International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 07 – Issue 06, November 2018

Towards Improving Rule-Based Arabic Root Extraction Algorithm for Non-Vocalized Text

Nisrean Thalji
Department of Computer Engineering, School of Computer and Communication Engineering
University Malaysia
Perlis, Malaysia
Email: nnthalji1980 [AT] gmail.com

Zyad Thalji
Department of Management Information System
Imam Abdulrahman Bin Faisal University
Kingdom of Saudi Arabia

Sohair Al-Hakeem
Department of Computer Science
Ajloun National University
Ajloun, Jordan

Nik Adilah Hanin
Department of Computer Engineering, School of Computer and Communication Engineering
University Malaysia
Perlis, Malaysia

Walid Bani Hani
Department of Computer Science
Higher Colleges of Technology
Ras Al Khaimah, United Arab Emirates.

Abstract—**Rooting algorithms are used to remove affixes from different words, and extract the root from which the inputted word is derived. Rooting process helps to standardize terms referring to the same concept. These algorithms are widely used in Arabic language applications, such as information retrieval systems, indexes, text mining, text classifiers, data compression, spelling checkers, text summarization, question answering systems, machine translation, part of speech tagging systems, stemmers, and morphological analyzer ...etc. Khoja's algorithm is a standard Arabic root extraction algorithm, which has a number of flaws. The proposed algorithm extends Khoja's algorithm and resolves most of its flaws. The testing process was conducted on Thalji's corpus, which was mainly built to test and compare Arabic roots extraction algorithms. This corpus contains 720,000 word-root pairs from 12,000 roots. The performance of the proposed algorithm is then compared with Khoja's algorithm, the proposed algorithm obtained higher accuracy than Khoja's algorithm. The result shows that Khoja algorithm achieved 63%, and the presented algorithm achieved 92% accuracy of root extraction.**

Keywords-component; Root Extraction, stem, rules, pattern, prefix, suffix, infix. (key words)

I. Introduction

Arabic texts are mainly two types, the first one is known as Classical Arabic e.g. the Qur'an text. The second type called Modern Standard Arabic (MSA) and it is the form that is used in all Arabic-speaking countries' publications, media, and books [1]. In addition, Modern Standard Arabic is classified into three main types, the first type is the fully vocalized like the elementary textbooks. The second type is the partially vocalized like the newspapers, and the third type is non-vocalized text, it depends on the writer who writes the vowel or not. Vowels are used in Arabic to ensure the exact meaning of the words. If the word is non-vocalized in many cases it is an ambiguous word, and then we need to read the sentence and sometimes the whole text to understand the exact meaning, these vowels are written above or under the letter.

Root extraction is a very important initial step in many applications such as information retrieval systems, indexes, text mining, text classifiers, data compression, spelling checkers, text summarization, question answering systems, and machine translation.

The Arabic dialect contrasts from the Indo-European dialects morphologically, semantically, and grammatically. Building an Arabic stemmer is more complicated than building stemmer in any other European language such as English. English language stemmers are only concern with the removal of prefixes and suffixes [2]. In the contrast, Arabic stemmers have to deal with infix also.

Affixes in Arabic are prefixes, suffixes, and infixes. Prefixes attach at the beginning of the words, where suffixes attach at the end, and infixes attach in the middle of the words [3]. For example, with the Arabic word "كبيوتكم", which means (like your houses), "ك" is the prefix, which is a connected preposition, "كم" is the suffix, which is the subject, and "و" is the infix. The root is "بيت", where in English the preposition and subject are written separately. So, for word such as (houses) there is no prefix, no infix, (es) is the suffix, and the root is house.

In Arabic, words are made from roots and patterns. Patterns are non-constant letters, which can be interceded on as templates, in some cases as prefixes, suffixes, and in other cases as infixes. Patterns can be added to the root of the word or can be found within the roots of the word following well-defined models [4]. The grammatical system of the Arabic language is based on a root-and-pattern structure and considered as a root-based language [5]. Many words have the same pattern, if the word and the pattern are known, it is easy to extract the root, for example, the words “واستبدلته, واستجبرته, واستجهرته” have the same pattern “واستفعلته”, and the roots are “بدل, جبر, جهر” respectively.

There are two approaches that deal with the base form of the word, Light stemming and root extraction.

