki, sanki, şayet, vaktaki, velhasıl, velhasılıkelam, veya, veyahut, yani, zira
d	accelerando, acele, acep, acitato, adagio, adeta, affettuoso, agitato, ahir, ailecek, akabinde, akilane
e	aid, dair, hasebiyle, ila, itibaren, naşi, rağmen
i	abe, acayib, aferin, aleykümselam, alimallah, amenna, başkesen, billah, bismillah, cicoz, dandini, destur
n	abadi, abaküs, abandone, abani, abdest, abdiaciz, abdülleziz, abece, aberasyon, abide, abis, ablatif
p	birkaç, filan, filanca
s	milyar, milyon, trilyon
v	adçek, added, affeyle, ahzed, aşer, defol, eşkoş, hamded, hasred, hatmed, inan, kahred, kahreyle, kaybed, koyver, menol, merhabalaş, nakşed, rapted, sabreyle, sahileş, salıver, tabed, tayyed, vazgeç, zehrol
w	absürd, adfül, alameti, alikıran, ambale, amiri, ayini, çakar-söner, çintan, dişyuvasıl, etıbba, formüle, hanidir, hidroklorik, iner-çıkar, karaim, kito, konstrüktivist, mepsuten, monte, onikiparmak, sabote, taler, ürodel, veryansın, zevku

TABLE C.5. Root words that do not obey the secondary vowel harmony rule

Category	Words that do not obey the rule
a	abaşo, abullabut, abus, acul, aerodinamiğ, afyonkeş, agnostiğ, akaju, akkor, akromatiğ, aktüel, akut
b	au, eu
c	dolayısıyla, halbuki, oysaki, üstelik, veyahut, yahut
d	accelerando, acitato, adacyo, adagio, affettuoso, agitato, akşamüstü, akşamüzeri, alaminüt, alelhusus
e	dolayı, dolayısıyla
i	abu, agu, agucuğ, aleykümselam, alo, aport, aşkolsun, ayol, banko, bravo, cicoz, destur, desturun, elhamdülillah, estağfurullah, fesuphanallah, günaydın, hayrola, hodri, istop, müthiş, öhö, pardon, paydos, selamünaleyküm, şaralop, tünaydın, voyvo, yahu
n	abajur, abaküs, abandone, abanoz, abdülleziz, aberasyon, abluka, abone, abonman, aborda
p	öteki, şuracığ
s	milyon, trilyon
v	agula, alıkonul, alıkoy, avun, avut, başvur, defol, dolandır, düvesi, eşkoş, gadrol, gaşyol, göresi, hallol, indükle, kahrol, kavrul, kavr, kavuş, kaybol, kaydol, lağvol, mahvol, menol, nakşol, öngör, parpulla, savul, savun, savr, savuş, sırrol, süblimleş, yamrul, yamul, yavukla, zehrol
w	absürd, abuk, abur, adfül, afur, cambul, çakar-söner, çarpuk, çatur, dişyuvasıl, dolanlı, elüstü, hidroklorik, katur, kito, konstrüktivist, kumandı, mepsuten, onikiparmak, sabote, sakangur, sakulta, ürik, ürodel, yamru, yazboz, zevku

TABLE C.6. Root words that do not obey the last phoneme rule

Category	Words that do not obey the rule
a	homolog, had, şad, yad
b	ac, ag, b, c, cb, cc, cd, gd, hg, mg, nb, nd, pb, pd, rb, sb, sc, tb, tc, yb
l	b, c, d, g
n	ad, alg, antropolog, arkeolog, astrolog, aysberg, aysfild, bakteriyolog, biyolog, brifing, bumerang
v	ged, yed
w	absürd, hod

TABLE C.7. Words that do not obey the vowel harmony rule

Word	Categories
acul	a
akropol	n
alg	n
alkol	n
amal	n
ametal	n
amfibol	n
amiral	n
ampersaat	n
ampul	n
anasaat	n
anormal	a n
arzuhal	n
atol	n
atonal	a
ayal	n
bandrol	n
barkarol	n
basketbol	n
başrol	n

TABLE C.8. Words that do not obey the consonant harmony rule 1

Word	Categories
ad	n
ağ	n v
ağıryağ	n
alg	n
antikçağ	n
antropoloğ	n
arkeoloğ	n
astroloğ	n
aysberg	n
aysfild	n
bağ	n
bakteriyoloğ	n
baliğ	a
başbuğ	n
beliğ	a
biyoloğ	n
boğ	v
böğ	n
brifing	n
buluğ	n

