DESIGN AND IMPLEMENTATION OF A SPELLING CHECKER FOR TURKISH

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

By
Ayşin SOLAK
June 1991

DESIGN AND IMPLEMENTATION OF A SPELLING CHECKER FOR TURKISH

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCES

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Ayşın Solak
tarafından bağışlanmıştır.

By

Ayşın Solak

June 1991

B. 2860

QA
76.6
.S65
1991

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

[illegible]

Assoc. Prof. Dr. Kemal Oflazer(Principal Advisor)

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

[illegible]

Asst. Prof. Dr. Halil Altay Güvenir

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

[illegible]

Asst. Prof. Dr. M. Mete Bulut

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

DESIGN AND IMPLEMENTATION OF A SPELLING CHECKER FOR TURKISH

Ayşın Solak
M.S. in Computer Engineering and Information Sciences
Supervisor: Assoc. Prof. Dr. Kemal Oflazer
June 1991

Proliferation of personal computers and workstations that bring computing power to users of all levels has influenced how people prepare documents. Word processors offer numerous functionalities for formatting documents, and in general improving their presentation quality. In Turkey, computers are increasingly being used for document production; but word processors used lack various tools like spelling checkers specific to Turkish. The problem of spelling checking is very interesting in itself, as Turkish, being very different from many languages, presents special challenges and problems. In this thesis, the design and implementation of a spelling checker for Turkish, which can be incorporated into word processing applications, is presented.

ÖZET

TÜRKÇE METİNLERDE SÖZCÜK YAZIMI KONTROLÜNÜN TASARIMI VE GERÇEKLEŞTİRİMİ

Ayşın Solak
Bilgisayar ve Enformatik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Assoc. Prof. Dr. Kemal Oflazer
Haziran 1991

Günümüzde, kişisel bilgisayarların ve iş istasyonlarının kullanımının gittikçe artması doküman hazırlamakta kullanılan yöntemleri de etkilemektedir. Kelime işlemciler, dokümanları düzenlemek ve genel olarak kalitelerini arttırmak için pek çok işlev sunmaktadırlar. Bilgisayarların doküman hazırlamak için kullanımı Türkiye'de de gittikçe artmaktadır; ancak kullanılan kelime işlemcilerde Türkçe için sözcük yazımı kontrolü gibi bazı işlevler bulunmamaktadır. Türkçe pek çok dilden farklı bir dil olduğu ve bir takım zorluklar çıkardığı için, bu dilde sözcük yazımı kontrolü başlı başına ilginç bir problemdir. Bu tezde, Türkçe metinlerde sözcük yazımı kontrolü için gerçekleştirilen ve değişik kelime işlemcilere uyarlanabilecek bir yazılım ve tasarımı sunulmaktadır.

ACKNOWLEDGEMENT

I wish to express my considerable gratitude to my supervisor Assoc. Prof. Dr. Kemal Oflazer who has given me guidance and encouragement throughout the development of the thesis. I would also like to thank to my cousins Seval, Zuhal and Vural, and to my brother Yalçın for their help in the preparation of the dictionary. I owe a great dept of thanks to my family who have stood by me and supported me well beyond the call of duty. Finally, I would like to thank to my friends Bilge, Özlem, Shoeleh, Zeliha, and to all of my office-mates for their morale support during this study.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Proliferation of personal computers and workstations that bring computing power to users of all levels has influenced the ways in which people prepare documents. Word processors of all kinds offer numerous functionalities for entering and formatting documents according to the users' requirements and preferences. However, it has long been noted that the use of computers in this application area need not be limited to just formatting, but can extend to helping the user in improving the quality of the document. A number of tools have been developed for analyzing the text and suggesting changes that improve the readability of the documents.

Spelling checking is one of the functions that improve readability. Spelling checkers analyze documents word by word, and detect misspelled words. Solving this problem manually is usually a boring and an error-prone job as it requires a careful and fast reading, and a good memory. However, it is ideally suited for computers.

The reasons for us attacking the problem of spelling error detection for Turkish are manifold: More and more documents in the Turkish business and government work are being prepared using computers and word processors, and it is clear that such usage will increase significantly in the years to come. However, although many spelling checkers for English and some other languages have been developed, so far no such tool was present for Turkish. The reason for this is probably the complexity of the job, since being an agglutinative language, Turkish has rather complex word structures. In Turkish, words are combinations of several morphemes.[1] There is a root, and several suffixes are combined to this root in order to extend the meaning or create other classes of

[1] *Morphemes* are the smallest units of speech bearing a meaning.

words. There are certain rules that must be obeyed during the concatenation of morphemes. Wrong ordering of morphemes and errors in vowel or consonant harmonies may cause the wrong spelling of Turkish words. Consequently, in order to check the spelling of a Turkish word, it is necessary to make significant phonological[2] and morphological[3] analyses. During these analyses, the root and suffix morphemes must be determined, the necessary morphophonemic checks must be done, and the validity and the order of the morphemes must be controlled. This property of Turkish is its most important difference from other languages in the Indo-European group (e.g., English, French, German etc.), so the techniques for spelling checking developed for those languages are not readily applicable to Turkish. Thus, Turkish poses challenging issues not encountered in other spelling checkers, and therefore, understanding and solving the problem of spelling error detection for Turkish is itself an interesting research issue.

This thesis work involves the design and implementation of a first version of a spelling checker for the Turkish written language. The scope is the development of a spelling checking kernel that can be integrated to a variety of applications. The approach to spelling error detection is based on checking each word individually, with no attention to the semantics or to the context. Besides, no suggestions are given about the most likely correct words after detecting a misspelled word, i.e., spelling correction is not done.

The outline of the thesis is as follows:

General information on the properties of spelling programs and some historical information about various spelling programs, together with some examples are given in Chapter 2.

The major part of this work depends on a detailed and careful research on some features of Turkish that make the spelling checking problem for this language especially hard and interesting. Chapter 3 presents a short history of the language, detailed information on the syllable structure of Turkish words and on some basic morphophonemic aspects of the language, such as vowel and consonant harmony, and root deformations. The correct ordering of Turkish suffixes, and the rules that must be obeyed during their concatenation can be found in the same chapter.

[2] *Phonology* is the sound system of the language.
[3] *Morphology* is the word construction rules in the language.

In Chapter 4, the approach of the thesis to the problem is presented along with a description of the implementation.

Finally, a performance evaluation of the implementation is made depending on the results of some test runs of the spelling checker.

Chapter 2

SPELLING PROGRAMS

2.1 Causes of Spelling Errors

Spelling errors can be introduced in many ways. The following three are probably the most important ones [33]:

- **Author Ignorance:** Such errors can lead to consistent misspellings and are related to the difference between how a word sounds and is actually spelled.
- **Typographical Errors:** These are less consistent but perhaps more predictable, since they are related to the position of the keys on the keyboard and probably result from errors in finger movements during typing.
- **Storage Errors:** These are related to the specific problems in encoding and transmission of text.

In the context of Turkish and similar languages we can add the following to the ones above:

- **Morphological Errors:** Such errors occur during concatenation of morphemes forming words. Wrong ordering of morphemes and errors in vowel or consonant harmonies and root deformations can be considered among these errors.[1]

[1]These concepts will be explained later in the following chapter.

2.2 Types of Spelling Programs

Spelling programs are classified into two groups [33]:

1. **Spelling Checkers:** They identify potentially misspelled words in an input text file.
2. **Spelling Correctors:** They suggest a list of most likely correct words after detecting a misspelled word. Obviously, a spelling corrector is significantly more complicated than a spelling checker.

2.3 Two Structures of a Spelling Program

There are two canonical structures for spelling programs as shown in Figure 2.1 [3]. The one on the left is a *batch* program, and the other is an *interactive* program. In the batch structure, usually the input words are sorted and the duplicates are eliminated. One pass through the list and dictionary is enough to check all input tokens. The online program on the right looks up each word as it is encountered. A spelling checker may use either structure, but an interactive corrector is usually restricted to be online. Similarly, a random access dictionary may be used by either structure, but a sequential access dictionary is only suitable for a batch program.

There are some problems with a batch checker [33]. First, a substantial real-time wait may be required while the program is running and this can be rather annoying to a user at an interactive console. Second, the output list of misspelled and unknown tokens lacks context. It may be difficult using text editors to find these tokens in the text.

Such difficulties can be easily overcome with an interactive spelling checker. When a spelling error is found, an interactive checker asks the user what to do. The following list indicates some options that can be available to the user [33]:

- **Replace:** The unknown token is taken to be misspelled and will be replaced. The token is deleted and a correctly spelled word is requested from the user.
- **Replace and Remember:** The unknown token is replaced by a user

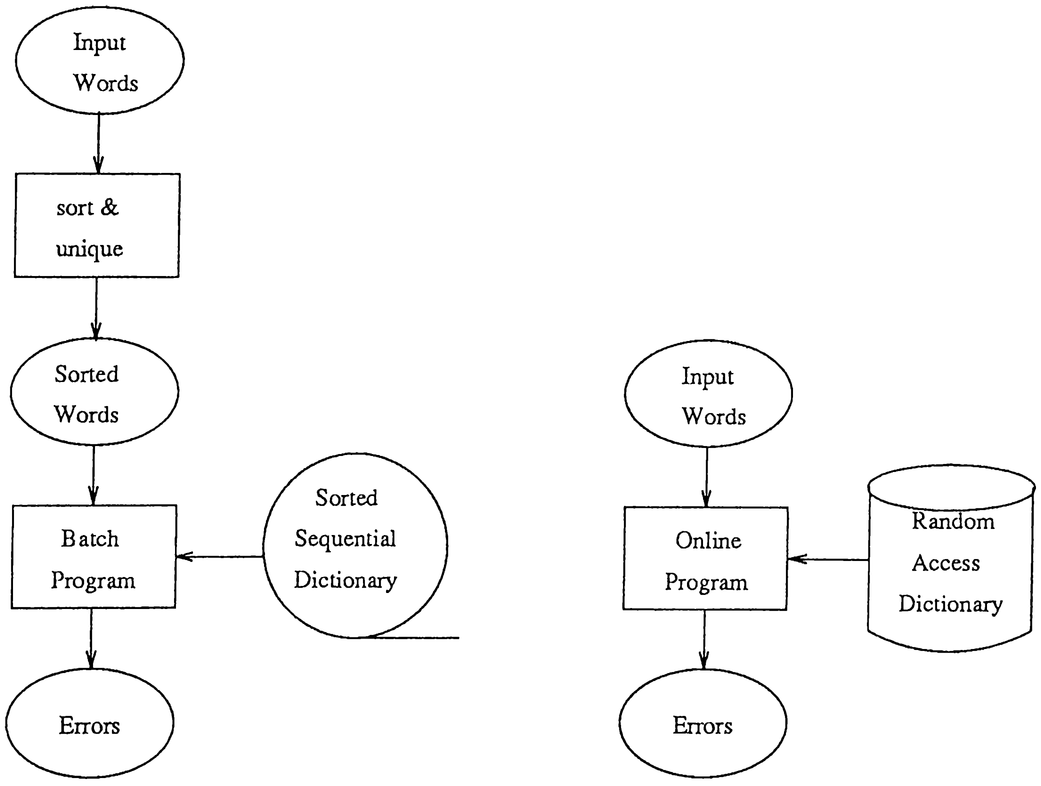

Figure 2.1: Two structures for a spelling program

specified word, and all future uses of this token are also replaced by the new word.

- **Accept:** The token is to be considered correct (in this context) and left alone.

- **Accept and Remember:** The unknown token is correct and will be left alone. In addition, all future uses of this token are correct in this document; the user should not be asked about them again.

- **Edit:** An editing submode is entered, allowing the file to be modified in its local context.

The performance of an interactive spelling checker is important. The user is waiting for the checker and the checker must therefore be sufficiently fast to avoid frustration. Also, unlike batch checkers which need to look up each distinct token once, an interactive checker may have to look up each occurrence in the order in which they are used. Thus, the interactive checker must do more work.

2.4 The Dictionary

All spelling checkers must use an external list of correctly spelled words in a data structure that serves the function of a *dictionary.* The content and the structure of the dictionary are both very important for the spelling checker to be useful and complete.

2.4.1 Content of the Dictionary

Assembling the list of correctly spelled words presents some difficulties. One must be very careful not to produce too large a dictionary, as it may include rare, archaic, or obsolete words.

One way to create the dictionary is to use the output of the spelling checker, i.e., a list of tokens which are not in the dictionary. Starting with a small dictionary, many correctly spelled words will be output by the checker. By deleting spelling errors from this list, a list of new words which can be added to the dictionary can be obtained. A new dictionary can easily be created by merging the two. In order to be successful with this method, the person who deletes the spelling errors must have an excellent knowledge of spelling and linguistics, because s/he has to decide which word is really misspelled and which one is correct.

The best sources for the list of correctly spelled words are the vocabulary items listed in the dictionaries for that language. This is a reasonable beginning, but it may cause a large dictionary to be created. Thus, certain criteria must be applied to select the necessary items. For instance, those words which are rarely used, or the stems derived from others should be deleted from the list. Another problem is that such dictionaries usually don't include proper names such as personal names, nationality names, countries and their cities. Such names must be assembled from different sources and added to the list. Many technical terms from different sciences are also not included in most of the dictionaries. These terms must also be added to the list of correctly spelled words. But, while adding care must be taken not to add too much.

The output of the checker can be examined on real runs, the checker may log after each run, and these logs may be analyzed by the developers who can recognize the problems with the checker, and correct them appropriately.

2.4.2 Structure of the Dictionary

The structure of the dictionary is also of great importance. A simple data structure will ease development and maintenance, but performance may be crucial in the interactive versions. The structure must allow very fast searches. The correct structure must be determined for the configuration of the computer system. Such factors as memory size, file access methods, and existence of virtual memory can be significant in determining appropriate structures. If the memory is large enough, the entire dictionary can be kept in memory, making operations easier and faster. It can be represented as a hash table, a binary search tree, or a trie. If the memory is virtual as opposed to physical, the dictionary structure should minimize page faults while searching. If the memory is too small, a two-layer structure may be needed, keeping the most frequently referenced words in memory, while accessing the remainder with disk reads as necessary. In this case, a B-tree or disk hash table is more suitable. There is no best algorithm; each machine and system make different demands on the dictionary data structure.

The dictionary will be most useful if it is sorted. Sorting can be done either alphabetically or by frequency [33]. Attaching a frequency count to each table entry provides the number of uses of each token. This can speed the process by searching higher frequency items first (a self-modifying table structure), and it may also help in determining misspellings. Typographical errors are generally not repeated, so tokens typed incorrectly will tend to have a very low frequency. Any token with low frequency is thus suspect. Consistent misspellings due to the author not knowing the correct spelling are not as easily found using this technique. Hence, alphabetical sorting is generally preferred.

A suggested structure for an alphabetically sorted dictionary is based on tries. A large tree structure represents the dictionary. The root of the tree branches to as many different nodes as the number of characters in the alphabet, one for each of the possible first letters of words in the dictionary.[2] Each of these nodes would branch according to the possible second letters, given the known first letter associated with the node. These new nodes branch on possible third letters, and so on. A special flag would indicate the possible ends of words. With this structure, searching for a token of *WL* characters requires following the tree *WL* levels down and checking the end-of-word bit.

A two-level search strategy for the dictionary lookup is given by Sheil [39]:

[2]This number is 28 for Turkish, since ğ never occurs in the beginning of a word.

The token is first compared with entries in a small in-core table of most frequently used words. If the token is not in this table, a search of the remaining words is made. This larger table might be stored on a secondary storage, or in a separate part of virtual memory, requiring longer access and search times.

Another improvement in search time can be made by noticing that the total number of distinct tokens in a document tends to be small to moderate, and often words of particular interest to the subject area are used. This means that for each specific document, there exist a small table of words which occur frequently in that document. Thus, it is wise to build another table of words which have been used in this specific document. By this three-table structure, the token is searched first in the small table of most common words, next in the table of words which have already been used in this document, and finally in the large list of the remaining words in the main dictionary. If a word is found at this level, it is added to the table of words for the document. Distinct data structures may be appropriate for these three tables since they exibit the following different properties:

- Most common words: static, small (100–200 items);
- Document specific words: dynamic, small to moderate (1000–2000 items);
- Secondary storage dictionary: static, very large (10,000–100,000 items).

2.4.3 Compression Techniques

For performance reasons, it is desirable to keep all the dictionary in main memory. Thus, compact representation of the dictionary is also an important issue, and serious thought has been given to ways of reducing the size of the dictionary.

Robinson and Singer [34] compressed an English dictionary something like a 50 percent using the fact that most words share the same initial letters as their predessors. Initial letters common to the previous entry in the dictionary are replaced by a count of common letters. Thus the sequence of words *eiderdown*, *eigen-value*, *eigen-vector*, *eight* appears as *eiderdown*, *2gen-value*, *7ector*, *3ht*. However, this technique has a disadvantage as it requires a sequential search of the dictionary.

Nix [30] suggested a space efficient way to store a dictionary. To illustrate the technique involved, suppose that we wish to test membership in a dictionary of 1000 words. The algorithm represents the dictionary as a 20,000 element bit table T and accesses T through ten independent hashing functions h1, h2,...,h10, that map words to numbers in the range 1 to 20,000. T is initially all zero; each word w in the dictionary is inserted by setting bits T[h1(w)], T[h2(w)],..., T[h10(w)] to 1. A word x is looked up by testing T[h1(x)], T[h2(x)],..., T[h10(x)]. If any of these bits are 0, then x is definitely not a part of the dictionary. If all of the bits are set, we say that x is in the dictionary. This method compresses the dictionary very much, but it has the disadvantages that it can produce erroneous results and it does not support affix analysis.

A similar technique is used by Dodds [11]. This technique involves replacing each dictionary entry by a hashed version. This hashed version is referred to as the *check hash.* The check hash can be significantly shorter than the average dictionary entry, thereby reduces storage requirements. The length of the check hash is fixed, which simplifies the dictionary structure and the routines required to create and access it. The fundamental limitation of the check hash is that it introduces the potential for error through collisions, because two strings may produce the same check hash. The frequency of such check hash collisions can be reduced to any desired level by increasing the length of the check hash, at the expense of increasing storage requirements.

Another compression technique is affix analysis. An affix is either a suffix or a prefix. By removing affixes and storing only the root word, the dictionary size can be reduced significantly. Two approaches are possible. In the simplest case, each token is examined for a list of common affixes. These are removed if found. The order of search can assure that larger suffixes are removed first. Then the dictionary is searched. If found, the token is judged correct. A major problem with this approach is that it does not catch misspellings which are the result of correct affixes incorrectly applied to correct words. This can allow misspelled tokens which are formed by invalid root-affix combinations to go undetected. A solution to this problem is to flag each word in the dictionary with its legal affixes. Then, after the root and the affixes are found, the flags of the root can be examined to see whether the particular affix is legal for this root [33]. Although such solutions are applicable in languages like English where the number of affixes is rather limited, they are not readily applicable in the case of Turkish where the number of possible affixes is upwards of 300 [18].

2.5 Example Spelling Programs

The original motivation for research on spelling checkers was to correct errors in data entry, and much early work was directed at finding and correcting errors resulting from specific input devices in specific context. Peterson [33] investigated the basic structure of several such existing programs:

> Davidson [7] was concerned with finding the (potentially misspelled) names of passengers for a specific airline flight. Either the stored or inquiry name (or both) might be misspelled. Carlson [6] was concerned with names and places in a genealogical database. Freeman [13] was working only with variable names and keywords in the CORC programming language, while McElwain and Evans [27] were concerned with improving the output of a system so that it would recognize Morse code.

Each of these projects considered the spelling problem as only one aspect of a larger problem, and not as a separate tool. Many academic studies on the general problem of string matching and correction algorithms have been conducted, but not with the aim of producing a working spelling program for general text.

Recently, several spelling checkers have been written for the sole purpose of checking text. Research dates back to 1957, but the first spelling checker written as an application program (rather than research) appears to have been **SPELL** for the DEC-10. This program and its revisions are widely available today. The UNIX operating system provides two spelling checkers for English: **TYPO** and **SPELL**, both of which represent different approaches.

2.5.1 SPELL for DEC-10

The first spelling program for DEC-10, **SPELL**, was written by Ralph Gorin at Stanford in 1971. It is an interactive program which searches the dictionary for each input token and asks the user what to do if the token is not in the dictionary. It uses a hash chain table of 6760 entries as its dictionary. The hash function for a token, which uses the first two letters ($L1$ and $L2$) and the length (WL) of the token (2, 3, .., 11 and over) is

$$(L1 * 26 + L2) * 10 + \min(WL - 2, 9).$$

Each hash table entry is a pointer to a chain of words, all of which have the same first two letters and the same length. This program assumes that all tokens of length 1 are correctly spelled.

There are four kinds of errors that the correction portion of the program attempts to correct:

1. one wrong letter,
2. one missing letter,
3. one extra letter,
4. two transposed letters.

For a wrong letter in the third or subsequent character, all words which are candidates must exist on the same chain that the suspect token hashes to. Hence, each entry on that chain is inspected to determine if the suspect differs from the entry by exactly one character. For a wrong letter in the first or second character, the program tries varying the second letter through all 26 possible values, searching for an exact match. Then all 26 possible values of the first letter are tried, after setting the second letter to its original value. This means that 52 more chains are searched for possible matches.

To correct one missing letter, $WL + 1$ copies of the token are made, each time inserting a null character in a new position in the suspect. The null character is never part of any word, so the suspect token augmented by an embedded null can be thought of as a word with one wrong letter (the null). Then the algorithm matching one wrong letter is used. If the first character is omitted, all 26 possible first characters are tried. Also, 26 more tokens are formed by varying the second character in case that had been omitted. To correct one extra letter, WL copies of the token are made, each with some letter removed. Each of these is looked up in the dictionary. This takes WL searches. To correct transposed letters, all combinations of transposed letters are tried. There are only $WL - 1$ such combinations, so it is fairly cheap to do this. Correction based upon these four rules is quite successful and relatively cheap, leaving the more difficult corrections to the user.

2.5.2 TYPO

One of the spelling checkers developed on UNIX is **TYPO** [29]. This program resulted from research on the frequency of two-letter pairs, *digrams*, and three-letter triples, *trigrams*, in English text. If there are 28 letters (alphabetic, blank and apostrophe), then there are 28^2 (= 784) digrams and 28^3 (= 21,952) trigrams. However, the frequency of these digrams and trigrams varies greatly, with many being extremely rare. In a large sample of text, only 550 digrams (70 percent) and 5000 trigrams (25 percent) actually occurred. If a token contains several very rare digrams or trigrams, it is potentially misspelled.

TYPO computes the actual frequency of digrams and trigrams in the input text and a list of the distinct tokens in the text. Then for each distinct token, an index of peculiarity is computed. The index for a token is the root-mean-square of the indices for each trigram of the token. The index for a trigram xyz given digram and trigram frequencies $f(xy)$, $f(yz)$, and $f(xyz)$ is:

$$[\log(f(xy) - 1) + \log(f(yz) - 1)] \,/\, 2 - \log(f(xyz) - 1).^3$$

This index is a statistical measure of the probability that the trigram xyz was produced by the same source which yielded the rest of the text.

The index of peculiarity measures how unlikely the token is in the context of the rest of the text. The output of **TYPO** is a list of tokens, sorted by index of peculiarity. Experience indicates that misspelled tokens tend to have high indices of peculiarity, and appear toward the front of the list. Errors tend to be discovered since misspelled words are found quickly at the beginning of the list, and the list is relatively short. In a document of 10,000 tokens, only approximately 1500 distinct tokens occur. This number is further reduced in **TYPO** by comparing each token with a list of over 2500 common words. If the token occurs in the list, it is known to be correct and is not output, thereby eliminating about half the distinct input tokens and producing a much shorter list.

2.5.3 SPELL for UNIX

Another spelling checker for Unix, which is called **SPELL** was first written by Steve Johnson in an afternoon in 1975. His straightforward approach is shown

[3]The log of zero is defined as -10 for this computation.

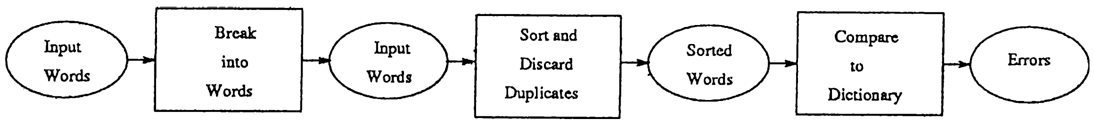

Figure 2.2: A simple spelling checker

```
prepare filename   #remove formatting commands
translit A-Z a-z   #map upper to lower case
translit !a-z @n   #remove punctuation
sort               #put words in alphabetical order
unique             #remove duplicate words
common -2 dict     #report words not in dictionary
```

Figure 2.3: Code for the simple checker

in Figure 2.2: Isolate the words in a document, sort them, and then compare the sorted list with the dictionary. The output is a list of all words in the document that are not in the dictionary.

