New Approach for Extracting Quadrilateral Arabic Roots

Ghassan Kanaan*, Riyad Al-Shalabi**, and Mohammed Naji Al-Kabi*

Received on June 6, 2004
Accepted for publication on Dec. 17, 2004

Abstract

This algorithm provides a new method for extracting the quadrilateral Arabic root (a four consonant string) from its morphological derivatives. Our stemming algorithm starts by excluding prefixes and checking the word starting from the last letter back to the first. A temporary vector is used to store the suffix letters being removed, and another vector is used to store the root. Particles and the definite article are removed before the suffix and root are partitioned. The algorithm has been tested on a sample of 145 words derived from quadrilateral Arabic verbs, with 95% accuracy for the initial results.

1. Introduction:

The Arabic language is the fifth most widely spoken language in the world. It belongs to the Semitic family; so it differs from the Indo-European languages morphologically, semantically, and syntactically. The Arabic alphabet contains twenty-eight letters, always written from right to left in cursive form. Diacritical marks (tashkil تشكيل) appear either above or below the letters, and play an essential role in many cases in distinguishing semantically and phonetically between two identical words with the same characters, but with different diacritics. Diacritical marks are used in holy books, poems, and children's literature; newspapers, journals and other books for adults are usually printed without diacritics, which mean that many strings are ambiguous. Most native Arabic words are derived from verbal roots. Arabized words, on the other hand, mainly nouns borrowed from other languages with a slight phonetic adjustment to suit the Arabic pronunciation, have no roots.

© 2005 by Yarmouk University, Irbid, Jordan.
* Department of Computer Information Systems, Faculty of Information Technology and Computer Sciences, Yarmouk University, Irbid, Jordan.
**Department of Computer Science, Faculty of Information Technology and Computer Sciences, Yarmouk University, Irbid, Jordan.
All Arabic words belong to three main categories: noun, verb or particle. Around 64% of Arabic words are derived from triliteral verbs (three consonants), but there are also biliteral verbs (two consonants), quadrilateral verbs (four consonants), and pentaliteral verbs (five consonants). Naturally these verbs represent the roots for which stemming algorithms typically search. This stemming process excludes words derived from nouns and particles. [1].

Previous studies of the Arabic language have shown that the majority of Arabic root verbs are of triliteral origin "ثلاثي"، while the remaining are of quadrilateral " رباعي", pentaliteral, and biliteral origin "ثنائي" [2]. In most cases roots with 2, 4, or 5 consonants represent various modifications of the original triliteral verbs.

These modifications are represented by special types of structures called templates "وزان". Arab grammarians use the verb "فعل" "faal" as the model to represent the templates that are used to form other verbs and nouns from a specific root. Each letter of the special verb "فعل" has a specific name and meaning applied in the formation of other forms of verbs and nouns, the first letter is called "ف" "fa", the second is "خ" "ain", and the third is "ل" "lam".

<table>
<thead>
<tr>
<th>Root Form</th>
<th>Root Pronunciation</th>
<th>Generated Form</th>
<th>Meaning of Generated Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>نحاس</td>
<td>Handz&lt;sup&gt;a&lt;/sup&gt;</td>
<td>لعالج</td>
<td>مهندس</td>
</tr>
<tr>
<td>علخ</td>
<td>Alal&lt;sup&gt;a&lt;/sup&gt;</td>
<td>ترجمان</td>
<td>معالجات</td>
</tr>
<tr>
<td>ترمج</td>
<td>Tarjama</td>
<td>زلزال</td>
<td>مترجمان</td>
</tr>
<tr>
<td>زالج</td>
<td>Zalzala</td>
<td>بيرم</td>
<td>زلزال</td>
</tr>
<tr>
<td>برمج</td>
<td>Barmaja</td>
<td>برمج</td>
<td>مبرمج</td>
</tr>
<tr>
<td>برمج</td>
<td>Barmaja</td>
<td>He is programming</td>
<td></td>
</tr>
</tbody>
</table>

