Proceedings of the IASTED International Conference
ARTIFICIAL INTELLIGENCE AND APPLICATIONS
February 16-18, 2004, Innsbruck, Austria

AN AFFIX STRIPPING MORPHOLOGICAL ANALYZER FOR TURKISH

Gülşen Eryiğit and Eşref Adalı
Dep. of Computer Engineering, Istanbul Technical University
34469 Ayazağa, Istanbul, Turkey

Abstract

This paper presents the design and the implementation of a morphological analyzer for Turkish. A new methodology is proposed for doing the analysis of Turkish words with an affix stripping approach and without using any lexicon. The rule-based and agglutinative structure of the language allows Turkish to be modeled with finite state machines (FSMs). In contrast to the previous works, in this study, FSMs are formed by using the morphotactic rules in reverse order. This paper describes the steps of this new methodology including the classification of the suffixes, the generation of the FSMs for each suffix class and their unification into a main machine to cooperate in the analysis.

Key Words
Natural Language Processing, Morphology, Affix Stripping, Turkish

1. Introduction

Morphological analysis, which deals with the subparts of the words, is one of the fundamental areas in natural language processing. Several different methods have been developed and implemented to make this analysis more efficient and effective. When these methods are investigated, it is seen that most of the studies were originated from an agglutinative language such as Hankamer's Keçi [1] for Turkish, PC_Kimmo [2] for Finnish, Ample [3] for Quechua. At the first glance, it seems possible to store all the word inflectional forms in a lexicon and do the language processing without any morphological analysis. This approach can be suitable for the languages, which are morphologically simple, but it is untenable to apply for agglutinative ones [4] where a word can take hundreds of different forms after the concatenation of affixes.

Ex: Some of the different forms of the noun "eye" in Turkish are listed below:

Göz	Eye
Göz-lem	Observation
Göz-lem-ci	Observer
Göz- lem-ci-lik	The job of the observer
Göz-lem-le-dik-ler-im	The ones that I observed

The work done in this field so far can be grouped into two classes: root driven and affix stripping. In the first approach, the stem of the word is found initially and then the affixes are determined. In the second approach, the determination of the affixes takes place first in contrast to the first one. After the removal of the affixes, the remaining part of the word can be assumed to be the stem or a lexicon can be used to approve this assumption.

Most popular morphological analyzers such as PC_Kimmo [2] and Ample [3] use the root driven approach and confirm the method's success with their customized versions for different languages. Root driven methods are also widely used in the studies done for Turkish [1] [5] [6]. However, for other agglutinative languages, some affix stripping methods [7] [8] have been developed and successful results were achieved.

In the root driven approach, the stem of the word should be firstly found in a lexicon before starting the morphological analysis. The major drawback of this approach is the cost of the searching process required to find the stem. In the work done by Solak and Oflazer [5], the word itself and its subparts, which have been obtained by removing the letters one by one from the end of this word, are looked up in a lexicon to find all the possible stems. The real stem is discovered after the morphological analysis made by using these possible stems. Even though there are different search methods improving the performance like letter tree encoding [2] in PC_Kimmo, the examining of each subpart is obviously a very time consuming process especially for the languages where the words can appear in very long forms. On the other hand, in the affix stripping approach, the searching process is relatively fast as the search is only done for affixes.

Turkish has a special place within the natural languages not only being a fully concatenative language but also having the suffixes as the only affix type [9]. In this language, words are formed from a stem and suffixes concatenated to this stem. This suffix concatenation can result in relatively long words, which are frequently equivalent to a whole sentence in English. Additionally, the morphotactic constraints are strictly defined. This allows Turkish to be modeled with FSMs [6] easily. Another feature of the language is that, someone who knows Turkish can easily analyze a word even if he/she does not know its stem. The phonological rules of Turkish are significant factors that influence this feature.

Ex: (any word)*lerim* (any word)-*ler-im*
"*ler*" plural suffix, "*im*" 1st singular person possessive.

When one sees this word, he understands that there is something called (any word), there are multiple of them and they belong to the speaker. If a Turkish speaker can make this analysis without knowing the meaning of the stem, a method can be formed to make the same analysis without using a lexicon.

With the aforementioned features and the rule-based structure of the Turkish language, in this study, an affix stripping morphological analyzer is developed for Turkish. Subsequent sections describe the details of the method, the generated FSMs and the collaboration of them. In the final section, some conclusions and suggestions for future work are given.

