Rule-based Approach for Arabic Root Extraction: New Rules to Directly Extract Roots of Arabic Words

Fatma Abu Hawas[1] and Keith E. Emmert[2]

[1] Department of Computer Science, Yarmouk University, Jordan
[2] Tarleton State University, Stephenville, Texas, USA

Extracting word roots in Arabic language is very problematic due to the specific morphological and structural changes in the language. To address this problem, several techniques have been proposed. This paper continues the problem of identifying and exploiting relationship amongst Arabic letters for Arabic root extraction begun in [1]. Eight different rules that detect the root letters according to other letters in the word have been proposed and tested, four of them benefiting from the idea of morphological substitution (MUTATION). The approach has been evaluated using the Holy Quran words. The evaluation results show a promising root extraction algorithm.

Keywords: rule-based stemmer, word root, suffixes, prefixes, words patterns

1. Introduction

The morphology of the English language has been well studied and documented. However, the nature of the Arabic language slows the development of effective stemmers. There are more than 223 million native Arabic language speakers, more than of any other Semitic language [6]. Arabic is a very rich language which makes root extraction a very complicated task. Some significant features of the Arabic language include:

- Arabic letters can be divided into two groups; the first one contains the affix letters[1] that form the word (سألتمونيها $s\text{'}altmwnyh\bar{a}$), and the second one includes the rest of Arabic alphabetic letters.
- The letters (ا، و، ي $\bar{a}, w, y$) are the long vowels and the rest are consonants.
- The gemination mark, ALShaddah, is a diacritical mark used to indicate a doubled consonant.
- Words are derived from basic building blocks with triconsonantal roots. For example, the word (استهتارهم $\bar{a}stht\bar{a}rhm$) which means *"their irreverence"* has the following structure: the prefix (است $\bar{a}st$), the infix (ا $\bar{a}$), the root (هتر htr), and the suffix (هم hm).

In spite of this difficulty, researchers have made some recent progress exploring the morphology of the Arabic language. This paper continues the problem of identifying and exploiting relationships amongst Arabic letters for Arabic word root extraction begun in [1]. The algorithms in this paper are compared with those found in [1].

This paper introduces an answer to the following research question: "Can we identify and exploit relationships between Arabic letters for Arabic word root extraction?".

Section 2 introduces a brief review of the various available Arabic root extraction techniques. Section 3 discusses the proposed approach. Experimental results, evaluations and discussions are found in Section 4. Finally, conclusions and future work can be found in Section 5.

[1] Letters act as prefixes, infixes, or suffixes in the Arabic language.

2. Previous Work

Many techniques have been developed to process languages such as English [8, 11, 12] and French [13, 14]. In [2], a detailed analysis of the morphological structure of Arabic words yielded classification strategies for the Arabic language. The Arabic language can be categorized into a more classical language, as found in the Holy Quran or poetry, a standardized modern language, and regional dialects. Two main types of stemming include light stemming, where a few prefixes and/or suffixes are removed, and heavy stemming, where the root is extracted after removing prefixes and suffixes [3]. Stemming can be further divided into several approaches: dictionary or table lookup, combinatorial, rule-based (also referred to as linguistic strategy), and pattern-based. Typically, one or more of these stemming approaches are selected to create new algorithms. A rule-based strategy is a commonly applied stemming technique. This linguistic strategy simulates the same process of an expert linguist during his analysis of a given Arabic word. The most common stemmers in this field are described in [7], [15] and [10]. In [3], the authors test a rule-based stemmer called the Arabic Rule-based Light Stemmer (ARBLS) which uses a series of rules and relationships to derive root words. Their approach differs from ours in that the rules and relationships used to derive root words depend only on the vowel letters (ي ، و ، ا *ā , w , y*).

The ARBLS algorithm finds only the triliteral word roots, it is not extended to deal with any other word root types. The ARBLS algorithm utilizes an Arabic dictionary of root words to aid in the classification process. The reliance on a dictionary is problematic because classification errors within the dictionary propagate into newly stemmed words. Moreover, each extracted root is checked against all the roots stored in the dictionary, which means that the execution time of the ARBLS algorithm will be consumed by searching the dictionary.

Stemming can also be used to reduce the curse of dimensionality. For example in [5] the authors pre-process the document as an intermediate step when classifying a new document against a known corpus. For root extraction, the authors used the method of Al-Shalabi, Kanaan, and Al-Serhan, that is also mentioned in [9]. Al-Shalabi method categorizes all letters according to six integer weights, ranging from 0 to 5, as well as the rank of the letter which is determined by the position this letter holds in a word. The weight and rank are multiplied together, and the three letters with the smallest product constitute the root of the word. See [4] for more details on this interesting approach to stemming. [9] claimed that Al-Shalabi did not explain or clarify why or on what basis did it use such ranking or weighting. In our work, only two integer weights (0 and 1) have been used to categorize some letters. We've also extended the number of categories of letters from six to eight, yielding a finer partition of the Arabic letters. The position of the individual letters are, of course, important and, additionally, we chose to utilize morphological substitution to further enhance the stemming process which utilizes some of the morphological characteristics of the Arabic language. Some rules are proposed to allow predicting separately for those letters which have no weight if they belong to the root.