TABLE C.9. Words that obey the consonant harmony rule 2

Word	Categories
antropoloğ	n
arkeoloğ	n
astroloğ	n
bakteriyoloğ	n
biyoloğ	n
dermatoloğ	n
diyaloğ	n
etimoloğ	n
etnoloğ	n
farmakoloğ	n
filoloğ	n
fizyoloğ	n
gastroenteroloğ	n
glasyoloğ	n
grafoloğ	n
hematoloğ	n
hidroloğ	n
jeoloğ	n
jeomorfoloğ	n
jinekoloğ	n

TABLE C.10. Words that obey the vowel insertion rule 1

Word	Categories
acz	n
adamcağz	n
ağz	n
ahd	n
akd	n
akl	n
aks	n
aln	n
asr	n
aşr	n
atf	n
azl	n
azm	n
bağr	n v
bahs	n
başşehr	n
bedr	n
benz	n
beyn	n
beyt	n

TABLE C.11. Words that obey the vowel insertion rule 2

Word	Categories
ayr	v
bağr	n v
buyr	v
çağr	v
çevr	v
çığr	n v
devr	n v
kavr	v
kayr	n v
kıvr	v
savr	v
sıyr	v
soğr	v
yoğr	v

TABLE C.12. Words that obey the double consonant rule

Word	Categories
af	n
ced	n
cer	n
emrihak	n
fek	n
fen	n
galatıhis	n
hac	n
had	n
hak	a n
hat	n
haz	n
his	n
kay	n v
ledün	n
lef	n
mahal	n
mas	n
med	n

TABLE C.13. Words that obey the phoneme deletion rule for verbs

Word	Categories
de	c v
ye	v

TABLE C.14. Words that obey the possessive suffix rule 1

Word	Categories
cima	n
elveda	i
emrivaki	n
enva	n
ibda	n
içtima	n
ihtira	n
ikna	n
imtina	n
indifa	n
inkıta	n
intiba	n
irca	n
irtica	n
irtifa	n
kablelvuku	n
kani	a
maktu	a
matbu	a
matla	n

TABLE C.15. Words that obey the possessive suffix rule 2

Word	Categories
bayi	n
cami	a n
mısra	n
sanayi	n
sema	n

TABLE C.16. Words that obey the rule for compound words 1

Word	Categories
acemborusu	n
acemlalesi	n
afrikamenekşesi	n
ağaçminesi	n
akşamsefası	n
altınsuyu	n
amerikaelması	n
amerikapiresi	n
arpasuyu	n
aslanpençesi	n
aşıboyası	a n
atbaklası	n
atkestanesi	n
ayıyoncası	n
babatatlısı	n
bademezmesi	n
balarısı	n
balıkyumurtası	n
başörtüsü	n
besisuyu	n

TABLE C.17. Words that obey the rule for compound words 2

Word	Categories
adabalığı	n
akyabalığı	n
amberağacı	n
amberbalığı	n
amberçiçeği	n
amerikaarmudu	n
antepfıstığı	n
armutkabağı	n
aslankuyruğu	n
asmakabağı	n
atbalığı	n
ateşbalığı	n
ateşböceği	n
ateşçiçeği	n
ateşkayığı	n
atkuyruğu	n
atlasağacı	n
atlasçiçeği	n
atsineği	n
avizeağacı	n

TABLE C.18. Words that obey the rule for compound words 3

Word	Categories
aslanağzı	n
balıkağzı	n
danaburnu	n
deveboynu	n
eloğlu	n
gökcismi	n
gökkutbu	n
güvercinboynu	n
herifçioğlu	n
hinoğlu	n
insanoğlu	n
itburnu	n
kargaburnu	n
kavuşumdevri	n
kişioğlu	n
kuloğlu	n
kurtağzı	n
kurtbağrı	n
kuşburnu	n
tavukgöğsü	n

TABLE C.19. Words that obey the rule for compound words 4

Word	Categories
adaçayı	n
adamkökü	n
adamotu	n
adasoğanı	n
adatavşanı	n
afyonruhu	n
akıldişi	n
alantopu	n
alaybeyi	n
alçıtaşı	n
almangümüşü	n
altınkökü	n
amerikabademi	n
amerikatavşanı	n
amerikaüzümü	n
anadili	n
anaokulu	n
anasınıfı	n
andızotu	n
aptesbozanotu	n