2. Suffix Classification

Turkish, which is an agglutinative language, has a very rich morphological structure. The Turkish words often contain some semantic information after the multiple affixations of suffixes. In some cases, this suffix concatenation helps to diminish the ambiguities and to understand the stem more quickly.

Ex:The word "*kale*" means castle and the word "*kalem*" means pencil in Turkish. When the word "*kalem*" is analyzed, there exist two possible solutions:

kalem	N(*kalem*)	pencil
kale-m	N(*kale*) + 1PS-POSS(*m*)	my castle

To decide which one of these two analyses is true, one should see the usage of the word in the sentence. But when the word "*kalemler*" is analyzed, there is only one valid solution:

kalem-ler	N(kalem) + PLUR(ler)	pencils
~~kale-m-ler~~	~~N(kale) + 1PS-POSS(m) + PLUR(ler)~~	

Here, the suffix concatenation rules help to find the right stem: the analysis cannot be "*kale-m-ler*" because in Turkish, a plural suffix cannot follow a possessive one.

The only affix type in Turkish is the suffixes. Therefore when an affix stripping method is used, the analysis is made by removing the suffixes from the end of the word. For Turkish, this method can also be called a right to left analysis. When the example given in the introduction "(anyword)*ler-im*" is analyzed, firstly the possessive suffix "-*im*" is removed, then the stem "(anyword)" is reached by removing the plural suffix "-*ler*" from the remaining part "(anyword)*ler*". In the above example "*kalemler*", no other stripping will be made after the removal of the suffix "-*ler*" and the stem "*kalem*" will be reached because the concatenation rules do not permit the suffix "-*m*" to precede the suffix "-*ler*". As it is not possible to reach the stem "*kale*" with this method, there will be no further processing for this stem. On the other hand, in the root driven approach, since the first step is to find the possible stems of the word from a lexicon, the analysis will be made for both "*kale*" and "*kalem*". The right stem will be decided at the end of the process.

With the purpose of composing an affix stripping analyzer for Turkish, the suffixes are firstly classified and then stored in a database. Table-1 shows the generated suffix sets.

Table-1: Suffix Classes

Class #	Class	Type
1	Nominal verb suffixes	Inflectional
2	Derivational suffixes	Derivational
3	Noun suffixes	Inflectional
4	Tense & person verb suffixes	Inflectional
5	Verb suffixes	Inflectional

A suffix in Turkish can have multiple allomorphs in order to provide sound harmony in the word to which it is affixed. For example, the first singular person possessive suffix with generic representation –(U)m has five allomorphs: -m, -ım, -im, -um, -üm. The abbreviations used to show suffixes in a generic way are shown below:

U: ı,i,u,ü	C: c,ç	A: a,e	D: d,t	I : ı,I

(): the letter between parentheses can be omitted

While the suffixes are put in the database, an item and arrangement method is implemented: all the allomorphs of the suffixes are generated and put in the database. With this approach the time, which is consumed by applying phonological rules in order to form the allomorphs of the affixes at runtime is reduced. Table-2 shows the number of suffixes in each class and the number of allomorphs put in different database tables.

Table-2: Number of Suffixes & Allomorphs

Class #	# Suffixes	# Allomorphs
1	15	73
2	10	40
3	19	89
4	29	133
5	44	184
Total	117	519

3. FSM Generation

After the classification of the suffixes, the next step is to design FSMs that make a right to left analysis on the word. There are four stages to create these FSMs:

- Creating a left to right FSM
- Labeling the suffixes
- Inverting the left to right FSM and obtaining a non deterministic finite state automaton (NFA)
- Converting NFA to a deterministic finite automaton (DFA) and constructing the right to left FSM

In the remaining part of this section, suffix sets and right to left FSMs are given for each suffix class. The above four stages are explained in detail on the “Nominal verb suffixes” class since its FSMs and NFA to DFA conversion operations are simpler than the other inflectional classes.

3.1 Nominal Verb Suffixes

1st Stage: Creating a left to right FSM

The suffixes are affixed to the stem according to definite ordering rules. At this stage, these rules are aggregated into a FSM [6] for current suffix class. This FSM is suitable for a root driven approach and serves in a left to right analysis.