Light stemming for Arabic words is the process of removing common affixes (suffixes and prefixes) from the word and minimizing it to its stem form but not the root form [6], one example of recent Arabic light stemmer for information retrieval system, is Al-Lahham, Matarneh and Hassan Arabic light stemmer [7]. For example, Arabic light stemmer removes the prefix “ال” and the suffix “ون” from the word “الـعـامـلـون” (workers), and do not deal with infix “ا,” so the result will be “عـامـل” (worker). Whereas, the root of the word “العـامـلون” is “عـمـل” (work). In English, light stemming process removes common affixes (suffixes and prefixes) from the word [8]. It minimizes the word to stem form but not the root form, for example, the word (workers), English light stemmer removes the suffix (s), and no prefix is founded, so it returns the word (worker) as stem. Many light stemmers have been conducted like the works in [9], [10], [11], [12], [13], [14], [15], [16], [17].

Root extraction is reducing the word to its root form instead of a stem form; a root is the part of the word-form that the most basic meaning of any word is contained after removing all the affixes [18]. For example, Arabic root extraction removes the prefix “ال” and the suffix “ون” from the word “العاملون” (workers) and infix “ا” so the result will be “عـمـل”. Many Arabic root extraction algorithms have been conducted like the works in [19], [20], [21], [22], [23] and [24]. Khoja and Garside algorithm [25] is the very popular Arabic root extraction algorithm. The algorithm starts by replacing initial “آ, أ, إ” with “ا”. Removes stop words, punctuation, non-letters, diacritics, finite articles from the beginning of the word “ال”, the letter “و” from the beginning of the word, “ة” from the end of the word, and remove prefixes and suffixes. Then compares the resulting word to patterns stored in the dictionary. If the resulting root is meaningless, the original word is returned without changes. After that, replaces weak letters “ا, و ,ي” with “و “and replaces all occurrences of Hamza”ء ,ئ ,ؤ” with”أ”. Finally, two letter roots are checked to see if they should contain a double character. If so, the character is added to the root. Khoja and Garside algorithm reported 96% accuracy of their stemmer using newspaper text.

In this study, a new rule-based Arabic root extraction algorithm (AREA) is proposed. This algorithm is for non-vocalized Arabic texts, which mean it may return more than one root for ambiguous words. The proposed algorithm shows the ability to outperform Khoja’s algorithm in both reliability and performance. Our proposed algorithm uses in both information retrieval systems (IRS) and Natural Language Processing (NLP) applications in an efficient effective way, it benefited from previous algorithms, merge the strength of it and extend it.

II. RESEARCH METHOD

The new root extraction algorithm proposes to find all possible roots for the nonvocalized Arabic words. The root is the base form of the word that gives the main meaning of the word, whereas khoja’s algorithm gives just one root. In this section our methodology is described in details.

A. Normalization

Normalization is the process that leads to remove the unwanted letters, punctuations, and non-letters, or unify some other letters in the dictionaries. The main steps of the proposed text normalization are the following:

1. Remove kasheeda symbol ("-").
2. Remove punctuations.
3. Remove diacritics.
4. Remove non-letters.
5. Replace Hamza’s forms “ء, آ ,ؤ, إ, ئ” with “أ ”.
6. Duplicate any letter that has the Shaddah: “ّ" symbol.

B. Extracting the Constant Letters in the Word

The proposed algorithm finds all possible roots of the word without removing prefixes and suffixes. It classifies the twenty-eight Arabic letters { ا ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي} in to two parts. The first part is the constant letters, which are sixteen letters { ث ج ح خ د ذ ر ز ش ص ض ط ظ ع غ ق}. Constant letters are surly part of root’s letters. So, if it is founded in the word, then it is simplifying the root extraction process. The second part is Non-constant letters, which are twelve letters {ا ب ت س ف ك ل م ن ه و ي}. The proposed algorithm starts by extracting the constant letters from the word and then converts Non-constant letters to constant letters. The starting process of the proposed algorithm differs from Khoja’s algorithm because it does not start by removing prefixes and suffixes from the derivation words.