TABLE C.20. Words that do not obey the aorist suffix rule

Word	Categories
aban	v
abart	v
acık	v
aksır	v
aktar	n v
al	a b n v
alçal	v
aldan	v
aldat	v
alıkonul	v
alış	v
andırış	n
anır	v
anıştır	v
apar	v
apış	n v
araştır	v
artakal	v

TABLE C.21. Words that obey the rule for the morpheme *su*

Word	Categories
ağırsu	n
akarsu	n
aksu	n
altınsuyu	n
arpasuyu	n
bengisu	n
besisuyu	n
billursu	a n
camsuyu	n
çiçeksuyu	n
ersuyu	n
gülsuyu	n
karasu	n
kenarsuyu	n
madensuyu	n
özsu	n
pissu	n
su	n
yüzsuyu	n

TABLE C.22. List of longest words in the corpus

gerçekleştirilebileceğini	gerçekleştirilemeyeceğini	anlamlandırabiliyordunuz
değerlendirilebileceğini	dinleyebileceklerimizden	etkisizleştirileceğinden
gerçekleştirebileceğiniz	karşılaştırılabildikleri	karşılaştırılamayacağını
sonuçlandırılamayacağına	ağaçlandırılabileceğini	cezalandırılabileceğini
demokratikleştirecektir	gerçekleştirememektedir	gerçekleştirememişlerdi
gerçekleştiremeyecektir	gerçekleştirilebileceği	gerçekleştirilemediğini
gerçekleştirilememiştir	gerçekleştirmediklerini	görevlendirilmeyecekler
kaydettirebileceklerini	kullandırabileceklerini	kullandıramadıklarından
mikrobiyolojistlerinden	randevulaşabiliyorsunuz	anlamlandıramıyorsunuz
barındırılabilecekleri	bilgilendirilemediğini	değerlendirilebilmekte
dönüştürülemeyeceğidir	gerçekleştirilebilmesi	gerçekleştirilememişti
gerçekleştirilmektedir	görevlendirebileceğini	gösterilmeyeceklerinin
içselleştirilebileceği	ilgilenebileceklerinin	karşılaştırabildiğimiz
kullandırılabileceğini	memnuniyetsizliklerini	olanaksızlaştıracağını
sağlamlaştıracaklardır	sevindirebileceklerini	silahsızlandırılmasına
silahsızlandırılmasını	sınırlandırılmamalıdır	yararlanabileceklerini
yararlanamamaktadırlar	yaygınlaştıracaklarını	yetiştirebileceklerini
acımasızlaştırıyorlar	ayarlayabileceklerini	başkonsolosluklarında
balkanlaştırılmasının	beslenemeyebileceğini	bindokuzyüzdoksandört
Bulgaristan'dakilerin	bulundurulamayacaklar	çarptırılabilecekleri
cevaplandırılmasından	cezalandırılmışlardır	cumhurbaşkanlığındaki
deneyimlenebileceğini	desteklemeyeceklerdir	değerlendirebilecekti
değerlendirebiliyoruz	değerlendireceklerini	değerlendirememesinin
değerlendirildiğinden	değerlendirilmediğini	değerlendirilmektedir
değerlendirilmesinden	değiştirebileceğimizi	değiştirebiliyorsunuz
değiştirmeyeceklerini	dönüştürülmeyenlerden	duygusuzlaştırıyorlar
eleştirilemeyeceğinin	etkilenebileceklerini	geçirilebileceklerini
geliştirilebilecektir	genelleştirilebilirse	genişletilebileceğini
gerçekleştirdiklerini	gerçekleştirileceğini	gerçekleştirilmesidir
gerçekleştiriyorsunuz	görevlendirileceklere	gülümseyebiliyordunuz
güncelleştirileceğini	hesaplayamayacağınızı	hissettireceklerinden
hızlandırabileceğinin	hoşgörüsüzlüklerimizi	ilgilendirebileceğini
irtibatlandırılmıştır	istikrarsızlaştırıyor	istikrarsızlaştırmaya
kademelendirilecektir	karşılaşabileceğimizi	karşılaştırılabilecek
karşılaştırılamayacak	karşılaştırılmaksızın	milletvekillerimizden
monotonlaştırıldığını	oluşturulabileceğidir	ödüllendiriliyorlardı
programlaştırılmasına	silahsızlandıracaktır	sürdürebilmektedirler
taksitlendirileceğini	tamamlayabiliyorsunuz	Türkçe'leştirildiğini
yaşayabileceklerinden	yakalanabileceklerini	yakalanamayacaklarını
yararlanabildiklerini	yaygınlaşabilmektedir	yaygınlaştırılacaktır
yerleştirebileceğiniz	yerleştirileceklerini	yetiştirilebileceğine
yumuşatılamayacağıdır		