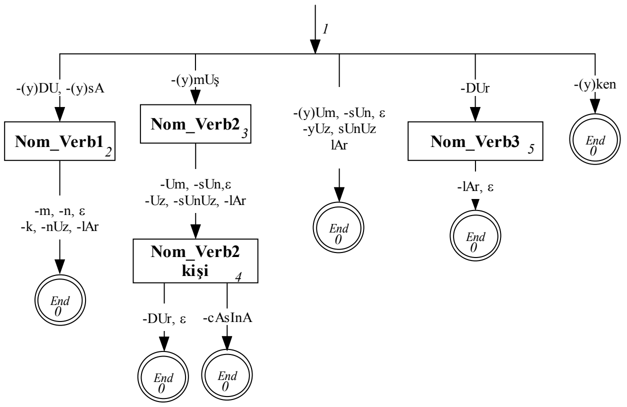


Figure-1: Nominal Verb Suffixes left to right FSM

In Figure-1, a numerical value is assigned to each state. In the following stages, the states will be expressed with these numbers: Ending states 0, Initial state 1 etc… The character “ε” shows the empty transitions between the states. When the analysis of the word “*güzel-miş-sin*” (you were beautiful) is made by using this FSM, firstly the stem “*güzel*” (beautiful) is found, then the first suffix “*-miş*” (-(y)mUş) makes a transition from the input state 1 to state 3, after that the suffix “*-sin*” (-sUn) carries the machine to state 4. The ending state is reached by the empty transition between state 4 and 0.

2nd Stage: Labeling the suffixes

In this stage, each suffix’ generic representation in the current suffix class is given a number. The suffixes’ allomorphs are then stored in the database with these numbers, their suffix classes and names. Table-3 shows the numbers and generic representations of the nominal verb suffixes. For example the two allomorphs (-lar, -ler) of the suffix –lAr will both have the same suffix number “5” in the database.

3rd Stage: Inverting the left to right FSM

This step is the first milestone to convert the left to right FSM to a right to left FSM. Figure-2 shows the obtained NFA after the reversion of the FSM in Figure-1. In this figure, the states are given inside circles and expressed with state numbers given in the 1st Stage. The inputs shown over the transitions are the suffix numbers given in Table-3. The initial state is shown with a ‘>’ sign beside and the ending state with a double circle around.

Table-3: Nominal Verb Suffixes

1	–(y)Um	6	–m	11	–cAsInA
2	–sUn	7	–n	12	–(y)DU
3	–(y)Uz	8	–k	13	–(y)sA
4	–sUnUz	9	–nUz	14	–(y)mUş
5	–lAr	10	–DUr	15	–(y)ken

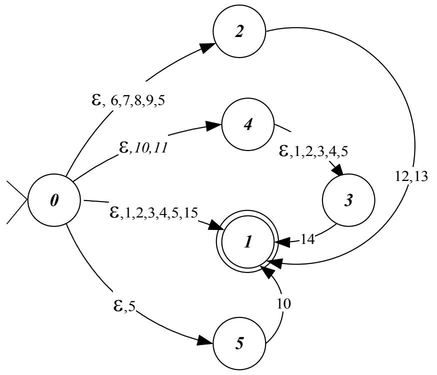


Figure-2: Nominal Verb Suffixes right to left NFA

4th Stage: Converting NFA to DFA

Multiple transitions for a single input and empty transitions make NFA hard to implement with computer programs. Therefore, the NFA in Figure-2 should be converted to a DFA in which there will be at most one transition for a single input and no ε-transition. An algorithm called “subset construction” [10] is used to make this operation. In this algorithm, each new DFA state corresponds to a set of NFA states. The idea is that all the states, which are connected by an ε-transition and reachable by a single input on the current state will be combined into a single DFA state.

Table-4: Nominal Verb Suffixes NFA to DFA Operations

A = {0, 1, 2, 3, 4, 5} “1, 2, 3, 4” : T={1, 3} → B “5” : T={1, 2 ,3 ,5} → C “6, 7, 8, 9” : T={2} → D “10” : T={1, 4} → {1, 3, 4}→ E “11” : T={4} →{3, 4} → H “12, 13, 14, 15” : T={1} → F	D = {2} “12, 13” : T={1} → F
	E = {1, 3, 4} “1, 2, 3, 4, 5” : T={3} → G “14” : T={1} → F
B = {1, 3} “14” : T={1} → F	G = {3} “14” : T={1} → F
C = {1, 2 ,3 ,5} “10, 12, 13, 14” : T={1} → F	H = {3, 4} “1, 2, 3, 4, 5” : T={3} → G “14” : T={1} → F