3. The Root Extraction Approach

Throughout this paper, $\wedge$ denotes logical "and" while $\vee$ denotes logical "or."

3.1. Overview of the First Stage of the Approach

The first stage of the new approach is the same as the one used in [1].

- Each word is represented in the form $W = (L_1, L_2, L_3, \ldots, L_n)$, where L_i represents a letter, and n is the length of the word.
- Let $W_{(segments)} = (S_1, S_2, S_3)$ be a partition of the word W into three approximately equal pieces with the following restrictions

  a) $\text{Length}(S_1) = \text{Length}(S_3) = \text{Round}\left(\frac{n}{3}\right)$.

  b) $\text{Length}(S_2) = n - 2 \cdot \text{Length}(S_1)$.
- The word root is represented in the form $R = (r_1, r_2, r_3, \ldots, r_n)$, where r_i is the i^{th}

letter's weight according to its use in the word root where

$$r_i = \begin{cases} 1, & \text{letter is present in the word root} \\ 0, & \text{letter is not used in the word root} \\ -1, & \text{letter is as yet undetermined.} \end{cases}$$

At the beginning of execution, $r_i = -1$ for $1 \leq i \leq n$, indicating no decision has been made yet about any letters found in the word W.

This approach also classifies Arabic word letters into eight sets or groups as shown in Table 1.

Example 3.1. *The word (سنكتب snktb) which means "we will write" and its root will be represented as follows:*

$n = 5$;
$W =$ (س ، ن ، ك ، ت ، ب b , t , k , n , s);
$\text{Length}(S_1) = \text{Length}(S_3) = \text{Round}\left(\frac{5}{3}\right) = 2$;
$\text{Length}(S_2) = 5 - 2 \cdot \text{Length}(S_1) = 1$;
$W_{(segments)} = (2, 1, 2)$;
$R = (0, 0, 1, 1, 1)$.

The first stage also proposed a definition for the permanent component (ال $\bar{a}l$) to distinguish between the definite article (ال $\bar{a}l$) and the permanent component (ال $\bar{a}l$).

Definition 3.1. *We do not consider the prefix (ال $\bar{a}l$) as a definite article, but rather a permanent component of a word, if at least one of the following cases is true. Let k be the position of (ال $\bar{a}l$) in the word.*

- Case one:

  $$L_{k+2} \in \{\textit{the sun\_ letters}\}^2 \wedge L_{k+3} \notin \{\textit{AL-Shaddah}\}$$

- Case two: $k = 2$ (one letter precedes ال $\bar{a}l$).

  $L_1 \notin$ {ب، ف، و، ك k ,w ,f ,b}
  $\vee$ ($L_1 \in$ {ب، ف، و، ك k ,w ,f ,b} $\wedge\, n < 6$).

- Case three: $k = 3$ (two letters precede ال $\bar{a}l$).

  – $L_1, L_2 \in$ {ب، ف، و، ك k ,w ,f ,b} $\Longrightarrow$
  ($L_1 = L_2$) $\vee$ ($L_1 \in$ {و w} $\wedge\, L_2 \in$ {ف f})
  $\vee$ ($L_1 \in$ {ب، ك k ,b}
  $\wedge\, L_2 \in$ {ب، ف، و، ك k ,w ,f ,b}).

  – L_1 or L_2 does not belong to
  {ب، ف، و، ك k ,w ,f ,b}

- Case four: $k > 3$ (more than two letters precede ال $\bar{a}l$).

From definition 3.1, the following rules are deduced. From case one, L_{k+2} is a root letter except when $L_{k+2} \in$ {ت t}. When working under case two, L_1 is considered a root letter. In the first condition of case three, L_2 is a root letter only when $L_1 = L_2$ or $L_2 \notin$ {و w}. In the second condition of case three, L_2 is considered a root letter unless $L_2 \in$ {و w}. Finally, in case four, the only prefix found in the holy Quran that negated this case is (أفب ʾafb). As a small example, consider (أفبالباطل ʾafbālbāṭl). In all cases above, L_{k+1} is considered a root letter. If (ال $\bar{a}l$) is not any of the cases mentioned above, then this approach considers it a definite article and applies the following rule:

("*Definite AL*") and $L_{k+2} \in$
{ب ،ف ، ك ، ن ، و ، ه ،ل ، ي ، س
s , y , l, h- , w , n , k , f, b}
$\Longrightarrow r_{k+2} = 1$.

3.2. The Supplementary Rules

Any letters not appearing in Table 1 will use the following supplementary rules to determine its weight. The idea behind using these rules to detect the word root is that some Arabic letters change their form (either as a root letter in some words or as a non-root letter in others) according not only to their positions in the word, but also according to the adjacent letters. An extensive linguistic analysis of patterns and affixes of the Arabic language were carried out to find these relationships. Eight rules will be proposed in the following subsections, four of them benefiting from the idea of morphological substitution (MUTATION).