Nouns can be classified as inflected "متر" (those that have variable diacritical endings due to their functional position in a sentence), or uninflected "مني". (those that have constant diacritical endings regardless of their position in a sentence). Nouns may have masculine "منكر" or feminine "مؤنث" gender. The definite article in the Arabic language is "ال" "al", which is written as a prefix to both nouns and adjectives. Nouns may have singular, dual, or plural number. Dual and plural nouns can be formed by adding suffixes to a singular noun, but some plurals are irregular in their grammatical formation.
New Approach for Extracting Quadrilateral Arabic Roots

Arabic particles include Explanations, i.e. (أي) "that is"; Prepositions, i.e. (في) "to"; Adverbial particles, i.e. (سوف) "shall"; Conjunctions, i.e. (و) "and"; Interjections, i.e. (يا) "you"; Exceptions, i.e. (عصرى) "Except"; Negatives, i.e. (لم) "Not"; Answers, i.e., (أجل) "yes"; Subordinates, i.e. (لا) "if". Traditional grammars include 81 such particles. [1]

A good Arabic stemmer is needed in many applications, such as: natural language processing, computerized language translation, compression of data, spell checking as well as in information retrieval. The majority of the existing Arabic stemming algorithms use a large set of rules, and many algorithms also refer to existing pattern and root files, and this requires a large amount of storage space, in addition to being time consuming [3].

The complexity of the stemming process depends on the complexity of the morphology of the language. Stemming Arabic words is very complicated, since the morphology of Arabic is very complex. Arabic words have many morphological variants, which cannot be recognized in a straightforward manner by using string-matching algorithms. In most cases, additional processing is needed since these variants have equivalent semantic interpretations, which can be treated equivalently for information retrieval. In this case, the stemming algorithms are used to reduce these variants to a root form.

Additional complexity is introduced by the diacritization of scripts, that is, vowels may or may not be included in the text. All of these factors and other factors such as its sensitivity to gender, number, case, degree, and tense, make Arabic very difficult to deal with [4]. A root in Arabic is the bare verb form which can be triliteral as in the overwhelming majority of Arabic words, and to a lesser extent, quadrilateral, pentaliteral, or (very rarely) hexaliteral; each root generates other verb forms and noun forms by the addition of derivational affixes [5]. The number of verb roots is listed as 7,500 to 10,000 by different authorities.

In summary, there are many factors that make stemming of Arabic words difficult: bare (nonvocalized) words are more ambiguous than vocalized words (with diacritics), this may cause mismatch with texts, lexicons, or word lists with diacritics. The Arabic language faces the problem of regional variation in spelling of some words, similar to the differences in spelling of some English words on opposite sides of the Atlantic. Other problems arise from the derivational and inflectional productivity of Arabic. Words can be found in a huge number of different forms, which should possibly be conflated for informational retrieval. A wide range of articles, conjunctions, prepositions, and other prefixes can be attached to the beginning of a word, and a large number of suffixes can be attached to the end. Furthermore, there is a common problem of irregularity of singular and plural nouns, which is not related to simple affixing. [6].

Researchers in computational linguistics have developed tools for the Arabic language, which can be used in information retrieval. Lexical databases can be used to
find stems also that is, a database can be constructed in both English and Arabic to provide a morphological analysis of any word in the lexicon and identify the base form.

In this paper, we present a detailed analysis of a new stemming algorithm suitable for use in extracting the quadrilateral Arabic root.

2. Previous Research in the Stemming of Arabic Quadrilateral Verbs

Extracting the quadrilateral Arabic root from its derivatives involves the problems already mentioned in the introduction, which contribute to the difficulties of the stemming process. There are dozens of published papers discussing stemming in Arabic, but they mainly rely on four basic approaches: statistical stemmers, manually constructed dictionaries, light stemmers, and morphological analyses.

The number of papers of stemming algorithms for English language exceeds those of stemming algorithms for Arabic language by many folds. We will present briefly some of the research dedicated to the Arabic language, and neglect English language stemmers, since their approaches are not fully appropriate for the development of Arabic stemmers, due to the morphological and semantic differences between these languages, refer to [7] for these differences.