In the example shown in Table-4, the DFA starting state A has as its first element the starting state 0 of the NFA. As all other states are reachable from state 0 by ε-transitions, the state A includes these also: A={0,1,2,3,4,5}. The numbers between quotes express the inputs, in other words the suffix numbers. The new DFA state is computed by finding the set of states having transitions on the inputs “1,2,3,4” from the members of A. Among the states 0,1,2,3,4,5 of A, only 0 and 4 have such transitions to 1 and 3. So the new state B is created from 1 and 3. If there were any states connected to these ones by an ε-transition, they would be included in the B set.

DFA states came out at the end of the operations in Table-4: A = {0, 1, 2, 3, 4, 5}, B = {1, 3}, C = {1, 2 ,3 ,5}, D = {2}, E = {1, 3, 4}, F = {1}, G = {3}, H = {3, 4.} They are combined to construct the ultimate FSM in Figure-3. The initial state of the NFA in Figure-2 is 0 and the ending state is 1. The new FSM's initial and ending states are determined according to these ones. The sets including 0 becomes the initial state and including 1 becomes the ending state.

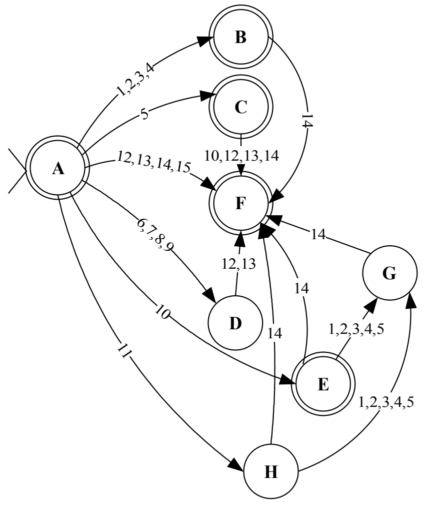


Figure-3: Nominal Verb Suffixes right to left FSM

3.2 Derivational Suffixes

In Turkish, there are hundreds of derivational suffixes, which change the meaning and in some cases the class of the word to which they are affixed. While the rules about the word types to which they can be affixed are defined in Turkish grammar books, the required morphotactic rules (ordering of morphemes) are not stated. The definite rule is that the derivational suffixes should be affixed to the word before the inflectional ones. This situation make hard to design a FSM for this class. Therefore, only the 2nd stage "Labeling the suffixes" is applied for some selected derivational suffixes (Table-5).

Table-5: Derivational Suffixes

1	–lUk	6	–lAn
2	–CU	7	–CA
3	–CUk	8	–lU
4	–lAş	9	–sUz
5	–lA		

3.3 Noun Suffixes

While doing the analysis of a word with the generated FSMs, the last state reached after the removal of the possible suffixes is controlled to be an ending state. If it is not the case, the last visited ending state is assumed to be the stopping point of the analysis. When the word "*etkilerden*" (from the effects) is analyzed by using the noun suffixes in Table-6 and FSM in Figure-4, the correct decomposition of the word is the following: *etki-ler-den* stem(*etki*) +suffix#1(-lAr) + suffix#15(-DAn). The last syllable "*-ki*" of the stem can also be considered as a possible suffix (suffix#18). This consideration will lead the analysis to reach a false stem "*et*" (meat). The following shows the visited states during the analysis:

$$\begin{tikzcd} A \arrow[r, "15"] & G \arrow[r, "1"] & L \arrow[r, "18"] & D \end{tikzcd}$$

As D is not an ending state, L being the last visited ending state is accepted as the stopping point and the correct stem "*etki*" (effect) is obtained.

