Journal of Engineering Science and Technology
Vol. 9, No. 6 (2014) 702 - 716

ARABIC LIGHT STEMMER (ARS)

ASMA AL-OMARI, BELAL ABUATA*

CIS Department, Faculty of Information Technology and Computer Sciences
Yarmouk University, 21163 Irbid – Jordan
*Corresponding Author: belalabuata@yu.edu.jo

Abstract

Stemming is a main step used to process textual data. It is usually used in several types of applications such as: text mining, information retrieval (IR), and natural language processing (NLP). A major task in stemming is to standardize words; which can be achieved by reducing each word to its base (root or stem). Arabic stemming is not an easy task. Unlike other languages, Arabic language is a highly inflected language, since it uses many inflectional forms. Researchers are divided on the benefit of using stemming in fields of IR, NLP...etc., since in Arabic the morphological variants of a certain word are not always semantically related. The aim of this paper is to design and implement a new Arabic light stemmer (ARS) which is not based on Arabic root patterns. Instead, it depends on well defined mathematical rules and several relations between letters. A series of tests were conducted on ARS stemmer to compare its effectiveness with the effectiveness of two other Arabic stemmers. Test shows clearly the effectiveness superiority of ARS compared to effectiveness of these two Arabic stemmers.

Keywords: Arabic stemming, Light Arabic stemming, Rule based stemming, Stemming errors.

1. Introduction

Stemming is an important process used in many fields of natural language processing like information retrieval systems, Web search engines, Question Answering Systems, textual classifiers, etc. Two preprocessing techniques are used to reduce the number of tokens in textual documents: stop word removal and stemming. Also, it is used to find semantically identical terms which are derived from the same stem/root. Two main types of stemming are usually used: Light stemming which is restricted to the removal of limited number of Arabic prefixes

and suffixes, to output a stem. Heavy stemming which is also known as root-based stemming, include implicitly the removal of Arabic prefixes and suffixes, besides extracting the appropriate root. Extracting the appropriate root for Arabic language is a difficult task compared with western languages due to the complicated morphological rules found in Arabic. As a result, there are many different studies carried out to get efficient stemming results [1]. Both orthography and morphology increase the amount of lexical variation in Arabic, which means that a word can be found in a huge number of different forms. This is not the case for many western European and English languages' stemming where the stemming process can be easily done by removing words' suffix in most cases [2]. Also, there are many differences between Arabic language from one side, and languages used in Europe and Northern and Southern America from the other side. Arabic language is a Semitic language written from right to left. It is based on 28 Arabic alphabets, which are also used by other languages such as Ottoman, Urdu, Persian, Kurdish, Sindhi, and Malay. Arabic language is highly inflectional, and the mother tongue of 5% of the world population.

Arabic language is the first widely spoken Semitic language, followed by: Amharic which is used in Ethiopia by 25 million native speaker, followed by Hebrew with 7 million native speakers, Tigrinya (6.7 million native speakers), and Aramaic (2.2 million native speakers) [3].

Some researchers describe Arabic language as an algebraic language, which means that its morphological analysis process is hard and difficult. Nouns and verbs in Arabic are generated from a set of roots – about 11,347 roots- classified according to their letters length as follows [4]:

- 2 letter Arabic roots: there are 115 roots.
- 3 letter Arabic roots: there are 7,198 roots.
- 4 letter Arabic roots: there are 3,739 roots.
- 5 letter Arabic roots: there are 295 roots.

There are several issues in Arabic language that complicate its stemming process. Some Examples of these issues are:

1. One word may have several meanings according to its position and the context of the word in the text, e.g., the word ["عين" – Ayn]:
   - "شربت من عين الماء - *Sharebt men Ayn Almaa*." Which means: I have drink from appointed water.
   - "عين الله تراعاك – *Ayn allah traak*" which means: god bless you.
   - "أمراض كثيرة تصيب العين – *Amrad Katherh toseeb al Ayn*" which means: Many diseases infect eye.
   - "أصبت عين الحقيقة – *Asabt Ayn alhakeka*" which means: You were appointed the truth.
2. Some words have the same root but different meaning, e.g., all of the following words have the same root "Computed, حسب":
   - "حسب - *Hasaba*" which means: computed.
   - "حاسوب – *Hasoob*" which means: Computer.
   - "محاسبة – *Mohasabeh*" which means: Accounting.
   - "حساب – *Hesab*" which means: Mathematics.
3. Some original letters in a word are changed in their form, these letters in most cases are the vowels (" (ا) *Alef*, (و) *Waw*, (ي) *Yaa*"), e.g.:

- "صام *Sama*" from "صوم *Sawm*" where (و Waw) is converted to (ا Alef).
- "سار *Sara*" from "سير *Sair*" where (ي Yaa) is converted to (ا Alef.)
- "رعى *Raa*" from "رعي *Raia*" where (ي Yaa) is converted to (ى Alef maqsorah).

