Rule Parser for Arabic Stemmer

Imad A. Al sughaiyer
Tel: 481-3217, fax: 481-3764
imad@kacst.edu.sa

Ibrahim A. Al kharashi
Tel: 481-3273, fax: 481-3764
kharashi@kacst.edu.sa

Computer and Electronics Research Institute
King Abdulaziz City for Science and Technology
P. O. Box 6086, Riyadh 11442, Saudi Arabia

ABSTRACT

Arabic language exhibits a complex but very regular morphological structure that greatly affect its automation. Current available morphological analysis techniques for the Arabic language are based on heavy computational processes and/or the existence of large amount of associated data. Utilizing existed morphological techniques greatly degrade the efficiency of some natural language applications such as information retrieval system.

This paper proposed a new Arabic morphological analysis technique. The technique is based on the pattern similarity of words derived from different roots. Unique patterns are extended and coded as rules that encode morphological characteristics. The technique does not require either complex computation or associated data yet adjustable to maintain enough accuracy. This technique utilizes a very simple parser to scan coded rules and decompose a given Arabic word into its morphological components.

This paper provides an introduction to Arabic language and its morphological characteristic followed by an overview of currently available morphological techniques. Explanation of the developed stemmer and its components including rule set and parser were given. Experimental results and the work conclusion were provided at the end.

Keywords: Natural Language Processing; Arabic language; Stemmers

1. INTRODUCTION

Morphological analysis techniques are computational processes that analyze natural words by considering their internal morphological structures. Stemming algorithms, on the other hand, are processes that gather all words sharing the same stem with some semantic relation. Stemming, as a term, is widely used by researchers dealing with languages with simple morphological systems while morphological analysis, as a term, is widely used by researchers in languages with complex morphological system such as Arabic and Hebrew. The main objective of the stemming algorithms and one objective of morphological analysis techniques is to remove all possible affixes and thus reduce the word to its stem [1, 2].

The major difference between Arabic and most of other languages resides mainly on its complicated, very regular and rich morphological structure. Arabic language is derivational while most of other languages are concatenative. Most of Arabic words are generated based on root-pattern structure. Arabic word generation is highly affected by its morphological characteristics [3, 4, 5]. Stems are generated from roots using one or more patterns. Suffixes, prefixes, and infixes can be added to a stem to generate an Arabic word. A reverse process is used to analyze Arabic words. Schematic diagram for analysis and generation processes is shown in Figure 1.

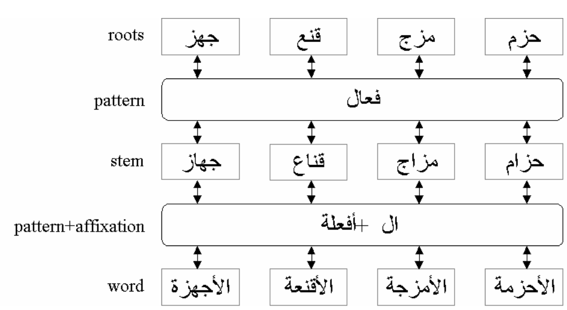

Figure 1. Arabic system for generating/analyzing words

Due to its non-concatenative nature, processing Arabic language is not an easy task. Tens or hundreds of words are generated using single root, few patterns and few affixes based on root-pattern schemes. Also, Arabic has a high degree of ambiguity because of many reasons such as missing of vowels and similarity between affixation letters and stem or root letters.

2. ARABIC MORPHOLOGICAL ANALYSIS TECHNIQUES

Arabic morphological analysis techniques are classified into table lookup, linguistic and combinatorial approaches [3, 6, 7]. Table lookup approaches utilize huge list that stores all valid natural Arabic words along with their morphological decompositions. A given Arabic word is analyzed by accessing the list and retrieving information associated with that entry [8].

Linguistic approaches, on the other hand, simulate the behavior of a linguist by considering Arabic morphological system and deeply analyzing input Arabic words accordingly to their morphological components. In such approaches, prefix and suffix of a given word are removed by comparing leading and trailing characters with a list of known affixes. Remaining part is either accepted as the required stem or modified using deletion, addition, or substitution of internal letters to generate the valid stem. The resultant stem is transformed to Arabic root simply by filtering process using valid list of patterns. Most of the published works are mainly linguistic based [3, 9, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18].

Finally, in combinatorial approaches all combinations of letters of tested word are generated. Resulting combinations are compared against lists of Arabic valid roots. On a match, valid root, stem and patterns are extracted. Otherwise other combinations will be tested [7, 19, 20].