Kernighan and Plauger reconstruct Johnson's program as in Figure 2.3. The input is the name of the file to be checked and the output is the list of misspelled words. The first program in the pipeline, **prepare**, deals with the fact that many computerized documents contain formatting commands. A spelling checker must ignore such commands. **prepare** copies its input to its output, with the formatting commands removed. **translit** transliterates its input to its output, performing substitutions on certain characters. Its first invocation in the pipeline changes uppercase letters to lowercase. The second invocation removes all nonalphabetic characters by mapping them into newline character. The result is a file that contains the words of the document in the order they appear, with at most one word per line. The next program **sorts** the words into alphabetic order, and **unique** removes multiple occurrences. The result is a sorted list of the distinct words in the document. **common**, with the cryptic **-2** option, uses a standard merge algorithm to report all lines in its (sorted) input that are not in the (sorted) named file, and the output is the desired list of spelling errors.

This program was far from perfect, but it demonstrated the feasibility of a spelling checker and gained a loyal following of users. Changes to the program over the next several months were minor modifications to this structure — a complete redesign would wait for several years.

Doug McIlroy wrote another version of **SPELL** in 1978. Its user interface is the same as Johnson's: typing *spell filename* produces a list of the misspelled words in the file. The two advantages of this program over Johnson's are a superior word list and reduced run time. It fits in a 64-kilobyte address space and it can check a 5000 word English document in 30 seconds of VAX-11/750 CPU time.

McIlroy's program is the same as Johnson's up to the point of looking up words in the dictionary (the **common** program above). The new program loops on each word, stripping affixes and looking up the result until it either finds a match or no affixes remain (and the word is declared to be incorrect). Because affix processing may destroy the sorted order in which the words arrive, the dictionary is accessed in random order.

Today, numerous spelling programs for several natural languages are available on all kinds of computers. Computer users are increasingly getting used to utilize these programs. Although it is obvious that such a tool for Turkish users is also necessary and will be very useful, no such program has been developed until recently. It may be because of some features of Turkish that makes it different from many other languages, and causes some difficulties in development of a spelling checker for this language. Turkish language, together with its features that make the spelling checking problem for it especially hard and interesting, will be discussed in the following chapter. The research and implementation presented in the subsequent chapters have solved this problem with a very satisfactory performance.

Chapter 3

THE TURKISH LANGUAGE

3.1 History and Classification

Turkish is a member of the Turkic family of languages, which extends over a vast area in southern and eastern Siberia and adjacent portions of Iran, Afghanistan and China. The more widely spoken Central Asian Turkic languages include Karakalpak, Kırghız, Uygur and Uzbek. To the east, there is another group of Turkic languages north of the Altai mountains, and this group includes Yakut in eastern Siberia. To the west, Tatar is spoken in the Volga area and in the Urals, and there is a group of related languages north of the Caucasus. Chuvash, descended from the language of Huns, is also spoken in the Volga region.

Turkish, in turn belongs to the Altaic family of languages, which also includes Mongol and the Manchu-Tunguz languages of north-eastern Siberia. There are some typological and lexical similarities between Altaic and Uralic languages, which include Finnish, Estonian, Hungarian and a number of Siberian languages, notably Samoyed. These similarities are evidence for a Ural-Altaic language family.

The southwestern or Oğuz subgroup of Turkic family includes the languages Türkmen, Azerbaijani or Azeri, Ghashghai, Gagauz and Turkish. The one that particularly concerns us is the language of the Republic of Turkey, i.e., Turkish. Turkish is also spoken in small areas throughout the Balkans, notably in Greece, Bulgaria and Macedonia, and on Cyprus. There is a Turkish speaking population in northern Iraq, in the area of Kirkuk, and smaller groups, including Turkish speaking Armenians, throughout the Middle East, particularly in Syria and Lebanon.

The history of Turkish is divided into three periods. Old Anatolian Turkish (Eski Anadolu Türkçesi) includes the 13^{th} through 15^{th} centuries. Ottoman Turkish (Osmanlıca) includes the period of the Ottoman Empire. The transition from Ottoman to Modern Turkish (Yeni Türkçe) is mainly by the political events connected with the fall of the Ottoman Empire, and by the Language Reform movement of the late 1920's and 30's.

The most important characteristic of Ottoman which distinguishes it from Modern Turkish is the very heavy influence of Arabic and Persian, a consequence of Arabic and Persian influence on Turkish literature and culture during that period. Ottoman Turkish was written with Arabic script, used a higher proportion of Arabic and Persian words, particularly in literary or learned writing, and borrowed certain syntatic rules from Persian. The modern language reform movement is considered to date from 1928, when the Arabic script was replaced by a Latin ortography. The current Turkish alphabet consists of 29 letters which are in sequence A, B, C, Ç, D, E, F, G, Ğ, H, I, İ, J, K, L, M, N, O, Ö, P, R, S, Ş, T, U, Ü, V, Y, Z.

During the decade following the ortographic reform, and continuing until the present time, the Turkish Language Society (Türk Dil Kurumu) has supervised a steady program aimed at the reduction of the Arabic and Persian loanwords. Turkish replacements were taken from non-standard dialects or other Turkic languages, constructed with Turkish derivational suffixes, or simply invented. The Arabic and Persian component of the vocabulary has been by no means eliminated; the current vocabulary still contains Arabic and Persian words. It is significant that there has been little attempt to reduce the number of European loanwords. Some words of Greek and Italian origin are very old, while more recently many French and English words have accompanied the westernization of Turkey.

Turkish spoken in different regions of Turkey also shows some differences. Spoken Turkish is divided into some *dialects* each of which is spoken in a certain region of Turkey. One of these dialects, namely *İstanbul Türkçesi*, which is the Turkish spoken in İstanbul area, is chosen as the written language for Turkish. Written Turkish has certain standard rules, hence a word may show differences while speaking, but it is written in a standard way.

Languages can be morphologically classified into three groups according to word construction rules [36]:

1. **Isolating Languages:** No suffix exists. No word can change in the sentence. Intonation and word order carry the information (e.g., Chinese).
2. **Agglutinative Languages:** Words are combination of several morphemes and suffixes. There is a root and several suffixes are combined to this root in order to extend its meaning (e.g., Turkish, Hungarian).
3. **Inflected Languages:** During root-suffix combination the vowel changes in the root. This fact is also observed in plural form of words (e.g., Indo-European languages such as English).

In this classification Turkish belongs to *agglutinative languages*, which means that it expresses syntactic relations between words or concepts through discrete suffixes, each of which conveys a single idea. For an agglutinative language such as Turkish, the concept of word is much larger than the set of vocabulary items. Words can grow to be relatively long by addition of suffixes and sometimes contain an amount of semantic information equivalent to a complete sentence in another language. A popular example of complex Turkish word formation is

ÇEKOSLOVAKYALILAŞTIRAMADIKLARIMIZDANMIŞSINIZ

whose equivalent in English is "You had been one of those whom we could not convert to a Czechoslovakian." In this example, one word in Turkish corresponds to a full sentence of 14 words in English. The word above has the following decomposition into suffixes:

ÇEKOSLOVAKYA/LI/LAŞ/TIR/AMA/DIK/LAR/IMIZ/DAN/MIŞ/SINIZ

Each suffix has a certain function and modifies the semantic information in the stem preceding it. In the previous example, the root morpheme ÇEKOSLOVAKYA is the name of the country *Czechoslovakia* and the suffix –LI converts the meaning into *person from [Czechoslovakia]*, while the following suffix –LAŞ makes a verb from the previous stem meaning *to become one of [the persons from [Czechoslovakia]]*.[1]

[1]From now on, we will indicate the English meaning of a word in Turkish in parentheses following it.

3.2 Syllable Structure

The phonemes of a language are almost never pronounced standalone — a number of them come together to form syllables. Meaningful words can be formed by combining one or more of these syllables. In Turkish, there are syllables that consist merely of a single vowel:

$\underline{\text{O}}$ (he/she/it) $\underline{\text{A}}$-ÇIK (open) $\underline{\text{İ}}$-Kİ (two)

but usually more than one phoneme combine and form a syllable.

Each syllable in Turkish must contain a single vowel, hence a word has as many syllables as the number of vowels it has. There are no words consisting of a single vowel except the third person singular pronoun O (he/she/it)[2] [1].

Syllable types in Turkish words can be classified into two groups as *regular* and *irregular*. Words of 'Pure Turkish'[3] contain only regular syllables, while irregular syllables are commonly used in transcriptions of words of foreign origin.

3.2.1 Regular Syllables

The regular syllable types of Turkish are as follows[4] [2, 9, 36]:

V VC VCC CV CVC CVCC.

We can give the following 6 mono-syllable words as examples to these syllable types:

O AK (white) ÜST (top) SU (water) KOL (arm) KURT (wolf).

As seen above, in a regular syllable, there can be at most one consonant before a vowel and at most two consonants after it. This means that words of Pure Turkish can begin with at most one consonant and end with at most two consonants.

[2]In Turkish, there is no distinction of gender (masculine, feminine, neuter), and there are no distinct personal pronouns or corresponding possessive suffixes for different genders. So, while giving the English translations, we will use the male correspondings (*he* and *his*) instead of listing all the three possibilities, i.e., *he/she/it* or *his/her/its*.

[3]Öztürkçe

[4]V represents a vowel and C represents a consonant.

In words of Pure Turkish, there is at least one consonant between two consecutive vowels, i.e., a syllable ending with a vowel can not be directly followed by a syllable beginning with a vowel. Portmanteau words — words that are formed by combining multiple words — form an exception to this rule. When a word ending with a vowel is directly combined with a word beginning with a vowel, two vowels follow each other without an intervening consonant: e.g., AÇIORTAY (bisection), BİLGİİŞLEM (information processing), CEZAEVİ (prison).

There rarely appears more than one consonant at the end of Turkish words, and when they do, the first of these consonants is L, N, S, Ş, or R [9]: e.g., ALT (bottom), RENK (color), ÜST, AŞK (love), DERS (lecture).

Since a regular syllable may end with at most two consonants and begin with at most one consonant, there may occur at most three consonants between two consecutive vowels in words of Pure Turkish: e.g., ABARTMAK (to exaggerate), RENKSİZ (colorless), TÜRKÇE (Turkish), YURTTAŞ (citizen).

3.2.2 Irregular Syllables

All of the rules above model the syllables of a word in Pure Turkish. As mentioned in the previous section, Turkish has many words assimilated from various other languages. Although most of these words have been given new equivalents in Turkish, there are still many of them that are used in daily conversation and writing. Some syllables in such words of foreign origin conflict with the Turkish phonetic system. Such syllables are called *irregular syllables.*

The following irregular syllable structures are commonly used in transcriptions of words of foreign origin:

CVCCC CCV CCVC CCVCC CCVCCC CCCV CCCVC.

We can give the following examples for such syllables:

SÖ-MESTR (semester) GRA-FİK (graphic) SPOR (sports)
BRANŞ (occupation) SFENKS (sphinx) STRA-TE-Jİ (strategy)
STRİP-TİZ (strip-tease).

In pronounciation, one usually inserts a vowel between some of the consonants, but such vowels are not written.

Some of these syllable types occur mostly at the beginning or at the end of the words. For example CCCV and CCCVC type syllables mostly occur at the beginning, while CVCCC and CCVCCC type syllables occur at the end of the words. This means that words of foreign origin can begin and/or end with at most three consonants. The lists of those words that begin or end with three consonants are given in Table 3.1 and Table 3.2 respectively.

As mentioned before, in Pure Turkish two vowels can not follow each other without at least one intervening consonant, but there are words of foreign origin that do not obey this rule: e.g., $\underline{\text{AO}}$RT(aorta), $\underline{\text{İA}}$DE (return), R$\underline{\text{Eİ}}$S (head, chief), S$\underline{\text{AA}}$T (hour, watch, clock), İPTİD$\underline{\text{Aİ}}$ (primitive), ŞAŞ$\underline{\text{AA}}$ (splendour). There are also a small number of words, again of foreign origin, where three vowels follow each other (see Table 3.3).

Since an irregular syllable may begin or end with up to three consonants, in some words of foreign origin one can find four or five consonants between two vowels (see Tables 3.4 and 3.5).

In Table 3.6 you can find a comparison of words of Pure Turkish and words of foreign origin with respect to their syllable structures.

3.3 Morphophonemics

Turkish word formation uses a number of phonetic harmony rules. Vowels and consonants change in certain ways when a suffix is appended to a stem, so that such harmony constraints are not violated.

3.3.1 Vowel Harmony

The best known morphophonemic process in Turkish is the *vowel harmony*. Turkish has an eight-vowel system (A, E, I, İ, O, Ö, U, Ü), made up of all possible combinations of the distinctive features front/back, narrow/wide, and rounded/unrounded. The resulting system is schematically shown as a cube by Jean Deny [10] (see Figure 3.1). When the eight vowels are placed at the eight corners of the cube, the opposite faces represent the above three classifications. Through this cube we can understand the three classes that each vowel belongs to. For instance, **A** is a back, wide and unrounded vowel, where **Ü** is a front, narrow and rounded one.

SKRAYPER	STRATOSFER	STRİPTİZCİ
SPREY	STRATUS	STRONSİYUM
STRATEJİ	STREPTOMİSİN	STRÜKTÜRALİST
STRATEJİK	STRİKNİN	STRÜKTÜRALİZM
STRATİGRAFİ	STRİPTİZ	STRÜKTÜREL

Table 3.1: Words beginning with three consonants

KİLOHERTZ
ROPDÖŞAMBR
SFENKS
SÖMESTR

Table 3.2: Words ending with three consonants

GEOİT
MAAİLE
MÜDDEİUMUMİ
SUİİSTİMAL

Table 3.3: Words with three consecutive vowels

ABSTRE	ENSTRÜMAN	KONTRBAS
DEKSTRİN	ENSTRÜMANTAL	KONTRFİLE
EKSPRES	ENSTRÜMANTALİZM	OBSTRÜKSİYON
EKSPRESYONİZM	FOKSTROT	SANTRFOR
EKSTRA	GANGSTER	TRANSKRİPSİYON
EKSTRAFOR	HORNBLENT	TUNGSTEN

Table 3.4: Words with four consecutive consonants

KONTRPLAK
GOLFSTRİM

Table 3.5: Words with five consecutive consonants

Words of Pure Turkish	Words of foreign origin
begin with at most one consonant and end with at most two consonants	can begin and/or end with at most three consonants
contain at least one consonant between two consecutive vowels (except for the portmanteau words)	can contain at most three vowels with no intervening consonants
can contain at most three consonants between two consecutive vowels	can contain at most five consonants between two consecutive vowels

Table 3.6: Comparison of words of Pure Turkish and words of foreign origin with respect to their syllable structures

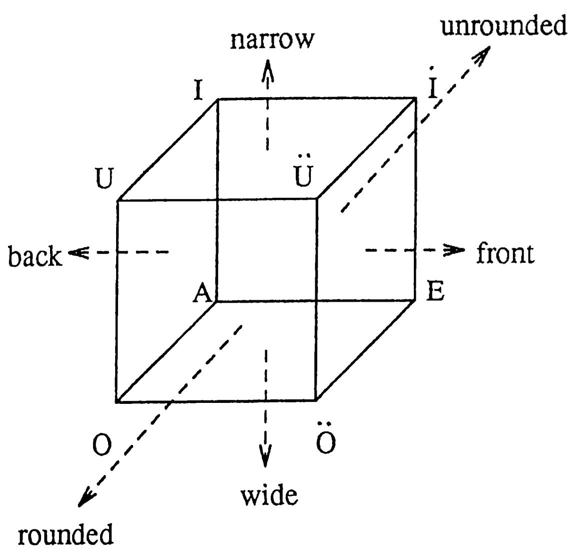

Figure 3.1: Vowel cube

Vowel harmony is a process by which the vowels in all syllables of a word except the first 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 [44]:

1. A non-initial vowel assimilates to the preceding vowel in frontness.
2. A non-initial narrow vowel assimilates to the preceding vowel in rounding.
3. A non-initial wide vowel must be unrounded; that is, O and Ö do not occur except in first syllables of the words.

Thus, while any of the eight vowels may occur in the first syllable of a word, the vowel of the following syllable is restricted to a choice of two. The features front/back and rounded/unrounded are entirely predictable, and only narrow/wide remains distinctive.

Since most of the loanwords do not obey to the vowel harmony rules, there exist many words whose vowels are not in harmony: e.g., İNAT (obstinance), ANTRE (entrance), EKONOMİ (economy), etc. Such words take suffixes conditioned by their last vowel: İNAT → İNATÇI (obstinate), ANTRE → ANTREDEN (from the entrance), EKONOMİ → EKONOMİMİZ (our economy). Thus, although some stems are not subject to vowel harmony internally, nearly all suffixes are in harmony with the vowel on their left.

Except the progressive tense suffix (-iyor), there are no suffixes in which the wide vowels O and Ö appear. Therefore, in citing suffixes, if we use the cover symbol {A} for a wide vowel and {I} for a narrow vowel, their allophones[5] will be as follows:[6]

$$\begin{array}{lcl} \{A\} & = & A \mid E \\ \{I\} & = & I \mid \dot{I} \mid U \mid \ddot{U}. \end{array}$$

Thus, the negation suffix can be shown as -M{A}, and the narrative past tense suffix as -M{I}Ş.

When a suffix is affixed to a stem, the first vowel in the suffix changes according to the last vowel of the stem. Succeeding vowels in the suffix change according to the vowel preceding it. If we denote the preceding vowel (be it in

[5]An allophone is any of the variant forms of a phoneme as conditioned by position or adjoining sounds.

[6]| indicates or.

the stem or in the suffix) by **V** then the two classes of vowels are resolved as follows:

$$\begin{array}{lcll}
\{\text{A}\} & = & \text{A}, & \text{if } \mathbf{V} \text{ is } \text{A} \mid \text{I} \mid \text{O} \mid \text{U} \\
 & = & \text{E}, & \text{if } \mathbf{V} \text{ is } \text{E} \mid \text{İ} \mid \text{Ö} \mid \text{Ü}.
\end{array}$$

$$\begin{array}{lcll}
\{\text{I}\} & = & \text{I}, & \text{if } \mathbf{V} \text{ is } \text{A} \mid \text{I} \\
 & = & \text{İ}, & \text{if } \mathbf{V} \text{ is } \text{E} \mid \text{İ} \\
 & = & \text{U}, & \text{if } \mathbf{V} \text{ is } \text{O} \mid \text{U} \\
 & = & \text{Ü}, & \text{if } \mathbf{V} \text{ is } \text{Ö} \mid \text{Ü}.
\end{array}$$

An allomorph is any of the variant forms of a morpheme. For example, the negation suffix –M{A} has two allomorphs, where narrative past tense suffix –M{I}Ş has four:

$$\begin{array}{lcl}
\text{–M}\{\text{A}\} & = & \text{–MA} \mid \text{–ME} \\
\text{–M}\{\text{I}\}\text{Ş} & = & \text{–MIŞ} \mid \text{–MİŞ} \mid \text{–MUŞ} \mid \text{–MÜŞ}.
\end{array}$$

The allomorph of a suffix that is to be used is determined according to the phonemes of the stem it is affixed. For example, when the suffix –M{A} is affixed to the root GÖR(MEK) ((to) see), the allomorph –ME is used, because as the vowel preceding the vowel {A} is **Ö** (V = Ö), {A} must resolve to an **E** ({A} = E):

$$\text{GÖR} \;+\; \text{M}\{\text{A}\} \;\rightarrow\; \text{GÖR}\underline{\text{ME}} \text{ (do not see).}$$

Similarly, the suffix –M{I}Ş takes the form –MÜŞ when it is affixed to the root GÖR(MEK):

$$\text{GÖR} \;+\; \text{M}\{\text{I}\}\text{Ş} \;\rightarrow\; \text{GÖR}\underline{\text{MÜŞ}} \text{ (he had seen).}$$

There are also some non-harmonic suffixes, such as –KEN and –{I}YOR, which are exceptions to harmonic conditioning from the vowel on their left: $\text{OKUR}\underline{\text{KEN}}$ (while reading), $\text{GEL}\underline{\text{İYOR}}$ (he is coming). Similarly, the second vowel in compound verbs (e.g., –Y{I}VER, –Y{A}BİL, –Y{A}DUR) do not change according to the preceding vowel either: $\text{OKU}\underline{\text{YABİL}}$ (can read), $\text{OKU}\underline{\text{YUVER}}$ (just read), $\text{GİYİN}\underline{\text{EDUR}}$ (go on dressing). Such suffixes condition the vowel on their right normally: $\text{GEL}\underline{\text{İYOR}}\text{UM}$ (I am coming), $\text{OKU}\underline{\text{YABİL}}\text{İR}$ (he can read).

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

SAAT + [Y]{A} → not SAATA but SAAT$\underline{\text{E}}$
ALKOL + L{I} → not ALKOLLU but ALKOL$\underline{\text{LÜ}}$.

When certain suffixes beginning with a consonant are affixed to the stems ending with a consonant, a narrow vowel is inserted between them.[7] This vowel is also determined similarly as explained before. For example the first person plural possessive suffix -[{I}]M{I}Z has eight different allomorphs:

-[{I}]M{I}Z = -IMIZ | -İMİZ | -UMUZ | -ÜMÜZ
= -MIZ | -MİZ | -MUZ | -MÜZ.

When this suffix is affixed to the root KAPI (door), it takes the form -MIZ. But when it is affixed to the root OKUL (school), the allomorph -UMUZ is used.

3.3.2 Consonant Harmony

Another basic aspect of Turkish phonology is *consonant harmony.* In one respect, consonants in Turkish may be divided into two groups as *harsh* (Ç, F, T, H, S, K, P, Ş)[8] and *soft* consonants (B, C, D, G, Ğ, J, L, M, N, R, V, Y, Z). Most of the consonant harmony rules listed below are based on this classification [5, 23]:

1. Turkish words mostly end with a harsh consonant; especially, the soft consonants B, C, D, or G are rarely found as the final phonemes of the originally Turkish words. If there is one of these consonants at the end of a foreign word, it changes to a corresponding harsh sound of P, Ç, T, or K respectively: e.g., KİTA$\underline{\text{B}}$ → KİTA$\underline{\text{P}}$ (book), İLA$\underline{\text{C}}$ → İLA$\underline{\text{Ç}}$ (medicine).

2. In multi-syllabic words and in certain mono-syllabic roots, the final harsh consonants P, Ç, T, K are mostly (not always) softened (i.e., it changes to B, C, D, or Ğ respectively) when a suffix beginning with a vowel is attached: e.g., AKOR$\underline{\text{T}}$ (tune) → AKOR$\underline{\text{D}}$U (its tune) but AOR$\underline{\text{T}}$ (aorta) → AOR$\underline{\text{T}}$U (his aorta).

3. 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 stem

[7]We will show such vowels as [{I}].

[8]An easy way to remember these consonants is through the famous mnemonic **ÇİFTE HASEKİ PAŞA**.

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.

If the last phoneme of the stem to which one of such suffixes is attached is a harsh consonant, the initial consonant of the suffix becomes harsh (Ç, T, or K respectively), otherwise it remains as C, D, or G. Thus, the allomorphs of the definite past tense suffix –{D}{I} can be listed as:

–{D}{I}	=	–DI	|	–Dİ	|	–DU	|	–DÜ
	=	–TI	|	–Tİ	|	–TU	|	–TÜ.

When this suffix is affixed to the root GEL(MEK) ((to) come), it takes the form –Dİ, and when it is affixed to the root KOŞ(MAK) ((to) run), the allomorph –TU is used:

GEL + {D}{I} $\rightarrow$ GEL$\underline{\text{Dİ}}$ (he came)
KOŞ + {D}{I} $\rightarrow$ KOŞ$\underline{\text{TU}}$ (he ran).

Furthermore some morphemes beginning with a vowel are affixed to the stems ending with a vowel with the insertion of one of the consonants N, S, Ş, or Y.[9] For example, the genitive suffix can be shown as –[N]{I}N, the third person singular possessive suffix as –[S]{I}, distributive numerical suffix as –[Ş]{A}R, and the acceleration suffix as –[Y]{I}VER. The allomorphs of these suffixes are as follows:

–[N]{I}N	=	–NIN	|	–NİN	|	–NUN	|	–NÜN
	=	–IN	|	–İN	|	–UN	|	–ÜN
–[S]{I}	=	–SI	|	–Sİ	|	–SU	|	–SÜ
	=	–I	|	–İ	|	–U	|	–Ü
–[Ş]{A}R	=	–ŞAR	|	–ŞER	|	–AR	|	–ER
–[Y]{I}VER	=	–YIVER	|	–YİVER	|	–YUVER	|	–YÜVER
	=	–IVER	|	–İVER	|	–UVER	|	–ÜVER.