TABLE C.23. List of longest roots in the corpus

allahaısmarladık	egzistansiyalist	gastroenteroloji
telekomünikasyon	elektrifikasyon	elektromanyetik
hindistancevizi	transplantasyon	antidemokratik
karbonmonoksit	konvertibilite	mikroorganizma
milletlerarası	rehabilitasyon	reorganizasyon
selamünaleyküm	toplumlararası	transformasyon
alacakaranlık	Anadoluhisarı	bibliyografya
cumhurbaşkanı	derinlemesine	dokümantasyon
elhamdülillah	endokrinoloji	enternasyonal
fesuphanallah	Gaziosmanpaşa	hipertansiyon
Kahramanmaraş	karakteristik	kendiliğinden
konfederasyon	konsantrasyon	konservatuvar
konsolidasyon	konvansiyonel	kordiplomatik
maalmemnuniye	maslahatgüzar	mikrobiyoloji
misafirperver	modernizasyon	mütekabiliyet
nitrogliserin	pastörizasyon	radyoaktivite
sinyalizasyon	sosyoekonomik	sosyokültürel
sterilizasyon	Şarkikaraağaç	şehirlerarası
terakkiperver	transatlantik	yükseköğrenim
yükseköğretim		

APPENDIX D. LIST OF SUFFIXES THAT ARE NOT USED IN THE SPELLING CHECKER

This appendix lists the suffixes that are affixed to a small number of words. Because of this, these suffixes were not inserted into the suffix lexicon. The suffixes are given in Table D.1.

TABLE D.1. Suffixes not included in the suffix lexicon

Suffix	Source category	Words
A	n	ad, boş, diş, ot, sığ, tür, yaş
(A)cAn	v	bil, ev, iv, sev
(A)C	n	ana, baba, bazlama, dar, doğa, düz, kaya, kır, kısa, orta, top
(A)ğAn	v	dur, gel, gid, küs, ol, piş, solu
AğI	n	diş, kıl, kır
(a)j	n	bloğ, bobin, doz, dren, estamp, gram, krom, model, pilot, pompa, silindir, sonda, ton, volt
(A)Ğ	n	ad, baldır, baş, bebe, ben, bor, dam, göl, koç, koy, orta, ot, öz, pas, sol, top, yan, yayla, yol
(A)lA	v	bağda, çalka, çek, deş, dön, dur, eş, gev, gez, güd, it, ov, saç, sars, serp, şaş
(A)lgA	v	bit, çiz, göç, kon
(A)m	v	diz, dön, kur, tut
(A)mAĞ	v	bas, kaç, tut, ye
(A)n	n	dil, güc, öz, üst
(A)nAĞ	v	daya, düz, ek, geç, gör, göze, it, kes, ol, öde, ört, seç, tak, tut, yaz, yet
(A)r	n	bol, boz, deli, ev, göç, göl, iç, ot, toz
(A)t	n	anane, baş, göl, hükm, öz
(A)v	v	işle, sına, söyle
(A)y	v	bit, dene, dol, dola, ol, uza
başı	n	ases, haham, hamal, koca, kol, mehter, mimar, oda, pazar, sekban, usta
baz	n	çene, dil, düzen, hile, kumar, kuş, küfr, oyun, sihr
CA	d	adamakıllı, alelacele, ayrı, başka, beraber, böyle, çabuğ, epey, erken, evvel, iyi, kasti, öyle, sade, şöyle, yalnız
CA	v	beğen, dinlen, dokun, dönen, düşün, eğlen, gerek, güven, imren, inan, sakın
cAcIĞ	d	hemen, kolay, usul, yavaş
cAğIz	n	abla, adam, çocuğ, kadın, kız
CAĞ	n	aile, ev
CAĞ	n	ayağ, bürüm, ılı, ısı, kol, oyun, pür, yalın, yavru
CAĞ	v	em, erin, salın
CAnA	d	açığ, beraber, böyle, iyi, kolay, öyle, şöyle
cIlAyIn	p	ben, biz, sen
C	v	avun, dayan, diren, edin, erin, gönen, güven, iğren, ilen, inan, kazan, kıskan, kıvan, özen, usan, utan, üşen
dAĞ	n	dingil, öz, yar
dAlIĞ	n	gün, on, ön
DAn	n	al, yel
DAn	a	can, dış, iç, ince, kök, öz, sıra, su, top, yüreğ