Two problems are facing table lookup approaches. First, compiling huge list of all valid natural Arabic words is impractical. Second, the storage overhead for the list and access time are problematic. Linguistic approaches require too many lists and faced with the problem of wrongly removing beginning and ending substring of the word as they have similarity with valid prefix and suffix. Combinatorial approaches are trail and error processes and hence require heavy computational processing. It should be noticed, however, that ambiguity is a common symptom of any word-based approach. Higher-level contextual processing is usually the solution for such symptom.

To analyze Arabic words, some researchers suggested to reach their roots [3] while others suggested analyzing them to their stems only [4, 5]. Analyzing words to their roots is useful in linguistic processing, while analyzing words to their stems is preferred in some other applications such as information retrieval-based systems.

3. RULE-BASED ARABIC STEMMER

This section introduces a new approach that utilizes the apparent symmetry of generated natural Arabic words. In this approach, a unique regular expression-based rule is generated for group of similar Arabic words as shown in Figure 2. Regular expressions are compact patterns of characters or simple keyword searching sets of symbols used to match complex patterns in text strings. They are powerful expressions that describe a pattern of text rather than literal sub-string. They can be used in a complex way for searching and matching patterns in texts and databases, and once found, patterns are modified, extracted, or replaced with different string.

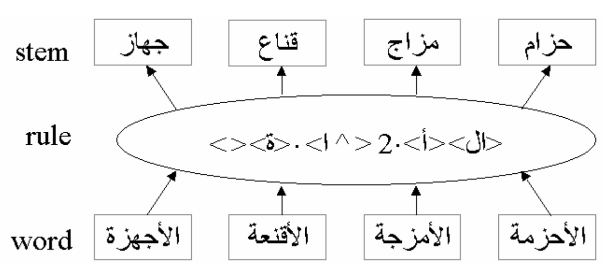

Figure 2. Sample rule usage.

Rules are used to describe the internal morphological structure of Arabic words and guide the decomposition process of a given word to its basic components i.e. stem, prefix and suffix. Rules are written from right to left to match script writing direction of Arabic language. Rule pattern may contain up to three distinct parts. The first and last parts describe affixation properties of the word while the middle part controls the stem extraction process. Pairs of angle brackets surround affixation parts. Absence of prefix or suffix in the rule patterns is sometimes denoted by empty angle brackets. This is necessary in order to distinguish them from an angle-bracketed part of the stem.

Rule complexity varies from very simple ones to very complicated rules that deal with complex morphological behaviors. Their syntax were generated after deep analysis of a randomly selected Arabic text and created with the following structure:

prefix-part stem-part suffix-part

where

prefix-part represents attached prefix, if any, and can be drawn from a finite list of prefixes.

stem-part represents stem structure and guide the process of extracting its original form.

suffix-part represents attached suffix, if any, and can be drawn from a finite list of suffixes.

Rule patterns are constructed using the following conventions:

- ***<str>*** to match the string ***str*** and delete it if in the stem part or consider it as prefix/suffix if in the prefix/suffix part.
- ***<s1^s2>*** to substitute ***s1*** by ***s2*** in stem and suffix parts. This notation is also used for insertion **<^*s2*>.**
- **< >** An empty bracketed string to indicate null prefix or suffix. This is necessary to distinguish the prefix/suffix from the start/end part of the stem part.
- ***.n*** to match ***n*** number of characters where ***n*** is an integer greater than one. Single letter is denoted by single dot. Matched characters are used to construct the stem.

Set of simple rules is created to handle words already in stem forms, isolated articles, proper names and foreign words. For example the rule ".4" matches any word with four letters and pass it as a valid stem with no further processing.

Other rules are used to treat words with morphological structure ranging from very simple to very complex. The rule "<و>.2<ا>.2<^ة><>" matches any six letters word with leading letter "و" followed by any two letters, letter "ا" and ending with any two letters. The letter "و" is extracted as prefix, letter "ا" is deleted and the letter "ة" is inserted in order to complete the stem creation process.

In the following pattern "<><ي>.3<أي^ء><>" the "<أي^ء>" part is used to substitute the substring "أي" with the letter "ء". Leading and ending empty bracketed parts denote the absence of both prefix and suffix. Table 1. lists few rules extracted from a list of about 1200 rules generated using the text collection.

Table 1. Sample Rules.

