DECSYM'92-Latest Trends in Computing

SPELLING CHECKING IN TURKISH

Selahattin Kuru and H. Levent Akın

Department of Computer Engineering, Boğaziçi University,
80815 Bebek, Istanbul, Turkey
E-MAIL: (KURU, AKIN02) at TRBOUN.BITNET

ABSTRACT

For most languages, including the Indo-European languages such as English and French, spelling checking is done simply by comparing the spelling of the written word against the dictionary. For the so-called agglutinative languages such as Turkish, Hungarian and Finnish, on the other hand, spelling checking requires morphological analysis of words. This paper is mostly devoted to the morphological analysis of Turkish since a morphological analyzer is the essential part of a spelling checker for an agglutinative language. The morphology of Turkish is investigated, approaches to the morphological analysis of agglutinative languages are reviewed, and the structure of a morphological parser is discussed. The spelling checker program that is under development is to be integrated with the All-In-1 office automation system of Digital.

1. INTRODUCTION: THE MORPHOLOGY OF TURKISH

Turkish is an *agglutinative* language, in the same category with Finnish and Hungarian. Words of these languages contain a linear sequence of *morphemes* which are the smallest units of speech bearing a meaning. There is a one-to-one correspondence between the morphemic and the semantic structure of the words formed and words derived by agglutination can still serve as new stems for further affixations. In some instances such words may be carrying an amount of semantic information equivalent to a message made up of several words of another language. A popular example of this is the Turkish word "Çekoslovakyalılaştıramadıklarımızdanmışsınız", which is equivalent to "it is said that you were one of those that we could not convert to a Czechoslovakian."

Rules governing the morphology of agglutinative languages are of two types: The *morphophonemic* rules determining the surface structure of suffixes, i.e. *allomorphs*, and the *morphotactic* rules determining the ordering of morphemes.

1.1 Morphophonemics of Turkish

Morphophonemic rules of Turkish are the *vowel harmony rule*, the *consonant harmony rule*, and the *root deformation rule*.

1.1.1 Vowel Harmony Rule

According to the vowel harmony rule in Turkish, the vowels in the suffixes are members of one of two basic vowel sets V1 and V2 where

V1 = {a,e}
V2 = {ı,i,u,ü}.

The membership is determined according to the final vowel of the root as follows:

V1 = {a}, if the previous vowel is in the set {a,ı,o,u},
V1 = {e}, if the previous vowel is in the set {e,i,ö,ü},
V2 = {ı}, if the previous vowel is in the set {a,ı},
V2 = {i}, if the previous vowel is in the set {e,i},
V2 = {u}, if the previous vowel is in the set {o,u},
V2 = {ü}, if the previous vowel is in the set {ö,ü}.

For example, "kitap" (book) + "-ı" (accusative suffix or 3rd person singular suffix) --> "kitabı" (book, acc. or his/her book). This is because two of the allomorphs of accusative and 3rd person singular suffixes are "-ı" and "-i", and since the last vowel of the word "kitap" is an "a", the allomorph "-ı" is selected in accordance with the rule. There are of course exceptions to the rule: An example is "saati" (clock, acc.) "saat"+"-i"

1.1.2 Consonant Harmony Rule

According to the consonant harmony rule, Turkish consonants are classified into two main sets C1 and C2 as voiceless and voiced consonants, respectively, where

C1={ç,f,t,h,k,p,s,ş}
C2={b,c,d,g,ğ,j,l,m,n,r,v,y,z}.

Then, the rule may be briefly stated as follows: In polysyllabic words and in certain monosyllabic roots, the final voiced consonant in the set {b,c,d,ğ} becomes respectively one of in the set {p,ç,t,k} in the surface form, when it is not directly followed by a vowel; except when an "n" precedes the final "g", "g" becomes a "k" although "g" is not in the domain set, e.g. "rengi" (its color) --> "renk" (color) + "-i" (third person possessive suffix).