4. Some letters are original in word and have a function in other word, e.g. the letter (و Waw) in the following words:
   - The letter here is original:
     - (ورق *waraq*) which means paper.
     - (وصل *wasal*) which means arrived.
     - (لون *lawn*) which means color.
   - The letter here means (And):
     - (درس وكتب *Darasa wa kataba*) which means: Study and write.
     - (أكل وشرب *Akala wa shareba*) which means: Eat and drink.

The novel algorithm presented in this paper is one of few algorithms which do not rely on Arabic root patterns. Its core idea is based on arithmetic calculations and relations between letters. The remainder of this paper is organized as follows: Section 2 presents a number of studies related to our study. Section 3 exhibits Arabic stemming algorithms approaches. Section 4 talks about the problem domain. Section 5 talks about the new stemming algorithm. Section 6 shows the results and section 7 presents conclusion and future plans to improve this study.

2. Related Work

There many algorithms and methods presented by researchers to find the appropriate Arabic stem/root of different Arabic words. A well known Arabic stemming algorithm is presented by Khoja et al. [5]. Khoja's stemmer first attempts to remove prefixes and suffixes, and then find the Arabic root. McNamee et al. [6] stemmer works by matching n-grams of multiple lengths with index words. Their stemmer produces a big size index. Other researchers like Larkey and Connell [7], Darwish and Orad [8], and Chen and Gey [9], used a light stemmer where they selected prefixes and suffixes to be removed from the Arabic word. A new idea used from Kadri and Nie [10], when they tried to find the core of a word and determine such cores as an index since each word in Arabic is formed as a sequence of (antefix, prefix, core, suffix, and postfix). These indexes will encode the basic semantics in Arabic Language.

Al-Shalabi et al. [11] used what is called a letter weight, but they have not show how to derive these weights. They extract the Arabic root of the word after applying a weight and rank values to each letter in the word according to some tables. Then they used special calculations to decide which letter has to be removed and which has to stay in order to get the root or stem. Other researchers like Al-Ameed [12] used two parts stemmer, where in the first part, he used rule-based light stemmer to remove the prefixes and suffixes and in the second part, he used pattern based infix remover to remove infixes from the word according to specific patterns.

Aitao and Fredric [13] build an MT-based Arabic stemmer. They translate Arabic word into English and based on this translation they divide the Arabic words into clusters. Then all Arabic words conflated to the same English stem (after translation) form the same cluster. All the Arabic words in this cluster are changed to the shortest word in the cluster. Manual technique was used by Al-Kharashi and Evens [14] to find stem of a word. They used a small text collection to build dictionaries of roots and stems for each word to be indexed. Buckwalter [15] created a set of lexicons of Arabic stems, suffixes and prefixes with truth table and BBN group used this table in TREC-2001. Statistical techniques were used by Goldensmith [16] to find the stem or root. He used an information theoretic measure to find the best set of frequently occurring stems and suffixes. Also Oard et al. [17] used statistical techniques but they used the most frequently occurring word-final n-grams to be suffixes.

From the above discussion we find that there are four different approaches used by researchers to get the stem or root of an Arabic word. These approaches are presented in the next section.

3. Arabic Stemming Algorithms Approaches

There are four main approaches for Arabic stemming [2]:

i) Manually constructed dictionaries: Al-Kharashi and Evens work with a small text collection and find a dictionary of roots and stems for each word [14].

ii) Light stemmer: works by removing prefixes (letters added at the beginning of root) and suffixes (letters added at the end of the root). It does not deal with patterns or infix (letters added within the root, it causes a serious issue in stemming Arabic documents because it is hard to differentiate between root characters letters and affix letters).

iii) Morphological analysis: It states that the best way to find an Arabic stem is to conduct morphological analysis correctly first. Then to use some valid morphological unit for indexing and retrieval. For Arabic, this unit has often been thought to be the root. Many morphological analyzers have been developed for Arabic; few of them received a standard IR evaluation. Most such morphological analyzers find the root, or any number of possible roots for each word.

iv) Statistical stemmer: This type of stemmers uses clustering techniques to group word variants. It is widely applied to automatic morphological analysis in the field of computational linguistics.

