Rule Merging in a Rule-Based Arabic Stemmer

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

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

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

Abstract

Semitic languages require more complicated systems for processing their morphology. Arabic language, for example, exhibits a very complex but very regular morphological struct ure. Many approaches were proposed to analyze Arabic language at the morphological level. Proposed approaches can be classified into table lookup, linguistic, combinatorial and rule-based techniques.

This paper proposes a new approach to enhance a rule-based Arabic stemmer. The enhancement is based on rule merging process to reduce number of rules, increase language coverage and maintain the same level of performance.

Introduction

Stemming and morphological analysis techniques are computational processes that analyze natural words by considering their internal structures. Stemming term is usually used by researchers dealing with languages with simple morphological systems while morphological analysis term, is widely used by researchers in languages with complex morphological systems.

Stemming and morphological analysis techniques can be viewed as clustering mechanisms and usually help in resolving the lexical ambiguity. 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. Both processes are very useful in many natural language applications such as information retrieval, text classification and categorization, text compression, data encryption, vowelization and spelling aids and automatic translation Lovins (1968), Dawson (1974).

Popovic showed that the effectiveness of a stemming algorithm of a given language is determined by its morphological complexity Popovic (1992). Semitic languages are derivational while most of other languages are concatenative. Arabic words are generated based on root-pattern structure Ali (1988), Alsuwaynea (1995), Al-Atram (1990). Stems are generated from roots using a few patterns. Affixes can be added to a stem to generate an Arabic word. A reverse process is used to analyze Arabic words. Figure 1. shows Arabic analysis and generation processes.

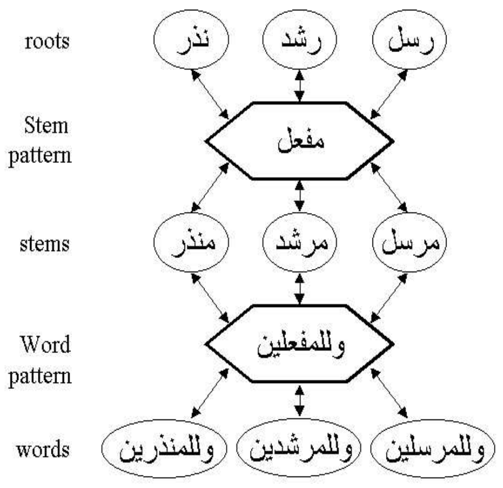

Figure 1. Generating/analyzing processes

1. Arabic Morphological System and Techniques

Arabic language exhibits a very complex but straightforward morphological system. Arabic root is the word's origin, usually trilateral or quadrilateral, before any transformation process. A Stem is a morpheme or a set of concatenated morphemes usually derived by applying pattern on a root and accepts affixes. A single root can create a set of stems using tens of patterns. Hundreds of natural Arabic words are created by applying affixation process on those stems. In addition, Arabic has a high degree of ambiguity because of many reasons such as missing of vowels and similarity between affixation letters and stem boundary letters. Any morphological analysis technique for Arabic language should consider such phenomena in order to correctly extract stem or root.

Computational Arabic morphology drew the attention during the last two decades. This, consequently, has led to the emerging of some morphological analysis techniques. Some researchers suggested analyzing Arabic words to reach their roots Ali (1988) while others suggested analyzing them to their stems only Alsuwaynea (1995), Al-Atram (1990). Analyzing words to their roots is preferred in linguistic-based applications while analyzing words to their stems is more useful in some other applications such as information retrieval-based systems.

Arabic morphological analysis techniques can be classified into table lookup, linguistic, combinatorial and rule-based approaches Ali (1988), El-Affendi (1991), Al-Fadaghi (1989).

Table lookup approaches uses huge list that stores all valid natural Arabic words along with their morphological decompositions. Arabic words are analyzed by searching for entries in the list and retrieving corresponding information.

Linguistic approaches, on the other hand, simulate the behavior of a linguist by considering Arabic morphological system and thoroughly analyzing Arabic words accordingly to their morphological components. Most of the published works were mainly linguistic based Ali (1988), Thalouth (1987), El-Affendi (1991), El-Sadany (1989), Kiraz (1995), Hegazi (1986), Hlal (1990), Gheith (1987), Aluthman (1990), Beesley (2001), Aref (1997), Albawab (1998).

In combinatorial approaches a given word is used to generate all combinations of letters. These combinations are compared against predefined lists of Arabic roots. On a match, root, stem, and patterns are extracted. Otherwise other combinations should be investigated Al-Fadaghi (1989), Al-Shalabi (1996), El-Affindi.

It should be noticed, however, that ambiguity of any word-based approach is a common symptom. Higher-level contextual analysis is usually the solution for such symptom.