TABLE D.1. Suffixes not included in the suffix lexicon (continued)

Suffix	Source category	Words
Dar	n	alem, bayrağ, defter, din, hazine, hükm, ıs, kin, makas, methal, mihman, ser
DAş	n	denk, düşün, ev, kan, karn, kol, kök, öz, pay, ses, şekl, tür, yan, yön, yurd
DAt	n	al, ser
dIrIĞ	n	boyn, burn, çiğin, oğl
GA	n	im, omur, öz, sim, soy, yar
GA	v	bil, böl, çiz, çökel, diz, göster, kavr, öner, ör, sömür, süpür, yon
GAC	v	bur, çevr, del, kıs, süz, utan, üşen, yüz
GAn	n	baş, belit, ceb, köşe, yel
GIC	v	bil, dal, del, kaz
GAn	v	çalış, eri, giriş, koru, sırıt, sürün
I	n	bar, berk, erg, palaz, şavk, taş, toz
I	v	berk, kak, kaz, sor
(I)Ğ	n	aç, geç, göz, kar
(I)kIA	v	dürt, say, sür
(I)l	n	ad, ağ, ard, as, bağ, buz, çoğ, diş, dişi, dörd, er, giz, iki, kas, kum, taş, tek, tirpit, yaban
(I)msA	a	anı, az, ben, çoğ, güc, kötü, öz, sayrı, yok
(I)msA	v	an, çek, duy, gül, kaç, soğr
(I)msAr	a	iyi, kara, kötü
(I)mtıraĞ	a	acı, beyaz, ekşi, gri, kahverengi, kırmızı, mavi, mor, pembe, sarı, siyah, yeşil
(I)n	v	ak, bas, çak, diz, ek, kıy, sat, say, yay, yaz, yığ
(I)t	n	anı, boy, bronş, döl, eş, karşı, kök, yaş
k	v	bur, sar, sil
keş	n	afyon, cefa, çile, esrar, gayret, keman, ser, sırma, tuğra
lA	i	agu, ah, deh, ıh, of, pehpeh, puf, püf, uf, yuha
lA	n	ay, buz, kum, kış, sin, tuz
lAĞ	n	av, büğe, büve, diş, güz, kış, kuş, ot, su, teker, tuz
lAm	n	bağ, boy, deng, düz, ek, en, eş, göz, iç, iki, iz, sağ, ses
leyin	n	akşam, gece, sabah
lI	n	bihuzur, eködeme, esas, hata, hız, hülya, içedönüş, kasvet, kuruntu, tad, tekparça, yanödeme, yapmacığ, zaman
mAn	n	ağ, ata, ev, göğ, göz, kamera, kir, koca, koskoca, köle, sağ, şiş
mAn	v	araştır, çek, çevr, danış, diz, eğir, eğit, eleştir, gez, göç, ılı, kat, öğret, say, seç, yaz
mAz	v	added, art, beğen, benze, değiş, doy, eksil, eski, gerek, ırgala, kırp, kızar, püskür

TABLE D.1. Suffixes not included in the suffix lexicon (continued)