[2] The sun_letters are: (ل ،ن ،ظ ،ط ،ض ،ص ،ش ،س ،ز ،ر ،ذ ،د ،ث ،ت t, $\underline{t}$, d, $\underline{d}$, r, z, s, $\check{s}$, $\d{s}$, $\d{d}$, $\d{t}$, $\d{z}$, n, l)

Group Names	The Letters	Location	Weight
G1	{ث، ج، ح، خ، د، ذ، ر، ز، ش، ص، ض، ط، ظ، ع، غ، ق}	anywhere in the word	1
G2	{ ه }	first letter of S_1	1
G3	G2 + {ف }	S_1	1
G4	G3 +{ب، ك، ن، ل ، أ ، ىء ، وء ، ء }	first letter of S_2	1
G5	G4 + {س، م}	S_2	1
G6	{ب، ف، س، ل، و، أ ، ىء ، وء}	last letter of S_3	1
G7	{ب، ف، س، ل، م، أ ، ىء ، وء ، ء}	S_3	1
G8	{ا}	S_1	0
	{ة}	the end of S_3	0

Table 1. Arabic word letter classifications according to the proposed algorithm.

3.2.1. Mutation rules

Mutation or morphological substitution can be defined as the changing or removing of one letter and replacing it with another. Our approach benefits from this idea by either finding some root letters or eliminating others. Four mutation rules are proposed.

1. $(L_i \in \{\text{ىء ، وءإ}\} \wedge L_i \in S_1) \Longrightarrow L_i = \text{أ}$.
   Next, apply the rule

   $(L_i \in \{\text{أ}\} \wedge L_{i-1} \in \{\text{ا}\} \wedge L_i \in S_1) \Longrightarrow r_i{=}1.$

   In this rule, change each one of {ىء ، وءإ} found in the first part of W with أ and then check the preceding letter. If it is ا then set أ to be a root letter. Examples include the words (اءتلَاف) and (إيتَاء).

2. $L_i \in \{\text{آ}\} \Longrightarrow (L_i = \text{أ} \wedge r_i = 1)$.
   Whenever آ is found in the word W, replace it with أ and set it as a root letter. Examples of words using this rule include (مآثر) and (الآيَات).

3. $(L_i \in \{\text{ص ، ض}\} \wedge L_{i+1} \in \{\text{ط}\})$
   $\Longrightarrow (L_{i+1} = \text{ت} \wedge r_{i+1} = 0)$.
   The letter ط is not considered a root letter when it appears either after the letters ص or ض. In these cases, it should be replaced with ت. Two examples using this rule are the words (يصطرخون) and (اضطرَاب).

4. $(L_i \in \{\text{ز}\} \wedge L_{i+1} \in \{\text{د}\})$
   $\Longrightarrow (L_{i+1} = \text{ت} \wedge r_{i+1} = 0)$.
   The letter د is not considered a root letter when it appears after the letter ز. When this occurs, it should be replaced with ت. Examples utilizing this rule include the words (ازديَاد) and (ازدرَى).

3.2.2. Other rules

The following rules are applied to those letters which are not mentioned in Table 1 and do not follow the above mutation rules.

1. $L_i = L_{i+1} \Longrightarrow (r_i = 0 \wedge r_{i+1} = 1)$.
   If any two successive letters are similar, then consider the first one an augmented letter and the second one to be a root letter.

2. $(L_i \in \{\text{ن ، ل ، م ، س}\} \wedge L_i \in (S_1 \cup S_2) \wedge r_j = 1 \wedge j < i) \Longrightarrow r_i = 1$.
   If one of the letters that form the word نلمس is found in S_1 or S_2, and one of the letters before is a root letter, then consider the letter a root letter.

3. $(L_i \in \{\text{أ ، ل ، و ، ي}\} \wedge L_i \in S_1 \wedge L_{i-1} \in \{\text{م}\}) \Longrightarrow r_i = 1$.
   In the first segment of the word, if one of the letters that form the word ألوي is preceded

by the letter م, then consider that letter a root letter.

4. $(L_n \in \{$ة$\} \wedge ((L_{n-1} \in \{$ي$\} \wedge n \leq 3) \vee (L_{n-1} \notin \{$ي$\}))) \Longrightarrow r_{n-1} = 1$.
   For all words ending with the letter ة, this rule considers the letter preceding ة a root letter. An exception to this rule is the letter ي, which is considered a root letter only when the word is three letters in length or less.

3.2.3. Calculating the distance between the word root letters

If during program execution two root-letters have been detected, then we may enhance the process of finding the third root letter by calculating root-distance which measures the number of letters between the two root letters being found. If the value of the root-distance is two or more, then the current two root-letters are the first and the last ones, and the third one is between them. Otherwise, the root-distance routine will start searching for the third root letter outside the two root-letters, either before the first one or after the second one, using the assumption that only one or two letters can be added between any two root letters [16].