2. Rule-Based Arabic Stemmer

Researchers, who proposed different morphological analysis techniques, were seeking for a high degree of accuracy. This caused proposed systems to be based on heavy computational processes and/or the usage of large amount of associated information.

The rule-based stemmer utilizes the apparent symmetry of generated natural Arabic words to suggest the stem of a given Arabic word. In this approach, a unique regular expression-based rule is generated for group of similar Arabic words as shown in Figure 2. To analyze Arabic words, some researchers suggested to reach their roots Ali (1988) while others suggested analyzing them to their stems only Al-Atram (1990), Alsuwaynea (1995). 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.

Rules are written from right to left to match script writing direction of Arabic language. Rules are used to describe the internal morphological structure of Arabic words and guide the decomposition process of a given word to its main parts i.e. stem, prefix and suffix. 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.

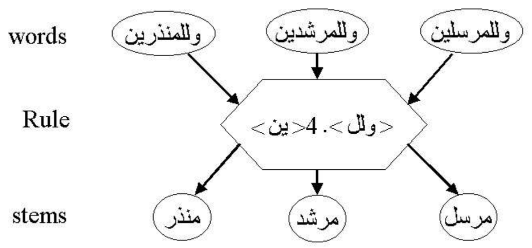

Figure 2. Regular expression-based rule.

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/suffix-part*** represents attached prefix/suffix, if any and ***stem-part*** represents stem structure and guide the process of extracting its original form.

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.

Simple rules are created to handle words already in stem forms, isolated articles, proper names and foreign words. Other rules are created to treat words with more complicated morphological structure. Table 1. Shows a small set of rules extracted from a list of about 1200 rules generated using the text collection.

To process rules and extract morphological components of word, a very simple rule parser was developed. The parser tries to match between rules and a given Arabic word. The matching process is achieved when the parser successively analyzes the word and decomposes it to its valid components according to the parsed rule.

The parser is divided into three distinct parts to treat prefix, suffix and stem. Interpreting the corresponding part of the rule mostly end up with extracting morphological components of a given word. Initially, the parser scans the suggested rule to identify boundaries of each part. The angle-bracketed substring at the rule boundaries 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 components.

Prefix and suffix are extracted using simple string matching process between word boundaries and prefix/suffix parts of the rule. Suffix may 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 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 morphological components. A given word should fire at least one rule and match only one rule. An Arabic data set Al kharashi (2002) has been used to test the correctness of the stemmer. A straightforward experimentation showed 80% correctness on identifying stems. The correctness of the stemmer can be improved by applying different firing policies.

Table 1. Small set of sample rules*.

Applied Rule	Word	Resultant		
		Prefix	Stem	Suffix
2. .2	في fy		في fy	
<ال>.4<ية> <al>.4<yp>	الهجومية alhjwmyp	ال al	هجوم hjwm	ية yp
<><ت>.<و^ا>. <><t>.<w^A>.	تزور tzwr		زار zAr	
<ل><ها> <l><hA>	لها lhA	ل l		ها hA
<ب>.5 <b>.5	بقيادة bqyAdp	ب b	قيادة qyAdp	
.6<ة> .6<p>	متسارعة mtsArEp		متسارع mtsArE	ة p

* Roman transliteration of sample rules is provided using the convention adopted by Beesley (1998)

3. Rule merging

It is clear that number of rules generated for pattern-based stemmer gets larger as language coverage increases. Such situation requires more management overhead and larger list of rules to maintain. Enhancement can be achieved by reducing number of rules while maintaining the same functionality.

Merging rules is one method that can be used for enhancing the rule-based stemmer. Rule merging is a clustering process performed on rule list to reduce its size and hence make it more manageable. Merging process basically decreases number of rules and makes single rule recognizes more than one pattern.

Many proposals for merging rules can be suggested. Stem-based merging class utilizes the similarity of affixation of different stem patterns while affixation-based merging class utilizes the similarity of stem patterns with different affixation. Based on its syntax, a given rule can be merged with either classes or left with no merging. For both merging approaches a new operator was introduced to express new merged rule. The operator "~" is used as logical *or* while merged parts of merged rules are listed between parenthesis and separated by the merge operator. Figure 3. shows two merging examples while Table 2. shows a small set of merged rules using both approaches.

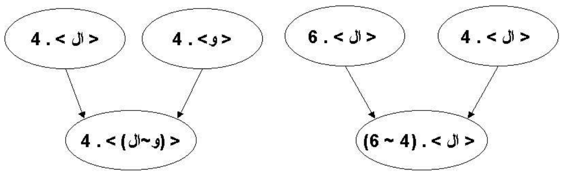

Figure 3. Example of merging rules.