Since the main goal for defining a stemmer is to increase and support the search effectiveness in IR systems. Stemming assists IR systems to match user's queries with relevant documents. Many researchers developed Arabic stemmers that depend on these approaches. A superior root-based stemmer is Khoja's stemmer [5]. This stemmer removes suffixes, prefixes and infixes and uses pattern matching using a dictionary to extract the roots. ISRI is a new stemmer which performed equivalently to Khoja stemmer in an IR environment. There is another algorithm that depends on Khoja's stemmer but it uses two dictionaries: one for roots and another for radicals (Infix and root form the core report also called radical of the word) [18].

Larkey et al. [2], created a group of light stemmers including Light 1, 2, 3,..., 8, and 10. The latest light10 stemmer outperformed the previous light stemmer versions. Darwish and Orad [8] presented a modified light stemmer called "Al-stem". It filters rootless words, and then removes suffixes and prefixes. It removes excessive letters such as these found in the word "سألتمونيها Saaltomoneeha" only if a letter occurs more than once in a word. Al-Shalabi, et al. algorithm does not use any dictionary [11]. Instead of removing prefixes and suffixes they analyze Arabic patterns by grouping patterns according to their lengths. The algorithm then extracts general rules for each group, which presents the possible positions of letters in order to consider them as excessive and thus remove them.

Saad et al. algorithm depends on using Back Propagation Neural Network to extract five letters from Arabic words roots. Encoded with binary digits, four types of input were created, one depends on the original letters, and the other three, classify the letters of the word according to their occurrence frequency as an affix letter [19].

4. Problem Domain

From the stemming methods and algorithms mentioned previously, all affixes of a word must be removed to get its stem or root. However, this is very complicated for Arabic words since there are four types of affixes [8] as shown in Table 1: antefixes, prefixes, suffixes and postfixes and the root of the word or its stem will be in the middle. Examples are the word "*Al-talabat* الطلبات" which means *the requisition* and the word "*le-uhadethokum* ليحدثونكم " which means *to talk to you*.

Table 1. Types and examples of Affixes in Arabic.

Word	Postfix	Suffix	Root (Core)	Prefix	Antefix
الطلبات		ات	طلب	ال	
ليحدثونكم	كم	ون	حدث	يـــ	لـــ

According to this partition, the root is typically in the middle of the parsed word.

Let us consider the Arabic word (الطلبات), and try to use a mathematical formula and not worry about any of the affixes. Instead we count the total number of letters (7 letters) and add to it 1 – to get an even number. Then divide the results (8) by 2 (to get the number 4). So our algorithm will consider the Arabic letter at position 4 with its two neighbors, and this leads directly to extract the correct Arabic root "Ask *Talab* طلب" in this case.. This example shows how our novel stemming algorithm is capable of finding the Arabic roots which lies in the middle. Arabic Roots not always lies in the middle, but they could lie around the middle as shown in the next example.

The second example is the word (ليحدثونكم). Its length is 9 then we add to it 1 and divide the sum by 2 to get the position to start with which is 5. The letter in this position with its two neighbors is (دثو).Since the third letter is vowel "Waw, و" we have to remove it and then move one letter to the right to get the Arabic letter "ح". The result root will be "*Hadath* حدث" which means happened.

The following four main steps summarize the new stemming algorithm (ARS):

- **Step 1**: Read a word from a document or from a word list and then finds its length (Number of its letters).
- **Step 2**: Identify a position to start with according to a mathematical formula. Afterward extract the letter at that position with the two neighboring letters.
- **Step 3**: Search for the extracted word from previous step in a dictionary of Arabic roots. If the extracted word from previous step is found in the dictionary of Arabic roots, then stop search. If not found then shift to the right by one letter to get a new word, then start searching in the dictionary. Repeat this step until either a root is found or no more letters exist to the right.
- **Step 4**: If the root is not found from the previous step return to the position computed in the second step and start shift to the left by one character and search for the extracted word in the dictionary of Arabic roots, until either a root found or not found.

This will be explained in detail in the next section.

5. The New Stemming Algorithm

The new proposed algorithm is an Arabic light stemmer that uses rules to find the word's root. Hence, it can be categorized as Arabic Light Stemmer (ARS). The detail of the new algorithm is as follows: (During these steps Vowel letter means one of the following "*Alef* ا, *Waw* و , *Yaa* ي ").