Particularly, removing prefixes and suffixes from the words leads to omitting many letters from the root, which leads also to wrong results. Khoja’s algorithm is removing the prefixes and suffixes from the words depending on expectation processes. This means do not sure exactly that prefixes and suffixes are affixes or not; for instance: with the word “استماع”, Khoja algorithm removes the prefix “است” from the word because it depends on the expectation processes that the prefix "است" is founded in its prefix's lists. So, it removes the prefix "است" directly. Whereas the proposed algorithm finds out the constant letters in the word; if the number of the constant letters is more than one letter, they will be considered as one of the expected roots. For example, for the input word "التقارير", the constant letters are "ر ,ر ,ق", then "قرر" is one of the possible roots for the word.

C. Converting Non-Constant Letters to Constant Letters

Constant letters are surly part of root's letters, whereas Non-constant letters are not always part of root's letters. A certain set of rules are applied to some Non-constant letters in order to convert these letters to constant letters. These rules detect if Non-constant letters are surly part of root's letters or not.

1) Rules for Letter "ب".

The letter "ب" is a Non-constant if it is a preposition letter, preposition letters attached just at the begging of the words. So, the algorithm converts "ب" letter to constant letters if it detects that it is not preposition letter by applying the following algorithm:

```
def B_rules(term, index, index_of_first_consonant,
index_of_last_consonent):
    is_consonant = False
    if index > index_of_first_consonant :
        is_consonant = True
    elif index >= (len(term) / 2):
        is_consonant = True
    elif (index >2) :
        is_consonant = True
    elif (index >= 2) and  (term[index-2]) == 'ا' and
(term[index-1] )== 'ل' and (index < (len(term) / 2)):
    is_consonant = True
    elif (index >= 1) and ( (term[index - 1] == 'ب') or
      (term[index - 1] == 'ا')  or (term[index - 1] == 'أ')
or (term[index - 1] == 'ن') or (term[index - 1] == 'ت')
or (term[index - 1] == 'س')
or (term[index - 1] == 'م') or (term[index - 1] == 'ه')
or (term[index - 1] == 'ي')):
        is_consonant = True
    return is_consonant
```

For example, the algorithm converts the letter "ب" to constant letter if it comes after constant letter, like the word "جواب" (answer); the letter "ب" comes after the constant letter "ج".

2) Rules for Letter "ف".

The letter "ف" is a Non-constant if it is a preposition letter. So, the algorithm converts "ف" letter to constant letters if it detects that it is not preposition letter by applying the following algorithm:

3) Rules for Letter "س".

The letter "س" is a Non-constant if it is a prefix letter. So, the algorithm converts "س" letter to constant letters if it detects that it is not a prefix letter by applying the following rules:

```
def S_rules(term, index, index_of_first_consonant,
index_of_last_consonant):
    is_consonant = False
    if index > index_of_first_consonant:
        is_consonant = True
    elif index >= (len(term) / 2):
        is_consonant = True
    elif len(term)>=3 and
      len(term)>=index+3 and   term[index + 1] =='ا'
         and term[index + 2] == 'ل':
      is_consonant = True
    elif (index >= 1) and ((term[index - 1] == 'ل')
      or (term[index - 1] == 'ب')or
          (term[index - 1] == 'س') or
           (term[index - 1] == 'ه') or
             (term[index - 1] == 'ك')) :
        is_consonant = True
    elif ((len(term) > index + 1) and
     ((term[index + 1] != 'ا') and
       (term[index + 1] != 'ن') and
        (term[index + 1] != 'أ')and
         (term[index + 1] != 'ي') and
          ( term[index + 1] != 'ت'))):
        is_consonant = True
    return is_consonant
```

4) Rules for Letter “ل”.

The letter “ل” is a Non-constant if it is a prefix letter. So, the algorithm converts “ل” letter to constant letters if it detects that it is not a prefix letter by applying the following rules:

```
Def
L_rules(term,index,index_of_first_consonant,index_of
_last_consonant):
    is_consonant = False
    if index > index_of_first_consonant :
        is_consonant = True
    elif index > (len(term) / 2):
        is_consonant = True
    elif (index >= 2) and  (term[index-2]) == 'ا' and
(term[index-1] )=='ل' and (index < (len(term) / 2)-1):
        is_consonant = True
    elif (index>= 1)and( (term[index - 1] == 'ن' )or
(term[index - 1] == 'ت' ) or  (term[index - 1] == 'م' )or
(term[index - 1] == 'ه')    or (term[ index - 1] == 'ي')or
(term[index - 1] == 'س')   or (term[index - 1] == 'ك')):
        is_consonant = True
    return is_consonant
```

5) Rules for Letter “ه”.