Table 2. A small set of merged rules.

```
Stem-based:
    (12~11~10~9~8~7~6~5~4~3~2).
    (10~9~8~7~6~5~4~3~2) <ال> .
    < > <^ة> (3~2) . <و> .
    < > (5~4~3) . <ت^ا><>
Affixation-based:
    <( تنا ~ تهم ~ كم ~ هم ^ ة ~ وها ~ يات ^ ة )>3.
    < >< ( ص ^ ة ~ ط ^ ة ~ م ^ ة ) > 2.<ال>
    < > < ( اء ~ ى ) > . < ي ^> 2. < ال >
    . < ( ^ ا ~ ائ ~ ايي ^ يا ) > 2. < بال >
```

4. Empirical Evaluations

About 1120 rules generated for more than 23000 Arabic words were manually investigated and then merged to produce list of about 560 merged rules. A rule preprocessor was developed to handle merged rules before parsing any given word for analysis.

Growth of merged and non-merged rules needed to analyze the Arabic data set was studied. The parser has access to a list of accumulated rules both merged and non-merged. The parser fires rules in sequence to analyze a given Arabic word. On match, the word structure will be updated with number of fired rules, the id of matched rule and its sequence. On mismatch, a new rule should be created and appended to the rule lists. For merged rule list, an existing expandable rule can be updated to correctly analyze the new word.

Figure 4. shows very rapid growth at lower number of words and a tendency to be stabilized as more words introduced. Merged rules, however, show a tendency to stabilize faster than non-merged rule case. Figure 5. depicts number of generated rules for every thousand words. It clearly shows that number of generated rules decreases as number of words increases in both cases but faster in the merged rules case.

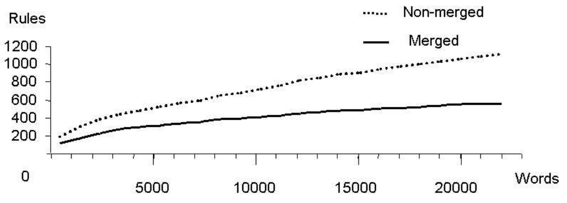

Figure 4. Growth of merged and non-merged rules per 1000 words

Time needed to analyze word collection using both merged and non-merged rules was studied. Figure 6. shows time needed for analyzing words in both cases per 1000 words. Figure 7. shows accumulated time to analyze words per 1000 words. The study shows a small time overhead needed to preprocess the merged rule set. Such overhead can be tolerated as long as a smaller set of manageable and expandable rule set can be produced.

Analyzer efficiency is greatly affected by the order of rule firing. It is desirable for any word to fire less number of rules and to maintain firing order in such a way that first fired rule is the matched one. In order to optimize the stemmer performance the average number of fired rules should be as low as possible. Although it is impractical to achieve ideal state, it is possible to have certain rule ordering that produces the best performance for such rule set. Figure 8. shows the average number of matched rules for each rule sequence per thousand words for both merged and non-merged cases. The figure shows that performance of merged rules outperforms the non-merged due to the higher number of matched rules with sequence zero. The curve shows a sharp drop for the merged rules indicating decrease in the number of matched rules with higher frequency sequences.

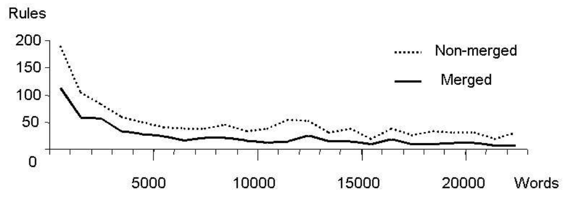

Figure 5. Number of merged and non-merged rules generated per 1000 words.

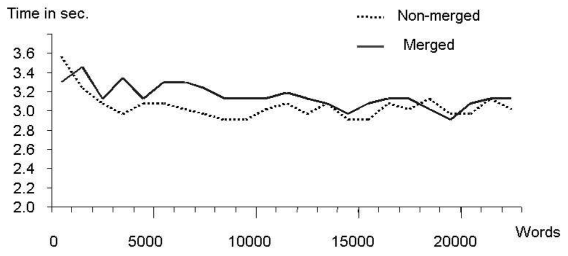

Figure 6. Time needed for analyzing words in merged and non-merged cases per 1000 words.

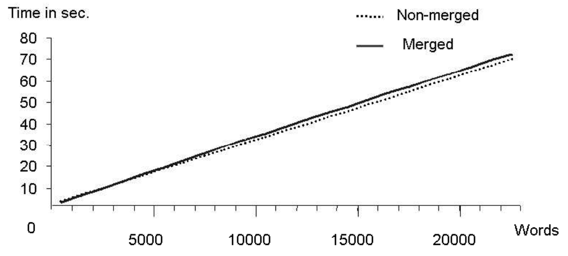

Figure 7. Accumulated time needed for analyzing words in merged and non-merged cases per 1000 words.