3.3. Root-extraction Algorithm

Algorithm 1 summarizes the paper's approach for root word extraction.

3.4. Examples Demonstrating the Proposed Technique

This section introduces three examples to demonstrate the proposed technique explained in this paper.

Example 3.2. *The word (وبَالوَالدين wbālwāldyn) which means "and with the parents"*

1. *At the initial stage the following parameters will be set:*

n=10;
W=(و ، ب ، ا ، ل ، و ، ا ، ل ، د ، ي ، ن);
$R=(-1, -1, -1, -1, -1, -1, -1, -1, -1, -1)$.

Algorithm 1 used for root word extraction

```
 1: $CountON \leftarrow CountOFF \leftarrow 0$
 2: $Flag \leftarrow true$
 3: $r_1\_Flag \leftarrow false$
 4: $r_n\_Flag \leftarrow false$
 5: $W \leftarrow inputword$
 6: $N \leftarrow wordlength$
 7: $R \leftarrow \{-1\}$
 8: Apply "identifyAL" routine
 9: $n \leftarrow N$
10: $W(segments) \leftarrow S_1, S_2, S_3$
11: Mutate W
12: for each letter $L_i \in G1$ do
13:     $r_i \leftarrow 1$
14:     Increment (CountON)
15: end for
16: for each letter $L_i \in G8$ do
17:     $r_i \leftarrow 0$.
18:     Increment (CountOFF)
19: end for
20: $start\_index \leftarrow 1$
21: $last\_index \leftarrow n$
22: if $CountON \geq 3$ then
23:     go to 54
24: end if
25: if $CountON = 2$ AND $Flag$ then
26:     Apply "root_Distance" routine
27: end if
28: if $r_{start\_index} = 0$ OR $r_{last\_index} = 0$ then
29:     if $r_{start\_index} = 0$ then
30:         $r_1\_Flag \leftarrow true$
31:         while $r_{start\_index} = 0$ do
32:             Increment (start_index)
33:         end while
34:     end if
35:     if $r_{last\_index} = 0$ then
36:         $r_n\_Flag \leftarrow true$
37:         while $r_{last\_index} = 0$ do
38:             Decrement (last_index)
39:         end while
40:     end if
41: end if
42: $n \leftarrow word\_length$
43: $W(segments) \leftarrow S_1, S_2, S_3$
44: for each letter $L_i \in G2, \ldots, G7$
45:     $r_i \leftarrow 1$
46:     Increment (CountON).
47: end for
48: $Changes \leftarrow false$
49: while Applying the rules $1, \ldots, 4$ no Changes do
50:     if Changes then
51:         go to 22
52:     end if
53: end while
54: Stop and output the root.
```

2. *Apply the "definite Al routine" and check for all occurrences of the component ال in the word. The first occurrence is found after the letters و ب.*
   The algorithm determines that ال is a definite article, since none of the cases is applicable. The algorithm removes ال along with all the letters preceding ال. Consider the letter و that followed ال a root letter. Since و ل is one of the letters, form the word سيلهون.
3. *Check again for ال. If ال is found for the second time, then ال in this case is a permanent component, and the letter ل is considered a root letter.*
4. *Update each of n; W; R and calculate* $W_{(segments)}$

   $n = 6$;
   $W =$ (و ، ا ، ل ، د ، ي ، ن);
   $Length(S_1) = Length(S_3) = Round\left(\frac{6}{3}\right) = 2$;
   $Length(S_2) = 6 - 2 \cdot Length(S_1) = 2$;
   $W_{(segments)} = (2, 2, 2)$;
   $R = (1, -1, 1, -1, -1, -1)$.
5. *Check W for all letters in Table 1, and update R.*
   $R = (1, -1, 1, 1, -1, -1)$.
   $R = (1, 0, 1, 1, 0, 0)$.
6. *Stop and produce the result.*

Example 3.3. *The word (ممبالغ mbālġ) which means "exaggerator"*

1. *At the initial stage the following parameters will be set:*

   $n = 5$;
   $W =$ (م ، ب ، ا ، ل ، غ);
   $R = (-1, -1, -1, -1, -1)$.
2. *Apply the "definite Al routine" and check for all occurrences of the component ال in the word. The only occurrence is found after the letters مب.*
   The algorithm decides that ال is a permanent component, since case three is applied to it. In this case, both the letter ب and ل are considered root letters.
3. *Update R and calculate* $W_{(segments)}$.