1.1.3. Root Deformation Rule

Phoneme insertion and deletion are the most frequently encountered deformations in Turkish. Some morphemes are affixed by inserting one of the auxiliary letters {y,s,n} when two vowels happen to follow each other. This is called phoneme insertion. For example, "bahçeyi" (garden, acc.) is "bahçe" (garden) + "-y" (auxiliary letter) + "-i" (accusative suffix) or "bahçesi" (his/her garden) is "bahçe" (garden) + "-s" (auxiliary letter) + "-i" (3rd person singular possessive suffix).

Some nouns, as well as verbs, both made up of two syllables, the second of which has a vowel in the set V2 also undergo a deletion, or insertion process. When these roots receive a suffix beginning with a vowel, the second vowel that belongs to set V2 drops. For example, "ağz" in the word "ağz-ım" (my mouth) should be in the form of "ağız" (mouth) when it is in isolation. When the root word doesn't take a suffix, the letter "ı" is inserted, in accord with the vowel harmony rules. This problem is, in fact, more complicated, and according to the semantics of the root, deletion may or may not occur.

There are also more systematic deletions in the context of verbal roots and stems ending with a phoneme in the set V1. We may give the following as examples of this: "ağlayacak" (he/she will cry) has an analysis of "ağla" (cry) + "-y" (helping phoneme) + "-acak" (future tense particle), but "ağlıyor" (he/she is crying) gives "ağl(a)" (cry) + "-ıyor" (present continuous tense particle) where phoneme "a" at the end of verb root drops.

1.2 Morphotactics of Turkish

As stated before, morphotactic rules determine affixation of suffixes to root and derived words. The suffixes in Turkish can be classified as *derivational* and *conjugational*. Derivational suffixes alter the meaning and sometimes the class of the word, whereas conjugational suffixes do not.

Conjugational suffixes are grouped according to the root class that they can be affixed to.

The class *noun* takes the plural suffix "-lar", the possessive suffixes "-ım", "-ın", "-ı","-ımız","-ınız", and "-ları", the internal case suffixes "-yı","-ya", "-da", "-dan", and "-nın", the external case suffixes "-yla", "-ca", "-lı", and "-sın", and the relative suffix "-ki", in this order. All of these suffixes are optional.

The class *verb* takes the reflexive voice suffix "-ın", the reciprocal voice suffix "-ış", the factitive voice suffixes "-dır", "-ıt", "-t", "-ır", and "-ar", the passive voice suffixes "-ıl", "-ın", and "-n", The negation suffixes "-ma" and "-ama", the compound verb suffixes "-abil", "-adur", "-ıver", "-agel", "-ayaz", "-akal", "-akoy", and "agör", the main tense suffixes "-dı", "-mış", "-acak", "-ar", "-ıyor", "-makta", "-sa", "-a", "-malı", and "-", the question suffix "-mı", the second tense suffixes "-dı", "-mış", and "-sa", and the person suffixes "-m", "-n", "-", "-k", "-nız", "-lar", "-ım", "-sın", "-ız", "-sınız", "-lım", "-ın", "-ınız", and "-sınlar", in this order.

The class *verbal noun* takes the question suffix "-mı", the tense suffixes "-dı", "-mış", and "-sa", the person suffixes "-m", "-n", "-", "-k", "-nız", "-lar", "-ım", "-sın", "-ız", and "-sınız", and the probability suffix "-dır", in this order.

Turkish allows the formation of nouns, adjectives and adverbs by adding suffixes to verb roots. The suffixes "-mak", "-ma", and "-ış" form nouns, the suffixes "-an", "-acak", "-ası", "-dık", and "-mış" form adjectives, and the suffixes "-ıy", "-arak", "-ınca", "-alı", "-ken", "-madan", "-maksızın", and "-casına" form adverbs.

