Journal of King Saud University – Computer and Information Sciences 29 (2017) 164–170

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Enhancing Arabic stemming process using resources and benchmarking tools

Younes Jaafar [a,*], Driss Namly [a], Karim Bouzoubaa [a], Abdellah Yousfi [b]

[a] Mohammadia School of Engineers, Mohammed Vth University – Rabat, Morocco
[b] FSJES, Mohammed Vth University – Rabat, Morocco

ARTICLE INFO

Article history:
Received 16 April 2016
Revised 4 November 2016
Accepted 21 November 2016
Available online 2 December 2016

Keywords:
Arabic stemming
Evaluation
Benchmark
Evaluation corpus

ABSTRACT

Many approaches and solutions have been proposed for developing Arabic light stemmers. These stemmers are often used in the context of application-oriented projects, especially when it comes to developing information retrieval (IR) systems. However, Arabic light stemming, as the process of stripping off a set of prefixes and/or suffixes, is a blinded task suffering from problems such as incorrect removal, vocalization ambiguity, single solution, etc. Moreover, each researcher claims that his/her stemmer reached a level of strength and accuracy quite high. However, in most cases, these stemmers are black boxes and it is not possible to access neither their source codes to verify their validity, nor the evaluation corpora that were used to claim such accuracy. Since these stemmers are very important for researchers, their comparison and evaluation is then essential to facilitate the choice of the stemmer to use in a given project. In this paper, we propose a new Arabic stemmer that gives solutions to the above mentioned drawbacks. In addition, we propose an automatic approach for the evaluation and comparison of Arabic stemmers that takes into account metrics related to the accuracy of results as well as the execution time of stemmers.

1. Introduction

Stemmers are basic tools used for many tasks that require text preprocessing, such as text categorization systems, text summarization systems, information extraction systems, etc. The stemming process includes the identification and the removal of affixes from a derived or inflected words and reducing them to their stems/roots. Different stemming approaches have been proposed for many languages including English, French, Turkish, and Chinese. Concerning the Arabic Language, there are two main stemming approaches (Otair, 2013): the root-based approach and the light stemming approach. Arabic is one of the Semitic languages, that differs from English, French, German, etc. Therefore, some Arabic stemmers reduce Arabic words to their roots instead of their stems (Al-Kabi and Al-Mustafa, 2006). In this article, we propose a third stemming approach that uses deeper validation of stems using lexicon resources.

The stemming process is important for researchers since it brings together words based on their lexico-semantic similarity. For example the words: "كتب" (he wrote), "كتبوا" (they wrote), "سيكتب" (he will write), "أكتبتم" (have you written?) have the same lexico-semantic content as "كتب" (he wrote) which leads to "the concept of writing". Thus, instead of dealing with four words, Arabic Natural Language Processing (ANLP) systems can handle one single word after reducing the list of words to the same stem. Therefore, queries or documents in IR systems can be represented using stems or roots rather than using the full original words. This operation reduces enormously the size of indexes of IR systems, which leads to a gain of space storage and processing time.

However, Arabic light stemming as the process of stripping off a set of prefixes and/or suffixes, is a "blinded" task suffering from problems such as:

- Incorrect removal: words starting with a string similar to a prefix, or ending with a string similar to a suffix will be truncated by mistake. For example, the analysis of the word "والده" ("his

* Corresponding author.
E-mail addresses: jayounes@yahoo.fr (Y. Jaafar), namly_driss@yahoo.fr (D. Namly), karim.bouzoubaa@emi.ac.ma (K. Bouzoubaa), yousfi240ma@yahoo.fr (A. Yousfi).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2016.11.010
1319-1578/

father") with the Light10 stemmer (Larkey et al., 2007) gives the stem "د" considering "وال" as a prefix and "ه" as a suffix, whereas removing "وال" is an incorrect choice because it is a part of the stem.

- Vocalization ambiguity: removal of diacritic marks from the stemming output could lead to an ambiguous meaning of words. For example, analysis of the word "فكتابه" gives the stem "كتاب" considering "ف" as a prefix and "ه" as a suffix, but the stem "كتاب" allows several alternatives such as "كُتّاب" (school or authors), "كتَاب" (book), ... etc.
- Single solution: most of the available Arabic stemmers provide one solution in the stemming output, but according to Arabic language morphology, a word admits one or more different stems. For instance, the word "لهم" (for them) must return: the verb "لَهَمَ" (greed), the noun "لَهِمَ" (glutton), the verb "هَمَ" (interested) and the empty stem for the combination of the prefix "لَ" (la) and the suffix "هُمْ" (they).

Moreover, it is important for researchers to make an optimal choice when choosing a stemmer in the context of a larger project. To help researchers making this choice, it is essential to propose tools and approaches to evaluate and compare Arabic stemmers. The literature shows that researchers classify metrics for evaluating stemmers into two categories: (1) metrics related to the "strength" which describe changes made to words in order to produce stems, i.e. stronger stemmers are intended to make more changes to words to produce stems by removing characters, (2) and those related to the accuracy which describe how much these stems are correct. However, the Arabic content in the digital world has become so large that is difficult to neglect execution time in running text processing software. To our knowledge, there is no research that takes into account execution time for evaluating Arabic stemmers.

Therefore, our objective in this paper is twofold:

- Propose a new Arabic stemmer: SAFAR-Stemmer. This new stemmer is a stem-based one with a stem validation process using a lexical resource. SAFAR-Stemmer gives answers to the above cited drawbacks through the "multi-solution" concept. By offering multiple possible stems, it resolves the three aforementioned troubles. First, to correct the "Incorrect removal" deficiency, SAFAR-Stemmer comes up with a stems collection including all possible alternatives. Secondly, to satisfy the "Vocalization ambiguity" SAFAR-Stemmer provides a diacritized output. Thirdly, it offers several possible solutions according to the stemmed word composition in compliance with the morphological particularities guiding affixes agglutination, which resolves the "Single solution" problem. It should be noted that in stemming coherent texts, a word can always be assigned a unique stem, as the context provides the clues needed for disambiguation. However, there are many other cases where researchers need to stem words out of their contexts. That's why we believe that an Arabic stemmer should return all possible stems for a given word. Let us also mention that both of these two aspects (multiple solutions and vocalization) are not taken into consideration while evaluating stemmers in this article. This is because all other stemmers do not provide this information and it will be not fair to perform a benchmark in this case. That is to say, in this special case of evaluation, there is no added value if a stemmer returns one or multiple solutions. The evaluation is performed based only on the common form of output of all stemmers.
- Present a new reusable and generic solution to evaluate and compare Arabic stemmers. This is achieved using an evaluation corpus dedicated specifically to this purpose. We propose also a new metric of evaluation that combines metrics related to the accuracy of stemmers as well as their execution time. This new metric will allow researchers to make the optimal choice even if the metrics returned by stemmers are disproportionate. To give a concrete example of our evaluation, we selected three light stemmers namely: Light10 (Larkey et al., 2007), Motaz stemmer (Saad and Ashour, 2010), Tashaphyne (Zerrouki, 2016) in order to be compared with our new stemmer (SAFAR-Stemmer). It should be noted that our benchmarking solution can also handle root-based stemmers' benchmark.

The rest of this paper is organized as follows. The next section presents some stemming approaches and algorithms. In Section 3, we present our approach for the new Arabic stemmer. In Section 4, we present some works that deal with evaluating and benchmarking Arabic stemmers. We present also the evaluation corpus and some common metrics. Then we present our new metric for evaluating stemmers. Experiments and results are presented in Section 5. Finally, we present the conclusion and future works in Section 6.

2. Related works

In this article, we focus on Arabic light stemmers rather than root-based ones. Indeed, researches have shown that light stemmers give better results comparing to root-based approaches (Larkey et al., 2002). Therefore, it would be more appropriate to focus on more promising approaches.

Several Arabic light stemmer approaches and algorithms have been already proposed. They consist of removing the most common affixes from words and producing stems. Below are some examples of Arabic light stemmers.

Larkey et al. (2007) proposed several Arabic light stemmers and assessed their effectiveness for information retrieval using standard TREC data. The light stemmer, Light10, outperformed the other approaches. It has been widely used in Arabic information retrieval (Larkey et al., 2007).

Aljlayl and Frieder (2002) studied the stemming impact in improving Arabic information retrieval systems. For this, they have proposed two stemmers: a root algorithm based on the work of Khoja and a light stemming (LS) algorithm. Authors affirm that the LS algorithm significantly outperforms the root algorithm in IR. However, they do not provide evaluations for the two stemmers themselves.

Chen and Gey (2002) proposed also two Arabic stemmers for information retrieval: a Machine Translation (MT) based stemmer and a light stemmer. The test shows that the light stemmer performed better than the MT based stemmer in IR, but no evaluations were made to compare the two stemmers in terms of accuracy of their stemming results.

Rogati et al. (2003) presents an unsupervised learning approach for building an Arabic light stemmer. Authors compare results of their stemmer to GOLD which is a proprietary Arabic stemmer built using rules, affix lists and human annotated text. They claim their approach results in 87.5% agreement with GOLD.

Saad and Ashour (2010) proposed a light Arabic stemming algorithm to address the impact of text preprocessing on Arabic text classification. The system was integrated into WEKA (Hall et al., 2009) and RapidMiner (Hofmann and Klinkenberg, 2013) platforms.

We have selected three light stemmers: Light10, Motaz stemmer and Tashaphyne in order to compare their results with our stemmer and give a concrete example of use of our benchmarking system. It should be noted that we have focused in this article only on stemmers and not on morphological analyzers (benchmarking Arabic Morphological Analyzers has been done elsewhere (Jaafar

et al., 2016). Moreover, we have selected stemmers that are free in order to integrate them within SAFAR framework. In the literature, other papers Al-Kabi et al. (2011), Majdi and Atwell (2008), Al-Shawakfa et al. (2010) were found but presented only in terms of approaches but no corresponding program to download is proposed.

3. Presentation of our Arabic stemmer

To gain more flexibility while developing our SAFAR-Stemmer, we decided to integrate it within the SAFAR (Software Architecture for Arabic language pRocessing) framework (Souteh and Bouzoubaa, 2011; Jaafar and Bouzoubaa, 2015) which is an integrated framework dedicated to ANLP. SAFAR has several layers: (1) the utilities layer includes a set of technical services, (2) the resources layer provides services for consulting language resources such as lexica, (3) the basic layer contains the three regular layers (morphology, syntax and semantics), (4) the application layer contains high-level applications that use the layers listed above, (5) and finally the client applications layer which interacts with all other layers providing users with web applications, web services, etc. Several articles about SAFAR and its different layers can be found in SAFAR website.[1] It should be noted that SAFAR-Stemmer is a new Arabic stemmer that we have included within the morphology layer of the SAFAR framework.

In the analysis of a word, SAFAR-Stemmer generates all likely possible combinations of clitics through querying the "clitics" API of SAFAR framework that returns a list of proclitic-enclitic couples. The list of clitics obtained is filtered in a second step by removing duplicates and invalid combinations. Then, the list of likely stems is generated. For instance, the word 'فقيد' has two likely stems. The first stem is 'قَيَدَ' "to tie" for which the first char in the word 'ف' is a likely proclitic. The second stem is 'فَقِيد' "missing" in which the first char in the word 'ف' is part of the stem. After this step, the role of the lexical resource "lexicon of stems" comes into play, which acts as a validator of obtained stems list to give two sub lists: list of checked stems "found in the lexicon" and list of non checked stems "not found in the lexicon". These steps are summarized in the example illustrated in Fig. 1.

With respect to SAFAR layered architecture Jaafar and Bouzoubaa (2015) composed of the Resources Services, Basic Services and Applications, the SAFAR-Stemmer belongs to the Basic Services layer and its process is made up out of four steps as explained in the figure above. The end user queries the service exploiting a dedicated application. For illustration purpose, we assume the user types "فكتابه" (and his book) as a stemming input. In the first step, the stemmer extracts from the clitics resource layer a list of all likely possible combinations of clitics ([–] "the empty clitic", [ف - هُ] "the conjunction 'then' and the pronoun 'his' ", [-فَك] "the conjunction "then" and the "kaf" of comparison", etc), then the stemmer as part of the Basic layer, filters this list of clitics in the second step by removing doubling and diacritics ([–] [-فك - ه] [فك-] [ف - ه] [- ه] [ف]) and extracts a list of likely stems in the third step (فكتابه "and his book", كتابه "his book", تابه "tAbh", كتاب "a book", تاب "forsake", فكتاب "and a book"). This is the particular feature offered by SAFAR-Stemmer and neglected by other stemmers that provide only a single solution. The fourth step in the SAFAR-Stemmer calls again the resource services layer to check the existence of obtained stems in the stems lexicon, to finally provide the application layer with a list of checked stems (كتاب "a book" and تاب "forsake") and a list of non checked stems (فكتابه "and his book", كتابه "his book", تابه "tAbh", فكتاب "and a book").

[1] http://arabic.emi.ac.ma/safar/?q=publications.

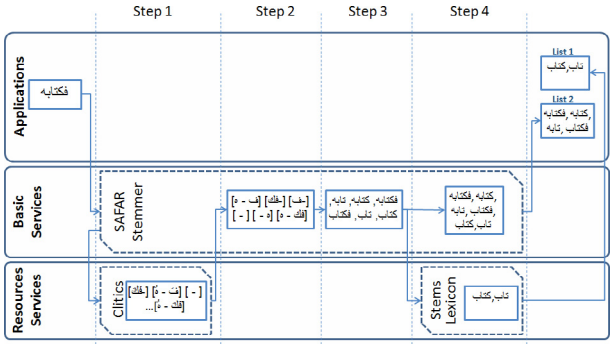

Figure 1. SAFAR-Stemmer steps workflow.

4. Stemmers evaluation and benchmark

Arabic stemmers are widely used in many text processing tasks such as text categorization, information extraction, text summarization and search engine indexing. Therefore, evaluating and benchmarking these stemmers are crucial for researchers to select the best stemmer that suits their needs in a given context. In addition, authors are generally supposed to compare their stemmers with other works in order to demonstrate their contributions and enhancements. That is to say, ANLP community should have common tools and resources to make fair evaluations of stemmers. However, in most cases authors present their stemmers and provide some evaluations statistics based on their own evaluation corpora without providing any open source tools or resources to verify the accuracy and strength of their stemmers. Most of them claim that their stemmers reach high rate of accuracy. Nevertheless, given the lack of fair common evaluation, we cannot be sure that such claims are true for all cases, independently of the used evaluation corpora.

Given this situation, some researchers tried to propose evaluations and benchmarks of several stemmers.

Majdi and Atwell (2008) proposed an evaluation of three root-based Arabic stemming algorithms namely: Khoja stemmer, Buckwalter morphological analyzer and Tri-literal root Extraction algorithm. The experiments were done by running the three stemmers on the chapter number 29 of the holy Qur'an "Sourat Al-Ankaboot" and a newspaper text taken from the Corpus of Contemporary Arabic. The results show that Khoja stemmer reaches the highest accuracy, followed by the tri-literal root extraction algorithm and finally the Buckwalter morphological analyzer. It should be noted that authors have used only metrics related to the accuracy of results in their experiments: errors, fault rate and accuracy.

Al-Kabi et al. (2011) proposed a work on benchmarking and accessing the performance of four root-based Arabic stemmers namely: Al-Mustafa stemmer, Al-Sahran stemmer, Rabab'ah stemmer and Taghva stemmer. The corpus used for this benchmark was randomly collected from 15 collections of Arabic documents taken from the internet. The experiments show that the stemmers are ranked as follows starting from the good one to the weakest: Rabab'ah, Al-Sahran, Al-Mustafa and finally Taghva. For that study, authors have used metrics related to the accuracy of results as well as metrics related to the "strength" of stemmers (stronger stemmers are intended to make more changes to words to produce stems).

Al-Shawakfa et al. (2010) presented a comparison study of six Arabic root-based stemmers algorithms. Authors unified the testing process by building a corpus of 3823 triliteral roots, obtained by combining 73 triliteral patterns with 18 affixes, to produce around 27.6 million words. The results of the evaluation were lower than what is reported by authors of the six stemmers. The comparison has shown that the best algorithm was that of Ghwanmeh et al., which achieved an accuracy rate of 39%, followed by

Khoja and Garside (34%), Al-Shalabi (32%), Sonbol et al. (24%), Taghva et al. (20%), and finally Al-Shalabi et al. (14%).

Maabid et al. (2015) proposed assessment criteria for benchmarking Arabic morphological analyzers and stemmers. In this study, authors do not propose any approach or tool for evaluating stemmers, they just try to sum up some assessment criteria to measure the accuracy and strength of stemmers.

These researches give evaluations of selected stemmers and provide fair comparisons based on new corpora. However, they do not offer any reusable and generic solution for benchmarking Arabic stemmers; they propose only examples of evaluating some known stemmers without any possibility of extension of their solutions to evaluate other stemmers or even changing the evaluation corpus. Thus, they should provide new solutions or alter their own ones every time they have to evaluate a new stemmer, which is not suitable for this kind of benchmarking.

Moreover, all of them neglect the execution time of stemmers when evaluating them. They only focus on metrics related to the accuracy and "strength" of stemmer. However, with the growing digital Arabic content, authors should optimize the execution time of their stemmers in order to process more data in less time.

Therefore, developing a new automatic, flexible and reusable evaluation system, which takes into account metrics related to accuracy and "strength" as well as the execution time of stemmers, will be very useful for the ANLP community.

4.1. Evaluation corpora

In order to perform the evaluation and benchmark process, the results returned by Arabic stemmers should be compared to results of an evaluation corpus that is annotated with some morphological information such as the stem or root. This corpus should be verified manually by linguists to maximize its precision and provide confidence in its content.

However, building these kinds of corpora is time consuming and requires experts with linguistic knowledge in Arabic language. It has been realized nowadays that the effort needed to build such corpora may exceed largely the effort needed to build tools that use those corpora. This justifies the lack of a gold standard for benchmarking the different Arabic tools.

Given this situation, we reused other works in order to propose an evaluation corpus for benchmarking both root-based and light Arabic stemmers. For this reason, we used the Quranic Arabic Corpus (Dukes and Habash, 2010). This latter is an online annotated linguistic resource with multiple layers of annotation including morphological segmentation, Part-Of-Speech tagging, syntactic analysis using dependency grammar. It consists of the holy Qur'an annotated according to the context of words. The main morphological information returned by this corpus is the stem, root, lemma, Part-Of-Speech, prefixes and suffixes. We processed this corpus in order to keep only the stem and root of words. The new corpus (Fig. 2) is in XML format and can be used to evaluate Arabic stemmers. It can be downloaded from our team website.[2]

For example, the word « يؤمنون » ("They believe") has one manually checked analysis according to its context. This analysis has three tags: the root "امن", the stem "يؤمن" ("he believes") and the lemma "ءامن" ("he believed").

4.2. Common metrics of performance used to evaluate stemmers

In the literature, authors separate metrics used to evaluate stemmers into two main categories namely: metrics related to the accuracy of results and metrics related to the "strength" of stemmers. For our benchmark solution, we use both types of metrics:

```
<?xml version="1.0" encoding="UTF-8"?>
<stemmerAnalysis total_words="18352">
  <word w_id="1" value="ألرحمان">
    <analysis root="رحم" stem="رحمان" lemma="رحمان" />
  </word>
  <word w_id="2" value="نعبد">
    <analysis root="عبد" stem="نعبد" lemma="عبد" />
  </word>
  <word w_id="3" value="يؤمنون">
    <analysis root="امن" stem="يؤمن" lemma="ءامن" />
  </word>
  ...
</stemmerAnalysis>
```

Figure 2. Example of the evaluation corpus used to compare Arabic stemmers.

Accuracy (Flores and Moreira, 2016): The accuracy of results returned by a stemmer expresses how these results are correct. Unlike classical precision and recall scores, accuracy is equal to 100% only if the stemmer returns all correct stems for all words, and in addition to that, it returns no additional incorrect stems. If the accuracy is equal to 100%, this means that the stemmer is perfect. In the case of stemmers, the accuracy can be calculated as follows:

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

where TP is the total number of correct stems, FP is the total number of incorrect ones and FN is the total of correct stems not returned by the stemmer.

Number of Words Per Conflation Class (WCC) (Galvez et al., 2005): it is the average number of words that match the same stem. For example if the words "مكتبة" ("library") "يكتبون" ("they write") and "كتاب" ("book") are stemmed to the same root "كتب", then the WCC score is three. The value of the number of words per conflation class reflects the strength of the stemmer. The higher the value is, the stronger the stemmer becomes. The *WCC* metric is calculated as follows:

$$\text{WCC} = \frac{C}{S}$$

where *C* refers to the total number of distinct corpus words before stemming, or number of word types. In the same way, *S* refers to the number of distinct stems returned by the stemmer.

Index compression factor (ICF) (Frakes and Fox, 2003): it represents the average decrease in index size achieved through the stemming process. For example, a corpus with 100,000 words and 80,000 stems would have an index compression factor of 20%. Stronger stemmers will tend to have higher *ICF* value. It is calculated as follows:

$$\text{ICF} = \frac{C - S}{C}$$

where *C* refers to the total number of corpus word types and *S* refers to the number of unique stems returned by the stemmer.

Word change average (WCA) (Al-Kabi et al., 2011): stemmers often leave words unchanged. For example, a stemmer might not alter the verb "كتب" ("he wrote") because it is already a root word. Stronger stemmers will change words more often than weaker stemmers in order to obtain the correct root/stem. This metric is calculated as follows:

$$\text{WCA} = \frac{C - U}{C}$$

where *C* refers to the total number of corpus word types and *U* refers to the number of unique words that have not been changed after the stemming process.

[2] http://arabic.emi.ac.ma/ibtikarat/?q=Resources.

Average number of removed characters after stemming (ARC) (Al-Kabi et al., 2011): stronger stemmers tend to remove more characters from words to form stems. For example, if the following words "مكتبة" ("library"), "يكتبون" ("they write"), "كتاب" ("book") and "مكتب" ("office") are stemmed to the same root "كتب", then the *ARC* value would be (2 + 3 + 1 + 1)/4 = 1.75 characters.

4.3. New metric for evaluating Arabic stemmers

When evaluating stemmers, researchers often use the usual metrics presented above. However, Arabic data on the digital world has become so large that it becomes impossible to neglect the execution time of tools. For example, indexing the huge amount of data available in the internet for information retrieval purposes is time consuming. Thus, researchers should take into account not only metrics related to the accuracy and strength of stemmers, but also their execution time. To remedy this, our solution of benchmarking calculates all the above metrics and also the execution time of each stemmer. It should be noted that we have previously developed a similar tool for benchmarking Arabic morphological analyzers (Jaafar et al., 2016). Therefore, researchers will have an idea about how long stemmers take to accomplish their stemming process.

However, returning separate metrics makes the selection of the best stemmer difficult for researchers. Indeed, the accuracy of the results and the execution time are two metrics that vary disproportionately. This causes a problem of comparison in the case where we have two stemmers that return asymmetric metrics.

Thus, in order to introduce the execution time while evaluating stemmers and to overcome the problem of metrics that vary disproportionately, we propose a new global metric called Gs-Score (for Global Stemming Score) which can be calculated as follows:

$$\text{Gs} - \text{Score} = \frac{\alpha . \sum T_w}{\beta . \sum \text{Accuracy}_w}.$$

where T_w is the time taken by the stemmer to stem the word "W", and $Accuracy_w$ is the accuracy of results returned by the stemmer for the word "W". The α and β parameters are used to adjust the weights of execution time and accuracy. Researchers can change their values in order to make one element more important than the other. For example, if the accuracy matters more for a researcher, he/she can set α to a low value and β to a high value. For our experiments, we have set both α and β to 1.

Indeed, execution time and accuracy are two metrics that reflect the real performance of the stemmer and the relevance of its results. A stemmer with high rate of accuracy and reduced execution time is considered a good stemmer. In contrast to this, a stemmer with a high number of Words per Conflation Class (WCC) and a high index compression factor (ICF) is not necessarily a good stemmer since these kinds of metric describe only how many changes were made to words in order to produce stems (number of added/removed characters), and do not reflect the accuracy. That is why we believe that both the execution time and the accuracy should be taken into account when evaluating Arabic stemmers. A fast and accurate stemmer is better than a fast and not accurate stemmer. If two stemmers have the same execution time, then their accuracies will determine which one is the best. If two stemmers have the same accuracy, then their execution times will determine which one is the best. That is to say that both accuracy and execution time should be considered when comparing and benchmarking stemmers.

It should be noted that our Gs-Score metric is considered better when its value tends to 0, and vice versa. Therefore, this new metric will help researchers make the optimal choice even if the metrics returned by stemmers are disproportionate.

```
word1 : stem1[, stem2, stem3...]
word2 : stem1[, stem2, stem3...]
word3 : stem1[, stem2, stem3...]
word4 : stem1[, stem2, stem3...]
...
```

Figure 3. Example of input stemming results to evaluate.

4.4. Presentation of our benchmarking solution

Our solution for the benchmark is integrated in the utilities layer of SAFAR framework. This solution can be called and reused easily in any other project without any modification (only the integration of a new stemmer to benchmark will be required). It should be noted that SAFAR proposes a layer for Arabic stemmers with several models and utilities to facilitate the development or the integration of new Arabic stemmers within the framework.

The selected stemmers presented in this article were integrated in SAFAR in order to gain more flexibility while benchmarking them. The benchmark over these stemmers can be run directly since they are already taken into account by the system. For evaluating and benchmarking new stemmers, researchers have then two possibilities: (1) integrate the new stemmer in SAFAR morphology layer and benefit from its flexibility, the system of benchmarking can then be called as usual without any modification, (2) if for any reason the researcher prefers not to integrate his/her stemmer within SAFAR, he/she can simply provide a text file (Fig. 3) containing the results of the stemmer, the system of benchmarking will then compare it against other stemmers results.

Each line of the custom stemmer results concerns one word. For example, in line 1, the word "word1" could have at least one stem "stem1" and probably other stems according to all word contexts: "stem2", "stem3", etc. Stems of each word are separated by a comma.

Thus, every stemmer can be evaluated and benchmarked with others either by integrating it within SAFAR or by providing its results in a separate file. There are no changes to make on the benchmark system side.

In addition, any other annotated corpus can be used to evaluate the Arabic stemmers. Researchers have only to normalize their corpora according to the XML file in Fig. 4. Therefore, researchers will have a large choice of Arabic stemmers to compare and evaluation corpora to use without having to change the benchmark system every time they want to evaluate a new stemmer.