Applied Rule	Word	Resultant		
		Prefix	Stem	Suffix
.3	على		على	
<ال>.3<ية>	النارية	ال	نار	ية
<ال><أ>.2<^ا>.<ة><>	الأتربة	ال	تراب	
<><ت>.<و^ا>.	تموت		مات	
<ب><ه>	به	ب		هـ
<وال><أ>.<ئ^ؤا>.<ة><>	والأفئدة	وال	فؤاد	
<><آ^أ>.<ا>.<ها>	آمالها		أمل	ها
<ال>.4<ية>	السريرية	ال	سرير	ية
<ولل>.3<ات^ة>	وللمسرات	و	مسرة	ات
.2<^ي>.<ائ><هم>	وزرائهم		وزير	هم
<><أ>.2<^ا>.<ة><>	أجهزة		جهاز	
<ال>.2<و>.	الرموز	ال	رمز	
<ب>.5	بزراعة	ب	زراعة	
.6<ة>	متكاملة		متكامل	ة
<وس><ي>.<و^ا>.	وسيقول	وس	قال	
<وال>.2<ى^ية><>	والرؤى	وال	رؤية	
<><أ>.2<^اء><ية><>	أوعية		وعاء	

4. RULE PARSER

A very simple rule parser was developed to perform the analysis to process and extract word morphological components. The parser is used to perform matching between input rule and a given Arabic word. The matching process is achieved when the parser successively analyze the input word and decompose it, according to the parsed rule, to its valid components.

The parser is divided into three distinct parts to treat prefix, suffix and stem. Extracting morphological parts of a given word is merely done by interpreting the corresponding part of the rule. Initially, the parser scans the suggested rule to identify boundaries of each part. The angle-bracketed substring at the beginning/end of the rule string distinguishes prefix/suffix parts. The remaining middle part of the rule is the stem part. Each part guides the parser during the process of extracting word morphological parts.

Prefix and suffix are extracted using simple string matching process between the beginning/end of the word and the string in the prefix/suffix part of the rule. Suffix may contain a code that affect extracted stem. Stem part is generated by sequential copying from the middle of the word with the possibility of going through insertion, deletion and/or substitution. A simplified pseudo code of the parser is shown in Listing 1.

A rule is said to be fired if it has the same length as the length of the processed word. A match is achieved if and only if a fired rule produces the correct prefix, stem and suffix. A given word should fire at least one rule and match only one rule.

Listing 1.

```
parser(word)
   for every rule
      if word length = rule length
         identify rule prefix boundaries
         identify rule stem boundaries
         identify rule suffix boundaries
         if rule prefix = word beginning
            copy word beginning to prefix
         else
            match fail
         end if
         while rule stem
            if dot
               copy n symbols from the
               word proper position to stem
            end if
            if  angle-bracketed  ^ expression
               copy to the stem with substitution
            end if
         end while
         if rule suffix = word end
            copy word end to suffix
            if ^ expression
               append to stem
         else
            match fail
         end if
         if empty rule AND empty word
            match succeed
         else
            match fail
         end if
      end if
   end for
end parser
```

5. EXPERIMENT

Rule generation process is performed by inspecting about 22,000 Arabic words. Words were extracted from 100 short Arabic articles collected randomly from the internet. Extracted words were normalized by removing vowels, if any, and then stored in binary file in the same order as the original natural text. Since word order was preserved, it is very easy to deduce the contextual meaning of the word by listing few words before and few words after the current word. Each word in the file was manually investigated where stem, prefix and suffix were manually generated and stored in the same file. The stem in the work is defined as a singular, masculine and past tense Arabic word without affixes.

The first part of the experiment was designed to study rule growth in a natural text. In this part each word passed to the parser for analysis. The parser has access to list of accumulated rules. The parser tries to fire rules in sequence. On a match, the word structure will be updated with number of fired rules, the id of matched rule and its sequence. On a mismatch, a new rule should be created and appended to the rule list then parser will be executed again.

The growth of rules is shown in Figure 3. It shows very rapid growth at lower number of words and a tendency to be stabilized as more words introduced. The figure is also shows number of generated rules for every thousand words. It clearly shows that number of generated rules decreases as number of words increases.

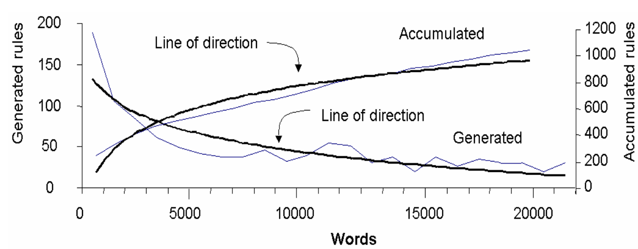

Figure 3. growth of generated rules

Order of rule firing plays an important role in the efficiency of the stemmer. For a given word, it is desirable to fire less number of rules and to maintain firing order in such a way that first fired rule is the matched one. Figure 4. shows the firing behavior of the stemmer for the set of rules arranged according to their generation order. Despite the uncontrolled list of rules in terms of its order, the experiment revile promising behavior. For a given word that fires a set of rules, it is most likely that the first fired rule will be achieve a match.