As an example, the suffix –[S]{I} takes the form –İ when it is affixed to the

[9]We will show such consonants as [N], [S], [Ş], and [Y] respectively.

root EV (house), but the allomorph –SI is used when it is affixed to the root KAPI (door):

EV + [S]{I} → EVİ (his house)
KAPI + [S]{I} → KAPISI (his door).

There may be some exceptions to such 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 [23]:

SANAYİ + [S]{I} → not SANAYİSİ but SANAYİİ.

For some such words both forms are valid:

CAMİ + [S]{I} → either CAMİSİ or CAMİİ.

A similar case happens when a case suffix comes immediately after some pronouns such as BU (this), ŞU (that), O (it), KENDİ (self), after the pronomial suffix –Kİ, or after the third person possessive suffixes –[S]{I} or –L{A}R{I}. In such cases an N is inserted in between:

BU	+	[Y]{I}	→	not	BUYU	but	BUNU
KENDİ	+	{D}{A}N	→	not	KENDİDEN	but	KENDİNDEN
SENİNKİ	+	[Y]{A}	→	not	SENİNKİYE	but	SENİNKİNE
KAPISI	+	{D}{A}	→	not	KAPISIDA	but	KAPISINDA.

When all the rules above are considered, we reach the result that Turkish suffixes tend to have a highly protean nature. As an extreme example, the participial suffix –{D}{I}{K}[10] has 16 allomorphs:

-{D}{I}{K}	=	-DIK	-DİK	-DUK	-DÜK
	=	-TIK	-TİK	-TUK	-TÜK
	=	-DIĞ	-DİĞ	-DUĞ	-DÜĞ
	=	-TIĞ	-TİĞ	-TUĞ	-TÜĞ.

In the word SATTIĞIN ([the thing] that you sell) that suffix takes the form –TIĞ, because it follows the root SAT(MAK) ((to) sell) which ends with the harsh consonant T (i.e., {D} = T) and whose last vowel is A (V = A → {I} = I), and it is followed by a suffix beginning with a vowel (i.e., {K} = Ğ).

[10]The allophones of {K} are K and Ğ.

3.3.3 Root Deformations

Normally Turkish roots are not flexed. However, there are some cases where some phonemes are changed by assimilation or various other deformations [23]. 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	+	[Y]{A}	→	not	BENE	but	BANA (to me)
SEN	+	[Y]{A}	→	not	SENE	but	SANA (to you).

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

BEN	+	L{A}R	→	not	BENLER	but	BİZ (we)
SEN	+	L{A}R	→	not	SENLER	but	SİZ (you).

These are individual cases and can be treated as exceptions.

A more systematic change occurs when the suffix -[{I}]YOR comes after the verbs ending with the wide vowel {A}. In such cases, the wide vowel at the end of the stem is narrowed:

KAPA	+	[{I}]YOR	→	not	KAPAYOR	but	KAPIYOR.

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(MEK) ((to) say) or YE(MEK) ((to) eat), they change as Dİ and Yİ respectively:

DE	+	[{I}]YOR	→	not	DEYOR	but	DİYOR
DE	+	[Y]{A}N	→	not	DEYEN	but	DİYEN [11]
YE	+	[Y]{I}P	→	not	YEYİP	but	YİYİP.

One of the most important deformations in roots and stems occur as the result of the second consonant harmony rule. This rule says that when some words ending with one of the harsh consonants P, Ç, T, K take a suffix beginning with a vowel, that consonant changes into B, C, D, or Ğ respectively:

DÖRT	+	{I}N{I}Z	→	not	DÖRTÜNÜZ	but	DÖRDÜNÜZ
TABAK	+	[{I}]M	→	not	TABAKIM	but	TABAĞIM.

If an N precedes a final K, the consonant K either stays as it is or it changes

[11]The verb DEMEK sometimes shows exception to this exception either. For example: DE + [Y]{I}P →not DİYİP but DEYİP.

into a G:

TANK + [Y]{A} → TANKA
RENK + [Y]{A} → not RENKE but RENGE.

A similar change occurs when a suffix beginning with a vowel is affixed to a word ending with -LOG. In such a case, the final G changes into Ğ:

PSİKOLOG + [Y]{A} → not PSİKOLOGA but PSİKOLOĞA.

Another root deformation occurs as a vowel ellipsis. When a suffix beginning with a vowel comes after some nouns, generally designating parts of the human body, which has a vowel {I} in its last syllable, this vowel drops:

AĞIZ + [{I}]M{I}Z → not AĞIZIMIZ but AĞZIMIZ.

Similarly, when the passiveness suffix -{I}L is affixed to some verbs, whose last vowel is {I}, this vowel also drops:

AYIR + {I}L → not AYIRIL but AYRIL.

When a noun which has to face with vowel ellipsis receives the first person singular or plural suffixes, i.e., -[Y]{I}M or -[Y]{I}Z, although these suffixes begin with vowel, the last vowel of the root does not drop:

OĞUL + [Y]{I}Z → not OĞLUZ but OĞULUZ.

When a suffix beginning with a vowel is affixed to some originally Arabic roots ending with a consonant, or when such a root is combined with another word beginning with a vowel, the final consonant of the root is duplicated:

HAK + [{I}]M → not HAKIM but HAKKIM
ZAN + ETMEK → not ZANETMEK but ZANNETMEK.

When the plural suffix -L{A}R is affixed to the portmanteau words which were originally indefinite compounds, a deformation occurs. This suffix, coming before the possessive suffix at the end of the stem, forms a 'mid'fixing:

GÖZYAŞI + L{A}R → not GÖZYAŞILAR but GÖZYAŞLARI.

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

KAYIT + {I}N → neither KAYITIN nor KAYTIN but KAYDIN

ZIT + [Y]{I} → neither ZITI nor ZITTI but ZIDDI

RENGEYİĞİ + L{A}R → neither RENGEYİĞİLER
nor RENGEYİĞLERİ
but RENGEYİKLERİ

ADEMOĞLU + L{A}R → neither ADEMOĞLULAR
nor ADEMOĞLLARI
but ADEMOĞULLARI.

3.4 Morphology

Turkish roots can be classified into two main classes: *nominal* and *verbal*. The verbal class comprises the verbs (GÖR(MEK), KOŞ(MAK), SAT(MAK), etc.), while nominal class comprises nouns, pronouns, adjectives (KAPI, BİZ, DÖRT, etc.), and adverbs (AZ, DÜN, YOK, etc.). The suffixes that can be received by either of these groups are different, i.e., a suffix which can be affixed to a nominal root can not be affixed to a verbal root with the same semantic function. There exist some roots which never take suffixes. Some interjections, conjunctions and postpositions can be given as examples to such roots (HEY (hi!), VE (and), RAĞMEN (despite), etc.).[12] There are also some roots which can take all the suffixes either the nouns or the verbs can take (TAT (taste), DİK (set up, sew, vertical), ŞİŞ (swell, swelling), etc.).

Turkish suffixes can be classified as *derivational* and *conjugational*. Derivational suffixes change the meaning and sometimes the class of the stems they are affixed to, while a conjugated verb or noun remains as such after the affixation. Conjugational suffixes can be affixed to all of the roots in the class that they belong to. On the other hand, the number of roots each derivational suffix can be affixed to differs. For example, -ZİR can only be used with the verb EMMEK (to suck), i.e., EMMEK→ EMZİRMEK (to nurse).

Conjugational suffixes may be divided into two groups according to the root class that they can be affixed to, i.e., a *noun* paradigm and a *verb* paradigm.

[12]Some adverbs, such as HEMEN (immediately), GALİBA (perhaps), never take suffixes either.

nominal root	plural suffix	possessive suffix	case suffix	relative suffix

plural suffix	-L{A}R	
possessive suffixes	-[{I}]M	-[{I}]M{I}Z
	-[{I}]N	-[{I}]N{I}Z
	-[S]{I}	-L{A}R{I}
case suffixes	internal	external
	-[Y]{I}	-[Y]L{A}
	-[Y]{A}	-{C}{A}
	-{D}{A}	-L{I}
	-{D}{A}N	-S{I}Z
	-[N]{I}N	
relative suffix	-Kİ	

Figure 3.2: The nominal model

3.4.1 Noun Paradigm

The elements of the noun paradigm, in order, can be shown as in Figure 3.2 [1, 35, 36, 44]. All of these elements (except the root) are optional.

The plural suffix -L{A}R is added directly to the nominal root before any other suffix or ending. In the plural forms of the pronouns BU, ŞU, O an N is inserted between the word and the suffix:

BU	+	L{A}R	→	not	BULAR	but	BUNLAR (these)
ŞU	+	L{A}R	→	not	ŞULAR	but	ŞUNLAR (those)
O	+	L{A}R	→	not	OLAR	but	ONLAR (they).

Possesive pronouns (in English: my, your, his/her/its, our, your, their) are represented by suffixes in Turkish: e.g., EVİM (my house), ARABAN (your car). If the possessed noun is plural, possessive suffixes come after the plural suffix: e.g., EVLERİM (my houses), ARABALARIN (your cars). When the third person plural possessive suffix -L{A}R{I} comes after a plural noun, two L{A}R's combine and one of them drops:

EV + L{A}R + L{A}R{I} → not EVLERLERİ but EVLERİ.

In the above example, the word EVLERİ means *their houses* because LERİ is the combination of the plural suffix –L{A}R and the third person plural possessive suffix –L{A}R{I}. The same word can be used in other meanings since it can be formed by different suffixes as:

EV	+	L{A}R{I}			→	EVLERİ (their house)
EV	+	L{A}R	+	[S]{I}	→	EVLERİ (his houses).

Portmanteau words which were originally indefinite compounds have the third person singular possessive suffix already in their structure: e.g., ATEŞBÖCEĞİ (fire-fly), SAFRAKESESİ (gall bladder). Such words receive the possessive suffixes after removing the possessive suffix which is already in their structure:

ATEŞBÖCEĞİM
ATEŞBÖCEĞİN
ATEŞBÖCEĞİ (not ATEŞBÖCEĞİSİ)
ATEŞBÖCEĞİMİZ
ATEŞBÖCEĞİNİZ
ATEŞBÖCEKLERİ (not ATEŞBÖCEĞİLERİ),

SAFRAKESEM (not SAFRAKESESİM)
SAFRAKESEN (not SAFRAKESESİN)
SAFRAKESESİ (not SAFRAKESESİSİ)
SAFRAKESEMİZ (not SAFRAKESESİMİZ)
SAFRAKESENİZ (not SAFRAKESESİNİZ)
SAFRAKESELERİ (not SAFRAKESESİLERİ).

The nominal roots SU (water) and NE[13] (what) create some irregular cases when they receive possessive suffixes [1]:

SUYUM (not SUM)	NEYİM
SUYUN (not SUN)	NEYİN
SUYU (not SUSU)	NEYİ
SUYUMUZ (not SUMUZ)	NEYİMİZ
SUYUNUZ (not SUNUZ)	NEYİNİZ
SULARI	NELERİ.

[13]The regular forms for the root NE are also valid: NEM, NEN, NESİ, NEMİZ, NENİZ, NELERİ.

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:

-[Y]{I}	accusative
-[Y]{A}	dative
-{D}{A}	locative
-{D}{A}N	ablative
-[N]{I}N	genitive.

Declensions of pronouns have some irregular forms. In the dative cases of BEN and SEN, the front vowels become back (see page 29). In the genitive cases of BEN and BİZ, -İM is used instead of the regular form -İN:

BEN	+	[N]{I}N	→	not	BENİN	but	BENİM (my)
BİZ	+	[N]{I}N	→	not	BİZİN	but	BİZİM (our).

Additionally, as mentioned on page 28, 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 portmanteau words having the characteristics mentioned above:

ATEŞBÖCEĞİ	+	[Y]{A}	→	not	ATEŞBÖCEĞİYE
				but	ATEŞBÖCEĞİNE
SAFRAKESESİ	+	{D}{A}	→	not	SAFRAKESESİDE
				but	SAFRAKESESİNDE.

The nominal 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	+	[N]{I}N	→	not	SUNUN	but	SUYUN
NE	+	[N]{I}N	→	not	NENİN	but	NEYİN.

The relative suffix -Kİ may be added only to genitive or locative suffixes. When it is affixed to a noun in genitive case, it forms a possessive pronoun: e.g., KAPININKİ (the door's), BİZİMKİ (ours). When it is affixed to a noun in locative case, it makes an adjective determining the location of the thing under question: e.g., KAPIDAKİ ([the one] that is at the door), BİZDEKİ ([the one] which is in our [hand, home]).

It is possible to affix the relative suffix directly to a temporal adverb or a noun adverbially used (e.g., DEMİNKİ (of a while ago), YARINKİ (tomorrow's)), or to a directional adverb or an adverb of place (e.g., KARŞIKİ ([the one] on the opposite side), AŞAĞIKİ (the lower one)). The number of such roots are quite limited.

In general, the relative suffix is not subject to vowel harmony. However, in the following cases -Kİ changes into -KÜ: DÜNKÜ (yesterday's), BUGÜNKÜ (today's), [O] GÜNKÜ ([that] day's), ÖBÜRKÜ (the other one).

A noun stem that received the relative suffix may take the plural suffix and any case ending: e.g., BURADAKİLER (those who are here), BURADAKİLERLE (with those who are here). In its singular form an N is put between -Kİ and the case-ending: e.g. BURADAKİNDEN (from the one who is here).

3.4.2 Verb Paradigm

The verb paradigm is more complex than the noun paradigm. Its elements, in order, are shown in Figure 3.3. Among these elements, the obligatory ones are the root, the main tense suffix, and the person suffix.

There are four voices of verbs in Turkish: reflexive, reciprocal, causative, and passive. Combination of these suffixes are possible, but they must appear in the indicated order, and the reflexive and reciprocal are mutually exclusive: e.g. GÖRMEK (to see) → GÖRÜŞMEK (to see each other) → GÖRÜŞTÜRMEK (to cause to see each other) → GÖRÜŞTÜRÜLMEK (to be caused to see each other).

Neither the reflexive nor the reciprocal can be affixed to all verb roots; thus, they can be considered as derivational suffixes: DÖVMEK (to beat) → DÖVÜNMEK (to beat oneself), but not KOŞMAK → KOŞUNMAK. ANLAMAK (to understand) → ANLAŞMAK (to understand one another), but not OKUMAK (to read) → OKUŞMAK.

The factitive voice of verbs takes various forms as follows [23]:

verbal root	voice suffixes	negation suffix	compound verb s.	main tense s.	question suffix	second tense s.	person suffix

voice suffixes	<u>reflexive</u>	<u>reciprocal</u>	<u>factitive</u>	<u>passive</u>
	-[{I}]N	-[{I}]Ş	-{D}{I}R	-{I}L
			-{I}T	-{I}N
			-T	-N
			-{I}R	
			-{A}R	
negation suffixes	-M{A}	-[Y]{A}M{A}		
compound verb suffixes	-[Y]{A}BİL	-[Y]{A}YAZ		
	-[Y]{A}DUR	-[Y]{A}KAL		
	-[Y]{I}VER	-[Y]{A}KOY		
	-[Y]{A}GEL	-[Y]{A}GÖR		
main tense suffixes	-{D}{I}	-S{A}		
	-M{I}Ş	-[Y]{A}		
	-[Y]{A}C{A}{K}	-M{A}L{I}		
	-[{I}]R	-Φ		
	-{A}R			
	-[{I}]YOR			
	-M{A}KT{A}			
question suffix	-M{I}			
second tense suffixes	-[Y]{D}{I}	-[Y]S{A}		
	-[Y]M{I}Ş			
person suffixes	-M	-[Y]{I}M	-[Y]{I}N	
	-N	-S{I}N	-[Y]{I}N{I}Z	
	-Φ	-[Y]{I}Z	-S{I}NL{A}R	
	-K	-S{I}N{I}Z		
	-N{I}Z	-L{I}M		
	-L{A}R			

Figure 3.3: The verbal model

Allomorph	Verb Stems Accepting The Allomorph	Examples
-{A}R	ÇIK, ÇÖK,[14] GİT, KOP, ON	ÇIKAR, GİDER
-{I}R	AŞ, BAT, BİT, DOĞ, DOY, DUY, DÜŞ, GEÇ, GÖÇ, İÇ, KAÇ, PİŞ, ŞİŞ, TAŞ, YAT, YİT	AŞIR, DOYUR, DÜŞÜR, YİTİR
-{I}T	AK, ÇARP,[15] KOK, KORK, SAP,[16] SARK, ÜRK	AKIT, KORKUT, ÜRKÜT
-T	all polysyllabic stems ending with a vowel or one of the consonants L or R	AĞLAT, YÖNELT, AĞART
-{D}{I}R	all other stems	YEDİR, GÖRÜŞTÜR

Table 3.7: Usage of allomorphs of the factitive verb suffix

-{D}{I}R = -DIR | -DİR | -DUR | -DÜR
= -TIR | -TİR | -TUR | -TÜR

-{I}T = -IT | -İT | -UT | -ÜT

-T = -T

-{I}R = -IR | -İR | -UR | -ÜR

-{A}R = -AR | -ER.

Although five different groups of these 19 different forms do not present any relationship at first sight, a close examination shows clearly that they are allomorphs. The set of rules applied to determine which allomorph is to be chosen for a given verb is given in Table 3.7 [23]. The irregular factitive forms are GEL (come) → GETİR (bring), GÖR (see) → GÖSTER (show), KALK (stand up) → KALDIR (lift), EM → EMZİR.

Allomorph	Verb Stems Accepting The Allomorph	Examples
-{I}L	all stems ending with a consonant other than L	SEVİL, YAPTIRIL
-{I}N	all stems ending with the consonant L	BULUN, BİLİN
-N	all stems ending with a vowel	DEN, BAĞLAN

Table 3.8: Usage of allomorphs of the passive voice verb suffix

The factitive verb suffixes can be used repeatedly:

İÇ	→	İÇİR	→	İÇİRT		
AŞ	→	AŞIR	→	AŞIRT	→	AŞIRTTIR
YAP	→	YAPTIR	→	YAPTIRT		
YÜRÜ	→	YÜRÜT	→	YÜRÜTTÜR		
KAPA	→	KAPAT	→	KAPATTIR	→	KAPATTIRT

The passive voice verb suffix also takes different forms as:

-{I}L = -IL | -İL | -UL | -ÜL

-{I}N = -IN | -İN | -UN | -ÜN

-N = -N.

The allomorph to be chosen for a given verb is determined by the set of rules listed in Table 3.8 [23]. Passive voice is also applied to impersonal use in Turkish. Thus, double passive structure may be used for a passive and impersonal verb. This is in fact superfluous. SÖYLENDİ (it was said) is clear enough but SÖYLENİLDİ may also be used.

The passive and reflexive forms of some verbs have the same structure, but

[14]The form ÇÖKTÜR is also sometimes used.
[15]The form ÇARPTIR is also sometimes used.
[16]The form SAPTIR is also sometimes used.

they differ in their meanings. For example, the verb YIKANMAK 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: -M{A} (not) and -[Y]{A}M{A} (can not). The suffix -[Y]{A}M{A} is used to express impossibility: SÖYLEMEM (I don't say), SÖYLEYEMEM (I can't say).

Compound verb suffixes can be affixed to verbs to add them some extra meanings. Among them the potentiality and possibility suffix -[Y]{A}BİL is the most frequently used one. It is used to express a physical or mental ability or capability (e.g., YAZABİLİR (he is able to write)), or permission or possibility (e.g., GİDEBİLİRSİN (you may go)). The acceleration suffix -[Y]{I}VER is the next frequently used one. It is used to express acceleration or quickness in an action (e.g., GELİVERDİ (he just came)), or to request someone to do something (e.g., AÇIVER (please open)). To indicate continuance in an action the continuance suffixes -[Y]{A}DUR, -[Y]{A}KOY, or -[Y]{A}KAL are used: e.g., YAZADUR (go on writing), YIKAYAKOY (go on washing), BAKAKALDI (he continued to look). The approximation suffix -[Y]{A}YAZ is rarely used. It indicates approximation to a state or situation: DÜŞEYAZDIM (I almost fell). More than one compound verb suffix may be added to a verb: SÖYLEYİVEREBİLİR MİSİN? (Could you please say?), YAZADURUVER (please go on writing).

Some compound verb suffixes can not follow negation suffixes. For example, except the potentiality suffix -[Y]{A}BİL none of them can be used after the impossibility suffix -[Y]{A}M{A} (e.g., YAZ**AMA**YABİLİR (he may not be able to write)). Similarly, the suffixes -[Y]{A}KOY, -[Y]{A}KAL, and -[Y]{A}YAZ are not used after negation suffixes.

Main tense suffix is one of the obligatory suffixes for the verbs. There are nine tenses: definite past (-{D}{I}), narrative past (-M{I}Ş), future (-[Y]{A}CA{K}), aorist (-{I}R, -{A}R, -R), progressive (-{I}YOR, -M{A}K-T{A}), conditional (-S{A}), optative (-[Y]{A}), necessitative (-M{A}L{I}), and imperative (-Φ). The last four are not tenses in the strict sense of the term, but their place in the verb model is the same as main tense suffixes.

In the tense system, the contrast between the definite past and the narrative past is of particular interest. The definite past is used to describe events which the speaker has personally witnessed, while the narrative past is used for actions about which the speaker knows through report or inference.

Allomorph	Verb Stems Accepting The Allomorph	Examples
{A}R	all mono-syllabic roots ending with a consonant except the following ones:[17]AL, BİL, BUL, DUR, GEL, GÖR, KAL, OL, ÖL, SAN, VAR, VER, VUR, and the compound verbs formed with the verb ETMEK	YAP<u>AR</u>, SEV<u>ER</u>, HİSSED<u>ER</u>, ZANNED<u>ER</u>
{I}R	all multi-syllabic stems ending with a consonant except the compound verbs formed with the verb ETMEK, and the mono-syllabic roots listed above	KAYBOL<u>UR</u>, SÜRÜL<u>ÜR</u>, AL<u>IR</u>, VER<u>İR</u>
R	all stems ending with a vowel	YE<u>R</u>, OYNA<u>R</u>

Table 3.9: Usage of allomorphs of the aorist suffix

As factitive and passive voice suffixes, the aorist suffix also changes according to some specific rules, which are listed in Table 3.9. 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:

VERMEM	(I don't give)
VERME<u>Z</u>SİN	(you don't give)
VERME<u>Z</u>	(he doesn't give)
VERMEYİZ	(we don't give)
VERME<u>Z</u>SİNİZ	(you don't give)
VERME<u>Z</u>LER	(they don't give).

The progressive tense suffix -[{I}]YOR causes a deformation on some stems it is affixed to (see page 29). The same deformation occurs in the negation suffix when it is followed by the suffix -[{I}]YOR:

$$\begin{array}{lllll} \text{SEV} + \text{M\{A\}} + \text{-[\{I\}]YOR} & \rightarrow & \text{not} & \text{SEVMEYOR} \\ & & \text{but} & \text{SEV}\underline{\text{Mİ}}\text{YOR.} \end{array}$$

The suffix -M{A}KT{A} 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

[17]This list of 13 exceptions is given by Lewis [25], page 116.

imperative form is understood through its person suffix. Every verb stẽm can be considered as in the second person singular imperative form (for positive orders positive stems, for negative orders negative ones): e.g., GEL! (Come!), KAPATMA! (Don't close!).

The question suffix –M{I} 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, conditional, or optative: e.g., GELDİN Mİ? (Did you come?), GELSEM Mİ? (Should I come?), GELSİN Mİ? (Do you want him to come?). For the remaining tenses, the place of the question suffix is between the main tense suffix and the person suffix: e.g., GELİR MİYİZ? (Do we come?), GELECEK MİSİN? (Will you come?). No matter in which tense the verb is, the question suffix comes after the third person plural suffix: GELMELİLER Mİ? (Must they come?), GELİYORLAR MI? (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 İ from the definite past, narrative past, and conditional forms of the verb İMEK, i.e., İDİ, İMİŞ, İSE: e.g., GELİYORDUM (I was coming), GELİRMİŞSİN ([I am told that] you come), GELECEKSEK (if we will 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: i.e., GELİYOR İDİM, GELİR İMİŞSİN, GELECEK İSEK. The second tense suffixes are affixed to verb stems ending with a vowel with the insertion of a Y in between: e.g., GELSEYDİ (if he came), GELEYMİŞ (I wish he had come), GELMELİYSE (if he 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: e.g., GELDİNDİ (you had come) or GELDİYDİN. GELDİKSE (if we came) or GELDİYSEK. In the third person plural, the first form is more frequently used than the second: e.g., GELDİLERDİ (they had come), GELDİLERSE (if they came). Similarly, no matter what the main tense suffix is, the third person plural suffix can be used either before or after all the second tense suffixes: e.g., GELİYORDULAR and GELİYORLARDI, GELMELİYMİŞLER and GELMELİLERMİŞ, and GELDİYSELER and GELDİLERSE are all valid.