Table-6: Noun Suffixes

1	–lAr	11	–(y)A
2	–(U)m	12	–nA
3	–(U)mUz	13	–DA
4	–(U)n	14	–nDA
5	–(U)nUz	15	–DAn
6	–(s)U	16	–nDAn
7	–lArI	17	–(y)lA
8	–(y)U	18	–ki
9	–nU	19	–(n)cA
10	–(n)Un		

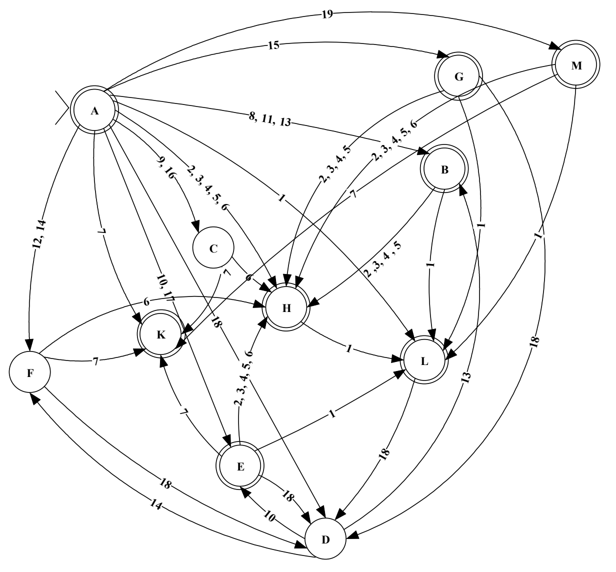


Figure-4: Noun Suffixes right to left FSM

3.4 Tense & Person Verb suffixes

There are two different types of ending states in the FSM (Figure-5) of this suffix class. This differentiation will be used in the unification of the FSMs.

Table-7: Tense & Person Verb Suffixes

1	–(y)Um	11	–mAktA	21	–(y)UnUz
2	–sUn	12	–mAlI	22	–(y)Un
3	–(y)Uz	13	–m	23	–sUnlAr
4	–sUnUz	14	–n	24	–DUr
5	–lAr	15	–k	25	–(y)DU
6	–mUş	16	–nUz	26	–(y)sA
7	–(y)AcAk	17	–DU	27	–(y)mUş
8	–(U)r	18	–sA	28	–cAsInA
9	–Ar	19	–lIm	29	–(y)ken
10	–(U)yor	20	–(y)A		

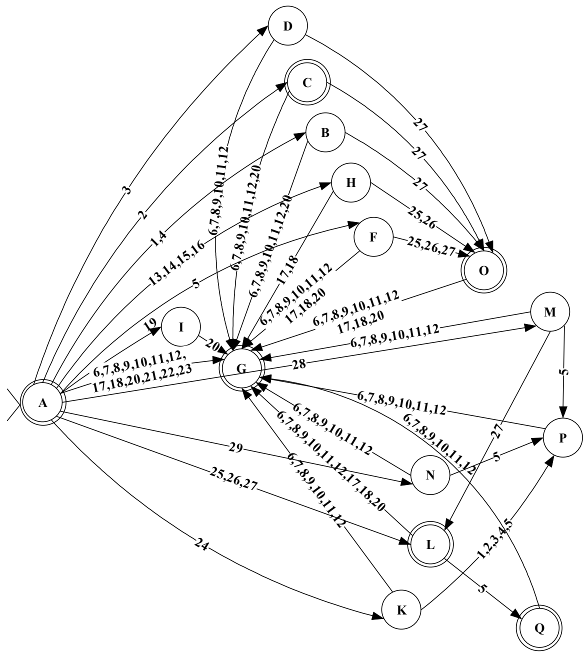


Figure-5: Tense & Person Suffixes right to left FSM

The states A, C, G, L, O and Q are the ending states of Figure-5. L, O, Q are acceptable endings for only the negative verbs and A, C, G are acceptable endings for only positive verbs. This control is done by the FSM of the following class "Verb Suffixes" in Figure-6. Table-7 shows the "Tense & Person Verb" suffixes in their generic representations.

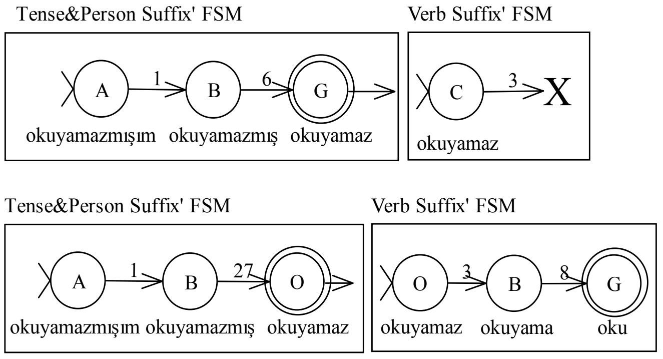
3.5 Verb Suffixes

Verb suffixes' FSM in Figure-6 is the most complex one having multiple entry states and the highest number of affixes (Table-8). When the FSMs are unified to work together, this class receives entries from different classes, which causes the FSM to have multiple starting states:

A and R normal entry
P entry from the noun FSM
C entry from tense&person verb FSM for positive verbs
O entry from tense&person verb FSM for negative verbs
Q entry from nominal verb FSM

Table-8: Verb Suffixes

1	–m	16	–(y)Akoy	31	–mAzlIk
2	–zsIn	17	–mAk	32	–mA
3	–z	18	–(y)UcU	33	–(y)Uş
4	–yIz	19	–(y)Up	34	–Dan
5	–zsInIz	20	–(y)AlI	35	–DA
6	–zlAr	21	–DUkçA	36	–(y)lA
7	–mA	22	–(y)ArAk	37	–(y)A
8	–(y)AmA	23	–(y)UncA	38	–mAksIzIn
9	–(y)Adur	24	–Dan	39	–mAdAn
10	–(y)Uver	25	–yA	40	–(U)n
11	–(y)Agel	26	–(y)An	41	–(U)ş
12	–(y)Agör	27	–(y)AcAk	42	–(U)l
13	–(y)Abil	28	–(y)AsI	43	–Dur
14	–(y)Ayaz	29	–DUk	44	-(U)t
15	–(y)Akal	30	–mUş		



The analysis of the word "*okuyamazmışım*" (it is said that I am not able to read) is given above. In this example the last two suffixes "*-mış*" and "*-ım*" are analyzed with the FSM of the previous suffix class "tense & person verb suffixes" in Figure-5. In this FSM the suffix#1 (Table-7) "*ım*" make a transition from the state A to the state B. The suffix "*mış*" can be either suffix#6 –mUş or suffix#27 –(y)mUş of the Table-7. This will make a transition from the state B to either the state G or the state O. As stated earlier, the state O of the tense&verb suffix' FSM can be an ending state only for the negative verbs whereas G for positive ones. After analysis is finished with the FSM in Figure-5, the stem "*okuyamaz*" is reached and then analyzed with the verb suffix' FSM in Figure-6. The second analysis with stem "*oku*" is the correct one and will have a higher success probability having a shorter stem than the first one.

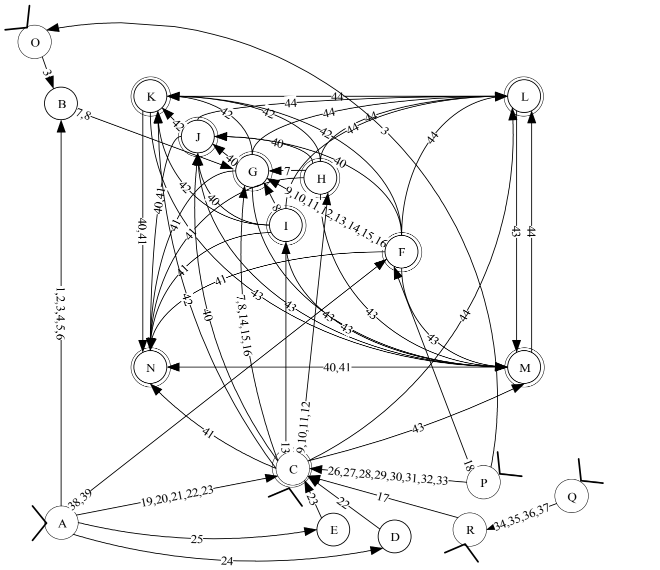


Figure-6: Verb Suffixes right to left FSM

4. Unification of the FSMs

Turkish words can be classified into two main categories being nominal and verbal words. The words can change their categories after the concatenation of suffixes from different classes: nouns can turn into verbs and verbs vice versa. For this reason, the discrete FSMs formed in the previous sections should be joined into a main FSM to cooperate in the analysis of a single word. In the main FSM shown in Figure-7, each discrete FSM firstly analyzes the intact word and then the outputs coming from the FSMs previously run.

