

# Turkish Word Segmentation Using Morphological Analyzer

M. Oğuzhan Külekci

National Research Institute of Electronics & Cryptology, Kocaeli /Turkey  
kulekci@uekae.tubitak.gov.tr

Mehmed Özkan

Boğaziçi University, İstanbul/Turkey  
mehmed@boun.edu.tr

## Abstract

This paper describes an algorithm to segment an input Turkish string without any spaces, which may be an output of a speech-to-text application, into words by using morphological analyzer. It is quite possible to use the algorithm on other languages, which has a morphological analysis component, as well. Turkish morphological analyzer is designed and implemented as the linguistic engine of the algorithm. The construction of the analyzer proposes a technique that attempts to achieve group wise morpheme recognition instead of searching suffixes one by one in a word.

## 1. Introduction

In speech recognition, segmentation is one of the main problems and it is very difficult to segment a sentence of speech into words merely based on their acoustic features [1]. Linguistic processing has to be deployed to overcome that difficulty. In this study, it is proposed to construct a system for Turkish that will take a character stream, which is supposed to be an output of a speech-to-text application, as the input and return the sentence segmented into words. The problem may be summarized as finding the word boundaries in an input string without any space character. The solution may be stated as recognizing substrings in the input whose concatenations construct the original string where each substring has a valid morphological parse in the language. Language processing is an important task in word recognition systems that are using words as units. For agglutinative languages it is much more difficult as a word can appear in many different forms according to inflections and derivations. It is almost impossible to store every inflected or derived form of a word in the dictionary. Morphological analyzers are the base components of applications to recognize words. Turkish morphology has been studied with different formalisms of finite state techniques. Hankamer [2] had given a finite state transition network implementation, and Güngör [3] implemented augmented transition network formalism. Oflazer [4] had described Turkish with two-level morphology [5]. Furthermore, Köksal [6], Solak [7], and Şehitoğlu [8] have investigated Turkish from a computational linguistics point of view. The morphological analyzers of the listed studies do root matching first and then search for possible suffixes that are concatenated to that root one by one as in example :

'evlerimdekilerden' (from the ones that are in my houses) = 'ev' (root) + 'ler' (plurality suffix) + 'im'(possessive suffix) + 'de' (case suffix) + 'ki' (relative suffix)+ 'ler'(plurality suffix) + 'den' (case suffix). In this study, morpheme search is performed in groups. The same example above will be parsed as 'ev'(root) + 'lerimde' (noun inflection morpheme group) + 'kilerden' (relative morpheme group) with the proposed technique. That type of searching is achieved by preparing all of the valid morpheme combinations in the language divided into some number of groups. Although the space requirements increase, the performance of the morphological analyzer is utilized as a consequence of group vice suffix recognition.

## 2. Turkish Language

Turkish is an agglutinative language with an alphabet of 29 Latin letters. The vowels are 'a', 'e', 'ı', 'i', 'o', 'ö', 'u', 'ü' and the consonants are 'b', 'c', 'ç', 'd', 'f', 'g', 'ğ', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'r', 's', 'ş', 't', 'v', 'y', 'z'. The word structure of the language can be described as suffixing of derivational and inflectional morphemes to the root word. Morphophonemic rules of the language serve some constraints for concatenation of the morphemes. Vowels in the affixed morpheme must be consistent with the preceding vowel in the word stem according to the vowel harmony. Furthermore, some vowel/consonant modifications or deletions in the root or in the affixed morpheme can take place. The detailed description of Turkish morphophonemic processes can be found in Oflazer [9] and it has been represented with two-level rules in Oflazer [4]. The morphotactics of the language is complex, but strictly obeys certain criteria that make it rather easy to be formulized. There are too many words assimilated from foreign languages, which causes exceptions in word formations.