Algorithm: Arabic Light Stemmer (ARS)
Input: Authentic Arabic Words.
Output: Arabic Stems/Roots/Words/ (No Stem or Root Found**)**

```
BEGIN
        WHILE Not All Arabic Words read
                Read Single Word (W)
                Check whether it is a stop word or Not
 // Compute (Len(W) ) the length of the word W
                Len(W) = Word Length
                 IF Len(W)  = 2 then

                        New Word (NW) = first letter + "و" + second letter.
                        OR  New Word (NW) = first letter + "ي" + second
letter.
                        Compare (NW) with WD(Word in dictionary)
                        IF found then
                                Exit While
                        ELSE
                        New Word (NW) = "و" + first letter + second letter.
```

```
                    OR New Word (NW) = "ا" + first letter + second letter.
                    Compare (NW) with WD(Word in dictionary)
                        IF found then
                            Exit While
                        ELSE
                            PRINT " Root not found"
                        Exit While
                        END IF
// Working with a word of a root of size three letters
            ELSEIF Len(W) >= 3
// the following are some procedure has to be done before start the algorithm steps
            Those special cases are found in Table 2.
// Compute the position (P) of a character to start with, according to this position
we will create a sub word
// of three characters: a character at position P in addition to its two neighbors
        IF Len(W) is a positive even number THEN
            P = (Len(W) / 2)
        ELSE
            P = (Len(W) + 1) / 2
        ENDIF
        Long_Arabic_Vowels_Set = { (Alif, " ا"), (Waaw, "و "), (Yaa', "ي ")}

// Char_Mid: Character in position P.
// Char_Right: Right neighbor in position P and Char_Left: Left neighbor in
position P.
// Rules for Vowel letters
// 1. The three letters are vowels.
            IF Char_Mid AND Char_Right AND Char_Left in
Long_Arabic_Vowels_Set THEN
                Char_Left = Char_Left +1
                Char_Right = Char_Right -1
            ENDIF
//2. Only a letter at the position P is a vowel.
            ELSEIF Char_Mid in Long_Arabic_Vowels_Set THEN
                Char_Mid = Char_Right
                Char_Right = Char_Right + 1
            ENDIF
//3. A letter at position P and its right neighbor are vowels.
            ELSEIF Char_Mid AND Char_Right in
Long_Arabic_Vowels_Set THEN
                Char_Right = Char_Right - 1
            ENDIF
```

```
//4. A letter at position P and its left neighbor are vowels.
                ELSEIF Char_Mid AND Char_Left in
Long_Arabic_Vowels_Set THEN
                        Char_Left = Char_Left +1
                ENDIF
//5. The two neighbors of a letter at position P are Vowels.
                ELSEIF Char_Right AND Char_Left in
Long_Arabic_Vowels_Set THEN
                        Char_Right = Char_Right - 1
                        Char_Left = Char_Left +1
                ENDIF
//6. Only the right neighbor is a vowel.
                ELSEIF Char_Right in Long_Arabic_Vowels_Set THEN
                        Char_Right = Char_Right - 1
                ENDIF
//7. Only the left neighbor is a vowel.
                ELSEIF Char_Left in Long_Arabic_Vowels_Set THEN
                        Char_Left = Char_Left +1
                ENDIF
// Concatenate the three leters to get a word of size 3.
                Sub Word (SW)=Char_Right + Char_Mid + Char_Left
                WHILE SW not found in dictionary
// while the root not found check for  more characters to the right of Char_Right
in the original word (W)
                IF more characters to right of Char_Right THEN
                        Char_Left = Char_Mid
                        Char_Mid = Char_Right
                        Char_Right =Char_Right - 1
// define a new SW word
                        SW=Char_Right + Char_Mid + Char_Left
                        Compare new(SW) with WD (word in dictionary)
                        ELSE
                                EXIT WHILE
                                ENDWHILE
WHILE SW not found in dictionary
// root not found check if we have more characters to the left of Char_Left in the
original word (W)
                IF more characters to left of Char_Left THEN
                         define a new SW word
                        Compare new(SW) with WD(word in dictionary)
                ELSE
                        EXIT WHILE
```

```
                    ENDWHILE
                IF SW found THEN
                    ROOT or STEM = SW
                ELSE
                    PRINT "No Stem or Root Found"
                ENDIF
            ENDIF
ENDWHILE
END
```

Table 2 shows how our algorithm handles special cases which need special treatments.

Table 2. List of Special Cases and the Processes to Handle them in ARS.