Derivational suffixes alter the meaning and sometimes the class of the word. They may be grouped according to the class of word that they are affixed to. Adjectives are changed to nouns by the suffix "-lık", to verbs by "-lan", and to adverbs by "-ca", nouns are changed to verbs by the suffix "-la", "-laş", and "-lan", to adjectives by "-lı" and "-sız", verbs are changed to nouns by the suffix "-yış", to adverbs by "-ınca", "-yıp", and "-ya". The suffix "-ımtrag" change the meaning of adjectives, and the suffixes "-cı", "-cık", "-ca", "-daş", "-lık", and "-lı" change the meaning of nouns.

The morphology of agglutinative languages can best be expressed in terms of a finite state transition network. The states (nodes) of such a network are stem (word) categories while the transitions represent suffixes. The transition network is given in the Appendix in terms of a state table of such a transition network.

1.3 Ambiguity in Morphological Analysis

Ambiguity is a complexity that has to be dealt with in morphological analysis of natural languges. As an example, the Turkish word "okuma" is either "ok-um-a" (to my arrow), noun root "ok" (arrow) + 1st person possessive suffix "-(i)m" + dative suffix "-(y)e", or "oku-ma" (do not read, or reading), verb root "oku" (read) + negation suffix or nominating participle "-me". If the root is presented with another morpheme in the form "okumak", the first solution becomes impossible, simply because there is no morpheme following a dative suffix. Then, the word "okumak" might be analyzed as "oku-mak" (to read), verb root "oku" (read) + infinitive suffix "-mek".

Since there exists more than one path leading to the generation of a given word, there are as many possible morphological solutions. The ambiguity brought about by the multiplicity of solutions can only be solved by syntactic or semantic analysis at the sentence level. There may even be cases where this ambiguity cannot be solved at all.

As another example consider the word "evleri". It carries three meanings: First, noun root "ev" + plural suffix "-ler" + accusative suffix "-i", meaning "houses, acc."; the second, noun root "ev" + plural suffix "-ler" + 3rd person singular suffix "-i", meaning "his/her houses"; third, noun root "ev" + 3rd person pluralizer "-ler" + 3rd person singular possessive suffix "-i", meaning "their house". If we analyze the word in the phrase "Ahmet ile Ayşe'nin evleri", an initial syntactic consideration easily indicates that the first cannot be possible in this context. But, one can never choose a meaning for this phrase between "Ahmet and Ayşe's houses" or "Ahmet and Ayşe's house", and consequently a morphological analysis between the second and the third analyses above. Such ambiguities can only be resolved by an additional semantic information coming before or after the phrase.

2. METHODS OF MORPHOLOGICAL ANALYSIS

Morphological analysis methods fall into two main categories, namely listing methods and computational methods. Listing methods are not efficient for agglutinative languages as reported by Hankamer [1,2] .

In the early 1980's morphological parsing of agglutinative languages were developed for Quechua [3], for Finnish [4], and for Turkish [5,6]. The approaches differ in their treatment of morphophonemic alternation, but they are essentially identical in the way they treat morphotactics. In all of these works there are two independent lexicons, one for root words and the other for the suffixes. The software keeps track of the morphotactics of the language and parses words according to its rules embedded within it. Still, these approaches may be divided into two categories depending on their way of approaching the problem of parsing an agglutinative language word. These approaches are named as *Suffix Stripping* (Right-to-Left) and *Root Matching* (Left-to-Right) algorithms.

The suffix stripping parser algorithm needs the whole word to be completely entered and begins the analysis at the end of the word. It tries to match a substring from the end of the word with a conclusive suffix from the suffix lexicon. If any match is found, then the algorithm strips that suffix off the word and tries to find the remainder in the root lexicon. If the remainder can be found in the root lexicon, the parse is successful; otherwise the parse continues by trying to match another substring from the end of the word with a suffix from the lexicon. Suffix matching should be limited by the morphotactics of the language that determines the order suffixes may take.

In the root matching parser, roots are sought in the lexicon that match the initial substrings of the word, and the grammatical category of the word determines what class of suffixes may follow. When a suffix in a permitted class is found to match a further substring of the word, grammatical information in the lexical entry for that suffix further determines 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.