None of the second tense suffixes can be used with the imperative suffix.

Additionally, the narrative second tense suffix can not be used with definite past tense suffix, and the conditional second tense suffix can not come after the optative and the conditional tense suffixes: i.e., OKUYDU, OKUDUYMUŞ, OKUSAYSA are not valid.

The last obligatory suffix for verbs is 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. For example, the second person plural suffix has 24 allomorphs which can be grouped in 4 as follows:

-N{I}Z	=	-NIZ	-NİZ	-NUZ	-NÜZ
-S{I}N{I}Z	=	-SINIZ	-SİNİZ	-SUNUZ	-SÜNÜZ
-[Y]{I}N	=	-YIN	-YİN	-YUN	-YÜN
	=	-IN	-İN	-UN	-ÜN
-[Y]{I}N{I}Z	=	-YINIZ	-YİNİZ	-YUNUZ	-YÜNÜZ
	=	-INIZ	-İNİZ	-UNUZ	-ÜNÜZ.

We can say that there are four different conjugations of person suffixes as shown in Table 3.10 [1]:

GELDİM	GELMELİYİM	GELEYİM	
GELDİN	GELMELİSİN	GELESİN	GEL
GELDİ	GELMELİ	GELE	GELSİN
GELDİK	GELMELİYİZ	GELELİM	
GELDİNİZ	GELMELİSİNİZ	GELESİNİZ	GELİNİZ
GELDİLER	GELMELİLER	GELELER	GELSİNLER

Different person suffixes may have the same form. For example, the suffix -S{I}N may be the second person singular suffix (see second and third rows of Table 3.10), or the third person singular suffix (see last row of Table 3.10).

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: GELDİ (he came), GELİRSE (if he 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 -S{I}N is used for the third singular person: e.g., GEL! ([You] come!), GELSİN! (Let him come!). Additionally, the imperative forms of the

Main Tense Suffix	Second Tense Suffix	Person Suffix					
		1st p.s.	2nd p.s.	3rd p.s.	1st p.p.	2nd p.p.	3rd p.p.
-{D}{I} -S{A}	-[Y]{D}{I} -[Y]S{A}	-M	-N	-Φ	-K	-N{I}Z	-L{A}R
-M{I}Ş -[Y]{A}CA{K} -[{I}]R -{A}R -{I}YOR -M{A}KT{A} -M{A}L{I}	-[Y]M{I}Ş	-[Y]{I}M	-S{I}N	-Φ	-[Y]{I}Z	-S{I}N{I}Z	-L{A}R
-[Y]{A}		-[Y]{I}M	-S{I}N	-Φ	-L{I}M	-S{I}N{I}Z	-L{A}R
-Φ			-Φ	-S{I}N		-[Y]{I}N -[Y]{I}NIZ	-S{I}NL{A}R

Table 3.10: Conjugation of person suffixes

first singular and plural persons are not present.

3.4.3 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 3.4. The only obligatory suffix in this paradigm is the person suffix.

Negation concept show differences in noun and verb sentences. In a verb sentence, it is obtained by adding a negation suffix to the verb of the sentence (see page 39): e.g., *Okula gitmedim.* (I didn't go to the school.). There is no such a suffix for the verbal noun of a noun sentence. Instead, the word DEĞİL is used for this purpose: *Öğrenci değilim.* (I am not a student).

nominal stem	question suffix	tense suffix	person suffix	probability suffix

question suffix	-M{I}	
tense suffixes	-[Y]{D}{I}	
	-[Y]M{I}Ş	
	-[Y]S{A}	
person suffixes	-M	-[Y]{I}M
	-N	-S{I}N
	-Φ	
	-K	-[Y]{I}Z
	-N{I}Z	-S{I}N{I}Z
	-L{A}R	
probability suffix	-{D}{I}R	

Figure 3.4: The verbal noun model

As for the verb sentences, interrogative noun sentences are formed by adding the question suffix: e.g., *Okula gittim mi?* (Did I go to the school?), *Öğrenci miyim?* (Am I a student?).

Time concept is given with the help of the tense suffixes in a noun sentence. As seen in Figure 3.4, there are three tense suffixes that can be added to a noun stem. They correspond to the second tense suffixes in the verb model. Thus, they are the definite past, narrative past, and conditional forms of the verb İMEK (see page 41), and they may also be used as independent words, i.e., İDİ, İMİŞ, İSE: i.e., *Öğrenciydim.* and *Öğrenci idim.* can both be used. To express remaining tenses and modes apart from these three tenses in noun sentences, the infinitive OLMAK (to become) is used: e.g., *Öğrenci olacağım.* (I will be a student.), *Öğrenci olmalıyım.* (I must be a student.).

The person suffixes used with a verbal noun are those listed in the first and second rows of the Table 3.10. Thus, when the tense is definite past or conditional, the person suffixes in the first row, when it is narrative, those in the second row are used. When no tense suffix exists, i.e., the sentence is in present tense, the second row of the table is active:

ÖĞRENCİ**YDİ**M	ÖĞRENCİYİM
ÖĞRENCİ**YDİ**N	ÖĞRENCİSİN
ÖĞRENCİ**YDİ**	ÖĞRENCİ
ÖĞRENCİ**YDİ**K	ÖĞRENCİYİZ
ÖĞRENCİ**YDİ**NİZ	ÖĞRENCİSİNİZ
ÖĞRENCİ**YDİ**LER	ÖĞRENCİLER.

As in the verb model, here also, the third person plural suffix may come either before or after the tense suffix: e.g. ORADALAR**DI** (they were there) and ORADA**YDI**LAR, ZAYIFLAR**MIŞ** ([I heard that] they were thin) and ZAYIF**MIŞ**LAR, and OKULDALAR**SA** (if they are at the school) and OKULDA**YSA**LAR are all valid. When this suffix should come after a plural noun, one of the -L{A}R's drops: e.g., ÖĞRENCİLER (students) → ÖĞRENCİLERDİ (they were students), not ÖĞRENCİLERLERDİ, or ÖĞRENCİLERDİLER.

The suffix -{D}{I}R 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 is here", but *Arkadaşınız buradadır.* means "he must be here (perhaps, I think, probably)". However, it is certainly used in statements which express permanent validities: e.g., *Kedi bir hayvandır.* (Cat is an animal.).

-{D}{I}R 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**MİŞ**TİR (he had [probably] come)	GEL**MİŞ**MİŞTİR
GEL**İYOR**DUR ([I think] he is coming)	GEL**İYOR**MUŞTUR
GEL**MEKTE**DİR ([probably] he has been coming)	GEL**MEKTE**YMİŞTİR
GEL**ECEK**TİR ([perhaps] he will come)	GEL**ECEK**MİŞTİR
GEL**MELİ**DİR ([may be] he must come)	GEL**MELİ**YMİŞTİR.

3.4.4 Participles

In Turkish, verb sentences can be transformed into a noun, adjective, or adverb clause by adding certain suffixes to the verb of the sentence. These suffixes can be listed in three groups as in Table 3.11.

During the transformation, the obligatory suffixes of the verb, i.e., main

Participles that form a(n)		
Noun	Adjective	Adverb
-M{A}{K}	-[Y]{A}N	-[Y]{I}P
-M{A}	-[Y]{A}C{A}{K}	-[Y]{A}R{A}K
-[Y]{I}Ş	-[Y]{A}S{I}	-[Y]{I}NC{A}
	-{D}{I}{K}	-[Y]{A}L{I}
	-M{I}Ş	-[Y]KEN
		-M{A}D{A}N
		-M{A}KS{I}Z{I}N
		-C{A}S{I}N{A}

Table 3.11: Participles

tense and person suffixes, are removed, and then the participles are affixed. However, most of the participles still denote the time characteristic of the verb. For example, -M{I}Ş and -[Y]{A}C{A}{K} still denote narrative past and future tenses, respectively. In addition, -{D}{I}{K} denotes past, -[Y]{A}S{I} future, and so on. Among the participles, only -M{A}D{A}N and -M{A}KS{I}Z{I}N can not be used with negation suffix since they include negation in themselves: OKUYACAĞINIZ (that you will read), GELMEYENLER (those who don't come), VERİLMEDEN (before/without being given).

-M{A}{K} forms the infinitive form of the Turkish verbs. The infinitive can be used as a noun, and may take any of the case endings but genitive. It never takes possessive suffixes: e.g., OKUMAĞ**A**, OKUMAK**TAN** are valid, but OKUMAĞ**IN**, OKUMAK**LARI** are not. Similarly, all the participles listed in the first and second columns of Table 3.11 may be used as a nominal root, i.e., they may take all the suffixes that a nominal root can take: e.g., GELİŞİNİZE (to your coming), VERDİKLERİNDENDİ (it was one of those that you gave).

The participles listed in the third column of Table 3.11 usually do not take any suffixes. Some of them can take only certain suffixes. For example -Y{A}R{A}K participle can take the suffix -{D}{A}N, which adds nothing to its meaning: YAPARAK (doing [something]) → YAPARAKTAN. In addition, -Y{I}NC{A} participle takes the suffix -[Y]{A} when it is used with the word

KADAR, and -[Y]{A}L{I} takes -{D}{A}N when it is used with the word BERİ: e.g., GELİNCEYE KADAR (until [the person] comes), GİDELİDEN BERİ (since [the person] has gone).

-[Y]KEN has a somewhat different usage than the other participles. Originally it is the -[Y]{A}N relative participle of the verb İMEK [5]. Like the other forms of this verb, it may be used as a suffix or as an independent word, i.e., İKEN. It is an invariable suffix, that is, it is not subject to the vowel harmony. In accordance with the general meaning of the sentence, it shows past, present, or future. It is affixed to a verb in the necessitative mode, or in any tense, except the definite past:

OKUMUŞKEN
OKUYACAKKEN
OKURKEN
OKUYORKEN
OKUMAKTAYKEN
OKUMALIYKEN.

It is not used with person suffixes, but it can follow the third person plural suffix -L{A}R: e.g., GELİRLERKEN (while they come). Second tense suffixes are not used with -[Y]KEN.

-[Y]KEN can also be affixed to a nominal stem causing a noun sentence transform into a noun clause: e.g. ÖĞRENCİYKEN, (when [the person] was a student), EVDELERKEN (when they are/were at home).

-C{A}S{I}N{A} shows some similarities with -[Y]KEN. It is affixed to certain tense bases, namely present, narrative past, and narrative of progressive and future:

UÇARCASINA
UÇMUŞCASINA
UÇUYORMUŞCASINA
UÇACAKMIŞCASINA,

and it can also be affixed to nouns and adjectives: e.g., ÇOCUKCASINA (as if a child), ÇILGINCASINA (as crazy).

3.4.5 Derivational Suffixes

Derivational suffixes are the suffixes which produce a new word having a different meaning than the word they are affixed to. As conjugational ones, derivational suffixes which can be added to nouns and verbs form different sets. Some derivational suffixes change the class of the word they are affixed to. Thus, they make nouns from verbs, or verbs from nouns. Others produce new nouns from nouns, or new verbs from verbs.

Some derivational suffixes may be received by all of the stems in the class that they belong to. The participles can be considered among them; i.e., they may be affixed to all verbs. Another group of the derivational suffixes can be attached to a great number, but not all, of the stems in their class. -{C}{I}, -L{A}Ş, -L{I}{K} are some examples to such suffixes. On the other hand, most of the derivational suffixes can be received by only a small number of stems. For example, the suffix -[Ş]{A}R can only be affixed to numerals to form distributive numerical adjectives: e.g., BİRER (one each), İKİŞER (two each). As an extreme example, the suffix -KEK can only be affixed to the noun ER (male), forming the noun ERKEK (man) [18].

There are hundreds of derivational suffixes in Turkish [1, 2, 18, 31, 23]. Some of them can only be added to some stems after they combine with some others. Such combinations should be examined as a single suffix. For example, the suffixes -L{A}N, -L{A}Ş, -L{A}T are the combinations of the suffix -L{A} with the suffixes -N, -[{I}]Ş, and -T, respectively [18]. Thus, although -L{A} can not be affixed alone to the nouns KUL (slave), YER (place), or KİR (dirt) to form verbs, the verbs KULLANMAK (to use), YERLEŞMEK (to settle down), and KİRLETMEK (to make dirty) are frequently used.

Examining all the derivational suffixes in Turkish necessitates a great effort and too much time. Even if we knew all the derivational suffixes, we should still examine all of the vocabulary of the language to determine which suffix can really be affixed to which roots. Below, you can find a small sample list of derivational suffixes, together with the class of the stems that they can be affixed and the class of the resulting word,[18] and a brief explanation about them:

–{A}L{A}: V → V

[18]**N** represents a nominal stem, where **V** stands for a verbal stem.

It can be attached only to a small number of verbal roots: e.g., KOVALAMAK (to run after), SİLKELEMEK (to shake off).

–{C}{I}: N → N

It is used to make the names of professions in the meaning of maker or seller of something: e.g., SUCU (water seller), BOYACI (painter). Additionally, it shows that one habitually or professionally occupies with something: e.g., EDEBİYATÇI (person who deals with literature), HAYALCİ (dreamer).

–[Y]{I}C{I}: V → N

It is used to form attributive adjectives in the meaning of "doing someting either continuously or temporarily"; they may also be used as nouns: e.g., SATICI (seller), DİNLEYİCİ (listener).

–L{A}Ş: N → V

Although it was formed by combining the suffixes –L{A} and –[{I}]Ş, it does not any more produce the reciprocal voice of the verbs formed by adding the suffix –L{A} to nominal roots, it is now a different suffix. –L{A}-Ş and –L{A}Ş may produce verbs of different meanings from the same root: e.g., TERS-LE-Ş-MEK (to scold each other), TERS-LEŞ-MEK (to become bad-tempered).

–L{I}{K}: N → N

It has too many usages [5]:

1. It is attached to adjectives and forms abstract nouns: e.g., İYİLİK (goodness), BOŞLUK (emptiness, blank, space).
2. It is attached to substantives to form adjectives showing the abundance of a thing in a place: e.g., AĞAÇLIK (grove), DAĞLIK (mountainous).
3. It is attached to nouns to make nouns and/or adjectives showing the purpose for which something is suitable or is intended: e.g., GÖZLÜK (eye glasses), YAZLIK (summer house).
4. It is added to some nouns to make names of containers: e.g., TUZLUK (saltshaker), ŞEKERLİK (sugar bowl).
5. It is attached to nouns preceded by a number to form adjectives meaning "of so many" or "for": e.g., BİN YILLIK (of thousand years), ALTI

SAYFALIK (of six pages).

6. It is added to some words showing the time to form temporal expressions: e.g., ŞİMDİLİK (for the time being), BUGÜNLÜK (for today).

–M{A}: V → N

It is added to verbs to form participles which can be used as nouns (see page 46): e.g., AYRILMANIZIN (of your departure), BAKMAN (your looking). It should not be confused with the negation suffix. They may be used together to form negative participles. For example, in the word BAKMAMA (not looking), the first MA is the negation suffix and the second is the participle:

$$\text{BAK} \quad + \quad \text{M\{A\}} \quad + \quad \text{M\{A\}} \quad \rightarrow \quad \text{BAKMA}\underline{\text{MA}}.$$

Another possibility is that the first MA is the participle, and it is followed by the first singular person possessive suffix –[{I}]M and then by the dative case suffix –[Y]{A}, giving the meaning "to my looking":

$$\text{BAK} \quad + \quad \text{M\{A\}} \quad + \quad \text{[\{I\}]M} \quad + \quad \text{[Y]\{A\}} \quad \rightarrow \quad \text{BAK}\underline{\text{MA}}\text{MA}.$$

Similarly, in the word SEVMEMEME (to my not loving) there are three ME's following each other. First of them is the negation suffix, second is the participle, and third is the combination of the first singular person possessive suffix and the dative case suffix:

$$\text{SEV} \quad + \quad \text{M\{A\}} \quad + \quad \text{M\{A\}} \quad + \quad \text{[\{I\}]M} \quad + \quad \text{[Y]\{A\}} \quad \rightarrow \quad \text{SEVME}\underline{\text{ME}}\text{ME}.$$

This suffix is casted in some words: e.g., AŞAMA (level), BALIKLAMA (headlong).

Although it is claimed that Turkish is characterized by a great regularity of patterns, the results of our research show that, in addition to its regularity, Turkish shows many irregularities that cause the problem of spelling checking for this language to become a very hard and interesting challenge. In the following chapter, our approach to the problem along with a description of our implementation will be presented.

Chapter 4

IMPLEMENTATION

4.1 General Structure

The scope of our current work is the implementation of a *spelling checking kernel* that can be integrated to different applications on a variety of platforms. Thus we have focused our efforts on solving the spelling checking problem instead of building a special application program for this purpose.

Our approach to spelling error detection is based on checking individual words in the text file by making a number of analyses with no attention to the semantics or to the context. Thus, if a word is spelled correctly but is the wrong word in the context, we have no intention for and way of flagging it as erroneous. For example, in the sentence "*Annem kardeşime dövdü.*", (My mother spanked my brother.) instead of the word *kardeşime* (kardeşim + DATIVE) the word *kardeşimi* (kardeşim + ACCUSATIVE) must be used. Since the word *kardeşime* is not misspelled when it is considered individually, we do not report it as misspelled. Thus, as in all other spelling programs, the text is examined with respect to words, not with respect to sentences. In addition, we do not yet give any suggestion about the most likely correct words after detecting a misspelled word, i.e., spelling correction is not done.

Figure 4.1 shows the general structure of the spelling checking kernel. A list of Turkish words is given as input to the program, and the program checks these words one by one, in the order they appear. The input words may be entered either from the keyboard or from a text file. If the spelling of an input word is incorrect, it is output as misspelled, either to the terminal or to a text file.

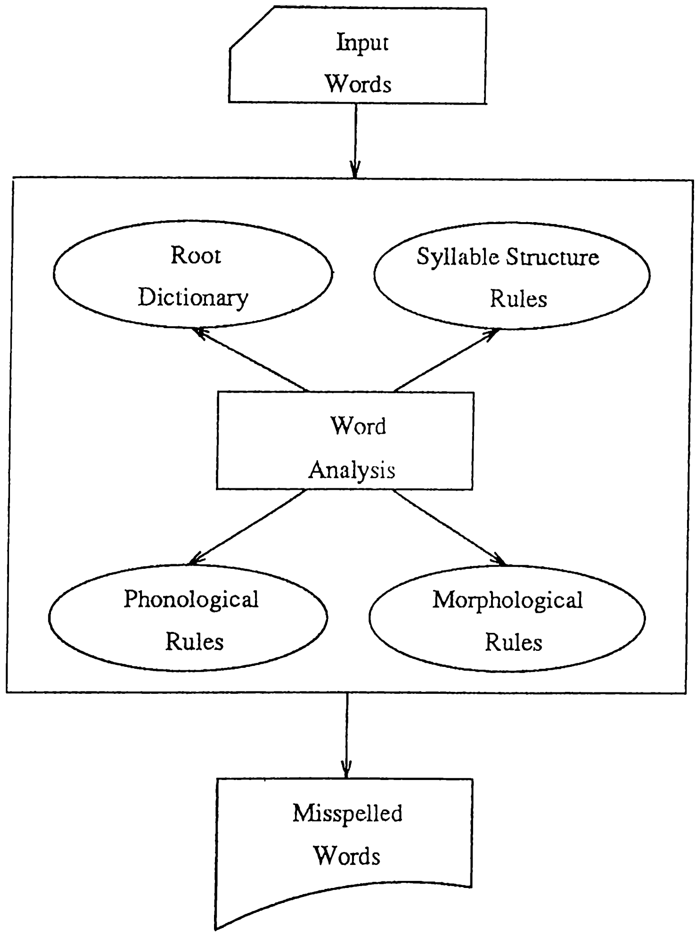

Figure 4.1: General structure of the Turkish spelling checker

Turkish alphabet contains some special letters shown by the symbols (ç, Ç, ğ, Ğ, ı, İ, ö, Ö, ş, Ş, ü, Ü) that do not exist on the standard character set of most of the computers, and on most of the keyboards. While a Turkish text is being entered such letters must be represented in a certain way. Most of the word processors have their own way to represent these letters. For example, in LaTeX ç is represented as \c{c}, Ü is represented as \"{U}, etc. In our kernel, such letters are required to be entered by preceding the letter that it follows in the alphabetical order with a special character, !. For example, the letter ç is entered as !c, Ü is entered as !U. Thus, the sentence "*Annem kardeşimi dövdü.*" must be entered as "*Annem kardel!simi d!lovd!u.*". When the kernel is to be integrated to a word processor, it is easy to replace the symbols used by that word processor to represent these letters using this convention. For example, if the checker is to be used with LaTeX files, all \c{c}'s appearing in them must be first replaced with !c's, and so on.

The external representation of the input word is converted into an internal representation before it is analyzed. In the internal representation, each letter that is not special is represented with its ASCII uppercase version while special letters are represented with lowercase letters preceding them in the alphabetical order. Thus, for example the letters ç and Ç are represented by the character **c**, where ü and Ü are represented by the character **u**. For consistency, the letters ı and I are represented by the character **i**, and i and İ are represented by the character **I**. The resulting character set is listed in Table 4.1.

After the external representation of the input word is converted into the internal representation, this word is analyzed in four steps:

1. Syllabification check,
2. Root determination,
3. Morphophonemic check, and
4. Morphological analysis.

During these steps a dictionary of Turkish root words, and a set of rules for Turkish syllable structure, morphophonemics, and morphology are used concurrently. All these steps will be explained in detail in the following sections, after a discussion of the data structures used in this implementation.

4.2 Data Structures

In the implementation, two main data structures are used. One of them is a *hash table* in which words that have already been checked are placed, and the other is an *ordered sequential array* in which the dictionary and the necessary flags are stored. Using these two tables, dictionary look-up is handled in two steps (see Figure 4.2).

4.2.1 Hash Table

The number of distinct words in a document often tends to be small. Therefore, building a table which contains distinct words that have been seen in processing a document helps to improve the analysis time: A word that has been examined already need not be examined for a second time. Each word whose spelling is checked is inserted into the table, together with a flag indicating whether its

Letters	External Representation	Internal Representation
a, A	a, A	A
b, B	b, B	B
c, C	c, C	C
ç, Ç	!c, !C	c
d, D	d, D	D
e, E	e, E	E
f, F	f, F	F
g, G	g, G	G
ğ, Ğ	!g, !G	g
h, H	h, H	H
ı, I	!i, !I	i
i, İ	i, I	I
j, J	j, J	J
k, K	k, K	K
l, L	l, L	L
m, M	m, M	M
n, N	n, N	N
o, O	o, O	O
ö, Ö	!o, !O	o
p, P	p, P	P
r, R	r, R	R
s, S	s, S	S
ş, Ş	!s, !S	s
t, T	t, T	T
u, U	u, U	U
ü, Ü	!u, !U	u
v, V	v, V	V
y, Y	y, Y	Y
z, Z	z, Z	Z

Table 4.1: External and internal representations of Turkish letters

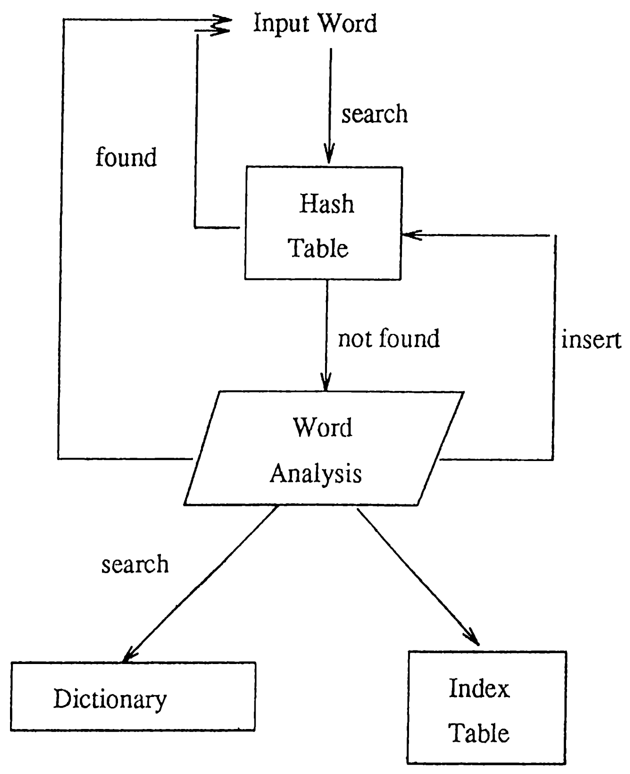

Figure 4.2: Data structures

spelling is correct or not. If this word occurs again in the same form as before, it is not examined since we know whether it is misspelled or not by checking the flag stored for this word into the table. Not to convert all of the words in the document into their internal representations, words are stored with their external representations in this table. If there is any difference between the external representations of two words, they are considered as different words even if they may be the same word. For example, if a word begins with a lowercase letter somewhere in the text and with an uppercase letter somewhere else in the same text (e.g., *bu* and *Bu*) they are considered as distinct words and inserted into the table.