It is worth noting here that all the following Arabic stemmers are unable to extract the quadrilateral Arabic root from words derived from such roots.

gheith/el-Sadany Morphological Analyzer

Gheith and El-Sadany morphological analyzer was developed for Arabic words with triliteral roots [8]. The analyzer was implemented on the IBM PC-AT machine. The main aim of developing this analyzer is to assist in research in Arabic natural language processing. This morphological analyzer is the work of a team consisting of Arabic grammarians and computer scientists who have developed a computerized morphological analyzer for Arabic words without diacritics [8]. The analyzer consists mainly of:

- A set of routines dealing with reading and building up a suitable data structure for each type of grammar rules;
- Lexicon;
- Interpreter which has several routines to process, analyze the input word and update the features of the words that have inflectional forms depending on the types of the affixes attached to their roots;
- A set of routines for building the lexicon, the search procedure, and general operations.

The lexicon is developed to achieve the following: the homogeneity of the lexicon entries, expandability of the lexicon and the efficiency of the retrieval process of a root. Each root has one or more lexical entries. Each entry contains category features, inherent-features and contextual-features. It also contains the allowed forms of the root
with their constraints. The roots themselves are not included in the lexicon, but only used to indicate the appropriate entry for the lexicon. This entry is calculated by hashing the root and this hashing value is used as the entry to the inverted file that represents the lexicon. For example, the root “اب” is used to find entry to the inverted file as follows:

\[ Entry \text{ no} = \text{dec}(\alpha) + \text{dec}(30 \times \text{پ} + \text{dec}(30 \times \text{ب}) \]

Where \( \text{dec}(\alpha) \) is the decimal value of the character \( \alpha \). Arabic has an alphabet of 30 characters and each alphabetic character has been given a decimal value from 1 to 30. The lexicon uses an Augmented Transition Network (ATN) to parse the input word and to find its root and its morphological analysis. If the lexical program encounters an ambiguous word, it will try all other possible choices for that word one at a time until a transition is enabled. Hence, this approach offers the possibility of successfully analysing the input word in accordance to the grammar utilization. In case the choice does not lead to a successful analysis, then the program backs up and tries again and this choice will be discounted. All and only the choices that lead to successful analysis will be output with these analyses. This morphological analyzer’s main aim is to solve ambiguity in Arabic words with triliteral roots. The system is tested for verb derivatives and articles, giving satisfactory results [8]. The disadvantage is that this analyzer was developed to process only words of triliteral root origin and does not apply to others.

El-Sadany/Hashish Morphological Analyzer

This morphological analyzer was implemented on an IBM PS/2 at Cairo’s IBM Scientific Center [7], using the PROLOG language. This analyzer is able to process voweled, semi-voweled, and non-voweled input. The analyzer is precisely aimed at analyzing only words of triliteral origin. This morphological analyzer consists of five main modules that together take a word as an input and then output a set of possible analysis.

Saliba/AL-Dannan Morphological Analyzer

This morphological analyzer was developed at the IBM Scientific Center in Kuwait [5]. This analyzer is somewhat similar to one developed by [7], but its lexicon has more affixes and the system has more extraction rules. This morphological analyzer can process types of non-voweled, semi-voweled and voweled Arabic words with emphasis given to modern Arabic words. The main modules of this analyzer are:

- Prefix lexicon
- Suffix lexicon
- Pattern lexicon
- Root lexicon
- Loan word lexicon
Lexicon structural word

Linguistic rules

Extraction rules

The process of analysis takes an Arabic input word and tries to produce the correct root for this word. The algorithm can be summarized as follows:

1. Find the valid prefix and suffix and remove them from the word;
2. Use the resulting stem from 1 to find the stem template;
3. Extract the root;
4. Compare with the lexicon to check for the applicability of the existing affixes and the root;
5. If needed, recode the root.