   $n = 5$;
   $W =$ (م ، ب ، ا ، ل ، غ);
   $Length(S_1) = Length(S_3) = Round\left(\frac{5}{3}\right) = 2$;
   $Length(S_2) = 5 - 2 \cdot Length(S_1) = 1$;
   $W_{(segments)} = (2, 1, 2)$;
   $R = (-1, 1, -1, 1, -1)$.
4. *Check W for all letters in Table 1, and update R.*
   $R = (-1, 1, -1, 1, 1)$.
   $R = (0, 1, 0, 1, 1)$.
5. *Stop and produce the result.*

Example 3.4. *The word (عسير ʿsyr) which means "very hard"*

1. *At the initial stage the following parameters will be set:*
   $n = 4$;
   $W =$ (ع ، س ، ي ، ر);
   $R = (-1, -1, -1, -1)$.
2. *Calculate* $W_{(segments)}$, *since the word does not contain ال.*

   $Length(S_1) = Length(S_3) = Round\left(\frac{4}{3}\right) = 1$;
   $Length(S_2) = 4 - 2 \cdot Length(S_1) = 2$;
   $W_{(segments)} = (1, 2, 1)$;
3. *Check W for all letters in Table 1, and update R.*
   $R = (1, -1, -1, 1)$.
4. *Check the rules in Section 3.2.2. Rule 2 is applicable.*
   Update R
   $R = (1, 1, -1, 1)$.
   $R = (1, 1, 0, 1)$.
5. *Stop and produce the result.*

4. Evaluation and Discussion

The Holy Quran words are used for evaluation, a preprocessing module which does the following is applied on a file consisting of all the 114 chapters of the Holy Quran[3]:

- Remove from the texts all the numerals and symbols found in the file, punctuation marks, assimilation marks, short vowels, function

[3] Information about the Holy Quaran can be taken from: `http://en.wikipedia.org/wiki/Quran`.

words, and diacritics except the gemination mark.

- Split the text into tokens.
- Exclude the stop words.
- Remove duplicate words.
- Save the remaining words in a file.

The file produced consists of 14,067 unique words. The length of these words ranges from 2 to 13 letters.

After executing the algorithm, the generated root letters of each word are compared to the ones stored in the roots file[4], which contains the positions of the root letters of each word in the words file in addition to the root letters that are missing from the words (if any). Table 2 illustrates the contents of the roots file.

If any match is found (either a whole match or sub match), then the root analysis is considered correct. On the other hand, if at least one letter produced by analyzing the tested word is wrong, the root analysis is considered incorrect.

The experimental system is evaluated using the following three phases:

- A summary of the final results obtained from [1].
- Evaluating the whole system with the rules being introduced in this paper.
- Discussing and analyzing the current version of the approach.

4.1. PHASE 1: A Summary of the Results in [1]

This section contains a brief summary of the results obtained after evaluating the system in [1]. The experimental system analysed 13,856 words and failed on 211. The generated roots letters of each word were compared to the ones stored into the roots file after taking into account that the system was in its first stage.

By using the evaluation method mentioned above, 13,193 results were considered correct, that is about 93.79% and only 663 of the results, that is 4.71% of the experimented words, were incorrect. Table 3 and Figure 1 show a summary of the results found in [1]. In Figure 1, the words with a correct root analysis are divided into the number of letters which are correctly reported. Thus, in Figure 1a, of the 13,129 words considered correct, the percentage of words with trilateral roots with one (14%), two (41.6%), or three (44.4%) root letters successfully identified is given. Similarly, Figure 1b is a visual representation of the number of correct letters discovered in the 89 correct matches of words with quadrilateral roots. In this case, the percentage of two (18%), three (57.3%), and four (24.7%) correctly identified letters is shown.

From Table 3, we see that:

- 5,360 words out of 14,067 were totally correct.
- 5,870 words out of 14,067 contained only one missing root.
- Only 1,963 words out of 14,067 contained two missing roots.

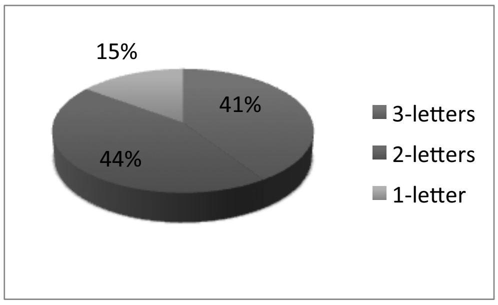
Word	Contents of the Root's file	Description
وَاقعدوَا *wāqʿdwā*	3, 4, 5	The third, fourth and the fifth letters form the root (قعد *qʿd*)
وَاعتصموَا *wāʿtṣmwā*	3, 5, 6	The third, fifth and the sixth letters form the root (عصم *ʿṣm*)
وقنَا *wqnā*	1, 2, { ى *y*}	The first two letters and the letter (ى *y*) form the root (وقي *wqy*)
وصدّ *wṣdd*	2, 3, 4	The diacritic (*AL-Shaddah*) is part of the root (صدد *ṣd d*)

Table 2. Description of the roots file format.

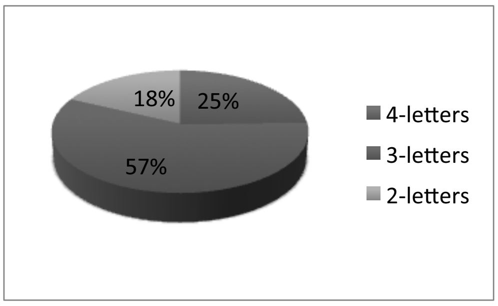
[4] We selected the root letters of each word by using the reliable Arabic dictionary Muktar Al_Sahah.