The table of distinct words is represented by a hash table in our implementation. This table has 256 elements, and the location of a word x in the table is obtained by computing an arithmetic function f of x. f is defined by

$$f(x) = (\sum_{i=1}^{length(x)} x[i])\, mod\, 256$$

thus, the ASCII values of each character in x are added, and then the modulus of this summation with respect to 256 is taken. As a result, $f(x)$ maps the words onto the integers 0 through 255.[1]

Several different words may hash to the same location. In such a case a *collision* is said to occur. The hash table contains one list for each possible value of f, each list containing all the words which cause the function f to give the same result. A search then involves computing the hash function $f(x)$ and examining only those elements in the list for $f(x)$. Since the sizes of these lists are not known in advance, they are maintained as linked chains. Each element of a chain consists of a word in its external representation, a flag showing whether the spelling of that word is correct or not, and a pointer to the next element of the chain. The end of the chain is identified with a null pointer. The head of each chain is held in the table, and they are all initialized to a null pointer.

When a new word x is to be inserted into the hash table, $f(x)$ is computed, and the corresponding chain of the table is searched. If its head contains a null pointer, then x does not collide with any previous word, and this null pointer is replaced with a pointer pointing x. If the head is not null then x is inserted at the front of the corresponding chain, i.e., the head points to x, while x points to the word which the head was previously pointing.

As an example, let's assume that the sentence "*Bu ev, bu arsa ve bu araba bizim.*" (This house, this field and this car are ours.) is to be checked. First, the hash table is initialized with null pointers. Then the first word, i.e., *Bu* (This) is examined and found to be correct. The hash function for this word results in the value $f(Bu) = (66 + 117)\,mod\,256 = 183$. HashTable[183] is replaced with a pointer pointing the element {Bu, CORRECT, nil}. Later comes the word *ev* (house) which results the element {ev, CORRECT, nil} to be pointed by the $(101+118)\,mod\,256 = 219^{th}$ element of the table. The next word to be checked is again the word *bu*, but since in this occurence it begins with a lowercase letter, f gives a different result for the word *bu* than for the word *Bu*: $f(bu) = (98 + 117)\,mod\,256 = 215$. So, this time HashTable[215] is replaced with a pointer pointing the element {bu, CORRECT, nil}, and later HashTable[167] is replaced with a pointer pointing the element {arsa, CORRECT, nil}. For the following word *ve* (and) f gives the same result as for the word *ev*, thus they collide. So the 219^{th} element of the table is replaced with a pointer to the element {ve, CORRECT, $p(ev)$}, where $p(ev)$ is a pointer to the element {ev, CORRECT, nil}. The next word *bu* is found in the table, further its spelling is correct, so it is not examined. Finally the words *araba* (car) and *bizim* (our)

[1]Obviously any other hash function that maps strings to integers can also be used (see [19]).

are inserted to the 247th and 27th chains in the table, and the hash table of Figure 4.3 is obtained.

4.2.2 Dictionary

The dictionary is stored in an ordered sequential array. The words are placed in a sorted order in this array. The order of the words is not really alphabetic, because of the special Turkish letters. To preserve the real alphabetical order of Turkish, a special coding system is to be used. But, for our purpose, it is not necessary to keep the real alphabetical order. It will be sufficient if the words are ordered in a certain consistent way. We chose the internal representation given in Table 4.1 for the letters in Turkish alphabet. Consequently, for instance, although the word GÜNEŞ (sun) comes before the word GÜNEY (south) according to Turkish alphabetical order, it appears after GÜNEY in our dictionary because its internal representation GuNEs comes after the internal representation of the other, i.e., GuNEY, according to the alphabetical order of the computer.

To search a word in the dictionary the *binary search* method is used. In the worst case, this method requires $O(\log n)$ key comparisons, where n is the number of elements in the table (see [19]).

Since the table is sorted, all words beginning with the same letter follow each other. Therefore, instead of searching the whole dictionary, it is enough only to search those words beginning with the same letter as the word being searched for. For this purpose, an index table based on the first letters of the words is prepared. For each letter in the Turkish alphabet[2] (without considering the case), this table holds the address of the first word in the dictionary beginning with that letter, and the number of words with the same initial letter. During a binary search only the portion of the dictionary indicated by the appropriate index entry is searched. With this method the number of comparisons decreases substantially since the number of elements to be searched, i.e., n, decreases.

Deciding the content of the dictionary presented some difficulties. It is obvious that for an agglutinative language such as Turkish, including all possible words of the language in the dictionary is neither an applicable nor a practical approach. Storing only the root words in the dictionary is enough. However, since not every root–suffix combination is valid, a misspelling which forms an

[2]Except the letter Ğ since no word in Turkish begins with this letter.

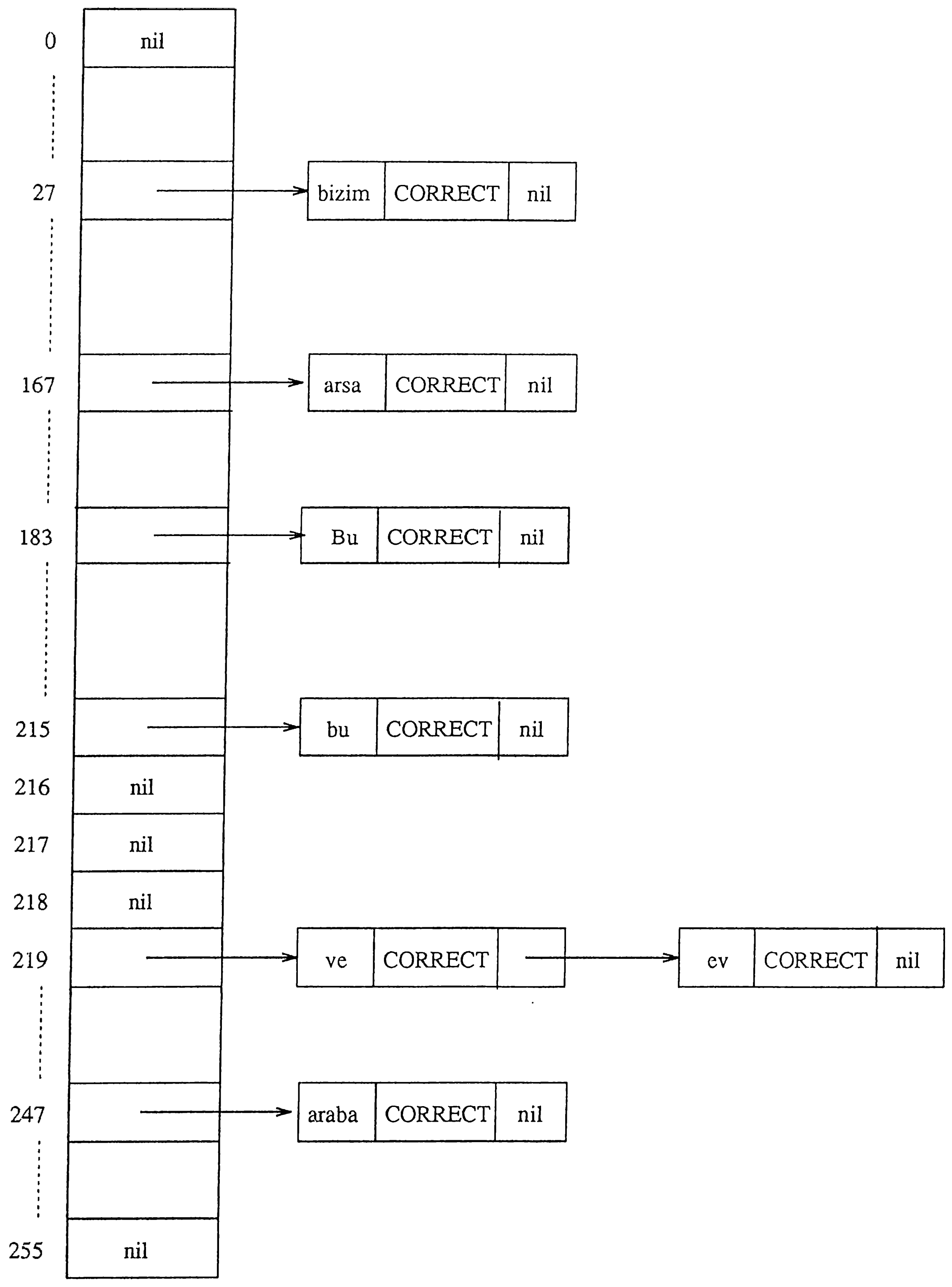

Figure 4.3: A sample hash table

invalid combination may go undetected. After the root and the suffixes of a word are found, it is necessary to examine whether they form valid combinations according to the Turkish word formation rules. For this purpose we have developed parsers which examine the suffixes in a word to determine whether they can really be affixed to that root, and if their order is valid. These parsers will be later explained in detail in Section 4.6.4.

The best source for the correct spelling of Turkish words is known to be the Turkish Writing Guide.[3] So, as the first step, we entered all the words appearing in this guide [46, 47] into our dictionary. This was a reasonable beginning as mentioned in Section 2.4.1, but the resulting dictionary was rather large (about 27,000 words). Later, we applied various criteria to delete the unnecessary entries.

The majority of the words listed in the Turkish Writing Guide are really the root words. However, there are still some words which can be derived by affixing certain suffixes (derivational or conjugational) to certain roots appearing in the guide. It is not necessary to include such words in our dictionary if they can be handled by the grammar rules of the parsers. For example, both the root word BÖYLE (such, so) and the word BÖYLESİNE (such) which can be derived from this root by affixing first the third person singular possessive suffix –[S]{I} and then the dative case suffix –[Y]{A} (see page 28)

BÖYLE + [S]{I} + [Y]{A} → BÖYLESİNE

appear in the guide, although we do not need to hold the word BÖYLESİNE in our dictionary. With careful analysis, most of such derived stems have been deleted from the dictionary. The following is a list of word categories that have been decided to be unnecessary to hold in the dictionary:

1. The passive forms of verbs: e.g., DUYULMAK (to be heard), ATANMAK (to be appointed), BULUNMAK (to be present).

2. The verbs having the compound verbs inside them: e.g., GELİVERMEK (to just come), BAKAKALMAK (to go on looking), ÖLEYAZMAK (to almost die).

3. The nominal stems that can be derived by affixing participial suffixes to verbal roots:[4] e.g., YAPMA (made, done), BAKIŞ (look), GELECEK

[3]Türkçe Yazım Kılavuzu

[4]Not all words ending with such suffixes belong to this catagory. For example, the word AŞAMA (level) is not deleted because there is no verb as AŞAMAK.

(future), GEÇEN (last), TANIDIK (acquaintance).

4. The group names (mostly biological) formed by the help of the plural suffix: e.g., KARINCALAR (ants), SÜRÜNGENLER (reptiles).

5. The adverbs formed by affixing the ablative case suffix –{D}{A}N to nominal roots: e.g., SONRADAN (subsequently), YENİDEN (again).

6. The adverbs formed by affixing the case suffix –YL{A} to nominal roots alone or together with the third person singular possessive suffix: e.g., ÇOĞUNLUKLA (mostly), SIRASIYLA (respectively).

7. The adverbs formed by affixing the dative case suffix after the third person singular possessive suffix to nominal roots: e.g., BÖYLESİNE (such), TERSİNE (on the contrary).

8. The location words formed by affixing the locative case suffix –{D}{A} to nominal roots: e.g., ORADA (there), YUKARIDA (above).

9. The pronouns formed by affixing the third person singular possessive suffix to nominal roots: e.g., BAŞKASI (someone else), ÇOĞU (the most).

10. The stems derived from numeral roots: e.g., ALTINCI (the sixth), BİRİNCİLİK (being in first position), İKİŞER (two each), KIRKLAMAK (to reach the 40th day after the birth of a baby).

11. The nominal and verbal stems derived by the derivational suffixes included in the grammar rules of the parsers: e.g., KAPICI (doorman), İYİLİK (goodness), ZORLAŞMAK (to become harder).

In the Turkish Writing Guide all verbs are listed in their infinitive forms (e.g., OKUMAK (to read), SEVMEK (to love)). We decided that we did not need to put the suffix -M{A}{K} at the end of the verbs; storing only the root part was enough. So, the infinitive part of the verbal roots has been deleted in our dictionary, thus the verbal roots are stored in their imperative forms as, for instance, OKU. SEV, and they are marked as being verbal roots. A careful analysis had to be made during this process because

1. Not all of the words ending with M{A}K are infinitives (e.g., BASAMAK (step), TOKMAK (mallet)). Such nouns should not be mixed with the infinitive forms of the verbs, i.e., the substring M{A}K at their ends should not be removed.

2. Some infinitives are homonyms with some nouns (e.g., EKMEK (to sow) and EKMEK (bread)). The nominal one is left as it is in our dictionary while the root of the infinitive one (i.e., EK) is marked as a verbal root.

3. After the deletion of the infinitive part from the end of the verbs, a root which can be used both as a nominal and a verbal root, thus, a nominal root which is the homonym of a verb in the imperative form, happens to occur twice in our dictionary (e.g., AK (white) and AK(MAK) ((to) flow), TAT (taste) and TAT(MAK) ((to) taste)). One of such roots has been deleted, and the remaining one is marked as being both a nominal and a verbal root.

In the Turkish Writing Guide the proper nouns are represented by writing their first letter in uppercase (e.g., Ankara (capital of Turkey), İngiliz (English [person])). The words which can be used both as a proper noun and as an unproper noun are listed twice, one beginning with an uppercase letter, the other one beginning with a lowercase letter (e.g., Ağrı (a city in Turkey) and ağrı (pain), Mısır (Egypt) and mısır (corn)). Since no case distinction is present in our dictionary representation, one of such words has been deleted while the other has been marked as being both a proper and an unproper noun.

After all these removals the size of the dictionary decreased substantially (about 5 thousand words have been removed). With a more detailed analysis it is still possible to delete many other unnecessary words, most of which are words of foreign origin that are rarely used today.

Another disadvantage of selecting the Turkish Writing Guide as our dictionary was the absence of certain classes words. There are many commonly used words which do not appear in the guide. One class of such words comprise the technical terms from different areas of science and engineering. There are many dictionaries published by Turkish Language Society listing the technical terms for such areas. Analyzing all of them to select the terms to include in our dictionary would have taken a substantial amount of time, and probably the size of the dictionary would grow too much. Instead of this, we added only some frequently used terms which appeared in an unpublished dictionary that had been prepared by a group in Middle East Technical University. Thus, our dictionary still lacks a number of words.

Turkish Writing Guide includes many proper names such as nationality names, countries and cities. In spite of this, it was necessary to include some other proper names in our dictionary,.such as personal names. Without using any sources, we added a great number of personal names that we knew into the dictionary, but of course still remain many others.

In order to determine the remaining words that should be added to our dictionary, we have examined the output of the checker on real runs. A copy of the output of the checker has been mailed to us after each test run. Examining these words, we have determined which words indicated as misspelled are in fact those which do not appear in our dictionary, and we have added these words into our dictionary. Obviously, this is an ongoing process.

Nearly 23,500 words, each having 7 letters on the average, are listed in our current dictionary. This amount may change (increase or decrease) in the future.

As mentioned above, some items in the dictionary have to be marked as having a certain property. For example, some must be marked as being a verbal root, some must be marked as being a proper noun, and so on. For this reason, for each word in the dictionary a series of flags representing certain properties of that word are held. Thus, each entry of the dictionary contains a word in Turkish and a series of flags showing certain properties of that word. It is possible to hold 64 different flags for a single word because two long integers are allocated for each dictionary item. If the bit corresponding to a certain flag is set for an entry then it means that the word which this entry belongs to has the property represented by that flag. Only 41 flags have been used yet, but later it may be necessary to use the remaining ones. The list of these flags together with some examples for which that flag is to be set is given in Table 4.2.

For each flag, the list of the words which appear in our dictionary and which have the property that is represented by that flag is prepared. Some of these lists contain a large number of elements, while only a few words exist in some of them. The lists for the flags IS_UDD, IS_STT, IS_KU, and F_UD are given in Tables 4.3, 4.4, 4.5, and 4.6 respectively as examples to the lists containing a small number of elements.

The flags for each entry of the dictionary is set by the help of a program. This program loops on each word of the dictionary, searchs it in all lists, and set the flags whose lists contain that word. When a new word is added into the dictionary, it should also be added into the lists of the flags which must be set for this word, and the program which sets the flags must be run.

Flag	Property of the word for which this flag is set	Examples
CL_NONE	belongs to none of the two main root classes	RAĞMEN, VE
CL_ISIM	is a nominal root	BEYAZ, OKUL
CL_FIIL	is a verbal root	SEV, GEZ
CL_BOTH	can be used both as a nominal and a verbal root	TAT, YAZ
EK	is a suffix that must be written separate from the word it follows	Mİ, İDİ
IS_OA	is a proper noun	AYŞE, TÜRK
IS_OC	is a proper noun which has a homonym that is not a proper noun	MISIR, SEVGİ
IS_SAYI	is a numeral	BİR, KIRK
IS_LIK	is a nominal root which can take the suffix –L{I}{K}	SENE, TUZ
IS_LAS	is a nominal root which can take the suffix –L{A}Ş	KENT, UYGAR
IS_LAT	is a nominal root which can take the suffix –L{A}T	AYDIN, KİR
IS_CI	is a nominal root which can take the suffix –{C}{I}	DAVA, KAVGA
IS_CILIK	is a nominal root which can take the suffix –{C}{I}L{I}{K}	KAR, ÜMMET
IS_CA	is a plural noun	BAKLAGİLLER
IS_KI	is a nominal root which can directly take the relative suffix –Kİ	BERİ, ŞİMDİ
IS_KU	is a nominal root which can directly take the relative suffix –KÜ	BUGÜN, ÖBÜR
IS_UU	is a nominal root which does not obey the vowel harmony rules during agglutination	SAAT, NORMAL
IS_UUU	is a nominal root which has a homonym that does not obey the vowel harmony rules during agglutination	SOL, YAR

Table 4.2: List of flags

Flag	Property of the word for which this flag is set	Examples
IS_SD	is a nominal root ending with a consonant which is softened when a suffix beginning with a vowel is attached	AMAÇ, PARMAK, PSİKOLOG
IS_SDD	is a nominal root ending with a consonant which has a homonym whose final consonant is softened when a suffix beginning with a vowel is attached	ADET, KALP
IS_KG	is a nominal root ending with the consonant K which changes into a G when a suffix beginning with a vowel is attached	ÇELENK, RENK
IS_ST	is a nominal root ending with a consonant which is duplicated when a suffix beginning with a vowel is affixed	HAK, TIP
IS_STT	is a nominal root ending with a consonant which has a homonym whose final consonant is duplicated when a suffix beginning with a vowel is affixed	HAL, ŞIK
IS_UD	is a nominal root which has a vowel {I} in its last syllable that drops when a suffix beginning with a vowel is affixed	AĞIZ, OĞUL
IS_UDD	is a nominal root which has a vowel {I} in its last syllable and which has a homonym whose last vowel drops when a suffix beginning with a vowel is affixed	HAYIR, METİN
IS_UDOD	is a nominal root whose last vowel drops and the consonant preceeding it changes when a suffix beginning with a vowel is affixed	ZABIT
IS_SI	is a nominal root ending with a vowel, but when it takes the third person singular possessive suffix the consonant S is not inserted in between	BAYİ, SANAYİ
IS_SII	is a nominal root ending with a vowel, and when it takes the third person singular possessive suffix the consonant S may or may not be inserted in between	CAMİ, MEVKİ

Table 4.2 continued.

Flag	Property of the word for which this flag is set	Examples
IS_BILEŞ	is a portmanteau word which was originally an indefinite compound	ADAÇAYI, YILBAŞI
IS_B_SI	is a portmanteau word ending with the third person singular possessive suffix -S{I}	ALINYAZISI, BALARISI
IS_B_SD	is a portmanteau word whose last consonant was softened during combination	AYAKUCU, RENGEYİĞİ
IS_B_UD	is a portmanteau word whose last word faced with a vowel ellipsis during combination	ADEMOĞLU, GÖKCİSMİ
IS_SU	is a nominal root which shows the irregularities that the root SU shows	AKARSU
IS_ZM	is a pronoun which shows some irregularities	BU, BİZ
F_SD	is a verbal root ending with a consonant which is softened when a suffix beginning with a vowel is attached	EMRET, GİT
F_UD	is a verbal root which has a vowel {I} in its last syllable that drops when the passiveness suffix -{I}L is affixed	AYIR, SAVUR
F_GUD	is a verbal root ending with a vowel {A} that changes into a {I} when the progressive suffix is affixed	ANLA, BENZE
F_GUDO	is a verbal root ending with a vowel {A} that changes into a {I} when a suffix beginning with a Y is affixed	DE, YE
F_GIR	is a monosyllabic verbal root which takes the suffix -{I}R as the aorist suffix	GEL, KAL
F_GER	is a polysyllabic verbal root which takes the suffix -{A}R as the aorist suffix	HİSSET
F_DIR	is a verbal root ending with a consonant but does not take the suffix -{D}{I}R as the factitive suffix	GİT, ÖKSÜR

Table 4.2 continued.

AKİT
HAMİL
HAYIR
KADİR
KATİL
KOYUN
METİN
NEFİS

Table 4.3: Word list for the flag IS_UDD

AD
HAL
ŞAK
ŞIK

Table 4.4: Word list for the flag IS_STT

BUGÜN
DÜN
GÜN
ÖBÜR

Table 4.5: Word list for the flag IS_KU

AYIR
ÇEVİR
KAVUR
YOĞUR
ÇAĞIR
DEVİR
KAYIR
SAVUR
SIYIR
KIVIR

Table 4.6: Word list for the flag F_UD

4.3 Syllabification Check

The syllable types which can be found in Turkish words are of limited amount (see Sections 3.2.1 and 3.2.2). The number of vowels and consonants which can follow each other in the words formed by combining such syllables is also limited (see Table 3.6). Analyzing all the words in Turkish Writing Guide [46, 47] and all the suffixes in Turkish [1], we have constructed a regular expression and a corresponding finite state automaton for validating if a word matches the syllable structure rules of Turkish[40].

Previously a number of studies on Turkish syllable structures and hyphenation have been made [2, 14], however they are inadequate in various aspects. They only consider the basic syllable types and fail in some words of foreign origin. For instance, Gönenç [14] hyphenates the word KONTRBAS as KONT-RBAS, however correct hyphenation is KONTR-BAS. Our analysis spans the syllable structures of "Pure Turkish", and handles all words of foreign origin used in Turkish, not handled by previous studies.

The regular expression constructed for proper Turkish syllable structures is used as a heuristic in our spelling checker. The heuristic is *if a word does NOT have the proper syllable structure of Turkish, it is misspelled.* 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 letters of the word do not form valid sequences according to Turkish syllable structures. On the other hand, if it can be matched, it is further analyzed as it may still be a non-Turkish or misspelled word. This expression can be given as follows:

```
(BEGV ((MIDC (MIDV MIDC)* ENDV) | ((MIDC MIDV)* ENDC))) |
(BEGC ((MIDV (MIDC MIDV)* ENDC) | ((MIDV MIDC)* ENDV))) |
ONEV
```

A simplified form of the finite state automaton for this expression is given in Figure 4.4.

According to the regular expression and the corresponding automaton, a word in Turkish can be formed by a single vowel (ONEV). Otherwise, if a word begins with vowels (BEGV) then it may either just end with consonants

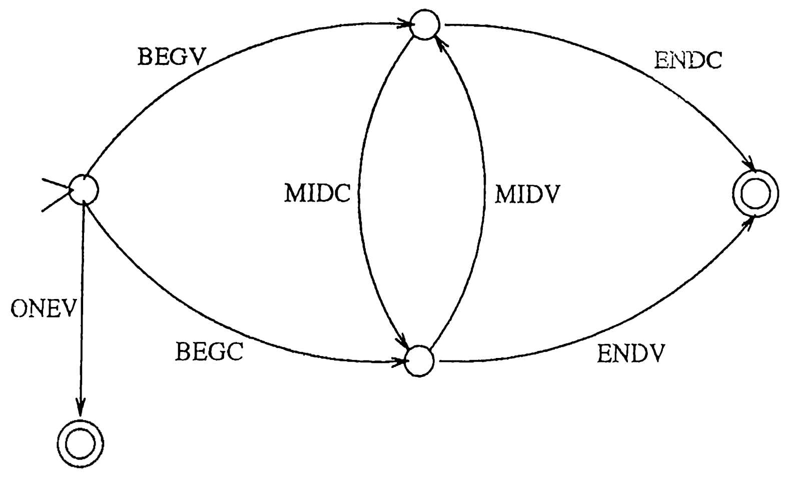

Figure 4.4: The simplified finite state automaton for proper Turkish syllable structure

(ENDC), or may be followed by some consonants (MIDC) and end with vowels (ENDV), or can be followed by any number of consonants-vowels series ((MIDC MIDV)*) and end with consonants, or can be followed by any number of consonants–vowels–consonants series (MIDC (MIDV MIDC)*) and end with vowels. On the other hand, if a word begins with with consonants (BEGC) then it may either just end with vowels, or may be followed by some vowels (MIDV) and end with consonants, or can be followed by any number of vowels–consonants series ((MIDV MIDC)*) and end with vowels, or can be followed by any number of vowels–consonants–vowels series (MIDV (MIDC MIDV)*) and end with consonants.