Hilal Morphological Analyzer

This morphological analyzer was implemented on an IBM PC. The analyzer consists of three main modules:

- An Article (stop-words) analyzer module,
- A Word analyzer module,
- A Lexicon module,

The article module checks whether an input Arabic word is an article or not. If the word is found to be an article, then the analysis process will stop, otherwise, it will be passed to the word analyzer module. The word analyzer uses the longest-affix match removal approach to remove the longest possible prefixes and postfixes from the word. The resulted stem after the removal of the affixes is analyzed to produce the correct root. This analysis process is carried as follows:

- Identify the applicable rule (stored in the lexicon) by checking the longest prefix, longest postfix, and the stem;
- Load the rule into the memory to speed up the process;
- Use the information that are associated with the rule, apply the rule against the word and get the root.

Al-Omari Morphological Analyzer

The morphological analyzer was implemented at the Department of Computer Science, University Kebangsaan Malaysia in 1993 as part of a multilingual document retrieval system. The analyzer uses a set of rules that must be loaded before the actual analysis begins. The analyzer consists of three main modules: the stop-word analyzer module, the word analyzer module, and the system knowledge base module. The stop-word analyzer module is where the input word is checked to see whether it is a stop word or not, similar to Hilal’s algorithm. This module contains a knowledge base that contains rules for determining valid stop words and non-valid ones. There are around 175 stop words defined in this analyzer. The second module is the word analyzer module.
New Approach for Extracting Quadrilateral Arabic Roots

which generates roots. This analyzer consists of three sub-modules: the prefix analyzer, the postfix analyzer, and the root extractor. The prefix analyzer finds all possible valid prefixes attached to the input word (which is not an article). Thus, this sub-module produces a list of possible prefixes that are attached at the beginning of the word. The postfix analyzer finds all the possible valid postfixes attached to the input word (which is not an article). Thus, this sub-module will produce a list of possible postfixes that are attached at the end of the word.

The root extractor module finds all the possible valid root(s) that can be extracted from the word after removing the prefixes and the postfixes. This module consists of five sub-modules. One of these sub-modules is the stem generator, which generates all valid stems from the triple (prefix, postfix, the stem). This process will be performed on all combinations of the prefix list obtained from the prefix analyzer and the postfix list obtained from the postfix analyzer. The output of this formation will be a set of possible stems. Another sub-module of the root extractor is the root generator. This sub-module finds the possible roots from the generated stems. All the information for the various modules and sub-modules of this morphological analyzer are stored in the knowledge base in a dictionary form. There is one dictionary concerned with the prefix rules and their applications to the word; another dictionary is used for the postfix rules and their applications to the word, and a few other dictionaries for the various sub-modules. Lists of many valid Arabic roots and templates are also found in the knowledge base dictionaries. This morphological analyzer performs quite well but with few limitations. The system refuses to stem words that do not have any matches in the knowledge base. The number of Arabic stop words, templates and roots handled by the system is relatively few.

Al-Fedaghi & Al-Anzi Morphological Analyzer

Al-Fedaghi and Al-Anzi [9] present an algorithm to generate the root and the pattern of a given Arabic word. The main idea in this algorithm is to locate the position of the root’s letters in the pattern and examine the letters in the same position in a given word to see whether they form a valid Arabic root or not. The algorithm examines all patterns that have the same length as the input word in case that word contains its full trilateral root. For those that lose one of their letters, the algorithm tests all patterns of lengths equal to one less than the length of the given word, and for the roots that lose two letters it examines all patterns of length equal to the length of the given word minus two. The algorithm has been tested by the authors in four modes as follows:

Mode1- The input word contains its full trilateral root.
Mode2- The third letter of the root of the input word is lost due to "التشدید" reduplication (which means the second and the third letters are identical).
Mode3- One letter of the root of the input word is missing.
Mode4- Two of the letters of the root of the input word are missing.
The algorithm has to check the input word against the four modes, if it fails to find the root of the word in mode 1, then it checks the input word in the second mode, and so on.

Al-Shalabi Computational Morphology System for Arabic Language

Al-Shalabi [10] developed and implemented an algorithm for finding the roots of Arabic words; this algorithm can be summarized as follows:

1- Remove the longest prefix that precedes the first root letter in the input word.

2- Denote the new word formed by new-word. The root letters of a given word will be within the new-word. In fact they will be within the first four or five letters.