The letter “ه” is a Non-constant if it is a suffix letter. So, the algorithm converts “ه” letter to constant letters if it detects that it is not a suffix letter by applying the following rules:

```
def
H_rules(term,index,index_of_first_consonant,index_o
f_last_consonant):
    is_consonant = False
    if index < index_of_last_consonant :
        is_consonant = True
    elif index < (len(term)/2)-1:
        is_consonant = True
    if len(term)>= 3 and index==len(term)-3 and
term[index+1]=='و'and term[index+2]=='ا' :
        return True
elif (len(term) > index+1 )and( (term[index + 1] == 'ب' )
 or(term[index + 1] == 'س' )or(term[index + 1] == 'ف' )
    or (term[index + 1] == 'ك')or(term[index + 1] ==
'ل' )or(term[index + 1] == 'ه' )or(term[index + 1] == 'ة'
)):
        is_consonant = True
    return is_consonant
```

6) Rules for Letter “ة”.

Letter “ة” is always extra letter, no root contains the letter “ة”. Therefore, we remove this letter from the word.

7) Rules for Letter “ك”.

The letter “ك” is a Non-constant if it is a suffix letter or prefix letter. So, the algorithm converts “ك” letter to constant letters if it detects that it is not a suffix letter or prefix letter by applying the following rules:

```
def
K_rules(term,index,index_of_first_consonant,index_o
f_last_consonant):
    is_consonant = False
    if  index_of_first_consonant < index <
index_of_last_consonant:
        is_consonant = True
    elif (index < (len(term) / 2) - 1) and (index >= 1)
and term[index - 1] not in ['و' ,'ف']:
        is_consonant = True
    elif (index > (len(term) / 2)) and (index <
index_of_last_consonant):
        is_consonant = True
    elif (index > (len(term) / 2))and
(len(term)==index+3) and ((term[index + 1] == 'ا'))and
((term[index + 2] == 'ت')):
        is_consonant = True
    elif (index > (len(term) / 2))and
(len(term)==index+3) and ((term[index + 1] ==
'ي'))and ((term[index + 2] == 'ن')):
        is_consonant = True
    elif len(term)==index+2 and term[index+1]=='ت':
        is_consonant = True
    return is_consonant
```

For example, the algorithm converts the letter “ك” to constant letter if it comes between two constant letters, like the word “ذكرته” (I mentioned); the letter “ك” comes between the two constant letters “ذ” and “ر”.

8) Rules for letter “م”.

The letter “م” is a Non-constant if it is a suffix letter or prefix letter. So, the algorithm converts “م” letter to constant letters if it detects that it is not a suffix letter or prefix letter by applying the following rules:

```
def M_rules(term, index, index_of_first_consonant, index_of_last_consonant,len_consnonats):
    is_consonant = False
    # if len_consnonats>=3 and index>index_of_last_consonant:
    #    return False
    if len_consnonats>=3 and index<index_of_first_consonant:
        return False
    if index_of_first_consonant < index < index_of_last_consonant:
            is_consonant = True
    elif (index < (len(term) / 2)) and (index >= 1) and ((term[index - 1] == 'ت')
        or (term[index - 1] == 'ا') or (term[index - 1] == 'ه') or ( term[index - 1] == 'ي')):
            is_consonant = True
    elif (index >= (len(term) / 2))and (index<index_of_last_consonant):
        is_consonant = True
    elif (index < (len(term) / 2)) and (index >index_of_first_consonant):
        is_consonant = True
    elif (len(term) >= index_of_last_consonant + 2) and (index == index_of_last_consonant + 2) and (
         (term[index - 1] == 'ا') or (term[index - 1] == 'و') or (term[index - 1] == 'ي')):
            is_consonant = True
    elif (len(term) >= index_of_last_consonant + 1) and (index == index_of_last_consonant + 1):
            is_consonant = True
    elif (index > (len(term) / 2))and (len(term)==index+3) and ((term[index + 1] == 'ا'))and ((term[index + 2] == 'ت')):
            is_consonant = True
    elif (index > (len(term) / 2))and (len(term)==index+3) and ((term[index + 1] == 'ي'))and ((term[index + 2] == 'ن')):
            is_consonant = True
return is_consonant
```

For example, the algorithm converts the letter “م” to constant letter if it comes between two constant letters, like the word “ثمار” (fruits); the letter “م” comes between the two constant letters “ث” and “ر”.