In this study all the words that are not verbs will be labeled as nouns (substantives). The nominal word inflection structure of Turkish can be modeled as in Figure 1:

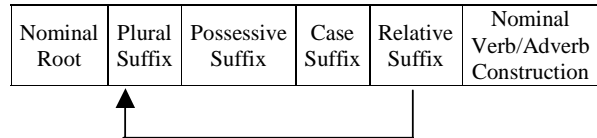


Figure 1: Turkish nominal inflection structure

The exceptional case of the above modeling is sometimes possessive suffix may come before plural suffix (especially to indicate location as in word ‘*babamlar* (home of my father)’). The verb inflection model of Turkish can be described with Figure 2. In written Turkish the suffix group beginning with question suffixes are separated from the word with space (gelmişler miydi?). In this study, the question suffixes ‘-mi’, ‘-mü’, ‘-mu’, ‘-mü’ are taken as separate root entities and marked in the root lexicon with a special tag. Sometimes person suffix and second tense suffix may interchange their positions in the verbal inflectional structure depicted in Figure 2 (‘*geliyorlardı*’ = ‘*geliyordular*’). Complete finite state machines of Turkish verbal and nominal morphotactics can be found in Oflazer [4,9].

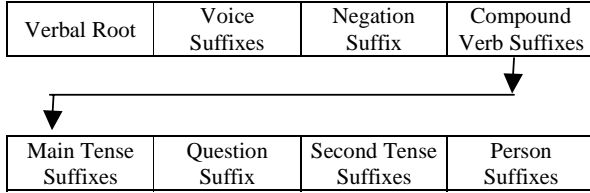


Figure 2: Turkish verbal inflection structure

The derivational word formation in Turkish is very productive and can be structured as in Figure 3.

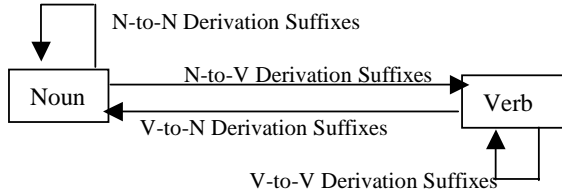


Figure 3: Turkish derivational word construction structure

### 3. Turkish Morphological Analyzer

#### 3.1. Motivation

Let a sample finite state machine be represented as in Figure 4. It is possible to eliminate the nodes B and C if we unify the arcs AB with BD and AC with CD by taking Cartesian products of the labels on each combining arcs. This implies  $\{a_1, a_2\} \times \{b_1\} = \{a_1b_1, a_2b_1\}$  is computed to unify arc AB and BD, and similarly  $\{a_3\} \times \{c_1, c_2\} = \{a_3c_1, a_3c_2\}$  for unification of AC with CD. The resulting schema is represented in Figure 5. Now the search process for a possible path from node A to node D is easy, but the number of labels on the path increased.

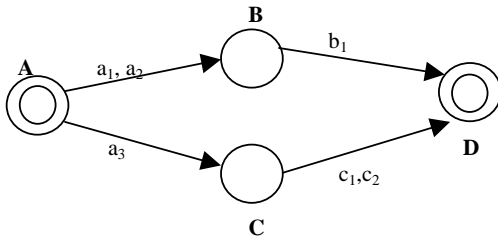


Figure 4: Sample finite state machine

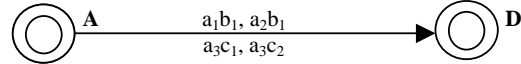


Figure 5: Reduced finite state machine of Figure X

The upside of the proposed method is that it will be efficient since the complexity of the search process is reduced and the morphemes are recognized in groups instead of single morpheme search. On the other hand, space requirements increase as a consequence of storing all possible suffix sequences that are computed with the preprocessing on the finite state machine describing the complete morphology of the language.

#### 3.2. Definitions

Following definitions are used to build up the morphological analyzer with the proposed method.

**Allophonic Allomorph Set (AAS):**

An *allophonic allomorph set* is defined as the collection of the different surface forms of morphemes that behave same morphotactically (allomorphs) and have the same phonemic characteristics. The necessary and sufficient condition for the elements of an allophonic allomorph-set can be stated as:

- a) If the affixation of a morpheme ‘X’ is valid for one of the elements in an allomorph-set, then it must also be valid for all the other elements in the same set.
- b) If one of the elements of an allomorph-set can be affixed to a morpheme ‘X’, then it must also be valid for all the other elements in the same set to be affixed to the morpheme ‘X’.