3- Check the first, second and third letter of the new-word to see whether it is in the root list or not.

4- If not, check the other possibilities of the trigraph formed by the first, second and fourth letters and so on.

3. The New Two-Stage Algorithm

Word is first submitted to light stemming to remove affixes and particles, and then the number of letters left is checked. The second stage derives the quadrilateral Arabic root. The algorithm used in this paper is described in the following pseudo-code:

Algorithm Quadrilateral:

Input: Any sequence of Arabic alphabetic characters with delimiters.

Output: A quadrilateral Arabic root, or an error message.

Read Arabic String $T$ (text) with $m$ characters
Remove the prefix (Ji) from $(T)$ if it appears
Remove the Arabic particles from $(T)$ if they appear
   { The end of first stage – Light stemming }
$NewT \leftarrow T$ after excluding the particles and (Ji) if they appear
$n \leftarrow$ size of $(NewT)$
If $n < 4$ then
   Ignore the whole string and exit
iTTemp $\leftarrow 0$
iRoot $\leftarrow 0$
New Approach for Extracting Quadrilateral Arabic Roots

if (n >= 4) then
for i ← 1 to n do
    if NewT[i] is in the word { سلمون 🌪} then
        iTemp ← iTemp + 1
        Temp_Array[iTemp] ← NewT[i]
        Temp_Post[iTemp] ← i
    else if (NewT[i] ≠ 'ג') and (NewT[i] ≠ 'ג') then
        iRoot ← iRoot + 1
        Root_Array[iRoot] ← NewT[i]
        Root_Post[iRoot] ← i
    else if (NewT[i] = 'ג') and (NewT[i−1] = 'ג') and (i > 1) then
        iRoot ← iRoot + 1
        Root_Array[iRoot] ← ג
        Root_Post[iRoot] ← i
    else if (NewT[i] = 'ג') and (i > 1) then
        iRoot ← iRoot + 1
        Root_Array[iRoot] ← /
        Root_Post[iRoot] ← i
while (iRoot < 4) do
    if (n = 5) and (Temp_Array[iTemp] = "ן") and (Temp_Post[iTemp] = 5) then
        iRoot ← iRoot + 1
        Root_Array[iRoot] ← Temp_Array[iTemp]
        Root_Post[iRoot] ← Temp_Post[iTemp]
        Temp_Post[iTemp] ← 0
        Temp_Array[iTemp] ← " "
        iTemp ← iTemp − 1

59
else if (n = 6) and (Temp_Array[iTemp] = "آ") and (Temp_Post[iTemp] = 6) then
  iRoot ← iRoot + 1
  Root_Array[iRoot] ← Temp_Array[iTemp]
  Root_Post[iRoot] ← Temp_Post[iTemp]
  Temp_Post[iTemp] ← 0
  Temp_Array[iTemp] ← ""
  Temp_Post[iTemp - 1] ← 0
  Temp_Array[iTemp - 1] ← ""
  iTemp ← iTemp - 2

else if (iTemp > 2) and (Temp_Array[iTemp] = "ن") and (Temp_Array[iTemp - 1] is in the word {هاء،اء}) and (Temp_Post[iTemp - 1] = Temp_Post[iTemp] - 1) then
  iRoot ← iRoot + 1
  Root_Array[iRoot] ← Temp_Array[iTemp - 2]
  Root_Post[iRoot] ← Temp_Post[iTemp - 2]
  Temp_Post[iTemp - 2] ← Temp_Post[iTemp - 1]
  Temp_Array[iTemp - 2] ← Temp_Array[iTemp - 1]
  Temp_Post[iTemp - 1] ← Temp_Post[iTemp]
  Temp_Array[iTemp - 1] ← Temp_Array[iTemp]
  Temp_Post[iTemp] ← 0
  Temp_Array[iTemp] ← ""
  iTemp ← iTemp - 1