Suffix	Source category	Words
mIĞ	v	çiğne, il, kıs, kıy, kus
nAĞ	v	der, değ, kas, öde
ol	n	def, gadr, gaşy, hal, kahr, kayb, kayd, lağv, mahv, sır, zehr
perver	n	cumhuriyet, hayr, menfaat, milliyet, misafir, sulh, terakki, vatan
rAĞ	n	acı, boz, dış, iç, kav, kısa, yeğ
sAĞ	n	bağr, burn, su, tüm, yakın, ırak
sAĞ	v	sav, tut
sAl	v	dokun, duy, düşün, eğit, gör, işit, uy
sever	n	barış, dil, hak, hayr, kitab, konuğ, müziğ, sanat, sinema, spor, sulh, ulus, vatan, yardım, yurd
sI	n	an, çiçeğ, yaban, yad, yan
sI	v	giy, sin, tüt, yat
sIl	n	dudağ, maya, ot, var, yoğ
sInA	d	afra tafra, bedava, böyle, öyle, şöyle, yanlama, yeni
sIz	n	besmele, eködeme, hata, kot, yanödeme, yapmacığ, zaman
şIn	a	ak, kara, sarı
tay	n	danış, kamu, kurul, yargı
(v)i	n	abide, aile, anane, anayasa, angarya, daire, efsane, fışkı, mide, mucize, terbiye, yaban
(y)ane	n	acz, akil, dost, dahi, hakim, halis, mahir, masum, mecnun, mest, muzaffer, naçiz, şair, zalim
(y)AsIcA	v	at, bat, çıldır, doy, eri, öl, zıbar
(y)In	n	ansız, çoğ, güc, güz, ilk, kış, öğle, örneğ, yaz
(y)sA	d	böyle, nasıl, neden, nerde, nerede, öyle, sıkı
zade	n	amca, asil, bey, dayı, hala, haram, helal, hemşire, kişi, paşa, teyze

BIBLIOGRAPHY

1. Charniak, E. and McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley Publishing Company, 1985.
2. Nilsson, N.J., Principles of Artificial Intelligence, Tioga Publishing Company, 1982.
3. Meskill, R.H., A Transformational Analysis of Turkish Syntax, Mouton and Co. N.V. Pub., The Hague, 1970.
4. Erguvanlı, E.E., The Function of Word Order in Turkish Grammar, University of California Press, California, May 1984.
5. Stoop, A.M., "Atmaca: Semantic Analysis by the Computer," Studies on Turkish Linguistics, pp. 539-564, 1988.
6. Hirst, G., "Semantic Interpretation and Ambiguity," Artificial Intelligence, vol.34, pp. 131-177, 1988.
7. Huber, W., Computational Linguistics: Tutorial, 1982.
8. Hankamer, J., "Morphological Parsing and the Lexicon," W.M. Wilson (ed), Lexical Representation and Processing, MIT Press, 1988.
9. Kibaroğlu, M.O., "Spell Checking in Agglutinative Languages and an Implementation for Turkish," M.S. Thesis, Boğaziçi University, İstanbul, 1991.
10. Ergin, M., Türk Dilbilgisi, 19th ed., Bayrak Yayınları, İstanbul, 1990.
11. Koç, N., Yeni Dilbilgisi, İnkılâp Yayınları, İstanbul, 1990.
12. Banguoğlu, T., Türkçe'nin Grameri, Türk Tarih Kurumu, Ankara, 1990.
13. Lees, R.B., The Phonology of Modern Standard Turkish, Indiana University Press, Bloomington, 1961.
14. Çotuksöken, Y., Türkçe'de Ekler-Kökler-Gövdeler, 2nd ed., Cem Yayınları, İstanbul, 1991.
15. Sebüktekin, H.İ., Turkish-English Contrastive Analysis - Turkish Morphology and Corresponding English Structures, Mouton Press, The Hague, Paris, 1971.
16. Güngör, T. and Kuru, S., "Representation of Turkish Morphology in ATN," Proceedings of the Second Symposium on Artificial Intelligence and Artificial Neural Networks, İstanbul, pp. 94-104, June 1993.
17. Akın, H.L., Kuru, S., Güngör, T., Hamzaoğlu, İ., and Arbatlı, D., "A Spelling Checker and Corrector for Turkish," Proceedings of the Second Turkish Symposium on Artificial Intelligence and Artificial Neural Networks, İstanbul, pp. 113-120, 24-25 June 1993.

18. Güngör, T. and Kuru, S., "Full Turkish Morphology Represented as an Augmented Transition Network," submitted to Literary and Linguistic Computing.

19. Packard, D., "Computer-Assisted Morphological Analysis of Ancient Greek," Computational and Mathematical Linguistics: Proceedings of the International Conference on Computational Linguistics, Pisa. Leo S.Olschki, Firenze, pp. 343-355, 1973.