	Total words	Total correct results	Number of correct letters discovered		
Trilateral roots	13,974	13,103 (93.7%)	3-letters: 5,337 (40.7%)	2-letters & one is missing: 5,819 (44.4%)	1-letter & two are missing: 1,947 (14.9%)
Quadrilateral roots	92	89 (96.7%)	4-letters: 22 (24.7%)	3-letters & one is missing: 51 (57.3%)	2-letters & two are missing: 16 (18%)
Pentliteral roots	1	1	5-letters: 1		
Total	14,067 (100%)	13,193 (93.79%)	5360 38.1%	5870 41.73%	1963 13.96%

Table 3. The experimental results of phase 1.

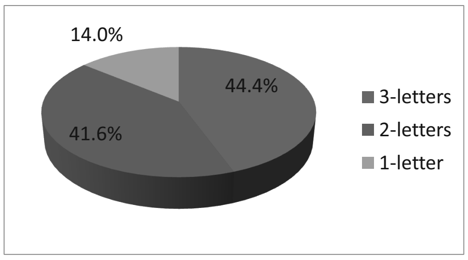


(a)

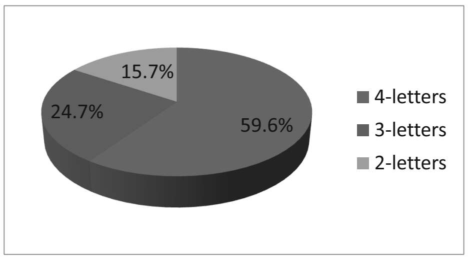


(b)

Figure 1. The experimental results of phase 1. (a) Percentage of root letters successfully found in words with Trilateral roots. (b) Percentage of root letters successfully found in words with Quadrilateral roots.



(a)



(b)

Figure 2. The experimental results of phase 2. (a) Percentage of root letters successfully found in words with Trilateral roots. (b) Percentage of root letters successfully found in words with Quadrilateral roots.

4.2. PHASE 2: Evaluating the Current Version of the Approach

In this phase, the experimental system analysed 13,876 words and failed on 191. By using the evaluation method mentioned above, 13,219 results were considered correct. This is an approximately 93.97% accuracy. Only 657 of the results, approximately 4.67% of the experimented words, were incorrect. Table 4 summarizes the results of this updated algorithm. Figure 2 again splits the percentage of the root analysis which is considered correct into the percentage of correctly identified letters. Thus, we see in Figure 2a, for trilateral roots, the percentage correctly identified (44.4%) is a small increase (3.7%) of the number trilateral roots completely identified from the previous algorithm. The number of roots with two letters correctly identified (41.6%)

	Total words	Total correct results	Number of correct letters discovered		
Trilateral roots	13,974	13,129 (94%)	3-letters: 5,835 (44.4%)	2-letters & one is missing: 5,467 (41.6%)	1-letter & two are missing: 1,827 (14%)
Quadrilateral roots	92	89 (96.7%)	4-letters: 53 (59.6%)	3-letters & one is missing: 22 (24.7%)	2-letters & two are missing: 14 (15.7%)
Pentalateral roots	1	1	5-letters: 1		
Total	14,067 (100%)	13,219 (94%)	5889 41.86%	5489 39.02%	1841 13.09%

Table 4. The experimental results of phase 2.

went down slightly, a small (2.8%) decrease, and the percentage of roots with a single letter correctly identified (14%) also decreased slightly (0.9%) when compared to the previous work. Figure 2b graphically summarizes the successes for quadrilateral roots. The percentage of complete identification of a root (59.6%) represents a large increase (34.9%) over the previous work. The percentage of roots with three letters correctly identified (39.02%) shows a small decrease (2.71%), and the percentage of two letter identifications (13.09%) is a quite small decrease (0.87%) when compared to the previous algorithm.

From Table 4, we see that:

- 5,889 words out of 14,067 were totally correct.
- 5,489 words out of 14,067 with exactly one root missing.
- Only 1,841 words out of 14,067 contained two missing roots.

4.3. PHASE 3: Discussing and Analyzing the Current Version of the Approach

Analysis of the final statistics in Table 5 and Figure 3, allow the following to be concluded:

- Slight improvement is achieved when only four rules are applied to the words. This result is very encouraging and it may yield a promising root extraction algorithm if more rules are added to the approach. Since 13,493 (95.9%) of the corpus contain long vowels, rules to identify relationships between the long vowels and the consonants should yield significant improvements. Table 6 shows some of the tested words after applying phase 1 and phase 2.

		Phase 1		Phase 2	
		Total Number	Percentage	Total Number	Percentage
Succeeded Results	Correct Roots	5,360	38.10%	5,889	41.86%
	Roots with 1 letter missing	5,870	41.73%	5,489	39.02%
	Roots with 2 letter missing	1,963	13.96%	1,841	13.09%
Failed Results	Wrong results	663	4.71%	657	4.67%
	Missing roots	211	1.5%	191	1.36%
Total		14,067	100%	14,067	100%

Table 5. Final results of phase 1 and phase 2.