In order to optimize the stemmer performance the curve in Figure 4. should show a sharp drop. Although it is impractical to achieve such optimum state, it is possible to have certain rule ordering that produces the best performance for such rule set.

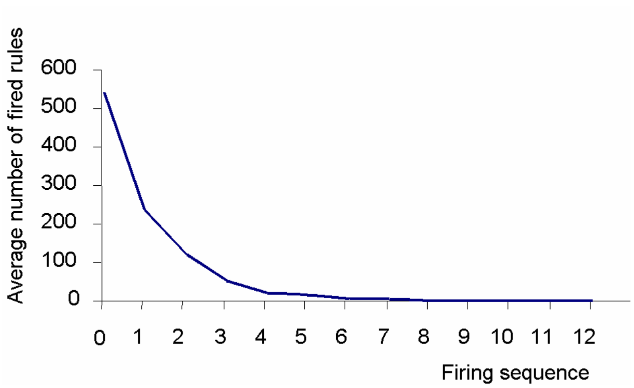

Figure 4. Average number of fired rules per 1000 words.

6. Conclusion

Available Arabic morphological analysis techniques suffer from few problems including slowness in processing and the need for prepared data. This paper introduced a new Arabic stemmer that requires neither prepared lists nor extensive computations. This work showed the practicality, simplicity and expandability of the proposed stemmer.

Firing policies should thoroughly be studied to enhance the accuracy and correctness of the proposed system. Furthermore, coverage of the system should be increased by introducing more rules. Rule merging and cascaded firing are currently under investigation.

7. References

[1] J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, No. 11, pages 22-31, March 1968.

[2] J. Dawson. Suffix removal and word conflation. *ALLC Bulletin*, 2(3): 33-46, 1974.

[3] N. Ali. *Arabic Language and Computer*. Ta'reeb,1988. (in Arabic)

[4] A. Alsuwaynea. *Information Retrieval in Arabic language*. King Fahad National Library, 1995 (in Arabic).

[5] M. Al-Atram. *Effectiveness of Natural Language in Indexing and Retrieving Arabic Documents*. KACST, AR-8-47. 1990. (in Arabic)

[6] M. El-Affendi. An algebraic algorithm for Arabic morphological analysis. *The Arabian Journal for Science and Engineering*. 16(4B):605-611, Oct 1991.

[7] S. Al-Fadaghi and F. Al-Anzi. A new algorithm to generate root-pattern forms. *Proceedings of the 11th National Computer Conference*, KFUPM, pages 391-400, March 1989.

[8] W. Frakes and R. Baeza-yates. Editors. *Information Retrieval: Data Structures & Algorithms*. Prentice hall, 1992.

[9] B. Thalouth and A Al-Dannan. *A comprehensive Arabic morphological analyzer/generator*. IBM Kuwait Scientific Center, February 987.

[10] T. El-Sadany and M. Hashish. An Arabic morphological system. *IBM Systems Journal*, 28(4):600-612, 1989.

[11] G. Kiraz. Computational analysis of Arabic morphology. Computer Laboratory, University of Cambridge, March 1995.

[12] N. Hegazi and A. Elsharkawi. Natural Arabic language processing. *Proceedings of the 9th National Computer Conference*, Vol. 2, Pages (10-5-1)-(10-5-17), Riyadh. October 1986.

[13] Y. Hlal. Morphology and syntax of the Arabic language. *Proceedings of the Arab School of Science and Technology*, pages 201-207, 1990.

[14] M. Gheith and T. El-Sadany. Arabic morphological analyzer on a personal computer. *Proceedings of the 1st KSU Symposium on Computer Arabization*, pages 55-65, April 1987.

[15] A. Aluthman. *A Morphological Analyzer for Arabic*. M. S. Thesis, KFUPM, Dhahran, 1990.

[16] K. Beesley. Finite state morphological analysis and generation of Arabic at Xerox research: status and plans in 2001. 2001. http://www.elsnet.org/arabic2001/beesley.pdf

[17] M. Aref. Object-oriented approach for morphological analysis. *Proceedings of the 15th National Computer Conference*. pages 5-11, KFUPM, Dhahran 1997.

[18] M. Albawab and M. Altabban. Morphological computer processing for Arabic. *Arabian Journal for Sciences*, No. 32, pages 6-13, 1998. (in Arabic)

[19] R. Al-shalabi. Design and implementation of an Arabic morphological system to support natural language processing. *Ph. D. Dissertation*. Computer Science Department, Illinois Institute of Technology. Chicago, 1996.

[20] M. El-Affindi. Performing Arabic morphological search on the internet: a sliding window approximate matching (SWAM) algorithm and its performance. Dept. of Computer Science. CCIS, KSU. Saudi Arabia.