In fact, the states in the automaton are non–deterministic FSA within themselves, whose corresponding regular expressions are as below:

$$
\begin{array}{lcl}
\text{BEGV} & = & \text{V} \mid \text{VV} \\
\text{BEGC} & = & \text{C} \mid \text{CC} \mid \text{CCC} \\
\text{MIDV} & = & \text{V} \mid \text{VV} \mid \text{VVV} \\
\text{MIDC} & = & \text{C} \mid \text{CC} \mid \text{CCC} \mid \text{CCCC} \mid \text{CCCCC} \\
\text{ENDV} & = & \text{V} \mid \text{VV} \\
\text{ENDC} & = & \text{C} \mid \text{CC} \mid \text{CCC} \\
\text{ONEV} & = & \textbf{O}
\end{array}
$$

where V and C represent a vowel and a consonant respectively. These expressions are also more complicated, because not all sequences of consecutive vowels or consonants are valid. Analyzing all root words and suffixes in Turkish, the restricted values for those sequences that appear at the beginning, middle and end of the words are determined and listed in Tables 4.7, 4.8, and 4.9

Transition	Sequence	Values	Examples
BEGV	VV	A[EİU][5] İA Oİ [AEO]O	AİLE, AUT İADE OİL AORT, OOSFER
BEGC	CCC	S[KPT]R	STRATEJİ
	CC	S[FKLMNPT] [BFGKP][LR] [DPT]R [KP]S	SKANDAL, SPOR BLUZ, KRAL DRAJE, TREN PSİKOLOJİ
	C	[BCÇDFGHJKLMNPRSŞTVYZ]	BEN, ZARAR

Table 4.7: List of valid letter sequences that appear at the beginning of the words

respectively with some examples. When all these restrictions are considered, the real finite state machine for the expression contains nearly *two thousand states* and more than *five thousand transitions.*

In the implementation of the finite state automaton, we have utilized one of the standard UNIX utilities, *lex* (see Section 4.6.2). The regular expression prepared has been given as the input specification to *lex*, and *lex* has produced the C program to match it.

With the help of the syllabification check, most of the typographical errors can be detected. For example, if the word YAPMAK (to make) were typed as YPMAK or YAPMKA, thus, if a "one missing letter" or a "two transposed

[5][xy] means the letter x or y.

[6](x | y) means an x or y.

[7]There appears four consonants within a word when a suffix beginning with a consonant is affixed to a word ending with three consonants.

[8]Set of the consonants that may appear in the beginning of a suffix.

[9]There appears three consonants within a word when a suffix beginning with a vowel is affixed to a word ending with three consonants.

[10]There appears three consonants within a word when a suffix beginning with a consonant is affixed to a word ending with two consonants.

Transition	Sequence	Values	Examples
MIDV	VVV	(AA | EO | Uİ)[6]İ EİU	MAAİLE MÜDDEİUMUMİ
	VV	[Aİ][AEIİOÖUÜ] [EOUÜ][AEİO] [EIU][ÖU] [EI]Ü I[AEIO] ÜÖ	SAUNA, ŞİİR REİS, MUAF SUÖRÜMCEĞİ ÖĞLEÜSTÜ AÇIORTAY VİRTÜÖZ
MIDC	CCCCC	LFSTR NTRPL	GOLFSTRİM KONTRPLAK
	CCCC	[BKN]STR KSPR N(GST | SKR) NTR[BF] RNBL (3 consonants that can appear at the end)[7][CÇDGKLMST][8]	ENSTRÜMAN EKSPRES GANGSTER KONTRBAS HORNBLENT ROPDÖŞAMBRLA, SÖMESTRDE
	CCC	BLD [ÇŞ]PL FT[YP] KSP L(DM | HP | K[BŞY] | PN) L(T[BFRŞY] | [FG]R) M[PT]R N(ÇP | D[RV] | G[PR] | JM) N(K[RNY] | SF | T[BFHPRY]) R(K[BY] | P[HR] | S[HPY] | T[BFNPRVY]) S(T[BNY] | [KP]R) Y[FR | SB] ([FR]D | [KP]T | NP | NP | ZB)R 3 consonants that can appear at the end[9] (2 consonants that can appear at the end)[10][CÇDGKLMST]	TABLDOT İÇPLAZMA NEFTYAĞI EKSPER CELPNAME TELGRAF EMPRESYONİST ARANJMAN HENTBOL SÜRPRİZ, ARTVİN ÜSTYAPI, ESPRİ BEYSBOL BORDRO, ELEKTRİK SÖMESTRİN KARTÇA, AHENKLİ, İLKMİŞ, DANSTA

Table 4.8: List of valid letter sequences that appear inside the words

Transition	Sequence	Values	Examples
ENDV	VV	[AIİOU]A	MÜDAFAA, DUA
		[AEİ]İ	MESAİ, Şİİ
		[AÜ]O	KAKAO, DÜO
		[AI]I	MISRAI
		E[EO]	ZATÜREE, STEREO
		UU	VUKUU
		ÜÜ	TEMETTÜÜ
ENDC	CCC	(MB | ST)R	ROPDÖŞAMBR
		NKS	SFENKS
		RTZ	KİLOHERTZ
	CC	[LMY]F	GOLF, TAYF
		[BKY]L	MONOKL, KOKTEYL
		[RY]N	MODERN, EBEVEYN
		[MS]P	KAMP, GASP
		[HY]Ş	SÜVEYŞ
		[NR][ÇDFGHKSŞPTZ]	AJANS, PARK
		[FKP][ST]	NEFT, ELİPS
		[LY][ÇHKMPST]	FELÇ, OFSAYT
		[MSŞ][KT]	DİSK, RÜŞT
		HT	TAHT
		R[JMV]	ŞARJ, ALARM
		V[ÇKMRT]	SEVK, NAKAVT
		TR	GUATR
		Z[KM]	RİZK, TURİZM

Table 4.9: List of valid letter sequences that appear at the end of the words

letters" error was made, the word would not be matched by the expression and its spelling would be reported incorrect. On the other hand, if it were written as YAPMEK, where a vowel harmony error is made, it would pass the syllabification check, and would not be reported as misspelled until morphophonemic checks. Most of the words from other languages (used within Turkish text) can not pass the syllabification check. If the checked document contains such words they are reported as misspelled during syllabification check, and no more analyzed. However, some foreign words whose structures also obey the Turkish syllabification rules (e.g., spelling, table) can pass this check, but are reported as misspelled in the subsequent steps of word analysis.

Our aim to construct a regular expression to capture the syllable structure of Turkish words is the creation of a heuristic for the spelling checker program. However, this work can later be integrated to different applications, such as development of an automatic Turkish hyphenation function for word processors.

4.4 Root Determination

Before analyzing the morphophonemic and morphological structures of a Turkish word, the root has to be determined. If the word passes the syllabification check, its root is searched in the dictionary using a maximal match algorithm. In this algorithm, first the whole word is searched in the dictionary. If it is found then the word has no suffixes and therefore its spelling is correct. Otherwise, we remove a letter from the right and search the resulting substring. We continue this by removing letters from the right until we find a root. If no root can be found although the first letter of the word is reached, the word is reported as misspelled.

The maximum length substring of the word that is present in the dictionary is not always its root. If further analyses show that the word is misspelled, a new root is searched in the dictionary, this time removing letters from the end of the previous root. If a new root can be found the same operations are repeated, otherwise the word is reported as misspelled. For instance, the root of the word YAPILDIN (you were made) is first determined as the noun YAPI (structure). However, the rest of the word does not form a valid sequence of suffixes for a nominal root. Instead of reporting the word as misspelled, a new root is searched, and the verbal root YAP (make, do) is found. Since this one is the real root, the word's spelling is found to be correct after the subsequent analyses.

As another example consider the word KOYUNLARMI? (are the sheep?) which has an incorrect spelling since the question suffix -M{I} has to be written separate (see page 41). The maximal match algorithm first determines the root as the nominal root KOYUN (sheep), which is the real root, but since the rest of the word can not be parsed correctly, it assumes that the root has been determined wrongly. Hence, a new root is searched and the nominal root KOYU (dark) is found. However, the rest of the word can not be parsed correctly with this root either. Next root determined is the root KOY. This root may either be the nominal root KOY (small bay) or the verbal root KOY (put). Both alternatives are tried but the results are unsuccessful. Since no other root can be found, the word is reported as misspelled.

Root determination presents some difficulties when the root of the word is deformed. For the root words which have to be deformed during certain agglutinations (see Section 3.3.3), a flag indicating that property is set in the dictionary (see Table 4.2). The individual cases such as the dative and plural forms of personal pronouns are inserted into the dictionary and treated as exceptions. For the other root deformations, the root of the word is found by making some checks and some necessary changes. In the following paragraphs, some examples are given to show how the real value of a deformed root is determined.

As the first example, let's consider the vowel ellipsis for nominal roots. In the word OĞLUN (your son) the nominal root OĞUL (son) has taken the shape OĞL when it received the second person singular possessive suffix -[{I}]N. In order to determine this root correctly, when the substring OĞL is not found in the dictionary, since it is followed by a vowel, its last two letters are consonants, and the third phoneme from its right end is a vowel, the possibility that it may be a deformed root by vowel ellipsis is considered. The new candidate for the root is obtained by inserting the proper vowel {I}, i.e., U, between the last two consonants of the current candidate, i.e., between Ğ and L. and the word OĞUL is searched in the dictionary. When it is found, the flag corresponding to vowel ellipsis for nominal roots, i.e., IS_UD, is checked. Since it is set for this word, the root of the word OĞLUN is determined as OĞUL, and remaining analyses are continued. If that word were written as OĞULUN, it should be reported as incorrect. In order to handle this case, when the root OĞUL is found in the dictionary, since it is followed by a vowel, the flag IS_UD is checked to see whether it is a root whose last vowel must drop when it is followed by a vowel. Since it is set for this word, but the last vowel of the word has not dropped, the algorithm decides that the root of the word OĞULUN is not the word OĞUL. Later, a new root is searched and since no root can be found, the word OĞULUN will be reported as misspelled. As another interesting case, both the words OĞULUM (I am a son) and OĞLUM (my son) have correct spellings, because in the first one the root OĞUL has received the first singular person suffix –[Y]{I}M (see page 30), while in the second one it received the first person singular suffix –[{I}]M. Not to report the word OĞULUM as misspelled, when it is realized that the root OĞUL is a root that has to deform when it is followed by a suffix beginning with a vowel, the algorithm checks whether that suffix may be one of the suffixes –[Y]{I}M or –[Y]{I}Z.

Another root deformation is the change of the last consonant in some roots. For example, in the word TABAĞIM (my dish), final consonant of the nominal root TABAK (dish), i.e., K, has changed into Ğ, when the first person singular possessive suffix is affixed. In this case, when the substring TABAĞ is not found in the dictionary, since it is followed by a vowel, and its last phoneme is one of the consonants B, C, D, G, and Ğ, the possibility that it may be a deformed root whose last phoneme has changed is considered. Since it does not end with the substring LOĞ,[11] and the final phoneme is not preceded by the consonant N,[12] the final phoneme Ğ is replaced with the consonant K, and the word TABAK is searched in the dictionary. When it is found, the flag corresponding to the change of the final consonant, i.e., IS_SD, is checked. Since it is set for this word, the root of the word TABAĞIM is determined as TABAK. If that word were written as TABAKIM, it would be reported as incorrect.

As another example, let's consider the duplication of the final consonant for some nominal roots. In the word HAKKINIZ (your right), the consonant K at the end of the root HAK (right) is duplicated when it received the second person plural possessive suffix. When the substring HAKK can not be found in the dictionary, since it is followed by a vowel, its last two phonemes are the same consonants, and the third phoneme from its right is a vowel, the possibility that its last phoneme may have been duplicated is considered. Its last phoneme is deleted and the word HAK is searched in the dictionary. When it is found, the flag corresponding to the duplication of the final consonant, i.e., IS_ST, is checked. Since it is set for this word, the root of the word HAKKINIZ is determined as HAK. If that word were written as HAKINIZ it would be

[11]If the word were PSİKOLOĞA (to the psycholog), this condition would hold and Ğ would be replaced not with a K but with a G.

[12]If the word were RENĞE, this condition would hold and no replacements would be done, and later it would be reported as misspelled.

reported as incorrect. As another interesting example, the root of the word TIBBIN (medicine's) is the word TIP (medicine) where its last phoneme is duplicated after changing into a B. In this case, as in the previous one, one of the B's is removed from the end of the word TIBB and the word TIB is searched in the dictionary. When it is not found, since its last consonant is B, it is changed into a P, and the word TIP is searched in the dictionary. When it is found, both the flags IS_ST and IS_SD are checked. Since both are set for this word, the root is determined as TIP. If that word were written as TIPIN, TIBIN, or TIPPIN, it would be reported as misspelled.

For all the other deformations such as vowel ellipsis in the verbal roots, narrowance of the final wide vowel in the verbal roots, midfixing of the plural suffix to the portmanteau words, etc., and their combinations, both the correct and incorrect usage of the roots are determined by using similar methods to the ones above.

For some roots both of the deformed and undeformed forms are valid. For example, both METNİ (accusative of text) and METİNİ (accusative of strong) are correct although the root of both words is METİN (text, strong) because this word can be used in two different meanings. Such cases are handled again by the help of certain flags, IS_UDD, IS_SDD, and IS_STT. For instance, to determine the root of the word METNİ as METİN, checking only the flag IS_UD is enough. On the other side, not to report the spelling of the word METİNİ as incorrect, when the root METİN is found, the flag IS_UDD is checked. Since it is set for this word, the root is determined as METİN. Similarly, none of the words ADEDİ (ADET: amount), ADETİ (ADET: custom), ŞIKKI (ŞIK: option), or ŞIKI (ŞIK: chic) is reported as misspelled.

The algorithm for root determination sometimes requires a lot of searches in the dictionary. To determine the root of the word OKULA (to the school), two searches (one for OKULA and the other for OKUL) are enough, but to determine the root of the word ALDIĞIMIZ (that we took), the dictionary is searched 13 times for the words ALDIĞIMIZ, ALDIĞIMI, ALDIĞIM, ALDIĞI, ALDIĞISI, ALDIĞ, ALDIK, ALDI, ALID, ALIT, ALT, and AL, respectively. Our tests has shown that, to determine the root of one word, the dictionary is searched 5–6 times on the average.

4.5 Morphophonemic Checks

After the root of the word is found, the rest of the word is considered as its suffixes. Vowels and consonants within suffixes should obey certain rules during agglutination (see Section 3.3). Therefore, the suffixes part of a word must be checked to see whether any of the morphophonemic rules are violated. The vowel harmony check may be done just after the root determination, but other morphophonemic checks should be done during morphological analysis.

4.5.1 Vowel Harmony Check

According to the vowel harmony rules of Turkish (see Section 3.3.1), the first vowel in the suffixes part must be in harmony with the last vowel of the root, while the succeeding vowels must be in harmony with the vowel preceding them. For example, the word YAPMEK (see page 72) can not pass the vowel harmony check because the vowel E can not follow the vowel A. On the other hand, special checks must be done for the suffixes, such as -KEN, whose vowels never change. So, when a disharmony is found, we check whether it is the result of such a suffix. For example, after the root of the word YANARKEN (while it is burning) is found as YAN (side, burn), the suffixes part, i.e., ARKEN, is checked to determine whether the word obeys vowel harmony rules. The first vowel A is in harmony with the last vowel of the root, but the next vowel E is not in harmony with the vowel preceding it. At this point, instead of deciding that the word does not obey vowel harmony rules, the phonemes preceding and following the current vowel are checked to determine whether that vowel belongs to one of the suffixes which do not obey vowel harmony rules, i.e., to -[Y]KEN, -[Y]{I}VER, or -[Y]{A}GEL. Since it does, the word passes the vowel harmony check. If this word was written as YANARKAN, it would pass the vowel harmony check, but it would not be parsed correctly during morphological analysis.

Before the vowel harmony check is done, some flags of the root must be checked. For example, if the word is a word of foreign origin that does not obey vowel harmony rules during agglutination (e.g., KONTROL (control)), the vowel harmony check must be applied inversely. Thus, the first vowel in the suffixes part must be in disharmony with the last vowel of the root (e.g., KONTROLLER (controls)). The flag IS_UU is checked to realize such cases. Some roots that may be used in two meanings (homonyms) present another interesting case. They obey vowel harmony rules when they are used with a certain meaning, but disobey them when they are used in the other meaning. For example, both SOLA (to the left) and SOLE (to the note sol) pass the vowel harmony check since their root SOL has two meanings as "left" and "a note in musics".[13] Such cases are handled by the help of the IS_UUU flag.

Another special case occurs when a root which does not obey vowel harmony rules within itself deforms by vowel ellipsis. For example, the root of the word NAKLİ (its transfer) is the noun NAKİL (transfer). If the vowel harmony check is done accepting the root as NAKL it fails because the vowel İ can not follow the vowel A. In such cases, not the deformed root but the real root appearing in the dictionary must be considered, and the suffixes part must be in harmony with the real root, i.e., in our example with the word NAKİL. The wrong form, i.e., NAKLI would also be realized, but not during the vowel harmony check, instead during root determination, because the proper vowel to be inserted between the consonants K and L would be determined as I, and the word NAKIL could not be found in the dictionary.

A more interesting case is caused by some roots which may deform or not depending on the meaning that they carry. Such roots obey vowel harmony rules when they are not deformed, but not when they are deformed (e.g., AD, KALP). For such roots, the flags to be checked are IS_UUU, IS_STT, and IS_SDD. Therefore, while all the words ADI (AD: name), ADDİ (AD: count), KALPI (KALP: unreliable), and KALBİ (KALP: heart) are correctly spelled, the words ADDI,[14] KALPİ, and KALBI can not pass the vowel harmony check.

4.5.2 Other Checks

To perform the other morphophonemic checks, the suffixes must be determined. Because of this, these checks are done during morphological analysis, after each suffix is isolated. During the lexical analysis, the suffixes are matched in their surface forms. Thus, if any of the allomorphs of a suffix can be matched, it is sent to the parser without checking whether the correct form of it is used. These checks are done within the parser. Since the vowel harmony check is done beforehand, only the remaining morphophonemic checks must be done at that point. The consonant harmony checks are among these checks (see Section 3.3.2).

Consider the words YAPDIKÇA, YAPTIĞÇA, YAPTIKCA, YAPTIĞCA,

[13]The word SOL is pronounced slightly different in the latter.

[14]The word ADİ passes the check because such a word is present in the dictionary.

and YAPTIKÇA. For all of them, the root will be determined as the verbal root YAP (do). Additionally, all will pass the vowel harmony check. Furthermore, for all of them the suffixes will be isolated as the participial suffix -{D}{I}{K} and the external case suffix -{C}{A}, respectively, and they form a valid sequence of suffixes for a verbal root. However, it is obvious that only one of them (YAPTIKÇA) has the correct spelling. In order to recognize the misspelled ones consonant harmony checks must be done. When the suffix -{D}{I}{K} is isolated, since it is a suffix whose initial phoneme changes depending on the phoneme preceding it, the last phoneme of the root YAP is checked. Since it is a harsh consonant, the suffix must begin with the consonant T. Therefore, the word YAPDIKÇA can not pass this check. In addition, the last phoneme of that suffix changes depending on the phoneme it precedes. Since it is followed by a consonant, it must end with the harsh consonant K. Hence the spelling of the word YAPTIĞÇA is also wrong. Later comes the suffix -{C}{A} whose first phoneme depends on the last phoneme of the stem it is affixed to. The word YAPTIKCA can not pass this check because although the suffix -{C}{A} comes after the harsh consonant K, it does not begin with the harsh consonant Ç. At this point, a shortcoming of the checker arises. If two consonant harmony errors immediately follow each other, the checker can not catch them. For example, in the word YAPTIĞCA, since both suffixes are used incorrectly, and this had caused a harmony between their consonants, the word will not be reported as misspelled, although it is.

Usage of passing vowels or consonants are also checked during morphological analysis (see Sections 3.3.1 and 3.3.2). For example, during the morphological analysis of the word GELİYORKEN (while [the person] is coming), when the first suffix is determined as the progressive tense suffix -[{I}]YOR, since the passing vowel {I} is used, the last phoneme of the root is checked to understand whether it really ends with a consonant. Later, the participial suffix -[Y]KEN is isolated. Since the passing consonant Y is not used, the phoneme preceding it is checked to see if it is really a consonant. If this word were written as GELYORKEN, GELİYORYKEN, or GELYORYKEN, it could not pass the morphophonemic checks, although it obeys to vowel harmony rules and the order of the morphemes are correct.

If a word can not pass any of the morphophonemic checks, considering the possibility that the root may have been determined wrongly, a new root is searched in the dictionary, and the process is repeated.

4.6 Morphological Analysis

What characterizes agglutinative languages is that stem formation by affixation to previously derived stems is extremely productive, so that a given stem, even though itself quite complex, can generally serve as the basis for even complex words. Consequently, agglutinative languages contain words of considerable morphological complexity, and spelling error detection for such languages necessitates a morphological analysis.

4.6.1 Morphological Parsing

Morphological parsing has attracted relatively little attention in computational linguistics until recently. This attitude is predictable from the fact that virtually all syntactic parsing research has been concerned with English, or with languages morphologically very like English. Major properties of morphological parsers can be given as follows [17]:

1. A morphological parser requires a morphophonological component which mediates between the surface form of a morpheme as encountered in the input text and the lexical form in which the morpheme is stored in the morpheme inventory, i.e., a means of recognizing variant forms of morphemes as the same.
2. A morphological parser also requires a morphotactic component which specifies which combinations of morphemes are permitted.

Morphological parsing algorithms may be divided into *affix stripping* and *root-driven* analysis methods. Both approaches have been taken from very early in the history of morphological parsing as we learn from the Hankamer [17]:

> Packard's parser [32] for <u>ancient Greek</u> 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 [4] apply a similar method to the analysis of <u>Finnish</u>, 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 [37], 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 parsing of agglutinative languages were developed independently: for Quechua [20, 21], for Finnish [22], and for Turkish [15]. These three approaches are identical in the way that they treat morphotactics. 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 iteration of this process, and if the last suffix analyzed is one which may end a word, the parse is successful.

Köksal, in his thesis [23], has also suggested the same approach for automatic analysis of Turkish words. We also use a very similar method. Our spelling checker has two separate sets of rules for the two main root classes. When the root of a word is found the class of the root determines which set of rules are to be used for further parsing.

4.6.2 Utilities Used

For the implementation of the lexical analyzers and parsers in which the rules are included, two standard UNIX utilities, *lex* and *yacc*, have been utilized respectively [26, 38]. *Lex* and *yacc* were designed as tools to help programmers writing compilers and interpreters, but they have a wide range of applications.

Lex, so called because it generates a lexical analyzer, reads a stream of bytes and groups them into tokens. The user provides a set of high-level, problem-oriented specifications for regular expression matching, and *lex* produces a program in C programming language which recognizes those regular expressions. We have used it to separate the suffixes of a word from left to right.

Yacc (which stands for Yet Another Compiler–Compiler) is used to codify the grammar of a language, and generates a parser. The parser examines the input tokens and groups them into syntactical units. The value of the tokens may be processed by action routines written in C. We have used *yacc* to parse the suffixes using morphological rules of Turkish grammar.

4.6.3 Lexical Analyzers

Two sets of *lex* specifications, one per each root class, are prepared to generate the lexical analyzers which are to be called by the parsers each time a new token is needed. The specifications contain regular expressions that match suffix tokens. The lexical analyzer corresponding to the category of the current stem sends, as the next suffix token, the maximum length substring from the left of the remaining suffixes part that matches to any allomorph of a suffix in the permitted class.

The following is a small section from the *lex* specification[15] for verbs:

```
A          [AE]
I          [iIUu]

.
%%

.
M{A}L{I}   return (MALI);
M{A}       return (MA);
```

Using this specification, the first suffix token of both the words YAPMALISIN

[15]This specification consists of two parts as *definitions* and *rules* section, which are seperated by the symbol %%. The definition part contains some *substitutions* which define regular expressions employed in the rules section. These definitions are then referenced by placing braces ({}) around the desired substitution string. For detailed information on *lex* specifications refer [26] or [38].