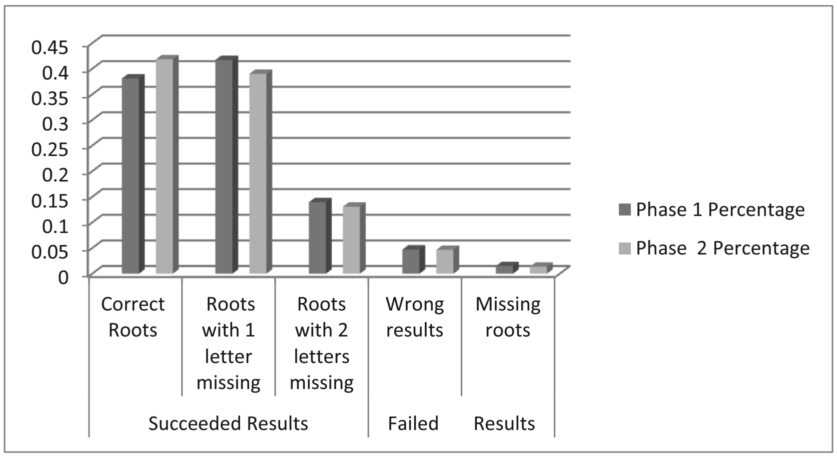


Figure 3. The experimental results of phases 1 and 2.

Word	Phase1	Phase2
الأَخسرون *ālʾaḫsrwn*	خر *ḫr*	خسر *ḫsr*
الأَيكة *ālʾaykh*		ك *k*
الآلهة *ālʾālhh*	ل *l*	أَله *ʾalh*
الآمرون *ālʾāmrwn*	ر *r*	أَمر *ʾamr*
الآية *ālʾāyh*		أَي *ʾay*
الحلقوم *ālḥlqwm*	حق *ḥq*	حلق *ḥlq*
الخنزير *ālḫnzyr*	خزر *ḫzr*	خنزر *ḫnzr*
السّمَاوَات *ālssmāwāt*	س *s*	سم *sm*
الشّمَال *ālššmāl*	شل *šl*	شمل *šml*
الملعونة *ālmlʿwnh*	الع *lʿ*	العن *lʿn*
الميمنة *ālmymnh*	م *m*	من *mn*
وَالرَّاسخون *wālrrāsḫwn*	رخ *rḫ*	رسخ *rsḫ*
وَالسّميع *wālssmyʿ*	سع *sʿ*	سمع *smʿ*
العنكبوت *ālʿnkbwt*	عكب *ʿkb*	عنكب *ʿnkb*

Table 6. Some of tested words of phase 1 and 2.

- Since the incorrect results obtained from phase 1 (4.71%) and phase 2 (4.67%) are considered poor, further analysis of the words used in the experiment was conducted.

  – Rule 2 was successful in classifying $L_i \in$ {ن، ل، م، س} as root letters when $L_i \in S_1 \cup S_2$ unless $L_i \in$ {ن} appeared as the last letter of S_2. Consider, for example, the word (اخترنَاهم) which means *"we chose them"*.

  – *G*5 in Table 1 classified the letters {ك، ن ، ه ، م} as root letters whenever they appeared in S_2. This behavior is correct except when one of these letters appears as the last letter of S_2. The words (أتّبعكمَا) which means *"to follow you"*, (اخترنَاهم) which means *"we chose them"*, the word (ادعهنّ) which means *"invite them"* and the word (دعوتموهم) which means *"you invited them"* are all examples of this behavior.

  – *G*7 in Table 1 classified the letter م as a root letter in S_3. Unfortunately, some of the results obtained contradict this fact. For example, the word (أتّبعكمَا) which means *"to follow you"*.

  – The system's failures are mostly found in some of the following ***Critical points*** of a word:
  - ∗ The first letter of S_2 (has been dealt with correctly in Table 1).
  - ∗ The last letter of S_2.
  - ∗ The last letter of S_3 (has been dealt with correctly in Table 1).

- To eliminate failures found in this algorithm, modifications are applied in both Table 1 and Rule 2, taking into account the ***Critical points*** of the word. Table 7 and the following Rule 2-A and Rule 2-B reflect this modification.

Group name	The letters	Location	Weight
G1	{ث، ج، ح، خ، د، ذ، ر، ز، ش، ص، ض، ط، ظ، ع ، غ، ق}	Anywhere in the word	1
G2	{ه}	First letter of S_1	1
G3	G2 + {ف}	S_1	1
G4	G3 +{ب، ك، ن، ل ، أ ، ىء ، وء ، ء}	First letter of S_2	1
G5	G4 + {س، م}	S_2	1
G6	**G5 –** {ك، ن ، ه ، م}	**Last letter of S_2**	**1**
G7	{ب، ف، س، ل، و، أ ، ىء ، وء}	Last letter of S_3	1
G8	{ب، ف، س، ل، أ ، ىء ، وء ، ء}	S_3	**1**
G9	{ا}	S_1	0
	{ة}	the end of S_3	0

Table 7. Arabic word classification according to the proposed algorithm.