As Hankamer [2] explained, one of the reasons that the left-to-right morpheme recognition algorithm is more universally adopted than the others is that the left-to-right recognition approach narrows the choice of possible suffixes that can combine with a stem of the current stem category at every step. It might be thought that a suffix-stripping strategy enjoys the same advantage, since the recognition of a suffix would narrow down the possible stems to which it could be attached. There is significant asymmetry, however. The set of suffixes determined by a stem is finite (and always very small), while the set of stems determined by a suffix is very large. Every time a suffix is stripped off, the remaining part

of the word needs to be analyzed as one of the stem categories that the morphotactics allows to precede the suffix just removed. Since most suffixes can attach directly to roots, this means that at almost every step in the stripping process, the lexicon must be searched to see if the current remainder is a root. Most initial substrings of a word will not be roots. So, most of these searches will be futile. The larger the lexicon, the more wasteful this process becomes.

3. THE MORPHOLOGICAL ANALYZER

Three major problems facing morphological analysis are that the input string contains no direct indication of where the morpheme boundaries are, that due to morphophonemic alterations, a given morpheme takes a shape dependent on its morphological and phonological environment, and that the morphological information may not be sufficient to resolve ambiguities. Below we discuss the two major components of a morphological analyzer: The parser and the lexicons.

3.1 The Parser

As stated above, morphotactic restrictions are encoded in a *finite state transition network* representation, which defines the classes of well-formed morphemic representations by each transition between nodes, or states. This is done by assigning each root to a basic stem category, which determines the class of affixes that may attach to it. Affixes are assigned to complex categories which combine with stem categories to yield other stem categories. Following Hankamer [7], these stem categories are represented by two characters of form XX, the first character of which denotes the main category a word belongs to, and the second of which denotes its level. For example, N0, V0, A0, and P0 represent nouns, verbs, adjectives, and postpositions, respectively. Note that these are at level 0, which means that they are root categories. The result is a categorial grammar which recursively defines the well-formed stems. This result is equivalent to a finite state transition network in which stem categories correspond to states and affix categories correspond to transitions from states to states. These transitions are represented in the form XXYY, where XX is the initial stem category, and YY is the final category reached after the transformation. All these transitions, altogether with the suffix forms, constitute the suffix lexicon. As stated before, the transition network representation of the lexicon is given in the Appendix.

The parser begins in a designated initial state. It can be in one of the states as a beginning state by recognizing an initial substring of its input matching to one of the roots in its root lexicon. The root category determines a class of affixes that are permitted in the next position. The parser searches an affix lexicon for an affix that is in the permitted class and matches the surface string at the current point. If one is found, a pointer is advanced to the new current point in the word and the parser jumps to the new state, corresponding to the derived stem category, determined by the affix. This process iterates. If the end of the input string is reached and the parser is in a designated final state, the string is said to be successfully parsed. The input word is a well-formed Turkish word.

The morphophonemic alternation is accounted for by a phonological component which mediates between the lexical and surface forms of morphemes. This process is a part of the matching process briefly expressed above. Both roots and affixes are listed in the lexicon in a "lexical" form, which is subjected to a set of phonological rules which convert lexical representations of candidate morphemes into forms consistent with surface environment.

As an example of a moderately complex word according to this morphotactics, consider the input,"çöplüklerimizdekilerden miydi" (was it from those that were in our garbage dumps). The analysis proceeds as follows. First, the root lexicon is consulted, and the form "çöp" (garbage) is found to match the first three segments of the input. The root "çöp" determines a stem category N0. Now, an affix is sought that can combine with a stem of category N0 and matches the initial substring of the remainder of the input form. Such an affix is the affix "-lıg" which matches input due to vowel harmony and final stop devoicing. "-lıg" is in the category N0N0, i.e. it combines with a stem of category N0 to yield a new