(you must do) and GELMELİYİM (I must come) is isolated as the necessitative suffix –M{A}L{I}. Thus, although the suffix –M{A} is also a substring of those words, since its length is less than the suffix –M{A}L{I}, the longest one is matched. If the wrong allomorph of the suffix were used in one of these words, for instance, if the first one were written as YAPMELİSİN, it would be recognized during vowel harmony check.

The morphotactic structure of some words can be analyzed in more than one form, but for our purpose, the real morphotactic structure of the word is not important. Thus, if a word can be analyzed correctly in one form, no other possible structures are analyzed. For example, the word EVİNİN may be analyzed into two morphotactic structures as

EV + [S]{I} + [N]{I}N → EVİNİN (his house's), and
EV + [{I}]N + [N]{I}N → EVİNİN (your house's).

But using the following *lex* specification prepared for nouns, it is analyzed as in the second form.

```
I       [iIUu]

.
%%

.
N{I}N   return (NIN);
{I}N    return (IN);
N       return (N);
```

Similarly, the maximum length suffix matched for the word KAPININ (the door's, or your door's) is the genitive suffix –[N]{I}N, although that word may have been formed by combining the suffixes –[{I}]N and –[N]{I}N.

Although all the conjugational suffixes have been included into the specifications and the rules, only a small subset of the derivational suffixes have been handled. The reasons for this are that majority of the derivational suffixes may be received by only a small group of roots, and determining such groups is a rather difficult and time-consuming job, and depends on various semantic criteria (see Section 3.4). The derivational suffixes that may be affixed to all of the roots in a class and those which can be affixed to large percentage, but not all, of the roots in their class are among the included suffixes.

The lists of all the suffixes included into the grammar rules for each root class can be found in Appendix A. Certain combinations of these suffixes are matched as if a single suffix token by the lexical analyzers, so that some rules can be simplified. For example, the combination of the negation suffix with the progressive tense suffix is matched as a single suffix -M{I}YOR, to eliminate the check for the deformation of the negation suffix (see page 40). On the other hand, some suffixes are formed by the combination of more than one tokens sent by a lexical analyzer. For example, instead of matching the third person plural possessive suffix -L{A}R{I} as a single suffix token, when the lexical analyzer for nouns sends the third person singular possesive suffix -[S]{I} after the plural suffix -L{A}R, their combination is treated as the suffix -L{A}R{I}.

4.6.4 Parsers

The grammar rules for morphotactics of Turkish words have been described in two *yacc* specifications, again one for each root class. The lexical analyzers described in the previous section produce the suffix token stream. *Yacc* generates the source files for the parsers. As a result, two parsers, a noun parser and a verb parser, have been constructed.

All the models in Section 3.4 have been utilized in for generating the rules used in the parsers. Additionally, all of the known exceptional cases, which are also mentioned in the same chapter, have been considered. The correct order of suffixes are coded as grammar rules, and necessary checks are done by the help of action routines associated with the rules. Those routines are executed each time the rule is matched. For example, when the lexical analyzer for the noun parser sends -{C}{I} as the suffix token for the word KİTAPÇI (book seller), first the IS_CI flag of the root KİTAP (book) is checked to understand whether that root can really receive the suffix -{C}{I}. This flag is set for this root, but one more check is necessary to determine whether the correct allomorph of the suffix is used. The value of the vowel in the suffix has been proven to be correct by the vowel harmony check, therefore, it is only necessary to prove that the suffix must really begin with the consonant Ç in its this usage. Therefore, the final phoneme of the stem it is affixed to is checked, and when it is seen that it is the harsh consonant P, Ç is proven to be the correct allophone for {C}, i.e., the correct allomorph of the suffix is used. If the word were written as KİTAPCI it would not have passed this check. On the other hand, the word SEVİNÇÇİ will not be parsed correctly because the nominal root SEVİNÇ (happiness) is not marked in the dictionary as a root which can receive the suffix –{C}{I}.

To check whether the correct allomorph of a suffix is used is relatively simple if only the phonetic conditions are to be considered. For the suffixes whose allomorphs change depending on certain rules, such as the factitive verb suffix, passive voice verb suffix, and aorist suffix (see Tables 3.7, 3.8, and 3.9), extra checks must be done. As an example, let's consider the aorist suffix. When the lexical analyzer for the verb parser sends the aorist suffix as the current suffix token, the parser controls whether the correct allomorph of the suffix is used depending on the stem it is affixed to. If the –R allomorph of the suffix is used, the final phoneme of the stem it follows must be a vowel (e.g., OYNA<u>R</u> (he plays)). If the –{I}R allomorph is used, the stem it is affixed to must end with a consonant, and must contain more than one syllables but must not be a compound verb formed with the verb ETMEK, i.e., the flag IS_GER must not be set for that root (e.g., KAYBOL<u>UR</u> (he disappears)), or must be a mono-syllabic root for which the IS_GIR flag is set (e.g., VER<u>İR</u> (he gives)). Otherwise, if the –{A}R allomorph is matched, the stem must again end with a consonant, but this time must be mono-syllabic and the IS_GIR flag must not be set (e.g., YAP<u>AR</u> (he does)), or it must be a compound verb formed with the verb ETMEK (e.g., HİSSED<u>ER</u> (he feels)). As a result of this check the incorrect words such as KAYBOLAR, VERER, YAPIR, HİSSEDİR will be detected.

As an example for difficulties faced during such checks, consider the passive voice suffix –{I}N, and the second person plural suffix for the imperative form of verbs, i.e., –[Y]{I}N. These two suffixes may sometimes take the same form as in the word BULUN. In this word, the suffix –UN may be either of the suffixes –{I}N or –[Y]{I}N. Since the passive voice suffix takes different forms depending on the stem it follows, some checks must be done when any of those forms are matched. If the suffix –UN is considered as the passive voice suffix, the check will be successful since the root BUL ends with the consonant L (see the second row of Table 3.8). If the other possibility is considered, the word will again be parsed correctly since the person suffix must be the last suffix. On the other hand, while the word KAPATIN is being parsed, if the suffix –IN is considered to be the passive voice suffix, it can not pass the check, where it will be parsed correctly if it is considered as the person suffix. To solve this problem, when the suffix –{I}N is matched as the last suffix of a word, it is decided to be the person suffix, and therefore, no check for the passive voice suffix is done. Otherwise, if there exists any suffix following that suffix, it is considered to be the passive voice suffix and the check is done.

The two parsers are alternatively used. First parser to be used is determined according to the class of the root, but as the parsing continues it may be necessary to switch from one parser to another and continue there, or again pass back to the previous one, since the class of a stem can change when it receives certain suffixes. For example, while parsing continues in the noun parser, if the derivational suffix -L{A}Ş, which makes a verb from a noun, is matched, a jump to the verb parser must be done. Such jumps are not possible using the C code generated by *yacc* as it is, so some modifications are done in that code automatically after each time it is generated.

The switches between parsers can sometimes be very complicated. Some suffixes can have two different usages. For instance, the suffix -M{A} can either make a verb a noun or negate it (see page 50). In such cases both possibilities have to be considered. For example, after the root of the word YAPMADIM (I didn't do) is determined as the verbal root YAP (do), the first suffix will be isolated as -M{A} in the verb parser. First considering the possibility that this suffix is used as a derivational suffix, the noun parser will be invoked. The remaining part of the word can not be parsed by this parser. So accepting -M{A} as the negation suffix, the verb parser will be returned to and parsing will be continued there. On the other hand, since the same suffix is used as a derivational suffix in the word YAPMANIZ (your doing), this word will be parsed successfully in the noun parser, thus returning to the verb parser will not be necessary.

If a word has received more than one derivational suffixes then many switches between parsers will be necessary. In Table 4.10 an example to such switches is given. In that example, the root of the word ÇEKOSLOVAKYALILAŞTIR-MADIKLARIMIZDANMIŞSINIZ (you had been one of those whom we did not convert to a Czechoslovakian) is found as the noun ÇEKOSLOVAKYALI (Czechoslovakian) in our dictionary. Then comes the suffix -L{A}Ş, therefore, a switch to verb parser has to be made. Parsing continues there until the suffix -M{A} is matched. Supposing that this suffix has changed the class of the stem, the noun parser will be returned back. Since the remaining part can not be parsed there, the verb parser is activated, and parsing will continue there considering -M{A} as the negation suffix. Then comes the suffix -{D}{I}{K}, which is also a suffix that makes a noun from a verb, therefore, again a switch to the noun parser will be made. Continuing in this parser, the word will be

Input Word: ÇEKOSLOVAKYALILAŞTIRMADIKLARIMIZDANMIŞSINIZ
Root: ÇEKOSLOVAKYALI

Input for Noun Parser	Input for Verb Parser
LAŞTIRMADIKLARIMIZDANMIŞSINIZ	TIRMADIKLARIMIZDANMIŞSINIZ
DIKLARIMIZDANMIŞSINIZ	DIKLARIMIZDANMIŞSINIZ
LARIMIZDANMIŞSINIZ	

Table 4.10: An example to parsing process and switch between parsers

parsed correctly.

For the roots that can take all the suffixes belonging to both nominal or verbal classes, if parsing is unsuccessful in the first parser chosen, the other one must also be tried. For example, the root of the word AÇLAR (hungry people) is AÇ. This root may either be used as a verb (open) or as a noun (hungry). Parsing is first attempted with the verb parser, but it results unsuccessfully. So we backtrack and use the other parser. With the noun parser the word can be parsed successfully.

In Figure 4.5 an example *yacc* specification[16] is given. These rules appear within the grammar rules for the nominal roots. They are used to parse a word whose root is a numeral. The terminal SAYI indicates that a numeral root has been matched. The rules for the suffixes that a numeral root can receive are represented by the non-terminal sayi_ek. The rules for the non-terminal sayi_isim says that a numeral root stays as a noun if it receives the suffixes -[{I}]NC{I} (the token INCI), -L{I}{K} (the token LIK), or a combination of them: e.g., BİRİNCİ (first), BEŞLİK (set of five), ÜÇÜNCÜLÜK (third place). The suffix -[{I}]NC{I} must take the form -NC{I} when it follows a root ending with a vowel (e.g., İKİNCİ (second)). Because of this, the usage of the passing vowel {I} is checked by the routine Check_I. The non-terminal sayi_fiil shows that by

[16]This specification consists of two parts as *declerations* and *rules* section, which are seperated by the symbol %%. Token definitions in the declerations section describe all possible tokens that the lexical analyzer will return to the parser, thus the *terminals*. The concatenation and/or union of these tokens form *nonterminals*, which may themselves be used as tokens in other rules. Actions can be associated with a rule. An action consists of C code that will be executed each time the rule is matched. For detailed information on *yacc* specifications refer [26] or [38].

affixing the suffix -L{A} or -L{A}T (the tokens LA and LAT respectively) to a numeral root, a verb can be derived: e.g., KIRK<u>LA</u>MAK, DÖRT<u>LET</u>MEK. The suffix -[Ş]{A}R (the token SAR) may be affixed to a numeral root either alone or after combining with one of the suffixes -L{I}{K} or -L{I} (the tokens LIK and LI respectively): e.g., ALTI<u>ŞAR</u> (six each), YEDİ<u>ŞERLİ</u> (with seven each), YÜZ<u>ERLİK</u> (able to contain hundred each). Since the consonant Ş is only used in this suffix when it is affixed to a root ending with a vowel, its usage is checked by the routine Check_SAR. If the suffix -L{I} comes immediately after a numeral root, if it is followed by the substring YOR it may be the deformed form of the suffix -L{A} (e.g., KIRK<u>LI</u>YORLAR), therefore, a call to the verb parser is done, otherwise the class of the stem remains as a noun.

In current implementation, the grammar for verb parser consists of 230 rules in which 80 terminals and 81 nonterminals are used, and in the grammar for noun parser, 263 rules, in which 68 terminals and 94 nonterminals appear, are present. Simplifying both grammars may be possible by examining all the rules carefully and eliminating the unnecessary ones (if any).

Figure 4.6 shows the details of the word analysis. Summarizing, first the syllable structure of the word is checked. If it is wrong, the word is added into the output list of misspelled words, otherwise the root is determined. If no root can be found the word is reported as misspelled. If a root is found, first the vowel harmony check is done. Then, according to the class of the root, one of the parsers is activated. In the parsers, as the suffixes are isolated one by one, necessary morphophonemic checks are done. Depending on the suffixes, switches between the parsers are possible. When the end of the word is reached, if no errors can be found then the spelling of the word is correct. If any error is found in any of the parsers or during morphophonemic checks, a new root is searched. If another root is found the same operations are done. If no successful parsing can be done although the first letter of the word is reached, the word is added into the output list.

```
.
% token SAYI SAR INCI LIK LAT LA LI

.
%%

.
ad          :  SAYI sayi_ek
            ;

sayi_ek     :  sayi_isim       { call_isim; }
            :  sayi_fiil       { call_fiil; }
            :  sar sayi_oth
            :  LI              { if (Next_YOR) call_fiil; else call_isim; }
            ;

sar         :  SAR             { Check_SAR; }
            ;

sayi_isim   :  INCI            { Check_I; }
            :  INCI LIK        { Check_I; }
            :  LIK
            ;

sayi_fiil   :  LAT
            :  LA
            ;

sayi_oth    :  LIK
            :  LI
            ;
```

Figure 4.5: *Yacc* specification for numerals

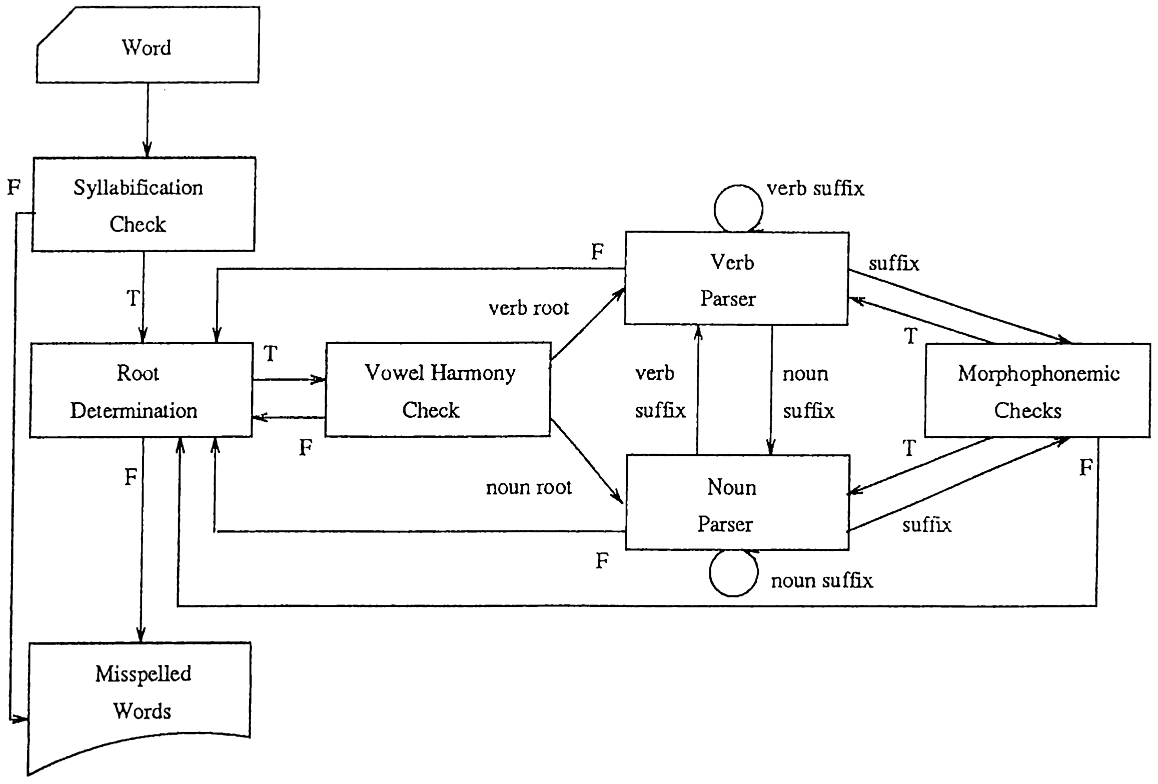

Figure 4.6: Word analysis

Chapter 5

PERFORMANCE EVALUATION

This spelling checker has been implemented using the C programming language in a UNIX environment, on SUN SPARC workstations at Bilkent University.

The current version of the spelling checker requires approximately 850 Kbytes of main memory space. More than 50 percent of this space is taken by the dictionary. Each entry of the dictionary takes 20 bytes on the average: 4 bytes for the pointer to the string which holds the word, 7 bytes on the average for that string, 1 byte for the end-of-string character, and 8 bytes for two long integers which hold the flags. Since the current dictionary contains nearly 23,500 entries, it requires nearly 470 Kbytes of memory space. In the present implementation the whole dictionary is kept in the main memory as it does not cause any problem, but the system is flexible so that when necessary, other storage techniques can be used.

The checking kernel can be integrated to different word processing applications or it can be used as a separate application. We have integrated it to GNU-EMACS text editor for use on LaTeX documents. In this form, the program is available for use within the university and around a number of sites on Internet.

It is also possible to obtain some statistical information by running the program with -s option. Table 5.1 presents certain statistical information obtained from the test runs of the checker with 10 different documents on different subjects, such as medicine, computer engineering, children psychology, etc. As can be seen, the number of distinct words within a document is relatively small, and more particularly, the percentage of distinct words to total words processed increases as the length of the document decreases. Most of those documents were prechecked manually, so the percentages of misspelled words

Input	Total # of words	Number of distinct words		Number of misspelled words		# of msp. wds. detected by syll. check		# of msp. wds. detected by other checks		Total time (sec)
DOC_1	13,677	3,111	(22.75%)	636	(20.44%)	245	(38.52%)	391	(61.48%)	4.91
DOC_2	7,430	3,312	(44.58%)	63	(1.90%)	19	(30.16%)	44	(69.84%)	4.64
DOC_3	6,519	2,473	(37.94%)	157	(6.35%)	51	(32.48%)	106	(67.52%)	3.51
DOC_4	6,349	1,984	(31.25%)	63	(3.18%)	35	(55.56%)	28	(44.44%)	2.71
DOC_5	5,311	2,063	(38.84%)	350	(16.97%)	128	(36.57%)	222	(63.43%)	2.59
DOC_6	4,303	1,486	(34.53%)	97	(6.53%)	51	(32.48%)	46	(67.52%)	2.27
DOC_7	2,658	1,292	(48.61%)	135	(10.45%)	57	(42.22%)	78	(57.78%)	1.53
DOC_8	1,175	546	(46.47%)	78	(14.29%)	37	(47.44%)	41	(52.56%)	0.81
DOC_9	941	701	(74.50%)	0						0.75
DOC_10	535	401	(74.95%)	66	(16.46%)	21	(31.82%)	45	(68.18%)	0.50

Table 5.1: Statistical information for test runs of the checker

are small. Approximately 40% of the misspelled words are detected by syllabification check and the rest are detected by other checks. The last column of Table 5.1 shows the total CPU times in seconds that the program spends while checking the documents. The number of distinct words affect the execution time more than the total number of words. As seen in Figure 5.1, the execution time increases proportionally as the number of distinct words within a document increases. This is an expected result, because a word is fully analyzed only once (see Section 4.2.1). The execution times listed in Table 5.1 are taken on SUN SPARC SERVER 490, which is a machine of 22 MIPS. In the second column of Table 5.2, the average CPU times in milliseconds, spent per word on the same machine are seen. These times include the times spent for input/output, preprocessing, etc. The CPU times spent per word analysis are listed in the third column. In general, the spelling checker can process at 1000-3000 words (roughly 2-6 pages) per second on this system, depending on the document.

In Table 5.3, some information on each function of word analysis, obtained from the tests of the first and last documents, are given. As seen in this table,

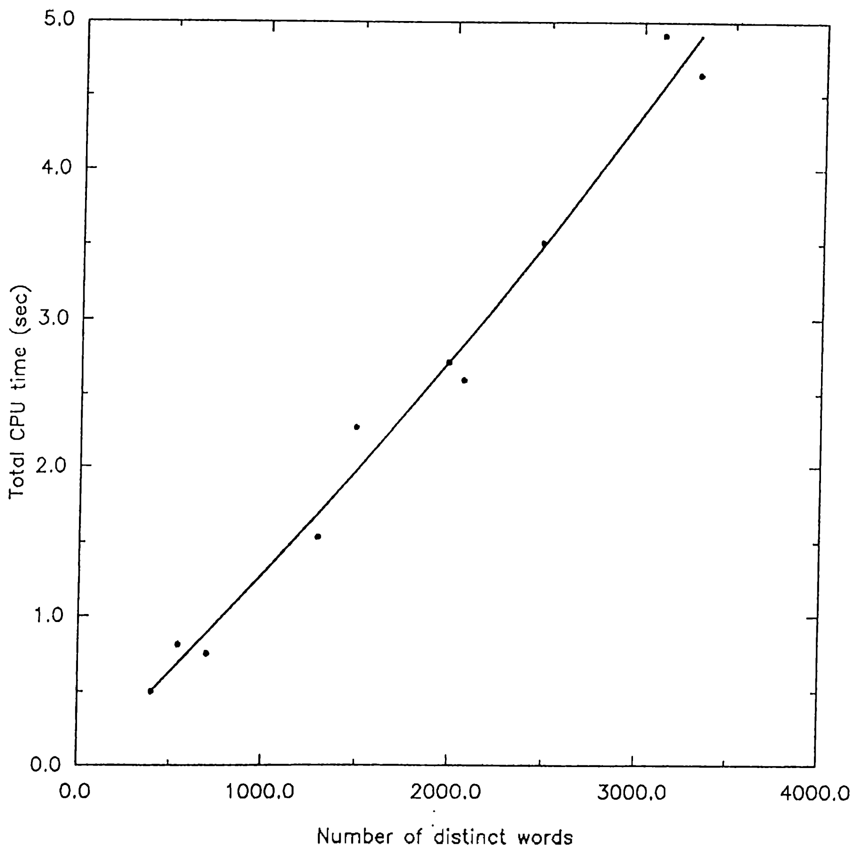

Figure 5.1: Change of execution time as an effect of the number of distinct words

the greatest percentage of the time taken by word analysis is spent for root determination, because root determination requires multiple searches in the dictionary (see page 75). Syllabification check may sometimes be an overhead, especially when the percentage of the misspelled words to distinct words is too small, since this check is applied to all of the distinct words.

The checker sometimes reports some correctly spelled words as incorrect since those words are not included in the dictionary. For example, although its spelling is correct, the checker reports the word *anatomik* as misspelled (see Appendix B) because that word is not included in the dictionary. This problem can easily be solved by adding the necessary words into the dictionary or by allowing a user dictionary to be created. On the other hand, there are still certain misspellings which may not be detected by the checker, because some

Input	SUN SPARC SERVER 490 (22 MIPS)		
	Total time (sec)	Time/ word (msec)	Time/ wd.an. (msec)
DOC_1	4.91	0.36	0.23
DOC_2	4.64	0.62	0.41
DOC_3	3.51	0.54	0.36
DOC_4	2.71	0.43	0.29
DOC_5	2.59	0.49	0.32
DOC_6	2.27	0.53	0.36
DOC_7	1.53	0.57	0.37
DOC_8	0.81	0.69	0.42
DOC_9	0.75	0.80	0.57
DOC_10	0.50	0.93	0.63

Table 5.2: Timings

flags are not fully set for all of the words in the dictionary. For example, after the verbal roots are marked correctly, all the remainings are marked as nominal roots even though some of those roots can not really receive all of the suffixes that a nominal root can take. The functional performance of the spelling checker can be fine-tuned by analyzing the word list and inserting the additional appropriate flags. Consequently, the reliability of the spelling checker can be improved by adding more words into the dictionary and increasing the number of flags.

Function	DOC_1			DOC_10		
	% of word analysis time	# of calls	Time/call (msec)	% of word analysis time	# of calls	Time/call (msec)
WordCheck	100.0%	13,677	0.23	100.0%	535	0.63
ExttoInt	2.8%	3,130	0.03	8.9%	403	0.07
SearchHash	6.7%	13,677	0.02	0.0%	535	0.00
InsertHash	2.6%	3,092	0.03	2.2%	399	0.02
CheckSyl	7.6%	3,111	0.08	8.9%	401	0.07
RootDet	36.4%	3,456	0.33	34.7%	452	0.26
VowelHarm	6.2%	2,483	0.08	7.4%	270	0.09
NounParser	13.1%	2,072	0.28	15.9%	193	0.28
VerbParser	9.3%	1,401	0.28	9.6%	130	0.25

Table 5.3: Some information on each function of word analysis

Chapter 6

CONCLUSIONS AND SUGGESTIONS

In this thesis, we have presented design and implementation of a spelling checker for Turkish.