Rule2-A: $L_i \in$ {ل ، م ، س} $\wedge L_i \in S_1 \cup S_2 \wedge r_j = 1 \wedge j < i \Longrightarrow r_i = 1$.

Rule2-B: $L_i \in$ {ن} $\wedge\, i < (Length(S_1) + Length(S_2) + k - 1) \wedge r_j = 1 \wedge j < i \Longrightarrow r_i = 1$, where k is the start index of W.

5. Conclusion and Future Work

A new technique for finding the Arabic word root without relying on a database of word roots, a list of patterns, or even a list of prefixes and suffixes of Arabic words has been introduced. This technique relies only on the use of some rules which benefit from the relationships among letters of a word. The results of the evaluation of this stage show a promising root extraction approach. The next stage of this approach will cover relationships to handle the long vowels of the Arabic alphabet. Since about 96% of the current corpus contain long vowels, rules which can identify relationships among the long vowels and/or between the long vowels and the consonants should yield significant improvements.

References

[1] F. Abu Hawas Fatma, Exploit relations between the word letters and their placement in the word for Arabic root extraction. *Computer Science Journal*, **14**(2) (2013), 327–341.

[2] I. A. AI-Sughaiyer, I. A. Al-Kharashi, Arabic morphological analysis techniques: A comprehensive survey. *Journal of the American Society for Information Science and Technology*, **55**(3) (2004), 189–213.

[3] A. Al-Omari, B. Abuata, M. Al-Kabi, Building and Benchmarking New Heavy/Light Arabic Stemmer. *The 4th International conference on Information and Communication systems (ICICS'13)*, (2013).

[4] R. Al-Shalabi, G. Kanaan, H. Al-Serhan, New Approach for Extracting Arabic Roots. *Proceedings of the International Arab Conference on Information Technology (ACIT'20003)*, (2003), pp. 42–59.

[5] Rehab Duwairi, Arabic Text Categorization. *The International Arab Journal of Information Technology*, **4**(2) (2007), 125–131.

[6] Ethnologue, `http://www.ethnologue.com/statistics/size`, Online accessed 16 January 2014.

[7] S. Khoja, R. Garside, Stemming Arabic text. Technical report, Computing Department, Lancaster University, 1999.

[8] R. KROVETZ, Viewing morphology as an inference process, Conference on Research and Development in Information Retrieval. In *Proceedings of the Sixteenth Annual International ACM SIGIR*, (1993), pp. 191–202.

[9] M. Y. AL-NASHASHIBI, A. A. YAGHI, Stemming Techniques for Arabic Words: A Comparative Study. *2nd International Conference on Computer Technology and Development(ICCTD 2010)*, (2010), pp. 270–276.

[10] M. MOMANI, J. FARAJ, A novel algorithm to extract tri-literal Arabic roots. In *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, (2007), pp. 309–315.

[11] C. D. PAICE, Another stemmer. *SIGIR Forum*, **24**(3) (1990), 56–61.

[12] M. F. PORTER, An algorithm for suffix stripping. *Program*, **14**(3) (1980), 130–137.

[13] J. SAVOY, Stemming of French words based on grammatical categories. *Journal of the American Society for Information Science*, **44**(1) (1993), 1–9.

[14] J. SAVOY, A Stemming Procedure and Stop word List for General French Corpora. *Journal of the American Society for Information Science*, **50**(10) (1999), 944–952.

[15] R. AL SHALABI, Pattern-based stemmer for finding Arabic roots. *Information Technology Journal*, **4**(1) (2005), 38–43.

[16] A. SOUDI, V. D. B. ANTAL, G. NEUMANN, Arabic morphological generation and its impact on the quality of machine translation to Arabic. *Arabic Computational Morphology; Text, Speech and Language Technology*, **38** (2007), 287–302. Springer.

Received: June, 2013
Revised: January, 2014
Accepted: February, 2014

Contact addresses:

Fatma Abu Hawas
Department of Computer Science
Yarmouk University
Jordan
e-mail: `fatmih@yu.edu.jo`

Keith E. Emmert
Tarleton State University
Stephenville, Texas
USA
e-mail: `emmert@tarleton.edu`

FATMA ABU HAWAS is currently an instructor in the Department of Computer Science at Yarmouk University, Jordan. She has received her M.Sc degree in Computer Sciences from Yarmouk University, Jordan in 2003. Her current research interests are artificial intelligence and natural language processing.

KEITH E. EMMERT received his PhD in Applied Mathematics from Texas Tech University in 2004. Dr Emmert is currently an Associate Professor of Mathematics at Tarleton State University located in Stephenville, Texas, USA. His current research interests include biomathematics, parallel algorithms, statistics, astronomy, pollution modeling, and data mining.