9) Rules for letter “ن”.

The letter “ن” is a Non-constant if it is a suffix letter or prefix letter. So, the algorithm converts “ن” letter to constant letters if it detects that it is not a suffix letter or prefix letter by applying the following rules:

```
def N_rules(term, index, index_of_first_consonant, index_of_last_consonant,len_consnonats):
    is_consonant = False
    if len_consnonats>=3 and index>index_of_last_consonant:
        return False
    if len_consnonats>=3 and index<index_of_first_consonant:
        return False
    if len(term)>= 3 and index==len(term)-3 and term[index+1]=='ا' and term[index+2]=='ء' :
        return True
    if index_of_first_consonant < index < index_of_last_consonant:
        is_consonant = True
    elif (index >= (len(term) / 2)) and (index < index_of_last_consonant):
        is_consonant = True
    elif (index >= 2) and (term[index - 2] == 'ا') and (term[index - 1] == 'ل')and (index < (len(term) / 2)):
        is_consonant = True
    elif (index < (len(term) / 2)) and (index >= 1) and
((term[index - 1] == 'ه')or (term[index - 1] == 'ن')):

        is_consonant = True

    elif (index > (len(term) / 2))and
(len(term)==index+3) and ((term[index + 1] ==
'ي'))and ((term[index + 2] == 'ن')):

        is_consonant = True

    return is_consonant
```

For example, the algorithm converts the letter “م” to constant letter if it comes between two constant letters, like the word “جناحية” (Wings); the letter “ن” comes between two constant letters “ث” and “ر”.

D. *Extracting all possible patterns for the word*

After extracting all constant letters from the word and converting Non-constant letters to constant letters, in some cases the algorithm can’t find all roots’ letters. To find all roots’ letter the algorithm uses the patterns. In this section the authors use (4320) patterns that were extracted by Thalji’s corpus [26]. The algorithm Extract all possible patterns of the derivation words. For example, with the word “استماع” (Listen), the algorithm finds just two constant letters. Then it extracts all possible patterns of the word by comparing the words with patterns’ list, in this case the possible patterns are “استفعال افتعال”. So, the roots are “ماع سمع”. Whereas khoja’s algorithm just find one pattern not all possible patterns. For this word khoja’s algorithm returns just “ماع” root.

E. *Finding all possible roots from the possible patterns*

After finding all possible patterns, now all the possible roots that match the patterns are extracted.

F. *Finding all possible roots by applying Ebdal rules*

With Ebdal, the letters ” ط” and ” د” are consider as infix letters, if "د" comes after "ز" letter in the word and the word match the pattern” افدعل”, and if "ط" comes after one of these letters "ض, ص, ط , ظ" in the word and the word match the pattern “افطعل”.

G. *Solving shaddah’s word problem* “ ◌ّ ”

This study is for non-vocalized text, so in many cases, the writers don’t write Shaddah above the letter, the algorithm is tried to check Shaddah’s symbol, it starts checking from the second letter in the word, except the vowel letters. For instance: the word "البر", the algorithm is generated the possible words of the Shaddah like, “اللبر, الببر البرر.”

H. *Solving the problem of missing the vowel in Ealal rules*

In the Arabic language, if the root has one or more vowel’s letters in derivation words, the long vowels may be deleted. For example, the root "قول", one of possible derivation’s word is "قل", during the derivation process the long vowel "و" is deleted. So, in this, the algorithm gives all possible cases of omitting the vowel letter for"قل". So, the algorithm is generated all these possible vowels’ roots "وقل, قول, قلو".

I. *Solving the problem of changing the vowel in Ealal rules*

In the Arabic language, if the root has one or more vowel’s letters, in derivation words these letters may be changed to another vowel’s letters. For example, the root "قول", one of the possible derivation words is "قال", during the derivation’s process, the vowel’s letter "و" is changed to another vowel’s letter "ا". Another possible derivation word is "قيل", during the derivation’s process, the vowel’s letter "و" is changed to another vowel’s letter "ي". So, the algorithm gives all possible cases of changing the vowel’s letters of"و" to “ي,ا”. Finally, for this case, the possible roots for the word “قول” are “قيل, قال ”.