else if (iTemp > 2) and (Temp_Array[iTemp] = "ل") and (Temp_Array[iTemp - 1] = "ل") and (Temp_Post[iTemp] = n) then
  iRoot ← iRoot + 1
  Root_Array[iRoot] ← Temp_Array[iTemp - 2]
  Root_Post[iRoot] ← Temp_Post[iTemp - 2]
  Temp_Post[iTemp - 2] ← Temp_Post[iTemp - 1]
Temp_Array[iTemp-2] ← Temp_Array[iTemp-1]
Temp_Post[iTemp-1] ← Temp_Post[iTemp]
Temp_Array[iTemp-1] ← Temp_Array[iTemp]
Temp_Post[iTemp] ← 0
Temp_Array[iTemp] ← ""
iTemp ← iTemp - 1

else if (iTemp > 1) and (Temp_Array[iTemp] is in the word \{κατ\}) and
(Temp_Post[iTemp] = n) then
    Temp_Post[iTemp] ← 0
    Temp_Array[iTemp] ← ""
iTemp ← iTemp - 1

else if (iTemp > 1) and (Temp_Array[iTemp] is in the word \{κατ\}) then
    iRoot ← iRoot + 1
    Root_Array[iRoot] ← Temp_Array[iTemp-1]
    Root_Post[iRoot] ← Temp_Post[iTemp-1]
    Temp_Post[iTemp-1] ← Temp_Post[iTemp]
    Temp_Array[iTemp-1] ← Temp_Array[iTemp]
    Temp_Post[iTemp] ← 0
    Temp_Array[iTemp] ← ""
iTemp ← iTemp - 1

else if (iTemp > 2) and (Temp_Array[iTemp] = 'ω') and (Temp_Array[iTemp-1] = 'ι') and
(Temp_Post[iTemp] = n) then
    iRoot ← iRoot + 1
    Root_Array[iRoot] ← Temp_Array[iTemp-2]
    Root_Post[iRoot] ← Temp_Post[iTemp-2]
    Temp_Post[iTemp-2] ← Temp_Post[iTemp-1]
    Temp_Array[iTemp-2] ← Temp_Array[iTemp-1]
    Temp_Post[iTemp-1] ← Temp_Post[iTemp]
Temp_Array[iTemp-1] ← Temp_Array[iTemp]
Temp_Post[iTemp] ← 0
Temp_Array[iTemp] ← " "
iTemp ← Temp-1

Else

iRoot ← iRoot + 1
Root_Array[iRoot] ← Temp_Array[iTemp-2]
Root_Post[iRoot] ← Temp_Post[iTemp-2]
Temp_Post[iTemp] ← 0
Temp_Array[iTemp] ← " "
iTemp ← Temp-1

Sort the letters in Root_Array according to their order in the original word, which stored in Root_Post array

Return Root_Array

4. Evaluation

The algorithm is fully implemented using Visual Basic. In order to test the accuracy of our algorithm, we selected a number of words randomly from a list of common words derived from quadrilateral Arabic verbs. In Table 2, we trace the execution of the algorithm manually on 12 words.
Table 2. Trace of the Manual Extraction of the Correct Quadrilateral Root from 12 Words