Letters found	Process	Example
(*Yata* يتـ, *Mota* متـ , *Nata* نتـ , *Tata* تتـ , *Ata* اتـ)	Convert it to (*Alef* ا) or (*Waw* و) and remove all letters preceding it. If no stem found remove.	(*Nataketh* نتخذ) become (*Akath* اخذ)
Any of Vowel letters	Remove all succession of vowels from left.	(*Alwan* الوان) become (*Lawn* لون)
(*lela* للا) (*ala* الا)	Remove it and remove all letters preceding it.	*(Wlelawqat* وللاوقات) become (*Awqat* اوقات)
(*ea'a* ائـ)	-Remove it if the word length > 4 -Convert it to (*Alef* ا) or (*Waw* و) otherwise.	(*Mada'en* مدائن) become (Madan مدن) (*Qa'em* قائم) become (*Qawama* قوم)
(*Non* ن, *Mem* م، *Alef* ا ,*Yaa* ي, *Taa* ت) followed be a letter then (*Taa* ت) or (*Daa* د) or (*Taa* ط)	Remove it and remove all letters preceding it and keep the letter between them.	(*Ertagala* ارتجل) become (*Ragol* رجل) (*Ezdada* ازداد) become (*Zada* زاد)
(*Non* ن , *Mem* م، *Alef* ا ,*Yaa* ي, *Taa* ت) followed be a letter (*Noon* ن)	Remove it and remove all letters preceding it	(*Endathara* اندثر) become (*dathara* دثر)
(*Ya a*ي, *Waw* و, *Alef* ا) followed by (*Hamzah* ء)	Remove Hamzah (ء).	(*Asha'a* عشاء) become (*Asha* عشا)
(*Non* ن, *Mem* م، *Alef* ا ,*Yaa* ي, *Taa* ت) followed be the two letters (*Sata* ستـ)	Remove it and remove all letters preceding it	(*Estakhdama* استخدم) become (*Khadam* خدم)

In Figs. 1(a) and 1(b), two words with different lengths are stemmed using the new stemming algorithm. The figures also show the steps applied on the input word to obtain the stem.

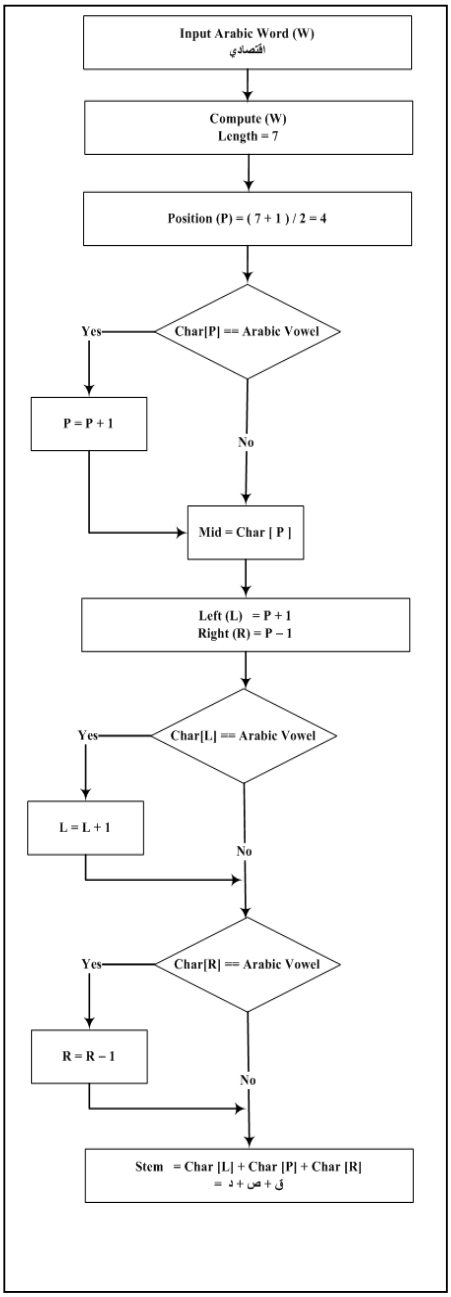

Fig. 1(a). Stemming Example for Word with Length 7 Using ARS.