J. *Minimizing possible roots by comparing them with roots’ list*

The algorithm generates a large number of possible vowel roots because vowel roots have many more cases that are special. The algorithm uses the list of (12000) Arabic roots that is extracted Thalji’ corpus, in order to minimize the possible roots. For example, the word "درهم" , the algorithm generates the following roots ادر, ودر, يدر, دار, دور, دير, درا, درو, دري, درر but the root " يدر" is excluded, because it is not founded in the original root’s list.

III. RESULTS & DISCUSSION OF THE PROPOSED ARABIC ROOT EXTRACTION ALGORITHM

In this section, the proposed algorithm is compared with Khoja’s Arabic root extraction algorithm, which is a very popular Arabic root extraction algorithm, and the only available Arabic root extraction algorithm for download. Khoja has tested her Arabic root extraction algorithm in the newspaper and achieved 95%. Thalji’s corpus has (720000) word-root pairs to test and compare Arabic root extraction algorithms. We select this corpus to test and compare our presented algorithm, because it has a large number of words, with more than (4320) words’ type. Pairing each word with its root simplify the testing process, which done automatically with no need to the experts to check the correctness of the algorithms. Specifically, we make a pure and completely comparison between Khoja’s Arabic root extraction algorithm and the proposed algorithm on Thalji’s corpus. The result of testing shows that the accuracy of Khoja’s algorithm is 63% and the accuracy of the proposed algorithm achieves 92%.

Table 1 illustrates some sample of results of Khoja’s root extraction algorithm, Khoja’s root extraction algorithm result is

(NOT STEMMED), where the present algorithm returns correct results.

TABLE 1 EXAMPLE OF NOT STEMMED WORDS BY KHOJA'S ALGORITHM

No	***Input word***	***Khoja's algorithm result***	***Present algorithm result***	***Exact root***
1	يحرمونهن	NOT STEMMED	حرم	حرم
2	اخلعوهن	NOT STEMMED	خلع	خلع
3	تستبدل	NOT STEMMED	بدل	بدل
4	حاسوب	NOT STEMMED	حسب	حسب
5	ازدهر	NOT STEMMED	زهر	زهر
6	اصطحب	NOT STEMMED	صحب	صحب
7	اصطلح	NOT STEMMED	صلح	صلح
8	لأصدقائك	NOT STEMMED	صدق	صدق
9	نستخدمها	NOT STEMMED	خدم	خدم
10	شركاؤنا	NOT STEMMED	شرك	شرك

Error! Reference source not found. shows examples of wrong results by Khoja's algorithm, where the present algorithm returns correct results

TABLE 2 EXAMPLES OF WRONG RESULTS BY KHOJA'S ALGORITHM

No	**Input word**	**Khoja's algorithm result**	**Present algorithm result**	**Exact root/ roots**
1	البثني	ثني	بثن	بثن
2	البسابس	سبس	بسبس, بسس	بسس,بسبس
3	والبغث	غثي	بغث	بغث
4	فالباقول	قول	بقل	بقل
5	البلسان	لسن	بلس	بلس
6	وجهوري	جهور	جهر	جهر
7	الحداد	حود	حدد	حدد
8	زعما	وزع	زعم	زعم
9	رجما	رجأ	رجم	رجم
10	رعابيب	عيب	رعب	رعب

The weaknesses points of Khoja's algorithm are listed below:

- Khoja and Garside's algorithm is missing the large number of roots, prefixes, suffixes, and patterns. Khoja and Garside's dictionary is restricting the result for just (4748) roots, (3,822) trilateral roots, (926) quadrilateral roots. Because Khoja and Garside's algorithm ignores (7252) roots, the result of ignoring these roots causes wrong results because if one uses any of ignoring roots, he/ she will not find the correct result or will not achieve the correct stemmed. For example, the word "مملوع" is stemmed to the wrong root "ولع", because it misses the root "ملع", also the word "لأصدقائك" is not stemmed because it is missing the pattern "لأفعلائك", the same thing will occur to the following words" اخلعوهن, تستبدل, يحرمونهن, حاسوب, ازدهر, اصطحب, اصطلح, لأصدقائك, نستخدمها, شركاؤنا".
- Khoja and Garside's algorithm suffers from affix ambiguity problems, for example, it returns "ميع" root for the word "استماع", but it should also return the root "سمع", this is because it starts by removing the longest suffix or prefix, but sometimes its neither prefix nor suffix, its root's letters.
- Khoja and Garside's algorithm, again, returns just one solution for non-vocalized words, ignoring other possible solutions, for example, the word "قل", the possible roots are"قول, قلل, قلو, وقل, قيل قلي", where the result is just "قلل".
- Khoja and Garside's algorithm replaces a weak letter with the letter "و", which occasionally produces a root that is not related to the original word, for example, it returns "وصل" root for the word "أصيل" which is the wrong root, the right root is"أصل".
- Khoja and Garside's algorithm may generate invalid roots or fail to find roots for words that contain Ebdal rule "ابدال" like "اصطحب, اصطلح" and ازدهر"".
- Khoja and Garside's algorithm, also, does not deal with Shaddah, for example, with the word "وأب", it returns the root "أبي", where the possible root also is "أبب".