stem of category N0. Hence "çöplük" (garbage dump) is analyzed as a stem of category N0. The process now iterates. Among the affixes that can combine with a stem of category N0 is the affix "-la" which matches "le" in the input. This will be tried, but the attempt will lead to failure, because "-la" is an N0V0 affix, and there are no affixes in the affix lexicon that match at the new point in the input and can combine with a stem category of V0. No overt affix leads to a successful parse from this point. The morphotactics, however, provide for certain free jumps between categories. In particular, any stem of category N0 counts as a stem of category N1, there is, thus, a free jump from N0 to N1 in the morphotactic network. Thus, "çöplük" counts as a category N1. From N1, there are jumps to N2 and to N3. If the jump to N2 is taken, the next affix must be the plural affix "-lar"; if the jump to N3 is taken, the next affix must not be plural. Both paths will be attempted, but the one which commences with a jump to N3 will fail, since there is no way to reach the end of the word along that path. The path commencing with a jump to N2 leads to the correct analysis. The only affix that can attach to N2 is the plural affix "-lar", which is the sole occupant of the category N2N3; this matches the "ler" after "çöplük", so "çöplükler" (garbage dumps) is analyzed as a stem of category N3. Omitting unnecessary details, the rest of the analysis proceeds with successive recognition of the affixes "-ımız" (N4N5, 1st plural possessive), "-da" (N7N8, locative), "-ki" (N8N9, relative), "-lar" (N2N3, plural), "-dan" (N7N9, ablative), "-mı" (Q1Q2, interrogative), "-y" (Q2Q3, auxiliary), and "-dı" (V4V5, past). There are free jumps from V5 through V6 and V7 to the final state WW, so the word is successfully parsed. A stem is accepted as a word only if there is a transition (or a chain of zero transitions) to a special category named WW.

The morphophonology is encoded in the functions which determine whether a given surface string matches a root or suffix entry. There are two different procedures, one for roots and one for suffixes, because rules are not exactly the same for roots and suffixes. What these functions do is modify the basic form of the morpheme to make it compatible with its surface environment.

3.2 The Lexicons

A morphological parsing algorithm must employ a well-designed database handler in order to effectively access the root and suffix lexicons. For, in a practical Turkish spell checking algorithm the number of roots may go up to as high as 10 to 20 thousand with the addition of frequently used people, place, brand names, etc. There are two problems associated with the usage of such a big root lexicon. First, minimizing the volume occupied by this root lexicon; second, minimizing the access time needed to locate a searched root. In the case of the suffix lexicon, since the space occupied and the number of suffixes are small compared to roots, there seems to be no need for a volume minimizing method. A search method is needed that can access suffixes having a stem category XXYY by a key code of XX. The current implementation uses an index of a pointer array type, each element of which is a header to a list of root words beginning with the same letter. This level of indexing is considered enough for the current algorithm, but an indexing made on the first two initial letters may increase the performance. The root words ending with one of the consonants in the set {p,ç,t,k} are represented in the form ending one of the consonants in the set {b,c,d,g} respectively. For example, "kitap" (book) has been entered as "kitab", "ağaç" (tree) as "ağac", "git" (go) as "gid", "yaprak" (leaf) as "yaprag", etc. This convention makes the application of the consonant rules easier.

The implementation of the suffix lexicon mirrors the transition network representation of the morphotactics of the language. The data structure employed is a bucket structure. The nodes in the linked lists each hold a stem category XXYY, the suffix form causing this transition, and a pointer to the next node. The bucket is an array of header pointers, each pointer pointing to the linked list of nodes with the same XX. The suffix lexicon is actually stored in a file and it is loaded into the bucket structure every time the parser is run. Note that this file actually contains some more information for the transitions. For instance N>V signifies that the transition converts a noun category into a verb category for the N0V0:la link. Similarly, PL signifies that "-lar" is the plurality suffix in the NLN1:lar transition. The following symbols are used for stem categories: N0 for noun roots, M0 for proper noun roots, V0 for verb roots, A0 for adjective roots, O0 for pronouns, E0 for interjections, C0 for conjunctions, P0 for

postpositions, Z0 for adverbs, K0 for deerminers, B0 for "ben" (I) and "biz" (we), S0 for "sen" (you,singular) and "siz" (you,plural), and D0 for "bana" (to me) and "sana" (to you).