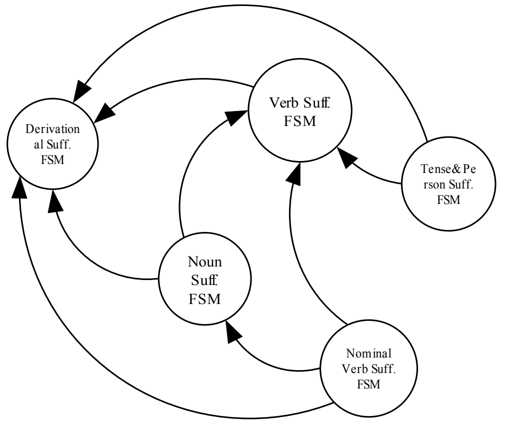


Figure-7: Main FSM

Ex: The following is the output taken after the analysis of the word "*çağırmadıklarımızdanmışsınız*" (You were the one of whom we did not call) with this main FSM. The classes of the suffixes are written at the end of each line.

"*Çağır-ma-dık-lar-ımız-dan-mış-sınız*"

Çağır	verb stem	
-mA	negative suffix	Verb Suffix
-dUk	adjective verb suffix	Verb Suffix
-lAr	plural suffix	Noun Suffix
-(U)mUz	1st plural possessive suffix	Noun Suffix
-DAn	case suffix	Noun Suffix
-(y)mUş	past tense	Nominal Verb Suffix
-sUnUz	2nd plural person suffix	Nominal Verb Suffix

At the end of the analysis, the first suffix removed from the end of the word determines the category of that word: nominal or verbal. The above example is a verb since its last suffix is a nominal verb suffix. This information is kept for being used in the syntactic analysis but it does not comprise the details about the nominal category such as noun, adjective or pronoun. In some cases, the affixes concatenated to the stem cause some deformation on it. The most widely seen of these cases happen when the suffixes starting with a vowel are affixed to a stem ending with the letter "p,ç,t,k" and an –(I)yor suffix is affixed to a stem ending with vowels "a" or "e". In the first case the letters are transformed to the letters "b,c,d,g" and in the second case the vowels turn into "ı" or "i". Some exception cases like these ones are handled within the software and the required changes are done over the stems reached at the end.

5. Conclusion & Future Work

In this paper, an affix stripping morphological analyzer is developed for Turkish by using the rule-based structure of the language. The software developed can be reached from the address http://www.cs.itu.edu.tr/~gulsen/nlp/nlp.html. This software aims to reach the stem of a word without using any lexicon while making the morphological analysis. To reach this aim, all the suffixes are grouped into five classes and stored in a database with their suffix classes and names. For each of the suffix class, a FSM describing the concatenation rules of the suffixes in reverse order is designed. A global FSM is formed to make the previously designed FSM's collaborate with each other. At the end of the analysis with the global FSM, the word is partitioned into its stem and suffixes. When one wants to add a new suffix to this analyzer, he should insert it to the related suffix class and update the related FSMs.

For Turkish documents, an information retrieval system needs essentially a stemming algorithm because in Turkish sentences, words are usually concatenated lots of suffixes. Especially in the web-based systems concerning the performance criteria, it is undesirable to use a lexicon to find the word stem. Originating from the method developed in this study, a Turkish stemming algorithm can be easily developed and this algorithm can be used in information retrieval systems for Turkish documents.

References

[1] J. Hankamer, Finite state morphology and left to right phonology, *Proceedings of the Fifth West Coast Conference on Formal Linguistics*, Stanford, CA, 1986, 29-34.

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

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

[4] D. Jurafsky & J.H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition* (Upper Saddle River, N.J. : Prentice Hall, 2000).

[5] A. Solak & K. Oflazer, Design and Implementation of a Spelling Checker for Turkish, *Literary and Linguistic Computing*, *8*(3), 1993, 113-130.

[6] K. Oflazer, Two-level Description of Turkish Morphology, *Literary and Linguistic Computing*, *9*(2), 1994, 137-148.

[7] B. Brodda & F. Karlsson, *An experiment with automatic morphological analysis of Finnish* (Stockholm: Department of General Linguistics University of Helsinki, 1980).

[8] H.J. Kaalep, An Estonian Morphological Analyzer and the Impact of a Corpus on Its Development, *Computers and the Humanities, 31*(2), 1997, 115-133.

[9] R. Sproat, *Morphology and computation* (Cambridge, MA: MIT Press, 1992).

[10] A. V. Aho, R. Sethi & J. D. Ullman, *Compilers: principles, techniques, tools* (Reading, MA: Addison-Wesley, 1986).