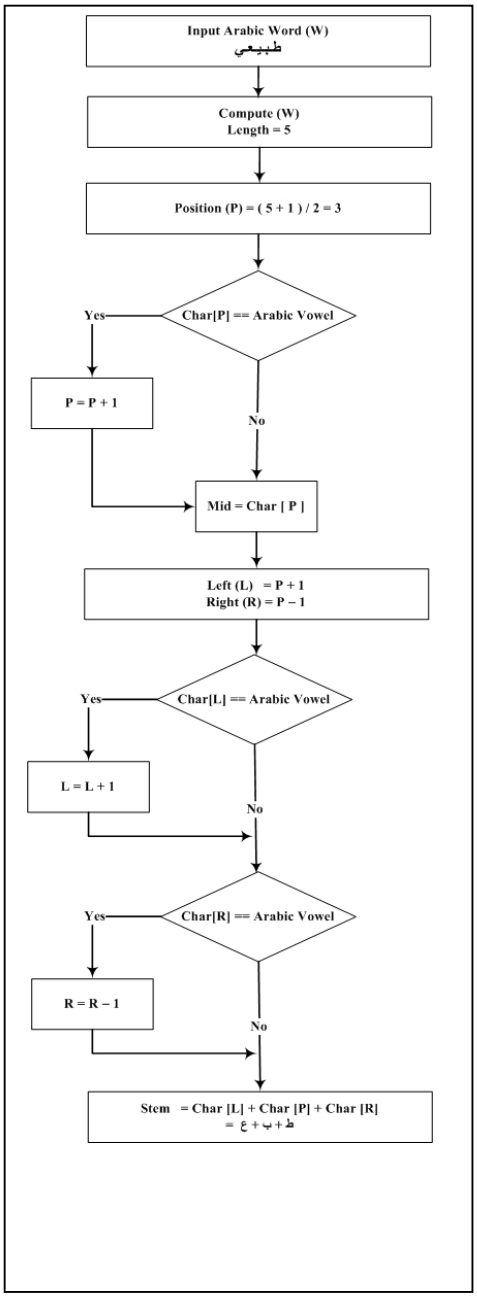

Fig. 1(b). Stemming Example for Word with Length 5 Using ARS.

6. Results and Discussion

This study is based on a dataset consists of 6,225 Arabic words. Inputting these words to our proposed algorithm (ARS) yield the results shown in Table 3 and Fig. 2. Table 4 shows some samples of Arabic words stemmed by ARS algorithm.

Several possible errors may occur in stemming. These errors are grouped into three types: over-stemming, mis-stemming and under-stemming [20]. These errors occur due to several language related issues or problems. For example, stemming two words with different stem to the same root is known as over-stemming, and the opposite is called under-stemming. Mis-stemming means "taking off what looks like an ending, but is really part of the stem" [20]. Table 5 shows examples of wrongly stemmed words according to these types of errors.

ARS is also tested and compared with two similar stemmers: Ghwanmeh's et al. [1] and Al- Kabi et al. [21]. The conducted tests had shown a major superiority in favor of ARS as shown in Fig. 3.

Table 3. Stemming Results of ARS Algorithm.

Total number of words	Correct Roots	Wrong Roots	
6225	5733	492	
		347 Wrong root	145 No root

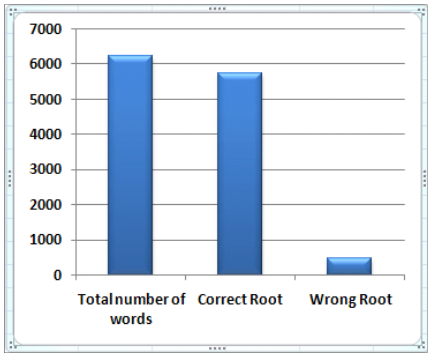

Fig. 2. Number of Correct, Wrong words Stemmed by ARS Algorithm.

Table 4. Examples of Words Correctly Stemmed by ARS Algorithm.

Word	Length	Root or Stem	Process
Sawm صوم (Fasting)	3	*Sawm* صوم	Nothing to do
Saa'e ساع (Runner)	3	*Sawa* سوع	Convert Vowel
Madda مدّ (Extend)	3	*Madada* مدد	Expand Shadeh

Doaa دعاء (Invocation)	4	*Daa'a* دعا	Remove Hamza (ء)
Ta'er طائر (Bird)	4	*Tayr* طير	Convert (ائ) to Vowel letter (ي)
Waheed وحيد (Lonely)	4	*Wahad* وحد	Remove ي to the left of middle.
Nawal نوال (Attain)	4	*Nawl* نول	Sequence of Vowel, remove to the left
Motakheth متخذ (Adopted)	4	*Akhath* اخذ	Convert (مت) to (ا)
Motawke'a متوقع (Expected)	5	*Waqa'a* وقع	Remove (مت)
Wasatah وساطة (Mediation)	5	*Wasad* وسط	Remove (ا) and shift to right
Ata'aat عطاءات (Tenders)	6	*Ata* عطا	Remove ء and shift to right, no need to remove (ات) because root found before that
Emtehanat امتحانات (Examinations)	8	*Mahan* محن	Remove ا in the middle then ت،ا in front. No removing from end because root fond before that
Wasayadrosonaha وسيدرسونها (And they will study it)	10	*Daras* درس	Nothing to be removed, first stem is right.