The data in the root lexicon is obtained from Hankamer [8] with around 1400 root words. The lexicon is refined and enlarged by the addition of 3500 new root words. The suffix lexicon as well is obtained from Hankamer [8], and refined by the authors. The vowels "a" and "ı" have been chosen as default vowels for the lexical entries. The vowel harmony procedure changes these vowels in accord with the rule. The only exception is the vowel "o" in the present continuous tense particle "-ıyor".

```
okullu
o O0
ok N0
oku V0
okul N0
**** N0-lı N0- N1- WW-
**** N0-lı N0- N1- NA- WW-
**** N0-lı A0- A1- N1- WW-
**** N0-lı A0- A1- N1- NA- WW-
**** N0-lı A0- A1- P1- P2- P4- P5- P6- WW-
**** N0-lı A0- A1- WW-

çöplüklerimizdekilerdenmiymiş
çöp N0
*** N0-lıg N0- NL-lar N1-ımız NA- N2-da NG-ki NK- NL-lar
N1- NA- N2-dan N3- P0- P1- P2-y P3-mış P4- P5- P6- WW

büyünün
büyü V0
**** V0-ın VR- VC- V1- VN- V3-yın V7- WW-
büyü N0
**** N0- N1- NA- N2-Nın NG- N3- P0- P1- P2- P4- P5- P6- WW-
**** N0- N1- NA- N2-Nın NG- N3- WW-

bağlarsın
bağ N0
*** N0-la V0- VR- VC- V1- Vn- V3-ar P1- P2- P4-sın P5- P6- WW
bağla V0
*** V0- Vr- VC- V1- VN- V3-ar P1- P2- P4-sın P5- P6- WW
```

Figure 1. A Sample Run of the Parser

Figure 1 gives a sample run of the parser. We see that three unsuccessful root forms, namely "o" (he), "ok" (arrow), and "oku" (read), have been tried while parsing "okullu" before the successful one, "okul" (school). Note that "bağlarsın" (you tie) and "büyünün" (of the magic/make yourself grow up) each have two successful parses. One of the parses is semantically meaningful for both of the words (stem category being N0 for "büyünün" and V0 for "bağlarsın". For "büyünün", the other one (stem category V0) may be accepted as semantically meaningful as well; but for "bağlarsın" (stem category N0) it is semantically nonsense. Details of the implementation and the complete transition network may be found in [9,10].

4. SPELLING CHECKING AND CORRECTION

A morphological parser is more than adequate for spelling checking purposes if it incorporates all of the morphological rules of the language. Because, most of the spelling errors that can be made while writing a text in Turkish language are due to typing errors, such as omitting a letter, interchanging two letters or writing an incorrect letter in place of letters having the same shape except the cedilla. For this reason, an efficient implementation of the aforementioned morphological parser will be sufficient to check the spelling of words in written text. However, there will still be the undetermined cases such as the correct placement of the suffix "-de" which in some case means "too" in which case it has to be written separately or in some cases have a different meaning where it must appear as a suffix. Such cases can only be determined by referring to the context.

Spelling correction can be performed by first detecting a misspelled word, then checking for the above mentioned typing errors and listing the words that result as a suggestion.

A spelling checker software for Turkish based on the aforementioned morphological parser is under development and will be integrated with the All-In-1 office automation system of Digital.

5. CONCLUSION

The present algorithm and its implementation has several shortcomings. For example, some noun roots that end in a consonant, double that consonant when they take a suffix. This problem is directly solved by including both forms with single and double consonants in the root lexicon, but the parser accepts the wrongly suffixed form, too.

The root lexicon is very small in its current state for a real spelling checker and it is to be enlarged to include a few ten thousand words. Internal data structures and file structures need to be redesigned for fast access to the enlarged lexicon.

ACKNOWLEDGEMENTS

This work is partly supported by TÜBITAK, grant number EEEAG-11, and by Digital Türkiye. The work is also submitted to Digital EERP and to Boğaziçi University Research Fund for support.