20. Brodda, B. and Karlsson, F., An Experiment with Morphological Analysis of Finnish, Papers from the Institute of Linguistics, University of Stockholm, Publication 40, Stockholm, 1980.

21. Sagvall, A., "A System for Automatic Inflectional Analysis Implemented for Russian," Data Linguistica, 8, Almqvist and Wiksell, Stockholm, 1973.

22. Kasper, R. and Weber, D., User's Reference Manual for the C Quechua Adaptation Program, Occasional Publications in Academic Computing, Number 8, Summer Institute of Linguistics, Inc., 1982.

23. Kasper, R. and Weber, D., Programmer's Reference Manual for the C Quechua Adaptation Program, Occasional Publications in Academic Computing, Number 9, Summer Institute of Linguistics, Inc., 1982.

24. Koskenniemi, K., Two-level Morphology: A General Computational Model for Word Form Recognition and Production, Publication No: 11, Department of General Linguistics, University of Helsinki, 1983.

25. Hankamer, J., "Turkish Generative Morphology and Morphological Parsing," unpublished paper, presented at Second International Conference on Turkish Linguistics, İstanbul, 1984.

26. Köksal, A., "Automatic Morphological Analysis of Turkish," Ph.D. Thesis, Hacettepe University, Ankara, 1975.

27. Taft, M. and Forster, K., "Lexical Storage and Retrieval of Prefixed Words," Journal of Verbal Learning and Verbal Behavior, vol.14, pp. 638-647, 1975.

28. Bradley, D.C., "Lexical Representation of Derivational Relation," Juncture, MIT Press, 1980.

29. Lukatela, G., Gligorijevic, B., Kostic, A., and Turvey, M.T., "Representation of Inflected Nouns in the Internal Lexicon," Memory and Cognition, vol.8, pp. 415-423, 1980.

30. Laudanna, A. and Burani, C., "Address Mechanisms to Decomposed Lexical Entries," Linguistics, An Interdisciplinary Journal of the Language Sciences, Mouton Press, pp. 775-792, 1985.

31. Hankamer, J., "Morphological Parsing," Conference on Lexical Representation and Processing, Nijmegen, 1986.

32. Hankamer, J., "Finite State Morphology and Left to Right Phonology," Proceedings of the West Coast Conference on Formal Linguistics, vol. 5, Stanford University, 1986.

33. Kibaroğlu, M.O., and Kuru, S., "A Left-to-Right Morphological Parser for Turkish," Proceedings of the Sixth International Symposium on Computer and Information Sciences, Antalya, 1991.

34. Ritchie, G.D., Russell, G.J., Black, A.W., and Pulman, S.G., Computational Morphology, MIT Press, Cambridge, 1992.

35. Solak, A. and Oflazer, K., "Parsing Agglutinative Word Structures and Its Application To Spelling Checking For Turkish," submitted to Computational Linguistics, 1992.

36. Darcan, O.N., "An Intelligent Database Interface for Turkish," M.S. Thesis, Boğaziçi University, İstanbul, 1991.

37. Dekkers, C., Koster, C.H.A., Nederhof, M.J., and Zwol, A.V., The Grammar Work Bench: A First Step Towards Lingware Engineering, Technical Report no. 92-06, University of Nijmegen, 1992.

38. Gazdar, G. and Mellish, C., Natural Language Processing in Lisp, Addison-Wesley Publishing Company, 1989.

39. McEnery, T., Computational Linguistics: A Handbook and Toolbox for Natural Language Processing, Sigma Press, Wilmslow, 1990.

40. Stoop, A.M., "Transit in the Word of Machine Translation: Towards an Automatic Translator For Dutch and Turkish," in H.E. Boeschoten and L.T. Verhoeven (eds), Studies on Modern Turkish Proceedings of the Third Conference on Turkish Linguistics, pp. 157-177, 1987.

41. Lewis, G.L., Turkish Grammar, Oxford University Pub, 1984.

42. Farkas, E., Koster, C.H.A., Köves, P., and Naszodi, M., "Towards an Affix Grammar for the Hungarian Language," Conference on Intelligent Systems, pp. 223-236, 1991.

43. Koster, C.H.A., Affix Grammars for Natural Languages, Technical Report no. 91-12, University of Nijmegen, 1991.

44. Koster, C.H.A. and Willems, R., "Towards an Affix Grammar for Turkish," Proceedings of the Sixth International Symposium on Computer and Information Sciences, pp. 1067-1076, Antalya, 1991.