Table 5. Examples of Arabic Words Failed to be Stemmed Correctly by ARS.

Word	Result	Root or stem	Type	Reason
Tamareen تمارين (Exercises)	*Tamara* تمر	*Marana* مرن	False positive	Over-stemming
Rayaan ريان (Chubby)	*Ryana* رين	*Rawa* روى	False Positive	Over-stemming
Qefoo قفو (Stand up)	No root	*Waqafa* وقف	False Negative	Under-stemming
Maoon ماعون (Vessel)	No root	*Ma'an* معن	False Negative	mis-stemming
Al-ekhwa الاخوة (The brothers)	No root	*Khawa* خوى	False Negative	Under-stemming
Rawabee روابي (Mounds)	*Rawaba* روب	*Rabaya* ربي	False Positive	Over-stemming
Mawazeen موازين	*Mawz* موز	*Wazan* وزن	False Positive	mis-stemming

(Scales)				
Masa'er مصائر (Fates)	*Masr* مصر	*Syara* صير	False Positive	Over-stemming
Al-mathameen المضامين (The Implying)	No root	*Thaman* ضمن	False Negative	mis-stemming
Zorna زرن (They visit)	No root	*Zara* زار	False Negative	Others
Nadros ندرس (We study)	*Nadr* ندر	*Daras* درس	False Positive	Others
Moseqa موسيقى (Music)	*Saqa* سقى	*Moseqa* موسيقى	False Positive	Over-stemming
Bakteria بكتيريا (Bacteria)	*Katar* كتر	*Bakteria* بكتيريا	False Positive	Over-stemming

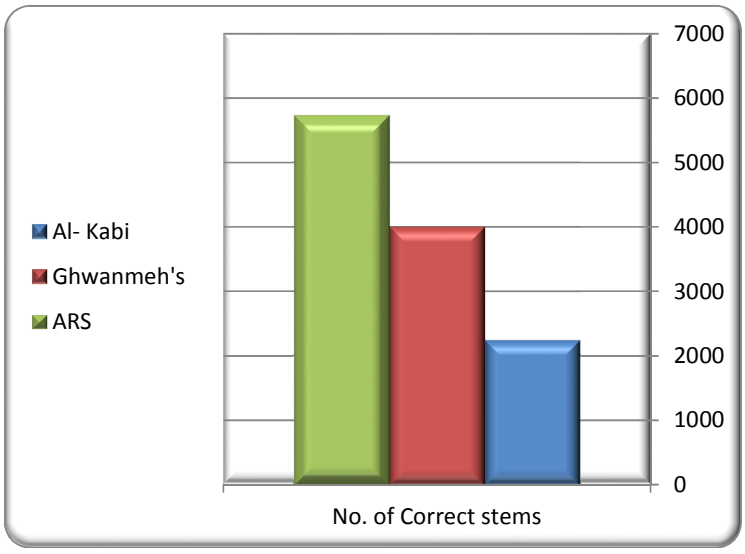

Fig. 3. Stemming Results of ARS Compared with Two other Arabic Stemmers.

7. Conclusion

This paper introduces a new Arabic stemming algorithm. This new algorithm is based on the use of well defined mathematical rules and the relations between Arabic letters for light Arabic stemming. This new algorithm (ARS) does not use at any stage any patterns or templates in the process of extracting the root/stem of an Arabic word. The main idea of ARS is based on performing a set of mathematical rules and matching the relations between some letters of Arabic word to find the stem/root of this word.

The evaluation of ARS had shown few wrong stems when applied on a set of 6,225 Arabic words. ARS is also tested and compared with two similar stemmers. The conducted tests had shown a major superiority in favor of ARS as shown in Fig. 3.

ARS needs to be enhanced and improve its capabilities to extract correctly the Arabic roots of a larger percentage of Arabic words. More tests for ARS are also required to justify its use in Arabic applications such as Arabic IR and Arabic translation.