**Morpheme Group (MG):**

The possible (morphotactically true) concatenations of allophonic allomorph-sets with each other result morpheme groups. The concatenation of two AASs (AAS<sub>1</sub> X AAS<sub>2</sub>) is defined as the Cartesian product of all the morphemes in AAS<sub>1</sub> with all the morphemes in AAS<sub>2</sub>.

**Morpheme Group Generation Table (MGGT):**

Morpheme Group Generation Tables are the look up tables that specify which AAS can be affixed with which AAS. MGGTs are two-dimensional where AASs in the same column are allomorphs and the AASs in the same row have the same phonemic characteristics as we name allophonic. The AASs in the same row can be concatenated. The morphotactic rules of the language specifies which columns of the MGGTs are to be affixed and all possible morpheme groups regarding to that are generated by computing the concatenation of all AASs in the same row of specified columns.

#### 3.3. Method

The morphemes of the language are listed with all of their different surface forms that can be observed according to the morphophonemic rules. Allophonic allomorph sets are constructed from that affix inventory. An example AAS is {‘*im*’, ‘*in*’, ‘*imiz*’, ‘*iniz*’}. The elements of this set are a subset of possessive suffixes in Turkish. Each of those suffixes has the same phonetic characteristics meaning that if one of them can be concatenated to or followed by a morpheme, so does the others. For example, plural suffix ‘-ler’ may be preceded

by all of these affixes ('-lerim', '-lerin', '-lerimiz', '-leriniz'), and also locative case suffix '-de' may be concatenated to them ('-imde', '-inde', '-imizde', '-inizde').

Morpheme group generation tables have to be built up to generate the morpheme groups automatically after forming the AASs. The morphotactics summarized in figure 1 and figure 2 serves the main guidelines that describe which type of AASs can come consecutively. For example, a plural suffix followed by a possessive suffix is valid as shown in figure 1. This implies that the plurality and possessive columns with the related AASs in their rows will exist in the corresponding MGGT. The AASs in the same rows will be concatenated to build the morpheme groups of the specified type.

In this study 7 types of morpheme groups are generated via MGGTs to cover Turkish nominal and verbal inflection structure:

1. Gerund morpheme groups include gerund suffixes.
2. Person suffix morpheme groups include person suffixes that can only be preceded by *Negation & Compound MGs*.
3. Voice morpheme groups include valid combinations of *reciprocal*, *reflexive*, *causative* and *passive* morphemes.
4. Negation & Compound morpheme groups include negation suffixes and the valid combinations of compound morphemes that are indicating *ability*, *impossibility*, *urgency*, *approximation* and *continuity* aspects of the verbal root in Turkish.
5. Time/Model inflection morpheme groups include the combinations of main and second tense/modal verb inflection morphemes along with their negative forms and with the person suffixes. Furthermore, the concatenation of the allomorphs of 'casına', 'ken' (adverb) and 'dır' (aspect) suffixes to any possible morpheme group in this class is also added.
6. Noun inflection morpheme groups include valid concatenations of plural, possessive and case morphemes.
7. Relative Suffix Morpheme Group involves all of the morpheme groups that are valid according to the structure: 'ki' suffix + Plural Suffix + Possessive Suffix + Case Suffix. It is possible to go around the loop a number of times, but that can take place only for some of the relative morpheme groups. For example '...-kiler + kiler' is not valid as 'kiler' morpheme group cannot be followed by another relative morpheme group. However, the case of '...-kilerde + kiler' is valid as 'kilerde' MG can be consecuted by the others. The relative morpheme groups that end with locative suffixes or genitive case suffixes can be nested.

The combined nominal and verbal morphotactics of Turkish with the generated MGs is as shown in Figure 6. While analyzing an input word, each node traversed in the figure represents a morpheme-group concatenation or an empty transition. Note that the machine begins with a nominal or verbal root and may last in any node in the figure.

The morphemes of noun-to-verb, verb-to-noun, noun-to-noun and verb-to-verb derivations are listed separately. In Figure 6 'Verbal stem to noun derivation suffixes' box contains only some of the verb-to-noun derivation suffixes.

The occurrence of derivations in the input word is searched at the beginning of the analysis. After finding possible root

candidates from the word lexicon, each candidate is investigated for a following derivational suffix concatenation in the input according to the figure 6. Every resultant word of derivational suffix concatenation is appended to the root candidate list. When there is no more derivational suffixing for any of the candidates, each of the recognized possible roots is fed into the machine depicted in figure 6 to find any valid parsing of the input.

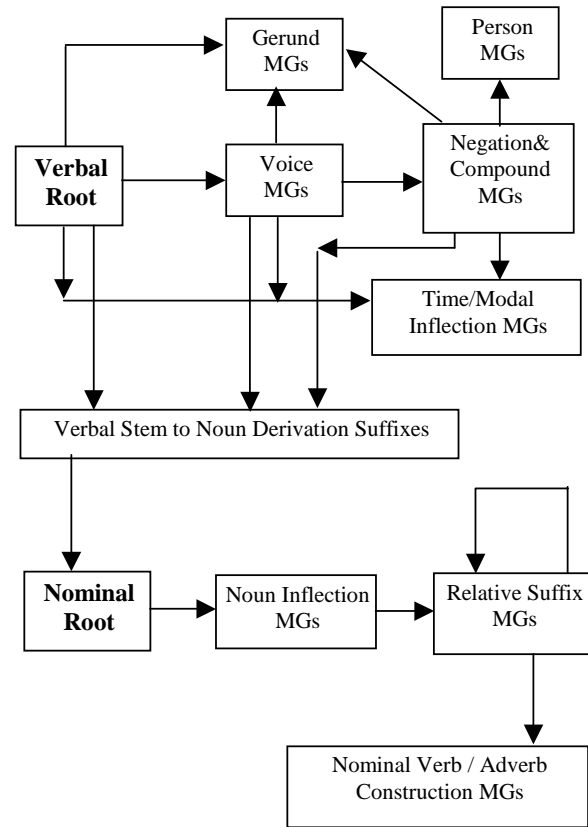


Figure 6: Turkish Morphology with Morpheme Groups

#### 4. Word Segmentation Algorithm

Let string  $S = c_0c_1c_2c_3c \dots c_N$  be the input stream where each  $c_j$  is a character. The step-by-step description of the algorithm to perform the word segmentation on the input string by using the morphological analyzer is as follows:

1. Search the root lexicon to find the words that appear in  $S$ . While doing that search, possible morphophonemic alterations of root words are also taken into consideration (such as Turkish word 'kitap' may be seen in the pattern  $S$  as 'kitab').
2. Construct the set of *cut points*. *Cut points* are defined to be the indices of possible word boundaries in the string  $S$ . That is;  $c_i$  is in the cut point set if there exists a word beginning at position  $i$  in  $S$ . Also add index  $N+1$  to this set to mark the end of the string. The elements of the set are shown as  $CP_i$ .

3. Let  $[c_a c_b]$  ( $a < b$ ) represent the string pattern  $c_a c_{a+1} c_{a+2} \dots c_{b-1}$ . Perform the morphological analysis of  $[c_a c_b]$  substrings for all numbers  $a$  and  $b$  in the cut point set where  $0 \leq a < N$  and  $a < b \leq N+1$ . If the analyzer returns a valid parse, then mark  $cp_a cp_b$  point with 1 in the matrix of Table 1.

$\begin{matrix} & cp_b \\ cp_a \end{matrix}$	$cp_0$	$cp_1$	$cp_2$	$cp_3$	..	$cp_b$	.	$cp_{N+1}$
$cp_0$	0							
$cp_1$	0	0						
$cp_2$	0	0	0					
...	0	0	0	0				
$cp_a$	0	0	0	0	...	1		
...	0	0	0	0	0	0		
...	0	0	0	0	0	0	0	
$cp_{N+1}$	0	0	0	0	0	0	0	0

Table 1. Possible word boundaries matrix

4. Table 1 represents a tree structure in adjacency matrix notation. Depth-first search of that tree for the paths, which are beginning with  $c_0$  and ending with  $c_{N+1}$ , implies the possible concatenations like  $[c_0 c_k] + [c_{k+1} c_{k+n}] + \dots + [c_h c_N]$ . The input string  $S$  is segmented by replacing '+' sign with space character.

*Example:*

Let the string  $S = 'kırmızıbaşlıklıkız'$  (the girl with the red hat) be the input. If we carry out the algorithm step by step:

- 1) The words 'kir', 'kırmızı', 'baş', 'aş' and 'kız' are detected in the input string after the root lexicon search.
- 2) The indices of the first characters of those words are put into possible cut points set along with the number 18 ( $N+1$ ). Now the set contains 0, 7, 8, 15 and 18.
- 3) The substrings  $[0,7[$  (*kırmızı*),  $[0,8[$  (*kırmızıb*),  $[0,15[$  (*kırmızıbaşlıklık*),  $[0,18[$  (*kırmızıbaşlıklıkız*),  $[7,8[$  (*b*),  $[7,15[$  (*başlıklık*),  $[7,18[$  (*başlıklıkız*),  $[8,15[$  (*aşlıklık*),  $[8,18[$  (*aşlıklıkız*), and  $[15,18[$  (*kız*) are fed into the morphological analyzer. The valid parses are returned only for  $[0,7[$ ,  $[7,15[$ ,  $[8,15[$  and  $[15,18[$ . The matrix is formed as in Table 2.

$\begin{matrix} & c_b \\ c_a \end{matrix}$	0	7	8	15	18
0	0	1	0	0	0
7	0	0	0	1	0
8	0	0	0	1	0
15	0	0	0	0	1
18	0	0	0	0	0

Table 2. Possible word boundaries matrix for the exam

- 4) The depth-first search of the tree (which is expressed in the matrix) for the pattern beginning with 0 and ending at 18 results the path  $[0,7[ \rightarrow [7,15[ \rightarrow [15,18[$ . So the word segmentation of the input  $S = 'kırmızıbaşlıklıkız'$  is solved to '*kırmızı başlıklıkız*' which is the correct parse.

## 5. Conclusions

The strength of such a word segmentation system very much depends on the morphological analyzer it employs. While processing an input string with  $X$  number of possible cut points, the system calls the morphological analyzer  $X(X+1)/2$  times. Therefore the speed of the analyzer has to be improved as much as possible. The morphological analysis technique proposed in this study serves for that purpose. Approximately 6000 morpheme groups are generated and used to achieve the construction of the analyzer with a root word list of about 45000. It has been observed that Turkish words are parsed within nanoseconds by the analyzer that is implemented in C on a computer with an Intel PIII processor and 128 MB RAM.

The word recognition algorithm may offer more than one suitable parsing for an input string. Although the resultant words are all valid in Turkish, there exists impropriety according to the context and sentence syntax. Further steps of natural language processing with syntactic and semantic features may be deployed to solve such possible ambiguities.

## 6. References

- [1] Furui, S., *Digital Speech Processing, Synthesis and Recognition*, pp.245, Marcel Dekker Inc., New York, 2001,.
- [2] Hankamer, J., "Turkish generative morphology and morphological parsing", Second International Conference on Turkish Linguistics, İstanbul, 1984.
- [3] Güngör, T., Kuru, S., "Representation of Turkish Morphology in ATN", Proceedings of Second Symposium on Artificial Intelligence and Artificial Neural Networks, pp. 92-104, İstanbul, 1993.
- [4] Oflazer, K., "Two-level Description of Turkish Morphology," Proceedings of the Second Turkish Symposium on Artificial Intelligence and Neural Networks, pp. 86-93, İstanbul, 1993.
- [5] Koskenniemi, K., *Two-level Morphology: A General Computational Model for Word Form Recognition and Production*, Publication No: 11, Department of General Linguistics, University of Helsinki, 1983.
- [6] Köksal, A., "Automatic Morphological Analysis of Turkish," Ph.D. Thesis, Hacettepe University, Ankara, 1975.
- [7] Solak, A., "Design and implementation of a spelling checker for Turkish", MS Thesis, Bilkent University, 1991.
- [8] Şehitoğlu, O. T., Bozşahin C., "Lexical rules and lexical organization" in *Breadth and Depth of Semantics*, E. Viegas (Ed.), Kluwer, Boston, 1999.
- [9] Oflazer, K., Göçmen E., Bozşahin, C., "An outline of Turkish morphology", Technical Report, Bilkent University, 1994