As we stated before, the accuracy of the proposed algorithm is 92%, it benefited from most of the existing Arabic root extraction algorithm and takes the strengths points and care for the weakness's points. So, when the proposed algorithm tests words it returns almost the true result and all potential roots, in this section the weak points of the proposed algorithm are listed below:

- The proposed algorithm fails to find the root of words with only one letter length. In Arabic, there are some few words with only one letter length like "قِ, عِ, رِ", these words are derived from vowel
- root with length three letters, and these vowel letters are deleted during derivation process.
- Another case that the proposed algorithm still fails to find the root of words is in the word "درهم" the algorithm result is "درأ, دري, دور, ودر درر, ". In this word's case, the algorithm finds two constant letters in the word, this leads to continually looking for the third constant letter in order to return trilateral roots. However, we stopping allowed the proposed algorithm to continue looking for the fourth one.
- Another case that the proposed algorithm still fails to find the root of words is in the word "ذيعوعة", the algorithm result is just "ذعع" root. In this well-known word's case, if the algorithm finds three constant letters, it returns them as trilateral root, which becomes the result. At the same time, the proposed algorithm is not deal with exchanging the constant letter with the vowel letter because this case has rarely happened.
- The proposed algorithm gives all possible roots of the word, so this cause a misunderstanding result for the researcher to find the exact root, which is needed. This fails coming up clearly, because the proposed algorithm deals with words rather than the completed meaningful sentences in a paragraph.

IV. CONCLUSION

In this study, we enhanced the rules of Khoja's Arabic root extraction. This study also analyses most previous Arabic root extraction algorithms, benefits from all their strong ideas, and overcome the limitations. This study continues what the others already started. The proposed algorithm contributes to enhancing the existing algorithms by increasing the rules and the datasets. The proposed algorithm solves Negative prefix, Negative suffix, one solution and some words are not stemmed.

The proposed Arabic root extraction algorithm is compared with Khoja's Arabic root extraction algorithm, which is a well-known Arabic root extraction algorithm, and the only one available on the websites to use and download, it has 95% accuracy when it was tested in the selective data set. However, we test Khoja algorithm on Thalji's corpus, the result of testing shows that the accuracy of Khoja algorithm is 63%. At the same time, we test the algorithm on the same corpus, it achieves 92%.

REFERENCES

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first . . ."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Kanaan, R. Al-shalabi and M. Sawalha, "Improving Arabic information retrieval systems using part of speech tagging," pp. 32-37, 2005.

[2] M. Y. Dahab, A. Al Ibrahim e R. Al-Mutawa, "A comparative study on Arabic stemmers," International Journal of Computer Applications, vol. 125, n. 8, 2015.

[3] T. M. T. Sembok e B. AbuAta, "Arabic word stemming algorithms and retrieval effectiveness," In Proceedings of the World Congress on Engineering , vol. 3, pp. 3-5, 2013.

[4] R. Kanaan e G. Kanaan, "An improved algorithm for the extraction of triliteral Arabic roots," European Scientific Journal, vol. 10, n. 3, pp. 346-355, 2014.

[5] B. Abuata e A. Al-Omari, "A rule-based stemmer for Arabic Gulf dialect," Journal of King Saud University-Computer and Information Sciences, pp. 104-112., 2015.