<table>
<thead>
<tr>
<th>Original Word</th>
<th>Root Array</th>
<th>Text Array</th>
<th>Result Array</th>
<th>Input Array</th>
<th>Output Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>متسلاسلة</td>
<td>متسلاسلة</td>
<td>لس لس</td>
<td>مس لس</td>
<td>لس لس</td>
<td>مس لس</td>
</tr>
<tr>
<td>السلاسل</td>
<td>سلاسل</td>
<td>لس لس</td>
<td>مس لس</td>
<td>لس لس</td>
<td>مس لس</td>
</tr>
<tr>
<td>فرعون</td>
<td>فرعون</td>
<td>ون فرع</td>
<td>ون فرع</td>
<td>ون فرع</td>
<td>ون فرع</td>
</tr>
<tr>
<td>الزلازل</td>
<td>الزلازل</td>
<td>لال زز</td>
<td>لال زز</td>
<td>لال زز</td>
<td>لال زز</td>
</tr>
<tr>
<td>وسواس</td>
<td>وسواس</td>
<td>وس وس</td>
<td>وس وس</td>
<td>وس وس</td>
<td>وس وس</td>
</tr>
<tr>
<td>السمسار</td>
<td>السمسار</td>
<td>رس مس</td>
<td>رس مس</td>
<td>رس مس</td>
<td>رس مس</td>
</tr>
<tr>
<td>الشياطين</td>
<td>الشياطين</td>
<td>شط يط</td>
<td>شط يط</td>
<td>شط يط</td>
<td>شط يط</td>
</tr>
<tr>
<td>يتبزخن</td>
<td>يتبزخن</td>
<td>يتن برزخ</td>
<td>يتن برزخ</td>
<td>يتن برزخ</td>
<td>يتن برزخ</td>
</tr>
<tr>
<td>المهندسات</td>
<td>المهندسات</td>
<td>مهندسات مس</td>
<td>مهندسات مس</td>
<td>مهندسات مس</td>
<td>مهندسات مس</td>
</tr>
<tr>
<td>معالجات</td>
<td>معالجات</td>
<td>عل جم</td>
<td>عل جم</td>
<td>عل جم</td>
<td>عل جم</td>
</tr>
<tr>
<td>المترجمات</td>
<td>المترجمات</td>
<td>رج مس مس</td>
<td>رج مس مس</td>
<td>رج مس مس</td>
<td>رج مس مس</td>
</tr>
<tr>
<td>سلاطين</td>
<td>سلاطين</td>
<td>ط س لس</td>
<td>ط س لس</td>
<td>ط س لس</td>
<td>ط س لس</td>
</tr>
</tbody>
</table>

Second, we show the execution of the program based on our algorithm. The inputs to this program consisted of 145 words derived from quadrilateral Arabic verbs, and stored in a text file, as shown in Figure 1.
5. Summary and Conclusions:

In this paper, we have described the design and successful implementation of a new algorithm suitable for extracting the quadrilateral (four consonants) Arabic root of words derived from such roots. The algorithm has been implemented using Microsoft Visual Basic 6.0.

Our approach is characterized by simplicity, and independence from any root and pattern files, as in other stemming algorithms. The methodology used proved to be more efficient in extracting the correct Arabic quadrilateral root than other methodologies. The algorithm was tested on a sample of 145 words derived from quadrilateral Arabic verbs; it successfully extracts the proper roots with 95% accuracy.

Future work will concentrate on enhancing the algorithm so that it can extract the roots of foreign words, irregular words, or words that do not have quadrilateral roots, which the present algorithm fails to recognize.
مختصر

تعود مصادر الكلمات في اللغة العربية إلى جذور (Roots)، حيث تتكون جذور اللغة العربية من ثلاثة أحرف أو أربعة أحرف ونادرًا ما تكون مكونة من خمسة أحرف. هناك عدة خوارزميات لاشتقاق جذور الكلمات في اللغة العربية الثلاثية، ولم يتطرق الباحثون إلى كيفية اشتقاق الجذور الرباعية للكلمات العربية و من هنا جاء التفكير في خوارزمية لإيجاد الجذور الرباعية تساعد في تحليل النصوص العربية وحالاتها حاسوبيًا، ويمكن استخدام هذه الخوارزمية لتكوين برنامجاً تعليمياً يستفيد منه في المدارس والجامعات.

يقدم هذا البحث خوارزمية جديدة لإيجاد مصادر الأفعال الرباعية، أي إيجاد الجذور الرباعي للكلمات العربية المشتقة من أصول ربعية. تبدأ الخوارزمية بحذف أول التعريف إن وجدت، ثم تقوم بتدقيق الكلمة بأسلوب مكسي، من النهاية إلى البداية، وتستخدم لهذا الغرض مصفوفة مؤقتة لتخزين اللاحقات (Suffix) التي تم حذفها، وتستخدم مصفوفة أخرى لتخزين الجذور الرباعي وأدوات التعريف.

تم اختيار هذه الخوارزمية على عينة تضم 145 كلمة عربية مشتقة من جذور رباعية، لتحصل على نتائج تصل دقتها إلى 95%.

References:


