International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:10, No:11, 2016

Arabic Light Stemmer for Better Search Accuracy

Sahar Khedr, Dina Sayed, Ayman Hanafy

***Abstract*—**Arabic is one of the most ancient and critical languages in the world. It has over than 250 million Arabic native speakers and more than twenty countries having Arabic as one of its official languages. In the past decade, we have witnessed a rapid evolution in smart devices, social network and technology sector which led to the need to provide tools and libraries that properly tackle the Arabic language in different domains. Stemming is one of the most crucial linguistic fundamentals. It is used in many applications especially in information extraction and text mining fields. The motivation behind this work is to enhance the Arabic light stemmer to serve the data mining industry and leverage it in an open source community. The presented implementation works on enhancing the Arabic light stemmer by utilizing and enhancing an algorithm that provides an extension for a new set of rules and patterns accompanied by adjusted procedure. This study has proven a significant enhancement for better search accuracy with an average 10% improvement in comparison with previous works.

***Keywords*—**Arabic data mining, Arabic Information extraction, Arabic Light stemmer, Arabic stemmer.

I. Introduction

THE need to tackle Arabic language in the computational linguistic field has grown rapidly in the past few years. Many reports anticipate the need to properly learn and engage more in Arabic language, as declared in a British Council report. The report stated Arabic as one of the 10 languages which will be of critical importance and impact in the world for the coming years [1]. Arabic was also ranked as the fourth most spoken language on the Internet according to Internet World Stats [2]. From the industry perspective, it is clear that many enterprises have begun to realize this fact. As a result, there is more attraction towards Arabic as a language growing in importance. For instance, the announcement by IBM that states "IBM supercomputer and Jeopardy champion Watson is learning Arabic…" [3]. This involves tremendous development for natural language processing and machine learning artifacts. Yet in the open source domain, there are still shortage in tools and libraries that provide adequate handling for Arabic language challenges. Stemming is one of these challenges. Stemmers are the fundamental unit that is used in query systems, data mining, text search and information retrieval. By chopping of few letters of the word you can increase the probability of matching the word with other similar ones. This work is inspired by extending previous work done in regard to the Light stemmer by [8], and enhanced the Light Stemmer proposed by [5]. By adding new patterns, rules and algorithm modification the approach was able to successfully achieve better accuracy than prior works. The paper starts by stating the related work done in this area, and then walks through the detailed implementation for the algorithm. It is followed by the experiment results compared with similar systems. At the end, the conclusion and future work are presented.

Sahar Khedr is a software developer working as an intern in IBM, Cairo, Egypt (e-mail: saharkhedr@gmail.com).

Dina Sayed is a senior software engineer specialized in Arabic support, she is with Cairo Technology and development center, IBM, Cairo, Egypt (phone: 02-353-61426; e-mail: dsayed@ eg.ibm.com).

Ayman Hanafy is a senior research engineer with Watson group Department, IBM, Cairo, Egypt, (e-mail: ahanafy@eg.ibm.com).

II. Background and Related Work

Stemming is the process of reducing all morphological variants of a given word to its basic form, which can be either stem or root. Therefore, stemming has a significant impact on the search effectiveness of Information Retrieval (IR) systems. Stemming assists IR systems to match user's queries with relevant documents. Good stemming results in effective IR.

Julie Beth Lovins [6] and Porter [7] published the first Stemming algorithms in 1968 and 1980, respectively. Both algorithms were developed for the English language. The algorithms were simple and based on suffix stripping, which removes word suffixes by applying context rules. Other stemmers have been developed for a range of languages like French, Turkish and German. The same stemming approaches that have been applied to most languages are not appropriate for Arabic. This is because Arabic has significant differences in terms of grammatical, lexical and rich morphological features compared to other languages [4]. For example, Arabic has different affixes structure from other indo-European languages like English [9], and a rich morphology in word form, where a word can take many different forms. Therefore, Arabic poses special problems and complexities to NLP. To build effective Arabic IR systems, Arabic requires a different stemming process from other languages. Many studies covered comparisons for different implementations of Arabic stemmers as presented in [16] and [17]; these studies elaborated diversity in approaches, results and applications for various Arabic stemming techniques.

The introduced work does not support stemming that is root-based, as in the Arabic Light Stemmer (ARS) by [12] and [13]. Instead the approach focuses on light stemming rather than heavy stemming. A light stemmer avoids over-stemming errors by carefully removing limited affixes. Accordingly, the drawback is having many under-stemming errors. On the contrary, a heavy stemmer aggressively removes more affixes, and thus introduces over-stemming errors; though light stemming proved to be more accurate, as in case study by Mustafa, Suleiman H [10].

Several works were done to implement Light stemmer for Arabic. For instance, Kadri et al. introduced a rule-based effective light stemmer [11]; however, it does not process

word infix, and only supports prefix and suffix removal. Similarly, infix support was not introduced in improving Arabic light stemmer by [8]. The approach also is not context aware as in the "Rule-Based Extensible Stemmer" by [14]. Instead, it operates depending on the mostly used Arabic patterns accompanied with the stem list. The proposed approach utilizes a stem list unlike Al-Shammari et al. [15], which did not use a dictionary. The stem list helped in boosting the results by storing only the stem word rather than the all its inflected forms. One of the significant characteristics for the current implementation is that it supports broken plurals, which is not covered in most light stemmers.

III. Algorithm Overview

The proposed approach extends the El-Beltagy et al. [5] stemmer by boosting the results through adding new patterns combined with algorithm modifications, which lead to increase the stemming accuracy. The approach of El-Beltagy et al. [5] builds upon the Larkey et al. [8] stemmer, as it is probably one of the most accurate existing stemmers. El-Beltagy et al. [5] introduced support for broken plurals and infix support. The algorithm proposed by this paper implements the same procedure which removes prefixes, suffixes and tackles infixes from the input word carefully to stem the word correctly to the closest stem, and in some cases the algorithm validates the result by a stem list before returning it. However, the former works basically target nouns. Additional work is done to handle more noun patterns and some patterns of verbs that help in improving the stemming accuracy. The proposed algorithm does not make a validation for the transformed word with a local context containing the input word, unlike El-Beltagy et al. [5]. The reason for that is the algorithm is designed to be generic and context independent. On the other hand, this validation step will increase the time of the stemming processing. Thus, the stemming procedures that have to validate their results only validate it by the stem list.

The proposed algorithm is not error-free, especially for broken plurals and verbs patterns. For example, the same rule that stems "وزراء" (ministries) to "وزير" (minister) will also stem "ابناء" (sons) to "بنين" (boys). The same rule that stems "تستشير" (consult) to "استشار" (consulted) will stem "تنصير" (Evangelize) to "انصار" (supporters). Accordingly, building a clear stem list that is free of conflations and rich with common or popular stems improves the accuracy of the stemming process.

IV. Methodology

In general, the presented algorithm is divided into two phases. The first phase is building a stem list domain specific or general stem list. The second phase is the stemming phase which will be detailed later.

A. Building the Stem List

According to the evaluation results that will be stated in Section V, the best way to build the stem list is to extract the stems from documents in the domain of the application or general documents. The procedure of this phase is as follows:

1) Extract unique words from documents.
2) Get the stems of these words either manually or automatically by referring to a dictionary. The approach used an automatic procedure to get the stems of the extracted words from ElixirFM [18]. The stem of the word is the smallest form of that word, for example: if we have the word "مصابيح" (lamps) and its inflections, as elaborated in Table I.

TABLE I
Example for Word "Lamb" Inflections

مصباح	مصباحا	المصباح	مصابيح	المصابيح	

Then, the stem that will be added to the stem list is "مصباح" (lamp).

3) Create a text file containing the extracted unique stems, i.e. the stem list.

B. Stemming Phase

The procedure of this phase begins by loading the stem list file to validate with it through the stemming steps. This step is performed only once.

Rather than performing the validation process after each stemming step, it is carried out after some steps that showed it is a must to be validated according to the analysis of the stemming rules. Other steps usually result in accurate stems, and thus the validating step will not be executed. This was concluded after observing that restricting the validation after each step or stemming rule may decrease accuracy as some rules usually result in accurate stems, while some of these stems may not exist in the stem list.

The general procedure of the stemming phase is as follows and a flow chart for the same procedure is shown in Fig. 1.
Given the normalized word S:

- *1st Step*: If an S occurs in the stem list, return to S. If not remove prefixes from S and generate S'.
- *2nd Step*: If S' is in the stem list, return to S'. If not, then remove the suffixes of S' and generate S''.
- *3rd Step*: If S'' is in the stem list, return to S''. If not, then remove the verb infixes from S'' and generate S'''.
- *4th Step*: If S'' does not match S''', return to S'''.
- *5th Step*: If S'' ends with "ئ" (YEH with HAMZA above) and S'' does not match S' and the character just before "ئ" is "ا" (ALEF), then change "ئ" (YEH with HAMZA above) to "ء" (HAMZA on the line) and generate S'''. If S''' is in the stem list, return S'''. If not, then let S'' equal S'''.
- *6th Step*: If the removed suffixes of S' contains one of the plural suffixes shown in Table II then return to S''.
- *7th Step*: Remove noun infixes from S'' and generate S''' and return S'''.

TABLE II
Plural Suffixes

Plural Suffixes	ان	ين	وا	ون	اته	ات	يات	

The reason behind changing "ئ" from the end of nouns before removing infixes (i.e. 5th Step) is that some nouns end

with "ء", and the "ء" is changed to "ئ" while attaching suffixes to it. Therefore, the algorithm has to return it back to its original spelling like "اشياء - اشيائهم" and "سماء - سمائها".

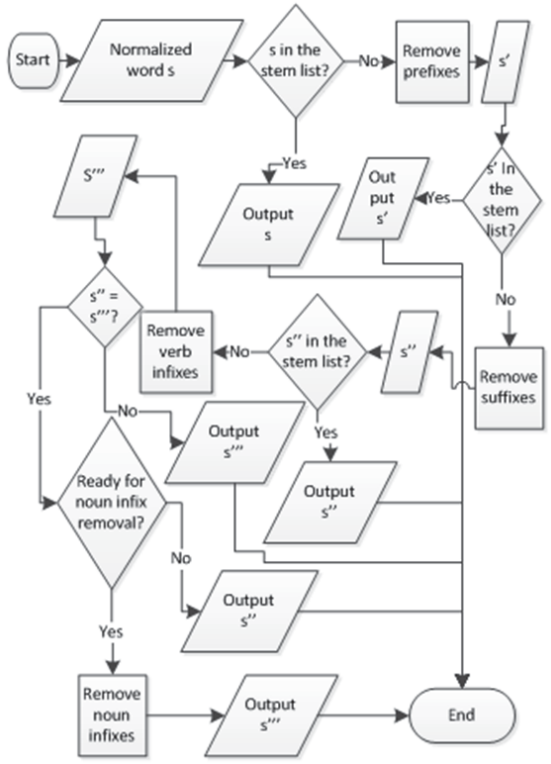

Fig. 1 Stemming procedure flow chart

C. Stemming Rules

The proposed stemming algorithm consists of three ordered levels of affix removal: Prefix, suffix and infix removal. Each of these levels has its own set of rules which will be described in following sub-sections i, ii and iii. For the removal of any affixes, the length of the resulting term should be two or more letters otherwise the affix will not be removed.

1. Prefix Removal

TABLE III
SINGLE PREFIXES AND THEIR MEANINGS

Prefix	Meaning
و	And
ك	Like
ف	Then
ل	For
ب	With/at

This algorithm uses the same procedure for prefix removal, as in [5] stemmer and removes the same prefixes. The prefixes are divided into single and compound prefixes, as shown in Tables III and IV, which are reproduced from [5].

The procedure used for the prefix removal starts with compound prefix removal. If the word remains unchanged, then single prefix removal will be executed.

TABLE IV
COMPOUND PREFIXES AND THEIR MEANINGS

Prefix	Meaning
لا	No
ال	The
وال	And the
بال	With the
كال	Like the
فال	Then the
لل	For
وبال	And with the
وكال	And like the
وفال	And then the

D. Compound Prefix Removal

There are no rules or conditions on removing compound prefixes, except for prefix "لا". Thus, if a word matches with a prefix in the compound prefix set, except prefix "لا", it is simply removed. There is no particular order in which these prefixes should be checked.

The removal of the prefix "لا" has to be validated by removing the first two letters of the word that starts with it and then checking for the resulting word in the stem list. If a match is found, the resulting word is returned. If no match is found, the first letter is removed from the original word and the validation method is called again. Once more, if a match is found, the resulting word is returned, but if no match is found, the original word is returned unaltered.

E. Single Prefix Removal

The prefix is removed and the resulting word is matched against entries in the stem list. If a match is found, the transformation is carried out safely. If not, the rest of the stemming procedure is invoked on the transformed word. If still no change occurs, then the stem is considered to be an original term.

The presented approach attempted to remove the prefix "س" (SEEN) if the word starts with "س" (SEEN) followed by "ا" (ALEF) or "ي" (YEH) or "ت" (TEH) to stem future verbs like "سيبدا" (will begin) into "يبدا" (begin), but that leads to more errors in stemming, like stemming "سيد" (mister) into "يد" (hand). Thus, this rule was not implemented. More details about prefix removal can be found in El-Beltagy et al. [5].

1. Suffix Removal

TABLE V
SUFFIXES SETS HANDLED BY THE ALGORITHM

Suffix set 1	El-Beltagy et al. [5]	يات، ات، اته
Suffix set2	El-Beltagy et al. [5]	ها، ون، وا، ين، ان، ية، يه، هم، ي، ه، ة، ا
Suffix set 3	New set	ن، نا، ت

After removing prefixes, the resulting word does not exist in the stem list, it is time to remove the suffixes. In [5], there are two sets of suffixes, while in the proposed algorithm a third set is added. All of the three sets of suffixes are shown in Table V.

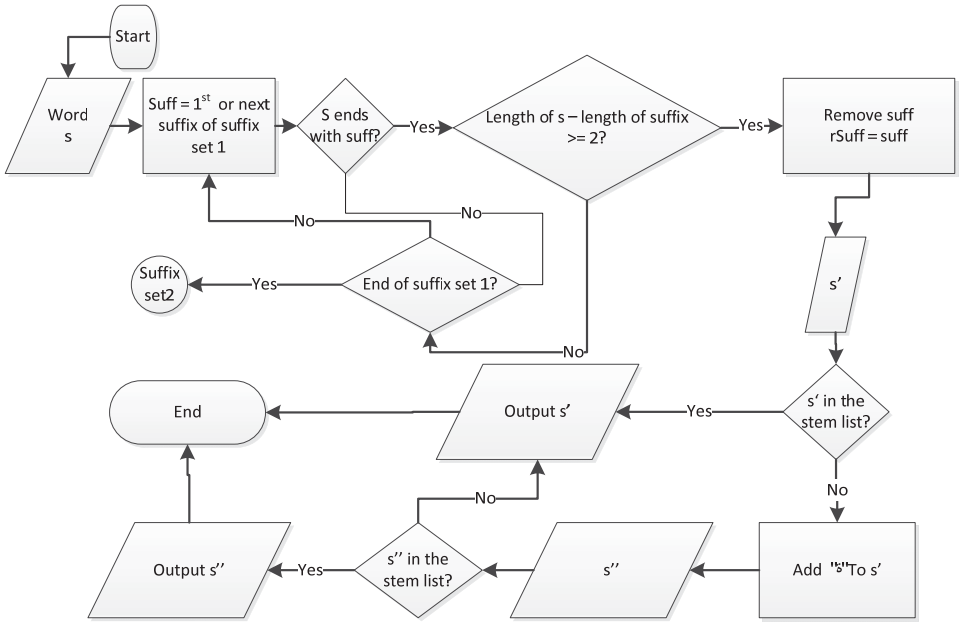

Fig. 2 Remove suffixes of suffix set 1 process flow chart

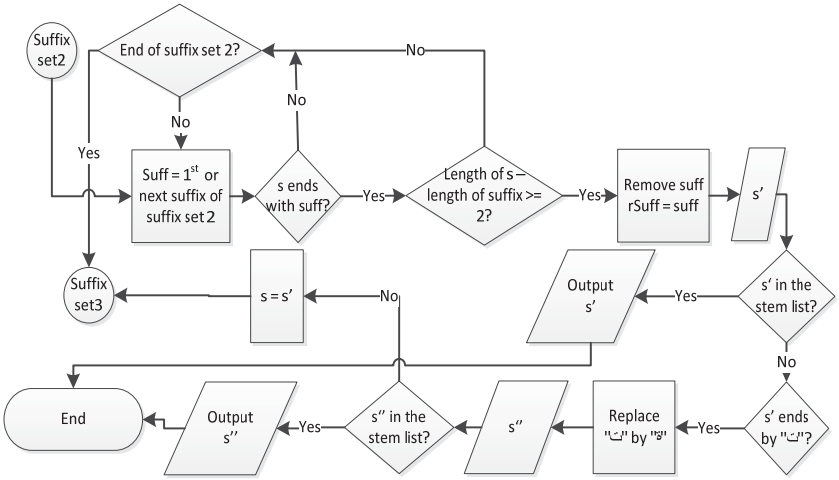

Fig. 3 Remove suffixes of suffix set 2 process flow chart

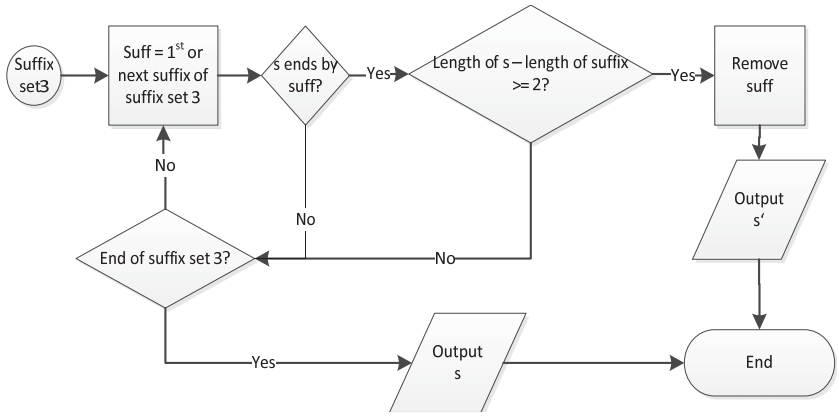

Fig. 4 Remove suffixes of suffix set 3 process flow chart

```
/*Start with the procedure for removing or replacing infixes of verbs
patterns*/
String stemInfixVerb (String S){
  If S matches P17
    Apply R17 and generate S'
    If S' in the stem list
      Return S'
    Return S
  Else
    If (prepareForNounInfix(S))
      stemInfixNoun (S) and generate S'
        Return S'
    Return S
}
/* The following procedure prepares the term S for the procedure of
*/
/*removing infixes from nouns and checks if the term S is a
transformed */
/*singular term of a plural one or not */
Boolean prepareForNounInfix (String S){
  If S ends with "ئ" and S has suffixes removed from it and 2nd char
from the end = "ا" {
    Replace "ئ" with "ء" and generate S'
    S = S'
  }
  If S isn't in the stem list {
    If S removed suffix matches one of the suffixes in the plural suffixes
set
      Return false;
    Else return true;
  }
/*the former means if S is already a stem and doesn't need more */
/*transformation like the word "اتصالات" */
/*after suffix removal will be "اتصال"*/
/* If we didn't prevent infix removal, it will returned */
/* as "تصل" because it will match R9 */
        Return false
}
/*the following procedure applies the other 15 rules for removing
infixes from nouns */
String stemInfixNoun (String S){
  Apply R4 on S and generate S'
  If S' in the stem list
    Return S'
  For each pattern P of pattern set PS (P1: P15
   in the order: P16, P3, P12, P1, P11, P6, P8, P9, P2, P13, P7, P10, P5,
  P14 and P15){
    If S matches P{
      Apply corresponding R and generate S'
      If R needs validation {
        If S' in the stem list
          Return S'
        Else Return S
      }
      Return S'
    }
  }
  Return S
}
```

Fig. 5 Infix removal procedure

For removal of any suffix in either of the suffix sets, the length of the resulting term must be at least two characters in length. If this condition is not met, the input term is returned as-is. If an input term ends with any of the suffixes in suffix set 1, the suffix is removed and validation is carried out. If a match is found, the resulting word is returned. If no match is found, the character "ة" (TEH_MARBUTA) is added to the resulting word and validation is carried out again. In case a match is found, the resulting word is returned. If no match is found in the proposed algorithm, unlike in El-Beltagy *et al.* [5], the resulting word after removing the suffix and before adding "ة" is returned. The reason for that was that this rule provided better accuracy in the results. If the input term does not end with any of the suffixes in suffix set 1, t is checked against each one of the suffixes in suffix set 2 in the order as shown in Table IV from right to left. If the word ends with the suffix, the suffix is removed and the resulting word is validated. If a match is found, it is returned. Otherwise, a check is made to see whether the resulting word ends with a "ت" (TEH). If it does, the "ت" is replaced with a "ة" and the resulting word is validated again. If the generated term match is found, it is returned as an output from this step. Otherwise, the term resulting from removing the suffix of suffix set 2 is checked for each of the suffixes in suffix set 3. If the word ends with the suffix, the suffix is removed and the resulting word is returned. If the input term does not end with any of the suffixes in suffix set 2. It is checked for each of the suffixes in suffix set 3. If the word ends with these suffix sets, the suffix is removed and the resulting word is returned. The flow chart shown in Figs. 2-4 clarifies the process of suffix removal.

In the step of checking the input term for each of the suffixes in suffix set 2, It was observed that some patterns should not continue the procedure of suffix removal, as these patterns are needed unaltered to be an input for the infix removal procedure to be stemmed correctly. The exceptional patterns are shown in Table VI.

TABLE VI
EXCEPTIONAL PATTERNS OF SUFFIX REMOVING

Matching suffix	Pattern	Example
ي	1st char = "ا" or "ي" or "ت"	ينبغي، اؤدي، تلاقي
ه	1st char = "ا" or 3rd char from the end = "ا"	بدايه، سياده، اتربه، اشاره
/	Last char = "ا" and 2nd char from the end = "ي" and 3rd char from the end = "ا" and 2nd char != "و"	ضحايا، هدايا، منايا

In the procedure of removing the suffixes of suffix set 3, there is also an exceptional case. The suffix "ن" should not be removed unless the suffix "ا" in suffix set 2 is removed from the input word to the stem, for example "ابلغنا، اتفقنا" into " ابلغ، اتفق".

2. Infix Removal

After suffix removal, infix removal takes place. Infixes usually occur as a part of broken or irregular plurals. Some of the commonly used broken plurals exhibit well-defined patterns that can be detected and transformed. Examples of patterns detected and handled by the developed stemming algorithm are shown in Table VII.

The first 11 patterns (P1 to P11) are the same as [5]. The 12th pattern, P12, is examined by [19], where they suggested four transformations for that pattern; this approach adds another new suggestion to them. The last five patterns (P13 to P17) and their rules are new patterns suggested by this algorithm.

TABLE VII
EXAMPLES OF PATTERNS HANDLED BY THE ALGORITHM

Pattern code	Examples	Stems
P1	جشائش، قصائد، دلائل	حشيشة،قصيدة، دليل
P2	روائح، فوائد، قوئم، سوائل	رائحة، فائدة، قائمة، سائل
P3	هدايا، وصايا، ضحايا	هدية، وصية، ضحية
P4	دول، نعام، اجاص	دولة، نعامة، اجاصة
P5	بذور، جذور، سطور، دروس	بذرة، جذر، سطر، درس
P6	مراكب، مدارس	مركب، مدرسة
P7	اشهر، احرف	شهر، حرف
P8	اجهزة، اتربة	جهاز، تراب
P9	اشجار، امراض، اقوال	شجر، مرض، قول
P10	سدود، خطوط، حدود	سد، خط، حد
P11	جوانب، عوائل، مواشي	جانب، عائل، ماشية
P12	**خبراء، اطباء، عقلاء، جبناء، سمحاء**	**خبير، طبيب، عاقل، جبان، سمح**
P13	**كلاب، سلال، ظلال**	**كلب، سلة، ظل**
P14	**سجان، رسام**	**ساجن، راسم**
P15	**تفاصيل، تقارير، اسابيع**	**تفصيل، تقرير، اسبوع**
P16	**امم، خطط، قطط**	**امة، خطة، قطة**
P17	**يري، يؤثر، يقترب، يشير، يسبعد**	**اري، اوثر، اقترب، اشار، استبعد**

For each of the previous patterns, a transformation rule is defined to transform that pattern to its stem. Some of these rules results are returned only if validated; otherwise, the input term is returned without transformation. And the other rules results are returned either validated or not, as correctness is proved their most of the time. All of these rules are outlined in Table VIII.

Through the analysis of infix stemming, it was noticed that the order in which the rules are applied affects the accuracy of the returned stems. Therefore, the procedure of infix removal was ordered, as shown in the pseudo code in Fig. 5 for the Infix removal procedure.

V. EVALUATION

The evaluation of the proposed stemming algorithm is done through stemming different data sets, using different stem lists and calculating the accuracy, precision, recall and F-score. To calculate these terms, the following equations are used:

$$Accuracy = \frac{no.of\ correct\ stems}{total\ no.of\ words\ to\ be\ stemmed} \quad (1)$$

$$Precision = \frac{no.of\ correctly\ stemmed\ words}{total\ no.of\ transformed\ words} \quad (2)$$

$$Recall = \frac{no.of\ correctly\ stemmed\ words}{total\ no.of\ words\ that\ should\ have\ been\ transformed} \quad (3)$$

$$F - score = \frac{2*Precision*Recall}{Precision+Recall} \quad (4)$$

ElixirFM dictionary was used to get the correct stem of any word by building a stem list. The list contains correct stems for the words that should be stemmed by the proposed algorithm to compare with and calculate the accuracy. As mentioned earlier, the stem of the word is the smallest form of the word where this word is mentioned. But some words are mentioned in different conflation forms. In this case, we took the smallest form of the word in each line where this word is mentioned and the output of the proposed algorithm is correct if it matches one of these stems. For example, for the word "اتفاقيات", it occurs in these two inflections forms as seen in Tables IX and X.

The output of the proposed algorithm is correct if it matches either "اتفاقي" or "اتفاقية".

In the evaluation process for each dataset, a comparison of the accuracy of the proposed algorithm versus the accuracy of the algorithm of El-Beltagy et al. [5] was made. Also, adding into the comparison the accuracy of [8] stemmer. The proposed algorithm resulted in better accuracy than both of these prior works. These results are detailed one by one in the following sub-sections A, B, C and D. One drawback that was noticed is that the proposed algorithm takes more execution time compare to prior works, but this expected as more rules and patterns are applied.

A. ElixirFM [3] Dataset

In this case, the dataset and stem list are extracted randomly from Elixir. The stem list is based on the smallest inflection form of the word. The stem list was created with different sizes of 5,000 words, 6,000 words and 10,000 words to conduct the evaluation. The testing data trials were extracted with different sizes from 1,000 words to 56,000 words. These datasets and stem lists helped in detecting more patterns. It also helped in improving the accuracy of the proposed algorithm. The evaluation results of these datasets are shown in Table XI.

The dataset of ElixirFm is rich in different words and patterns, but that does not mean that all of these patterns will be met by the stemmers frequently, as there are many words in the dictionary that are not used in real conversations or articles. Therefore, another evaluation was conducted on other types of documents as elaborated in the next subsections B, C and D.

B. Arabic Wikipedia Documents Dataset

In this experiment a set of random Arabic Wikipedia documents were collected. Approximately 2,500 unique words were extracted and normalized from the set. The equivalent stems were extracted from ElixirFM. In addition, stem list A of 4,758 unique stems was extracted from other random Arabic Wikipedia documents. Again the experiment included stemming the same data using the Larkey et al. [8] stemmer, and the algorithm of El-Beltagy et al. [5] and the proposed algorithm. The resulted accuracy, precision and recall of were calculated and recorded in Tables XII and XIII. The accuracy resulting from [8] stemmer is 59.4%.

TABLE VIII
PATTERN DETECTION AND TRANSFORMATION RULES

Rule ID	Condition	Rule	Example
R1	Length = 5 and 2nd char != "و" and 4th char = "ئ" and 3rd char = "ا" and 5th char != "ي" or "ه"	Replace 3rd char with "ي", delete 4th char. If no match is found, add "ة" to the end and return the result.	حشائش
R2	Length = 5 and 2nd char = "و" and 4th char = "ئ" and 3rd char = "ا" and 5th char != "ي" or "ه"	Delete 2nd char. If no match is found, add "ة" to the end and return the result.	روائح
R3	Length > 3 and last char = "ا" and 2nd char from the end = "ي" and 3rd char from the end = "ا" and 2nd char != "و"	Replace the last char with "ة" and delete 3rd char from the end and return the result.	ضحايا
R4	No conditions. *In El-Beltagy and Rafea paper the condition is length =3*	Add "ة" to the end of the word. If no match is found replace "ة" with "ه", if no match is found return the input term as is.	نعام
R5	Length = 4 and 3rd char = "و" and 4th char != "ه" or "ا" or "ي" or "ء" and 1st char != "ي" or "و" or "ا" or "ت" *The condition of not ending with "ء" is added by the proposed algorithm not to transform words like "لجوء" and "هدوء"*	Delete 3rd char. If no match is found, add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, return the transformed word before adding "ة".	جذور
R6	Length = 5 and 3rd char = "ا" and 1st char != "ت" and 5th char != "ه" *The condition of not ending with "ه" is added by the proposed algorithm not to transform words like "اشاره" and "سياده"*	Delete 3rd char and add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, return the transformed word before adding "ة".	مراكب
R7	Length = 4 and 1st char = "ا"	Delete 1st character. If no match is found, add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, return the input word as is.	اشهر
R8	Length = 5 and 1st char = "ا" and 5th char = "ه" and 3rd char != "ا" *The condition 3rd char != "ا" is added by the proposed algorithm not to transform words like "اشاره"*	If 3rd character = "ئ" replace it with "ؤ", -the following transformations are applied on the input term length of five characters, either the 3rd character = "ئ" or not - Replace the 5th character with the 4th character, replace the 4th character with "ا", delete the 1st character and return the result. *Changing "ئ" to "ؤ" is added by the proposed algorithm to transform words like "اسئلة" and "افئدة" correctly to "سؤال" and "فؤاد" not to "سئال" and "فئاد"*	اتربة
R9	Length =5 and 1st char = "ا" and 4th char = "ا" and 2nd char != "ي"	Delete the 4th character and the 1st character and add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, return the transformed word before adding "ة".	اشجار
R10	Length = 4 and 3rd char = "و" and 2nd char = 4th char	Delete the 4th character and the 3rd character and add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, return the transformed word before adding "ة".	سدود
R11	Length = 5 and 3rd char = "ا" and 2nd char = "و" and 1st char != "ت" or "ي" or "ا" and last char != "ه" The condition of not starting with "ا" or "ت" or "ي" is added by the proposed algorithm not to transform verbs like "انوازي" "اواعد", and "يواري". Also the condition of not ending with "ه" not to transform words like "شوايه"	Delete the 3rd character "ا". If no match is found, add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, delete the 2nd character of the input term before the transformation "و". If no match is found, add "ة" and return the result. The 1st trial of deleting the 3rd character and adding "ة" is added by the proposed algorithm to transform words like "طوارئ" and "قواعد" correctly to "طارئ" and "قاعدة" -if they are in the stem list- not to "طورئة" and "قوعدة".	جوانب
R12	Length = 5 and 4th char = "ا" and 5th char = "ء"	If 1st character = "ا", replace the 5th character with the 3rd character, replace the 4th character with "ي" and delete the 1st character. If no match is found, -the following transformations are applied on the input term of length five characters either starts with "ا" or not- replace the 4th character with the 3rd character and replace the 3rd character with "ي" and delete the 5th character. If no match is found, replace the 3rd character with "ا". If no match is found, replace the 3rd character with the 2nd character and replace the 1st character with "ا". If no match is found, delete the 2nd character. If no match is found return the input term as is.	**اطباء**
R13	Length = 4 and 2nd char = "ل" and 3rd char = "ا"	Delete the 3rd character. If no match is found and the 3rd character = the 2nd character, delete the 3rd character. If no match is found, add "ة"to the end. If no match is found, return the input term as is.	**كلاب**
R14	Length = 4 and 3rd char = "ا"	Replace the 3rd character with the 2nd character and replace the 2nd character with "ا". If no match is found, return the input term as is.	**رسام**
R15	Length = 6 and 3rd char = "ا" and 5th char = "ي"	If the 2nd character = "و", delete 2nd character and replace 4th character with "و". If no match is found, add "ة" to the end. If no match is found, replace "ة" with "ه". If no match is found, -the following transformations are applied on the input term of length six characters, either the 2nd character = "و" or not- delete the 3rd character. If no match is found, replace the 4th character with "ا". If no match is found, replace the 4th character with "و". If no match is found, replace the 4th character with "ي". If no match is found, return the input term as is.	**تفاصيل**
R16	Length = 3 and 3rd char = 2nd char	Replace the 3rd character with "ة". If no match is found, replace the 3rd character with "ه". If no match is found, return the input term as is.	**امم**
R17	Length is greater than or equal to 3 or less than 6 or length = 6 and 3rd char != "ا" and last char != "ه" and 1st char = "ي" or "ت"	Replace the 1st character with "ا". If no match is found and the 2nd character from the end = "ي", replace "ي" with "ا". If no match is found return the input term as is.	**يستعيد**

(R1 to R11 are the same rules of El-Beltagy et al. [5] with some modifications while R12 to R17 are new boosting rules)

TABLE IX
INFLECTIONS FORM FOR THE WORD اتفاقي

اتفاقي	**اتفاقي**	اتفاقيا	الاتفاقي	اتفاقيان	**اتفاقيات**	الاتفاقيات

TABLE X
INFLECTIONS FORM FOR THE WORD اتفاقية

االتفاقية	**اتفاقية**	الاتفاقية	اتفاقيتان	اتفاقيتا	**اتفاقيات**	الاتفاقيات

TABLE XI
EVALUATION RESULTS FOR ELIXIRFM DATASET

#of input words	Larkey *et al.* [8] Acc (%)	Time in ms	El-Beltagy [5] Acc (%)	Time in ms	The proposed algorithm Acc (%)	Time in ms	The proposed algorithm Acc (%)	Time in ms	The proposed algorithm Acc (%)	Time in ms
			stem list: 5000		stem list: 5000		stem list: 6000		stem list: 10000	
1000	35.4	6	53.1	0	67.7	16	70.6	15	70.6	16
2000	37.25	1	57.15	6	67.3	15	69.25	15	69.2	16
3000	33.9	3	58	11	66.83	16	67.97	16	68.6	16
4000	35.23	3	56.78	11	68.18	15	69.28	15	69.13	0
5000	35.3	0	56.34	9	66.76	16	67.72	15	67.8	16
6000	37.58	2	57.85	11	68.63	15	69.48	16	69.65	15
7000	34.87	1	55.41	9	67.66	16	68.16	16	68.06	15
8000	35.6	2	56.26	19	68.5	16	67.99	36	68.04	31
9000	35.1	4	55.77	14	68.61	31	68.22	16	68.28	16
10000	34.55	1	56.14	15	68.54	16	68.29	31	68.32	31
12000	35.28	1	55.61	15	67.98	47	67.73	31	67.89	62
14000	36.79	2	56.23	16	69.3	31	68.9	31	68.89	31
16000	35.13	2	54.99	31	68.33	32	68.19	33	68.05	31
18000	34.37	2	53.69	32	67.19	31	66.86	31	66.67	31
20000	35.23	0	54.54	47	68.42	31	68.22	31	68.12	31
23000	28.4	2	31.57	38	48.45	110	48.76	113	50.32	110
26000	37.03	5	56.19	37	68.82	46	68.72	47	69.09	63
29000	34.68	9	53.01	42	67.78	63	67.58	67	67.41	62
32000	36.43	4	55.06	43	68.53	62	68.5	63	68.71	65
36000	35.59	5	53.6	60	68.19	63	68.28	79	68.13	62
40000	38.09	7	52.64	64	68.97	93	69.49	94	70.19	79
45000	36.73	8	55.33	63	68.86	95	68.9	95	68.8	94
50000	35.76	15	53.08	70	67.68	94	67.64	93	67.77	95
56000	36.03	6	53.53	75	67.7	110	67.68	112	67.71	111
19666.67	35.43	3.79	54.24	30.75	67.29	45.00	67.60	46.29	67.73	45.79

(The last row contains the average values for the number of input words, accuracy and time)

C. Arabic Wikipedia Documents and Articles from Arabic Newspapers Dataset

In this experiment, a new set of words collected from Arabic newspapers [20], in addition to the previous set of input words from Wikipedia. Stem list A was used. The total number of input words was 5,400 unique words. The results were recorded in Tables XII and XIII, while the Larkey et al. [8] stemmer resulted in an accuracy of 54.1%.

For the same dataset, stem list B of 4,990 most used words of modern standard Arabic was collected from the web. The list of words was evaluated and found to be very common. It is noticed that the accuracy of Larkey et al. [8] did not change as it does not use a stem list, while the other two algorithms had different results, as shown in Tables XII and XIII.

D. Arabic Newspapers Articles Dataset

Using stem list B a new dataset collected from 60 Arabic newspapers articles of six different categories (4,689 unique words) [21]. After testing the calculated accuracy, precision and recall for El-Beltagy et al. [5] and the proposed algorithm, were reported in Tables XII and XIII, while the Larkey et al. [8] stemmer reported an accuracy of 53.657497%.

Finally, for the same dataset, stem list C was built containing stems of 5,000 of the most used words in Arabic subtitles [22] and used this stem list in stemming the input words. The Larkey et al. [8] stemmer accuracy did not change, as it does not depend on the stem list, while the other two algorithms recorded the accuracy, precision and recall shown in Tables XII and XIII.

TABLE XII
EVALUATION RESULTS FOR THE EL-BELTAGY ET AL. [5] ALGORITHM

Data set	B	C		D		Average
Stem list	A	A	B	B	C	
Accuracy	76.12%	67.8%	66.22%	65.57%	63.4%	67.83%
Precision	79.5%	72.91%	69.76%	70.11%	66.76%	71.81%
Recall	47.61%	43.84%	43.39%	43.62%	41.22%	43.94%
F-score	59.56%	54.75%	53.50%	53.78%	50.97%	54.52%

TABLE XIII
EVALUATION RESULTS FOR THE PROPOSED ALGORITHM

Data set	A	B		C		Average
Stem list	A	A	B	B	C	
Accuracy	84.6%	78.9%	77.39%	76.9%	75.68%	78.7%
Precision	82.61%	77.54%	75.31%	75.61%	74.07%	77.03%
Recall	59.53%	58.71%	59.21%	58.93%	57.1%	58.7%
F-score	69.2%	66.82%	66.29%	66.23%	64.49%	66.61%

Both of the proposed algorithm and the algorithm of El-Beltagy et al. [5] resulted in better accuracy when used with a domain specific stem list.

VI. Conclusion and Future Work

This work presented a new extension for Light stem in order to improve the Arabic IR system. The algorithm adequately utilized prior work done in this domain. The approach enhanced the algorithm and rules that governs the methodology for stem extraction. These enhancements included detecting new Arabic word patterns and built it on top of the previous works of El-Beltagy et al. [5] and Ababneh et al. [19]. The procedure was boosted for finding the stem and building a stem list from several resources. Comparing this work with former works, it has proven to achieve significant results for enhancing stem accuracy by an average 10%. The results were presented for each testing set. Since this work is intended to be part of an open source implementation, the factor of execution speed was recorded. It was noticed how the execution will take more time proportionally with the number of words undertaken by the stemming our algorithm. This is a drawback that can be tackled in future work. One of the approaches to tackle is issue is in having a massive amount of words to stem; however, in the massive parallel processing domain, different architecture and tools can help in maintaining a proper speed while still excelling in stem accuracy. The motivation for this work originated from the need to empower the open source community through providing effective and practical implementation of the algorithm for Arabic stemming. This target was successfully achieved by providing better search results compared to similar approaches.

References

[1] Tinsley, Teresa, and Kathryn Board. "Languages for the future: Which languages the UK needs most and why." British Council (2013).
[2] http://www.internetworldstats.com/stats7.htm.(accessed August 1, 2016)
[3] http://fortune.com/2015/07/14/ibm-watson-home-middle-east/ (accessed August 1, 2016)
[4] Larkey, Leah S., Lisa Ballesteros, and Margaret E. Connell. "Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis." Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. ACM,
[5] El-Beltagy, Samhaa R., and Ahmed Rafea. "An accuracy-enhanced light stemmer for arabic text." ACM Transactions on Speech and Language Processing (TSLP) 7.2 (2011): 2.
[6] Ding, Wei, and Gary Marchionini. "A study on video browsing strategies." (1998).
[7] Porter, Martin F. "An algorithm for suffix stripping." Program 14.3 (1980): 130-137.
[8] Larkey, Leah S., Lisa Ballesteros, and Margaret E. Connell. "Light stemming for Arabic information retrieval." Arabic computational morphology. Springer Netherlands, 2007. 221-243.
[9] El-Sadany, Tarek A., and Mohamed A. Hashish. "An Arabic morphological system." IBM Systems Journal 28.4 (1989): 600-612.
[10] Mustafa, Suleiman H. "Word stemming for Arabic information retrieval: The case for simple light stemming." Abhath Al-Yarmouk: Science & Engineering Series 21.1 (2012): 2012.
[11] Kadri, Youssef, and Jian-Yun Nie. "Effective stemming for Arabic information retrieval." The Challenge of Arabic for NLP/MT, Intl Conf. at the BCS. 2006.
[12] Al-Omari, Asma, and Belal Abuata. "Arabic light stemmer (ARS)." Journal of Engineering Science and Technology 9.6 (2014): 702-717.
[13] Khoja, Shereen, and Roger Garside. "Stemming arabic text." Lancaster, UK, Computing Department, Lancaster University (1999).
[14] Harmanani, Haidar M., Walid Keirouz, and Saeed Raheel. "A Rule-Based Extensible Stemmer for Information Retrieval with Application to Arabic." Int. Arab J. Inf. Technol. 3.3 (2006): 265-272.
[15] Al-Shammari, Eiman Tamah, and Jessica Lin. "Towards an error-free Arabic stemming." Proceedings of the 2nd ACM workshop on Improving non english web searching. ACM, 2008.
[16] El-Defrawy, Mahmoud, Yasser El-Sonbaty, and Nahla A. Belal. "Enhancing Root Extractors Using Light Stemmers." (2015).
[17] Dahab, Mohamed Y., Al Ibrahim, and Rihab Al-Mutawa. "A Comparative Study on Arabic Stemmers." International Journal of Computer Applications 125.8 (2015).
[18] http://quest.ms.mff.cuni.cz/cgi-bin/elixir/index.fcgi.(accessed May 15, 2016)
[19] Ababneh, Mohamad, et al.. "Building an Effective Rule-Based Light Stemmer for Arabic Language to Improve Search Effectiveness." International Arab Journal of Information Technology (IAJIT) 9.4 (2012).
[20] http://arabic-media.com/egypt-news.htm. (accessed July 12, 2016)
[21] https://sourceforge.net/projects/arabiccorpus/files/watan-2004corpus/. (accessed July 17, 2016)
[22] https://invokeit.wordpress.com/FREQUENCY-WORD-LISTS/ (accessed July 20, 2016)