[6] Y. Jaafar e K. Bouzoubaa, "Arabic natural language processing from software engineering to complex pipeline," Jaafar, Y., & Bouzoubaa, K. (2015, April). Arabic natural languagIn Arabic Computational Linguistics (ACLing), First International Conference on.IEEE, pp. 29-36, 2015.

[7] Y. A. Al-Lahham, K. Matarneh e M. A. Hassan, "Conditional Arabic Light Stemmer: CondLight," The International Arab Conference on Information Technology, pp. 1-5, 2017.

[8] R. Jayanthi e C. Jeevitha, "An approach for effective text pre-processing using improved Porters stemming algorithm," Int J Innov Sci Eng Technol , pp. 797-807., 2015.

[9] L. S. Larkey, L. Ballesteros e M. E. Connell, "Light stemming for Arabic information retrieval," Arabic computational morphology, ., pp. 221-243, 2007.

[10] M. Ababneh, R. Al-Shalabi e G. ,. A.-N. Kanaan, "Building an Effective Rule-Based Light Stemmer for Arabic Language to Improve Search Effectiveness," International Arab Journal of Information Technology (IAJIT) , vol. 9, n. 4, pp. 368-372, 2012.

[11] O. M. Elrajubi, "An improved Arabic light stemmer," International Conference on. IEEE, pp. 33-38, 2013.

[12] M. Y. Almusaddar, "Improving Arabic Light Stemming in Information Retrieval Systems," Thesis, Islamic University, Gaza, Palestine, 2014.

[13] A. Al-Omari e B. Abuata, "Arabic light stemmer (ARS)," Journal of Engineering Science and Technology, pp. 702-717.2014 ,.

[14] K. Abainia, S. Ouamour e H. Sayoud, "A novel robust Arabic light stemmer," Journal of Experimental & Theoretical Artificial Intelligence, vol. 29, n. 3, pp. 557-573, 2017.

[15] O. Aldabbas, R. Al-Shalabi, G. Kanan, M. A. Shehab, M. Albdarnah e N. Mahyoub, "Arabic light stemmer based on regular expression," Proceedings of the International Computer Sciences and Informatics Conference, pp. 1-82016 ,.

[16] S. R. El-beltagy e A. Rafea, "An accuracy-enhanced light stemmer for arabic text," ACM Transactions on Speech and Language Processing (TSLP), vol. 7, n. 2, 2011.

[17] S. Khedr, D. Sayed e A. Hanafy, "Arabic Light Stemmer for Better Search Accuracy," International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering, vol. 10, n. 11, pp. 3521-3529, 2016.

[18] N. Caka, "What is the difference between root word and stem word?," 25 8 2017. [Online]. Available: https://www.researchgate.net/post/What_is_the_difference_between_root_word_and_stem_word.

[19] F. Abu Hawas e K. Emmert E, "Rule-based approach for Arabic root extraction: new rules to directly extract roots of Arabic words," Abu Hawas, F., & Emmert, K. E. (2014). Rule-based approach for Arabic rJournal of Computing and Information Technology, vol. 22, n. 1, pp. 57-68, 2014.

[20] F. Abu Hawas e K. E. Emmert, "Rule-based approach for Arabic root extraction: new rules to directly extract roots of Arabic words," Journal of computing and information technology, pp. 57-68.2014 ,.

[21] S. Al-Fedaghi e F. S. Al-Anzi, "A new algorithm to generate Arabic root-pattern forms," proceedings of the 11th national Computer Conference and Exhibition, 1989.

[22] Q. Yaseen and I. Hmeidi, "Extracting the roots of Arabic words without removing affixes," Journal of Information Science, pp. 376-385, 2014.

[23] H. M. Al-Serhan, R. Al Shalabi and G. Kannan, "New approach for extracting Arabic roots," Proceedings of the 2003 Arab conference on Information Technology, pp. 42-59, 2003.

[24] I. Ali, "New Root-based Stemmer for Arabic Languag," Iraqi Journal of Science, pp. 2760-2766., 2016.

[25] S. Khoja and R. Garside, "Stemming arabic text," Lancaster, UK, Computing Department, Lancaster University, 1999.

[26] N. Thalji, N. A. Hanin, Y. Yacob e S. Al-Hakeem, "Corpus for Test , Compare and Enhance Arabic Root Extraction Algorithms," International Journal of Advanced Computer Science and Applications, vol. 8, n. 5, pp. 229-236, 2017.