References

1. Ghwanmeh, S.; Rabab'ah, S.; Al-Shalabi, R.; and Kanaan, G. (2009). Enhanced algorithm for extracting the root of Arabic words. *Sixth International Conference on Computer Graphics, Imaging and Visualization*, 388-391.
2. Larkey, L.S.; Ballesteros, L.; and Connell, M.E. (2002). Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. *SIGIR '02Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.* Tampere, Finland, 275-282.
3. Semitic languages (2011). Wikipedia, the free encyclopedia. Retrieved November 23, 2011, from http://en.wikipedia.org/wiki/Semitic_languages.
4. Kanaan, G.; Al-Shalabi, R.; Ababneh, M.; and Al-Nobani, A. (2012). Building as effective rule-based light stemmer for Arabic language to improve search effectiveness. *The International Arab Journal of Information Technology*, 9(4), 368-372.
5. Khoja, S.; and Garside, R. (1999). Stemming Arabic text. Computing Department, Lancaster University, Lancaster. Retrieved April 15, 2012, from http://zeus.cs.pacificu.edu/shereen/research.htm.
6. McName, P.; Mayfield, J.; and Piatko, C. (2000). A language independent approach to European test retrieval. *In Cross Language information retrieval and evaluation: Proceeding of the CLEF 2000 Workshop*, 129-139.
7. Larkey, L.S.; and Connell, M.E. (2001). Arabic information retrieval at UMass in TREC-10. *The Tenth Text REtrieval Conference TREC-10 conference*. Gaithersburg, Maryland, 562-570.
8. Darwish, K.; and Orad D.W. (2002). CLIR experiments at Maryland for TREC-2002: Evidence combination for Arabic-English retrieval. *The Eleventh Text REtrieval Conference TREC-11 conference*. Gaithersburg, Maryland, 703-710.
9. Chen A.; and Gey, F. (2002). Building an Arabic stemmer for information retrieval. *The Eleventh Text REtrieval Conference TREC-11 conference*. Gaithersburg, Maryland, 631-639.
10. Kadri, Y.; and Nie, J. (2006). Effective stemming for Arabic information retrieval. *In Proceedings of the challenge of Arabic for NLP/MT Conference.* The British Computer Society. London, UK, 68-74.

11. Al-Shalabi, R.; Kannan, G.; and Al-Serhan, H. (2003). New approach for extracting Arabic roots. *In Proc of 2003 International Arab Conference on Information Technology (ACIT'2003)*, 42-59.
12. Al-Ameed, H.K. (2006). *A proposed new model using a light stemmer for increasing the success of search in Arabic terms*. PhD Thesis, Bradford, UK: University of Brand ford.
13. Aitao, C.; and Fredric, G. (2003). Building an Arabic stemmer for information retrieval. *Special Publication SP*(*2003*), *Publisher*: Citeseer, 631-639.
14. Al-Kharashi, I.; and Evens, M.W. (1994). Comparing words, stems and roots as index terms in an Arabic information retrieval system. *Journal of the American Society for Information Science*, 45(8), 548-560.
15. Buckwalter, T. (2002). Buckwalter Arabic morphological analyzer Version 1.0. Linguistic Data Consortium, catalog number LDC2002L49. ISBN 1-58563-257-0. Retrieved April 15, 2012, from http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId= LDC2002L49.
16. Goldsmith, J. (2000). Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2), 153-198.
17. Orad, D.W.; Levow, G.A.; and Cabezas, C.I. (2001). CLEF experiments at Maryland: Statistical stemming and backoff translation. *In Cross-language information retrieval and evaluation: Proceedings of the CLEF 2000 workshop*, 176-187.
18. Al-Shammari, E.T. (2010). Improving Arabic document categorization: Introducing local stem. *In the 10th International Conference on Intelligent Systems Design and Applications*, 385-390.
19. Saad, E.M.; Awadalla, M.H.; and Alajmi, A. (2010). Arabic verb pattern extraction. *In the 10th International Conference on Information Science, Signal Processing and their Applications* (*ISSPA 2010*), 642-645.
20. Al-Shammari, E.T. (2009). A novel algorithm for normalizing noisy Arabic text. *World Congress on Computer Science and Information Engineering*, *IEEE*, 477- 482.
21. Al-Kabi, M.N; Kanaan, G.; Al-Shalabi, R.; Al-Sinjilawi, S.I.; and Al-Mustafa, R.S. (2005). Al-Hadith text classifier**.** *Journal of Applied Sciences,* 5(3), 584-587.