REFERENCES

1 Hankamer, J., "Morphological Parsing and the Lexicon", *Lexical Representations and Processing*, ed. W.M. Wilson, MIT Press, 1988.

2 Hankamer, J., "Finite State Morphology and Left to Right Phonology", *Proceedings of The West Coast Conference on Formal Linguistics*, Vol. 5, Stanford University, 1986.

3 Kasper, R. and D. Weber, (revised 1986 by Stephen McConnel) *User's Reference Manual for the C Quechua Adaptation Program*, Occasional Publications in Academic Computing, No 8 and 9, Summer Institute of Linguistics, Inc., 1982.

4 Koskenniemi, K., "Two-level Morphology", *University of Helsinki Department of General Linguistics Publication No. 11*, Helsinki, 1983.

5 Hankamer, J., "Turkish Generative Morphology and Morphological Parsing", unpublished paper presented at *the Second International Conference on Turkish Linguistics*, Istanbul, 1984.

6 Solak, A. and K. Oflazer, "Design and Implementation of a Spelling Checker for Turkish", *Proceedings of the Fifth International Symposium on Computer and Information Sciences*, ÜrgÜp, 1990.

7 Hankamer, J., "Parsing Nominal Compounds in Turkish", *Morphology as a Computational Problem*, UCLA Occasional Papers 7, ed. Karen Wallace, UCLA, 1988.

8 Hankamer, J., personal communication with, via BITNET.

9 Kibaroğlu, M. O., Spell Checking in Agglutinative Languages and an Implementation for Turkish, M.S. Thesis, Boğaziçi University, Istanbul, 1991.

10 Kibaroğlu, M. O. and S. Kuru, "A Left-to Right Morphological Parser for Turkish," *Proceedings of the Sixth International Symposium on Computer and Information Sciences*, Antalya, 1991.

APPENDIX

State Table for the Morphology of Turkish Words: A * denotes that a free jump is allowed.

present state	next state	suffix
A0	A1	*,-ımtrak
	N0	-lık
	V0	-lan
A1	N1,P1,WW	*
	Z1	-ca
O0	NL	-n
	WW	*
D0	WW	*
E0	WW	*
B0	P1	*
S0	P1	*
K0	WW	*
	N1	*
N0	N0	-cı,-cık,-ca,-cağız,-daş,-lık,-lı
	V0	-la,-laş,-lan
	NL,N1	*
	A0	-lı,-sız
NL	N1	-lar
N1	WW	*
	NA	*,-ım,-ımız,-ın,-ınız
	NB	-zı
NA	WW,N2,NM	*
NB	WW	*
	N2	-n
	NM	*
NM	NC	-yla
N2	NG	-nın,-da
	N3	-dan
	NC	-yı,-ya
N3	P0,WW	*
NC	WW	*

NG	N3	*
	NK	-ki
NK	WW,P0,NL,NB	*
0	P1	*
P1	P2	*,-mı
P2	P3	-y
	P4	*
P3	P4	-mış
	V4	*
P4	P5	*,-lar,-sın,-sınız,-yım,-yız
P5	P6	*,-dır
P6	WW	*
V0	N1	-yış
	VR	*,-ın
VR	VC	*,-ış
VC	VC	-dır
	V1	*,-ıl
V1	V2	* ,ya
	VN	*
V2	VN	-ma
VN	V3	*,-yabil
V3	A1	-mış,-yacag,-yan
	N1	-dıg,-ma
	NA	-mag
	P1	-ıyor,-ar,-mış,-malı,-yacag
	V4	*
	V6	-sın,-sınlar,-yayım,-yalım
	V7	*,-sana,-sanıza,-yın,-yınız
	Z1	-ınca,-yıp,-ya
V4	V5	-dı,-sa
V5	V4	-y
	V6	*,-k,-lar,-m,-n,-nız
V6	V7	*,-mı
V7	WW	*
Z0	Z1	*
Z1	WW	*