45. Oflazer, K., "Two-level Description of Turkish Morphology," Proceedings of the Second Turkish Symposium on Artificial Intelligence and Neural Networks, pp. 86-93, İstanbul, 1993.

46. Aho, A.V. and Ullman, J.D., The Theory of Parsing, Translation, and Compiling, Prentice-Hall International, 1972.

47. Gazdar, G. and Mellish, C., Natural Language Processing in Prolog, Addison-Wesley Publishing Company, 1989.

48. Krulee, G.K., Computer Processing of Natural Language, Prentice-Hall Inc., 1991.

49. Özsoy, S., Türkçe'nin Betimlemeli Sesbilimine Giriş, in preparation.

50. Özsoy, S., personal communication.

51. Töreci, E., "Statistical Investigations on the Turkish Language Using Digital Computers," M.S. Thesis, METU, Ankara, 1974.

52. Sezer, E., "The k/0 Alternation in Turkish," Harvard Studies in Phonology, vol.2, pp. 354-382, February 1981.

53. Aksoy, Ö.A., Ana Yazım Kılavuzu, 4th ed., Adam Yayınları, İstanbul, 1991.

54. Alkım, B., Redhouse Yeni Türkçe-İngilizce Sözlük, 11th ed., Redhouse Yayınları, İstanbul, 1990.

55. Eren, H., Türkçe Sözlük, Atatürk Kültür Dil ve Tarih Yüksek Kurumu, Türk Dil Kurumu Yayınları, Ankara, 1988.

56. Eren, H., Türkçe Sözlük, Türk Dil Kurumu Yayınları, Ankara, 1985.

57. Solak, A. and Oflazer, K., "Implementation Details and Performance Results of a Spelling Checker for Turkish," Proceedings of the Sixth International Symposium on Computer and Information Sciences, pp. 1057-1066, Antalya, 1991.

58. Sproat, R.W., Morphology and Computation, MIT Press, 1992.

59. Gönenç, G. and Töreci, E., "Türkçe'nin Bazı Özelliklerinin Bilgisayarla Sayımsal Çözümlenmesi," Bilişim Dergisi, pp. 42-78, 1975.

60. Pamuk, O., Kara Kitap, İletişim Yayınları, 1992.

61. Arslanoğlu, K., Yanılmanın Gerçekliği, İnsancıl Yayınları, İstanbul, 1994.

62. Kuru, S. and Akın, H.L., "Spelling Checking in Turkish," Proceedings of the DECSYM 92 Latest Trends in Computing Conference, Antalya, 1992.

63. Par, A.H., A'dan Z'ye Ansiklopedik Türk Adları Sözlüğü, Serhat Yayınevi, İstanbul, 1981.

64. Püsküllüoğlu, A., Çocuk Adları Sözlüğü, 4th ed., Özgür Yayınevi, İstanbul, 1992.

65. Sütçüoğlu, R. and Sütçüoğlu, A., İsimler Sözlüğü, Mey Yayınevi, İzmir, 1991.

66. Aysan, A. and Tuncay, S., Türk Adları Sözlüğü, V Yayınları, 1987.

67. Anonim, Büyük Dünya Atlası, Milliyet, 1992.

68. Solak, A. and Oflazer, K., "A Finite State Machine for Turkish Syllable Structure Analysis," Proceedings of the Fifth International Symposium on Computer and Information Sciences, vol.2, Nevşehir, pp. 1195-1202, 1990.

69. Demir, C. and Oflazer, K., "An ATN Grammar for a Subset of Turkish," Proceedings of the Second Turkish Symposium on Artificial Intelligence and Neural Networks, pp. 162-169, İstanbul, 1993.

70. Hovy, E.H., "Pragmatics and Natural Language Generation," Artificial Intelligence, vol.43, pp. 153-197, 1990.

71. Nirenburg, S., Carbonell, J., Tomita, M., and Goodman, K., Machine Translation, Morgan Kaufmann Publishers, California, 1992.

72. Gönenç, G., "A Finite-State Automaton for Syllabification of Turkish words," Proceedings of the Sixth International Symposium on Computer and Information Sciences, pp. 1039-1046, Antalya, 1991.