Today, numerous spelling programs for several natural languages are available as various word processors on the market. Computer users are increasingly utilizing such functionalities. Although it is obvious that such a tool for Turkish users is also necessary and will be very useful, no such program has been developed until recently. The reason is that, due to its agglutinative nature, Turkish presents special difficulties not encountered in spelling checkers for other languages such as English. In those languages, spelling errors are mostly caused by the difference between how a word sounds and is actually spelled, but Turkish words are written the same as they sound. In Turkish (and in other agglutinative languages) spelling errors are caused by some grammatical aspects of the language. Turkish words include an important amount of grammatical information embedded by the addition of suffixes to a certain root. Incorrect root-suffix combinations, wrong ordering of the suffixes, and errors in phonetic harmonies introduce spelling errors in Turkish text. Therefore, a series of phonological and morphological analyses have to be performed in order to detect wrong spelling of Turkish words.

Turkish words are formed obeying certain phonetic and morphological rules. It is claimed that those rules are well-defined and Turkish is a very regular language. However, the results of our research have shown that, in addition to its regularity, Turkish as used today shows many irregularities that cause the problem of spelling checking for this language to become a hard and very interesting problem. The results of our research on Turkish word formation rules and their exceptions are given in Chapter 3. These results may hopefully be helpful for future researchers on Turkish linguistics.

Many grammar books have been referred to collect Turkish word formation rules. 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 is softened, but there are some such words whose final consonant does 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. In order to implement that rule correctly in the spelling checker, all words having the indicated property have been examined, the list of the irregular ones have been obtained, and special checks have been done to catch those irregularities.

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 (say) and YE (eat) changes as Dİ and Yİ respectively when they receive a suffix beginning with the consonant Y", it is mentioned nowhere that the root DE does not always obey to this rule. For instance, it does not change when it receives the suffix –[Y]{I}P, i.e., the resulting word is not DİYİP, as said in the rule, but DEYİP. In order to implement that rule correctly, all the suffixes beginning with Y have been examined, those which do not cause DE to change have been somehow decided, and they have been handled specially.

In order to obtain reliable results from the spelling checker, all of the known rules and their exceptions have been implemented, but we have missed some rules. For example, it intuitively seems as if that the interrogative form of a verb in optative mood is not valid for some persons (e.g., GELESİN Mİ?), but that rule is not included in our rules since it is met in none of the grammar books. Hence, later it may be necessary to make minor modifications in our grammar rules.

Some misspellings caused by affixing certain suffixes to some roots, which in fact can not receive them, can not be detected by the spelling checker yet. The reason is that, in the current implementation, all of the roots outside the verbal ones are marked as nominal roots, and they are treated as if they can receive all the conjugational suffixes which can be affixed to nominal roots. However, this is not always true because some of those roots can not receive all of those suffixes. For example, the root HEP (all) does not take the first person singular suffix –[{I}]M although it takes the plural one,[1] i.e., HEPİMİZ (all of us) is correct but HEPİM is not, but the checker can not detect it. To solve this problem, the vocabulary of Turkish must be analyzed very carefully, the root classes must be determined correctly, the number of root classes must probably be increased, and which class can really receive which suffixes must be decided. Obviously, this is a very difficult and time consuming job which requires a good knowledge on Turkish vocabulary, and probably should be left to linguists.

The spelling checker sometimes reports correct words as incorrect. One reason of this is the absence of some words in our dictionary. Although the dictionary is reasonably complete, there still remains many technical terms and proper names which are not included. Adding more and more words will obviously increase the functional performance of the checker. Another reason is that, most of the derivational suffixes are not included into the rules. If a stem that is derived by such a suffix is not present in the dictionary, it is reported as misspelled. Additionally, for the derivation of suffixes that are included in our rules, the list of the roots that they can be affixed to may not be fully determined. This problem can also be solved by examining the dictionary.

The abbreviations are not considered in current implementation. Thus, the words such as *Dr.*, *vb.*, *T.B.M.M.* are reported as misspelled. While the abbreviations are written, both the punctuation and the case distinction are important, for instance, *tbmm* must be detected as incorrect. However all the punctuations in the input are removed before it is checked and no case distinction (except for the first character to check proper names) is present. This problem may be solved by holding the abbreviations in the external representation in a separate table and searching each word in that table before converting it into the internal representation and before removing the punctuation. Since this will form a great overhead for the execution time of the checker, the problem is left unsolved.

As seen in Chapter 5, the performance results of the checker are rather satisfactory. The current dictionary contains some words of usually Arabic or Persian origin which have lost their usage today. If such words are determined and deleted from the dictionary, its size will decrease substantially. Additionally, some of the currently used flags may be unnecessary, and removing them will reduce the dictionary size. Furthermore, some compression techniques may be applied to reduce the storage requirements of the dictionary, but those

[1]This rule is not written anywhere.

techniques must be carefully chosen so that the search time should not be increased. In fact, decreasing the search time may be possible using a different data structure for the dictionary but it may require more space.

The further work to extend the implemented spelling checker might be the development of a spelling corrector for Turkish. As it is known, a spelling corrector is more difficult to develop and maintain even for languages such as English. Some standard algorithms have been developed to give suggestions for typographical errors. Those can be used for Turkish too, but they may not be enough. For spelling correction of morphological errors in Turkish, some intelligent methods must be developed. For example, the word GELMEYECEĞİM (I will not come) should be suggested when the word GELMİCEM is detected as misspelled.

Appendix A

LISTS OF SUFFIXES

Suffixes included in grammar rules for nominal roots:

-L{A}R	-M{I}	-{C}{I}
-[{I}]M	-[Y]{D}{I} (İDİ)	-[{I}]N{C}{I}
-[{I}]N	-[Y]M{I}Ş (İMİŞ)	-[Ş]{A}R
-[S]{I}	-[Y]S{A} (İSE)	-L{I}{K}
-[{I}]M{I}Z	-[Y]KEN (İKEN)	-L{A}T
-[{I}]N{I}Z	-M	-L{A}Ş
-L{A}R{I}	-N	-L{A}
-[Y]{I}	-K	
-[Y]{A}	-N{I}Z	
-{D}{A}	-[Y]{I}M	
-{D}{A}N	-S{I}N	
-[N]{I}N	-[Y]{I}Z	
-[Y]L{A}	-S{I}N{I}Z	
-{C}{A}	-{D}{I}R	
-L{I}		
-S{I}Z		
-Kİ (-KÜ)		

Suffixes included in grammar rules for <u>verbal</u> roots:

-{D}{I}R
-{I}L
-{I}N
-N
-Z
-M{A}
-[Y]{A}M{A}
-[Y]{A}BİL
-[Y]{A}DUR
-[Y]{I}VER
-[Y]{A}GEL
-[Y]{A}YAZ
-[Y]{A}KAL
-[Y]{A}KOY
-[Y]{A}GÖR
-{D}{I}
-M{I}Ş
-[Y]{A}C{A}{K}
-[{I}]R
-{A}R
-[{I}]YOR
-M{A}KT{A}
-S{A}
-[Y]{A}
-M{A}L{I}
-M{I}
-[Y]{D}{I} (İDİ)
-[Y]M{I}Ş (İMİŞ)
-[Y]S{A} (İSE)
-[Y]KEN (İKEN)

-M
-N
-K
-N{I}Z
-L{A}R
-[Y]{I}M
-S{I}N
-[Y]{I}Z
-S{I}N{I}Z
-L{I}M
-[Y]{I}N
-[Y]{I}N{I}Z
-S{I}NL{A}R
-M{A}{K}
-[Y]{I}Ş
-[Y]{A}N
-[Y]{A}S{I}
-{D}{I}{K}
-[Y]{I}P
-[Y]{A}R{A}K
-[Y]{I}NC{A}
-[Y]{A}L{I}
-M{A}D{A}N
-M{A}KS{I}Z{I}N
-C{A}S{I}N{A}
-[Y]{I}{C}{I}

Appendix B

EXAMPLE RUNS

The following text is taken from a news which was pressed in the Hürriyet newspaper on June 1st, 1991. This text is written as it appears in the newspaper in a LaTeX file and checked by the spelling checker using -s option. The input file and the output of the checker can be found in the following pages. The same text is typed by a Turkish speaking foreigner in the required external representation and given as input to the spelling checker. It is a good example to see what kind of spelling errors can be made in a Turkish text, and which of those errors can be detected by the checker.

"Anne karnından hayata sarılış

Geçtiğimiz günlerde San Fransisco'daki California Üniversitesi'nde yapılan bir operasyon sırasında yaşanan ilginç bir olay, ameliyathanede büyük şaşkınlık ve heyecana yol açtı. Üniversitenin Çocuk ve Yeni Doğan Kliniği Şefi Dr. Michael Harrison ve ekibinin anne karnındaki bir bebek (Fetüs) üzerinde gerçekleştirdiği ameliyat sırasında, yaşamla ölüm arasında savaş veren beş aylık minik canlı, henüz gelişmesini tamamlamamış elini uzatarak, doktorunun parmağını sıkıca kavradı.

Çeşitli anatomik bozuklukları nedeniyle yaşam şansları zayıflayan ana karnındaki bebeklerin anomalliklerini düzeltmek için yapılan ameliyatlardan biri olan işlem sırasında, operasyonun gereği olarak embriyonun sağ kolu annenin rahmine yapılan kesikten dışarı çıkarıldı.

Dr. Michael Harrison, o güne kadar on beşten fazla bu tür ameliyat yaptığı halde, henüz anne karnındaki bir bebeğin adeta kurtarıcısına teşekkür ifadesi taşıyan bu sıcak tutunuşuyla, müthiş heyecanlanıp, duygulandığını belirtti.

Son yıllarda doğum öncesi teşhis ve tedavide atılan büyük adımlara ek olarak Dr. Harrison ve ekibinin gerçekleştirdiği, bebeğin çeşitli yapı bozukluklarının ana karnındayken yapılan ameliyatla giderilebilmesi, dünyanın ileri gelen tıp otoritelerince alkışlanacak bir başarı olarak değerlendiriliyor. On yıldır bu konu üzerinde çalışan ve yüzlerce gebe maymun ve koyunla bu tip ameliyatların klinik çalışmasını yapan Dr. Harrison, böylece operasyon tekniğini mükemmelleştirdiklerini söyledi. Dr. Harrison yapılan müdahalenin gerçekten yaşam şansını arttırdığını ve bu arada anneye de zarar verilmediğini ispatladıklarını belirtiyor."

Input LaTeX file:

```
Anne karn{\i}ndan hayata sar{\i}l{\i}\c{s}

Ge\c{c}ti\u{g}imiz g\"{u}nlerde San Fransisco'daki California
\"{U}niversitesi'nde yap{\i}lan bir operasyon s{\i}ras{\i}nda
ya\c{s}anan ilgin\c{c} bir olay, ameliyathanede b\"{u}y\"{u}k
\c{s}a\c{s}k{\i}nl{\i}k ve heyecana yol a\c{c}t{\i}.
\"{U}niversitenin \c{C}ocuk ve Yeni Do\u{g}an Klini\u{g}i
\c{S}efi Dr. Michael Harrison ve ekibinin anne karn{\i}ndaki
bir bebek (Fet\"{u}s) \"{u}zerinde ger\c{c}ekle\c{s}tirdi\u{g}i
ameliyat s{\i}ras{\i}nda, ya\c{s}amla \"{o}l\"{u}m aras{\i}nda
sava\c{s} veren be\c{s} ayl{\i}k minik canl{\i}, hen\"{u}z
geli\c{s}mesini tamamlamam{\i}\c{s} elini uzatarak,
doktorunun parma\u{g}{\i}n{\i} s{\i}k{\i}ca kavrad{\i}.

\c{C}e\c{s}itli anatomik bozukluklar{\i} nedeniyle ya\c{s}am
\c{s}anslar{\i} zay{\i}flayan ana karn{\i}ndaki bebeklerin
anomalliklerini d\"{u}zeltmek i\c{c}in yap{\i}lan ameliyatlardan
biri olan i\c{s}lem s{\i}ras{\i}nda, operasyonun gere\u{g}i
olarak embriyonun sa\u{g} kolu annenin rahmine yap{\i}lan
kesikten d{\i}\c{s}ar{\i} \c{c}{\i}kar{\i}ld{\i}.

Dr. Michael Harrison, o g\"{u}ne kadar on be\c{s}ten fazla
bu t\"{u}r ameliyat yapt{\i}\u{g}{\i} halde, hen\"{u}z anne
karn{\i}ndaki bir bebe\u{g}in adeta kurtar{\i}c{\i}s{\i}na
te\c{s}ekk\"{u}r ifadesi ta\c{s}{\i}yan bu s{\i}cak
tutunu\c{s}uyla, m\"{u}thi\c{s} heyecanlan{\i}p,
duyguland{\i}\u{g}{\i}n{\i} belirtti.

Son y{\i}llarda do\u{g}um \"{o}ncesi te\c{s}his ve tedavide
at{\i}lan b\"{u}y\"{u}k ad{\i}mlara ek olarak Dr. Harrison
ve ekibinin ger\c{c}ekle\c{s}tirdi\u{g}i, bebe\u{g}in
\c{c}e\c{s}itli yap{\i} bozukluklar{\i}n{\i}n ana
karn{\i}ndayken yap{\i}lan ameliyatla giderilebilmesi,
d\"{u}nyan{\i}n ileri gelen t{\i}p otoritelerince
alk{\i}\c{s}lanacak bir ba\c{s}ar{\i} olarak
de\u{g}erlendiriliyor. On y{\i}ld{\i}r bu konu \"{u}zerinde
\c{c}al{\i}\c{s}an ve y\"{u}zlerce gebe maymun ve koyunla bu
tip ameliyatlar{\i}n klinik \c{c}al{\i}\c{s}mas{\i}n{\i}
```

```
yapan Dr. Harrison, b\"{o}ylece operasyon tekni\u{g}ini
m\"{u}kemmelle\c{s}tirdiklerini s\"{o}yledi. Dr. Harrison
yap{\i}lan m\"{u}dahalenin ger\c{c}ekten ya\c{s}am
\c{s}ans{\i}n{\i} artt{\i}rd{\i}\u{g}{\i}n{\i} ve bu arada
anneye de zarar verilmedi\u{g}ini ispatlad{\i}klar{\i}n{\i}
belirtiyor.
```

<u>Output of the check with -s option:</u>

```
Fransisco
daki
California
nde
Dr
Dr Michael
Harrison
Fet\"{u}s
anatomik
anomalliklerini
Dr Michael
Checking this file took 1 seconds.
There were 206 words in this file.
162 ( 78.64%) of the words were unique.
11 (  6.79%) of the distinct words were misspelled.
2 ( 18.18%) misspelled words were detected by syllable structure check.
9 ( 81.82%) misspelled words were detected by other checks.
```

<u>Input file:</u>

```
Anne Karn!indan hayata sar!il!i!s

Ge!cti!gimiz g!unlerde San Fransisco daki California
!Universitesinde yap!ilan bir operasyon sirasinda ya!sanan
ilgin!c bir olay, ameliyathanede b!uy!uk !sa!skanl!ik ve
heycaqna yola!ct!i. !Universitinin !cocuk ve yeni do!gan
klini!gi !sefi Dr. Michael Harrison ve ekipinin anne
karn!indaki bir bebek (Fet!us) !uzerinde ger!cekle!sdirdi!gi
ameliyat siras!inda, ya!samla !ol!um aras!inda sava!s veren
be!s ayl!ik minik canl!i, henuz geli!smesini tamamlamami!s
elini uzatarak, doktorunun parma!g!in!i s!ik!ica kavrad!i.

!Ce!sitli anatomik bozukluklar!i nedenile ya!sam !sanslar!i
zeyiflayan anna karn!indaki bebeklerin anomaliklerini d!uzeltmek
i!cin yap!ilan ameliyatlarda bir olan i!slem siras!inda,
operasyonun gere!gi olarak embriyonun sa kolu annenin rahimine
yap!ilan kesiktan di!sar!i !c!ikar!ild!i.

Dr. Michael Harrison, o g!une kadar onbe!sten fazla butur
ameliyat yapt!i halde, henuz anne karn!indaki bir bebegin adeta
kurtar!ic!is!ina te!s!sekkur ifadesi ta!sayan bu s!icak
tutunu!suyla, muti!s heycanlan!ip, duguland!i!g!in!i belirtti.

Son yillarda do!ugum oncesi te!sis ve tedavide at!ilan b!uy!uk
ad!imlar!i ek olarak Dr. Harrison ve ekibinin
gerek!ceklestirdi!gi bebe!gin !ce!sitli yap!i bozukluklar!in!i
ana karn!indaken yap!ilan ameliyatla giderilebilmesi, d!unyan!in
ileri gelen tip otoritelerince alk!islancak bir ba!sar!i olarak
de!gerlendirliyor. On y!ild!ir bu konu uzerinde !cali!san ve
yuzlerce gebe maymun ve koyunla bu tip ameliyatlar!in kilinik
!cali!smas!ine yapan Dr. Harrison, boylece operasyon tekni!gine
m!ukemmelle!sdirdiklerini soyledi. Dr. Harrison yap!ilan
mudahalenin ger!cekten ya!sam !sans!in!i artt!ird!i!gn!i ve bu
arada anneyede zarar verilmedi!gini !ispatlad!iklar!in!i
belirtiyor.
```

Output:

```
Fransisco
daki
California
sirasinda
!sa!skanl!ik
heycaqna
yola!ct!i
!Universitinin
Dr
Dr Michael
Harrison
ekipinin
Fet!us
ger!cekle!sdirdi!gi
siras!inda
henuz
tamamlamami!s
anatomik
nedenile
zeyiflayan
anna
anomaliklerini
sa
rahimine
kesiktan
di!sar!i
Dr Michael
onbe!sten
butur
bebegin
te!s!sekkur
ta!sayan
tutunu!suyla muti!s
heycanlan!ip
duguland!i!g!in!i
yillarda
do!ugum
oncesi
```

```
te!sis
gerek!ceklestirdi!gi
karn!indaken
alk!islancak
de!gerlendirliyor
uzerinde
!cali!san
yuzlerce
kilinik
!cali!smas!ine
boylece
tekni!gine m!ukemmelle!sdirdiklerini
soyledi
yap!ilan mudahalenin
artt!ird!i!gn!i
anneyede
!ispatlad!iklar!in!i
```

References

[1] Adalı, O., "Türkiye Türkçesinde biçimbirimler", TDK, Ankara, 1979.

[2] Banguoğlu, T., "Türkçenin grameri", TDK, Ankara, 1986.

[3] Bentley, J., "A spelling checker", Communications of the ACM, Vol. 28, No. 5, 456 - 461, May 1985.

[4] Brodda, B., Karlsson, F., "An experiment with morphological analysis of Finnish", Papers from the Institude of Linguistics, University of Stockholm, Publication 40, Stockholm, 1980.

[5] Can, K., "Yabancılar için Türkçe-İngilizce açıklamalı Türkçe dersleri", METU, Ankara, 1987.

[6] Carlson, G., "Techniques for replacing characters that are garbled on input", Proceedings of 1966 Spring Joint Conference, AFIPS Press, 189 - 192, Arlington, 1966.

[7] Davidson, L., "Retrieval of misspelled names in airlines passenger record system", Communications of the ACM, Vol. 5, No. 3, 169 - 171, March 1962.

[8] Damerau, F. J., "A technique for computer detection and correction of spelling errors", Communications of the ACM, Vol. 7, No. 3, 171 - 176, March 1964.

[9] Demircan, Ö., "Türkiye Türkçesinde kök-ek bileşmeleri", TDK, Ankara, 1977.

[10] Deny, J., "Türk Dili Grameri (Osmanlı Lehçesi)", translated by A. Ulvi Elöve, İstanbul, 1941.

[11] Dodds, D. J., "Reducing dictionary size by using a hashing technique", Communications of the ACM, Vol. 25, No. 6, 368 - 370, Feb. 1982.

[12] Durham, I., Lamb, D. A., Save, J. B., "Spelling correction in user interfaces", Department of CS, Carnegie–Mellon University, December 1982.

[13] Freeman, D. N., "Error correction in CORC: The Cornell computing language", PH.D. Thesis, Department of Computer Science, Cornell University, Ithaca, New York, September 1963.

[14] Gönenç, G., Töreci E., "Türkçenin bazı özelliklerinin bilgisayarlarla sayısal çözümlenmesi", Bilişim'75, No. 9, 42 - 78, Ankara, 1975.

[15] Hankamer, J., "Turkish generative morphology and morphological parsing", a paper presented at Second International Conference on Turkish Linguistics, İstanbul, 1984.

[16] Hankamer, J., "Finite state morphology and left to right phonology", Proceedings of the West Coast Conference on Formal Linguistics, Vol. 5, Stanford University, 1986.

[17] Hankamer, J., "Morphological parsing and the lexicon", edited by William Marslen-Wilson, MIT Press.

[18] Hatiboglu, V., "Türkçenin ekleri", TDK, Ankara, 1981.

[19] Horowitz, E., Sahni S., "Fundamentals of data structures", The Pitman Press, Great Britain, 1981.

[20] Kasper, R., Weber, D., "User's reference manual for the C's Quechua adaptation program", Occasional Publications in Academic Computing, Number 8, Summer Institude of Linguistic, Inc., 1982.

[21] Kasper, R., Weber, D., "Programmer's reference manual for the C's Quechua adaptation program", Occasional Publications in Academic Computing, Number 9, Summer Institude of Linguistic, Inc.. 1982.

[22] Koskenniemi, K., "Two-level morphology", University of Helsinki, Department of General Linguistics, Publication No. 11, Helsinki, Finland, 1983.

[23] Köksal, A., "Automatic morphological analysis of Turkish", Ph.D. Thesis, Hacettepe University, Ankara, 1975.

[24] Köksal, A., "Türkçenin özdevimli biçimbilim çözümlemesi", Hacettepe University, Ankara, 1976.

[25] Lewis, G. L., "Turkish grammar", Oxford, 1967

[26] Mason, T., Brown, D., "lex & yacc", edited by Dale Dougherty, O'Reilly & Associates, Inc., USA, May 1990.

[27] McElwain, C. K., Evans, M. E., "The degarbler — a program for correcting machine-read Morse code", Inform and Control, Vol. 5, No. 4, 368 - 384, December 1962.

[28] Morgan, H. L., "Spelling correction in systems programs", Communications of the ACM, Vol. 13, No. 2, 90 - 94, February 1970.

[29] Morris, R., Cherry, L. L., "Computer detection of typographical errors", IEEE Trans. Professional Comm. PC-18, 54 - 64, March 1975.

[30] Nix, R., "Experience with a space efficient way to store a dictionary", Communications of the ACM, Vol. 24, No. 5, 297 - 298, May 1981.

[31] Özel, S., "Türkiye Türkçesinde sözcük türetme ve bileştirme", TDK, Ankara, 1977.

[32] 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, 343 - 355, 1973.

[33] Peterson, J. L., "Computer programs for detecting and correcting spelling errors", Communications of the ACM, Vol. 23, No. 12, 676 - 687, Dec. 1980.

[34] Robinson, P., Singer, D., "Another spelling correction program", Communications of the ACM, Vol. 24, No. 5, 296 - 297, May 1981.

[35] Sagay, Z., "Sözcük çekimi", Bilişim'78, Ankara, 1978.

[36] Sagay, Z., "A computer translation of English to Turkish", M.S. Thesis, METU, Ankara, 1981.

[37] Sagvall, A., "A system for automatic inflectional analysis implemented for Russian, Data Linguistica 8, Almquist and Wiksell, Stockholm, 1973.

[38] Schreiner, A. T., Friedman, Jr., H. J., "Introduction to compiler construction with UNIX", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[39] Sheil, B. A., "Median split trees : A fast look-up technique for frequently occuring Keys", Communications of the ACM, Vol. 21, No. 11, 947 - 958, Nov. 1978.

[40] Solak, A., 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, 1195 - 1202, 1990.

[41] Solak, A., Oflazer, K., "Design and implementation of a spelling checker for Turkish", Proceedings of the Fifth International Symposium on Computer and Information Sciences, Vol. 2, Nevşehir, 1203 - 1212, 1990.

[42] Solak, A., Oflazer, K., "Bilgisayar ortamında hazırlanmış Türkçe metinlerde yanlış yazılmış sözcüklerin bulunması", 8. Türkiye Bilgisayar Kongresi Bildiriler Kitabı, İstanbul, 1991.

[43] Solak, A., Oflazer, K., "Bilgisayarla Türkçe sözcük yazımı kontrolü", a paper submitted to Bilkon'91.

[44] Underhill, R., "Turkish", Studies in Turkish Linguistics, edited by Dan Isaac Slobin and Karl Zimmer, 7 - 21, 1986.

[45] "Türkçe sözlük", TDK, Ankara, 1988.

[46] "Yeni yazım kılavuzu", Ninth Edition, TDK, Ankara, 1977.

[47] "Yeni yazım kılavuzu", Eleventh Edition, TDK, Ankara, 1981.