Figure 9. shows the relationship between matched and total firings per rule for merged and non-merged cases. It reflects the firing behavior of the stemmer for the set of rules arranged according to their generation order. Having different rule orders will produce different plots. In order to achieve optimized performance the curve of Figure 9. should follow the horizontal line. Despite the uncontrolled ordering of rules, the experiment reveals promising behavior. For a given word that fires a set of rules, it is most likely that the first fired rule will achieve a match. Although it is impractical to achieve ideal state, it is possible to have certain rule ordering that produces the best performance for such rule set.

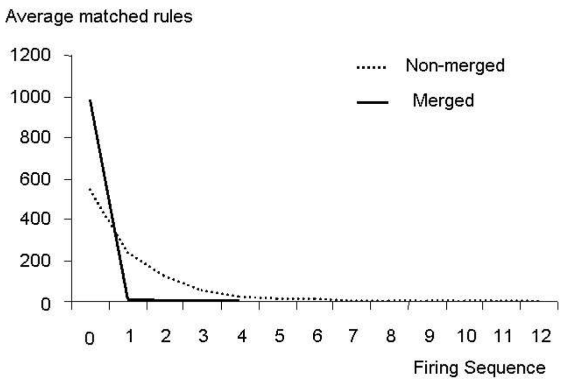

Figure 8. Average number of matched rules for each sequence per 1000 words.

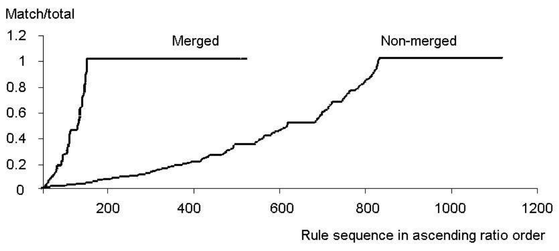

Figure 9. Relation between matched and total firings.

Conclusion

This paper introduced the concept of rule merging for rule-based Arabic stemmer. The stemmer has been expanded to handle merged rules. While keeping the time efficiency almost the same, merging rules reduces the rule list dramatically. Also merged rules shows a better performance in terms of average number of matched rules for every rule sequence and in terms of the match/total firing ratio. Merging concept opens the possibility to increase the system coverage by expanding most of the existing rules.

Rule merging can be utilized to enhance the stemmer performance. Better enhancement can be achieved by introducing better clustering mechanism such as cascading and back referencing.

References

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

Al-Kharashi I. and Al-Sughaiyer I. (2002) *Data Set for Designing and Testing an Arabic Stemmer.* Proceedings of Arabic Language Resources and Evaluation: Status and Prospects. Spain.

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

Al-Fadaghi S. and Al-Anzi F. (1989*) A new algorithm to generate root-pattern forms . Proceedings of the 11th National Computer Conference*, KFUPM, pp. 391-400.

Albawab M. and Altabban M. (1998) *Morphological computer processing for Arabic. Arabian Journal for Sciences*, 32, pp. 6-13. (in Arabic)

Ali N. (1988) *Arabic Language and Computer.* Ta'reeb. (in Arabic)

Aref M. (1997*) Object -oriented approach for morphological analysis. Proceedings of the 15th National Computer Conference*. Pp. 5-11, KFUPM.

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

Aluthman A. (1990) *A Morphological Analyzer for Arabic.* M. S. Thesis, KFUPM.

Beesley K. (1998) *Romanization, Transcription and Transliteration,* http://www.xrce.xerox.com/competencies/content-analysis/arabic/info/romanization.html

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

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

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

El-Affindi M. *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.

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

Gheith M. and El-Sadany T. (1987) *Arabic morphological analyzer on a personal computer. Proceedings of the 1st KSU Symposium on Computer Arabization*, pp. 55-65.

Hegazi N. and Elsharkawi A. (1986) *Natural Arabic language processing. Proceedings of the 9th National Computer Conference*, 2, pp. (10-5-1)-(10-5-17).

Hlal Y. (1990) *Morphology and syntax of the Arabic language. Proceedings of the Arab School of Science and Technology*, pp. 201-207.

Kiraz G. (1995) *Computational analysis of Arabic morphology.* Computer Laboratory, University of Cambridge.

Lovins J. (1968) *Development of a stemming algorithm. Mechanical Translation and Computational Linguistics*, 11, pp. 22-31.

Popovic M. and Willet P. (1992) *The effectiveness of stemming for natural-language access to Slovene textual data. Journal of the American Society for Information Sciences*. 43/5, pp. 384-390.

Thalouth B. and Al-Dannan A. (1987) *A comprehensive Arabic morphological analyzer/generator.* IBM Kuwait Scientific Center.