Moreover, researchers have the possibility to run the benchmark system using the web application.[3] This solution can be used either by developers or by linguists without writing any line of code.

Fig. 5 gives an overview of the proposed benchmarking solution and its different steps.

In step 1 of Fig. 5, all stemmers process the input text of the evaluation corpus. The results of each stemmer are then retrieved as memory objects in step 2. In step 3, the utility compares the results of each stemmer with those of the evaluation annotated corpus in order to calculate the accuracy of each stemmer.

5. Experiments and results

In order to give concrete examples of using our benchmark solution, we have selected three light stemmers (Light10, Motaz stemmer and Tashaphyne) and compared their results with our SAFAR-Stemmer. We selected these stemmers since they are available for download and are open source. The evaluations were

[3] http://arabic.emi.ac.ma:8080/SafarWeb_V2/BenchmarkController.

```
<?xml version="1.0" encoding="UTF-8"?>
<stemmer_analysis>
  <word value="word1" stem="stem1[,stem2...]" />
  <word value="word2" stem="stem1[,stem2...]" />
  <word value="word3" stem="stem1[,stem2...]" />
  <word value="word4" stem="stem1[,stem2...]" />
  ...
</stemmer_analysis>
```

Figure 4. Example of custom XML evaluation corpus.

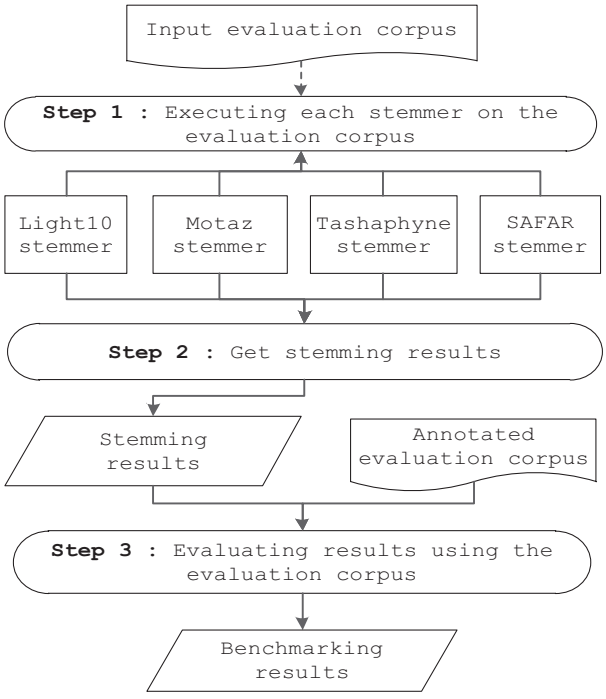

Figure 5. Stemmers benchmarking steps.

performed using the Quranic Arabic Corpus (Dukes and Habash, 2010) which contains 18,350 unique words.

Although SAFAR-Stemmer outputs several possible stems for one word, we modified our stemmer for it to output one stem per word form only. This made it possible to benchmark it against stemmers that return one stem only. However, we encourage researchers to consider the multiple stem solution also in their works.

Experiments were executed on a computer having the following characteristics: CPU = Core 2 Duo @2.13 GHZ, RAM = 4 GB, Operating System = Win7, 64bits. The following table gives an overview of the results.

Table 1 presents the results of comparing the four light stemmers using the Quranic Arabic Corpus as evaluation corpus. The results show that our SAFAR-Stemmer achieves the highest rate of accuracy with 33.7%, followed by Motaz stemmer with 18.59%, Light10 with 14.96% and finally Tashaphyne stemmer with 10.95%. For the execution time, Motaz stemmer takes less time with 0.36 s, followed by Light10 with 1.77 s, SAFAR-Stemmer with 3.73 s and finally Tashaphyne stemmer with 5.69 s. It should be noted that our stemmer takes more time than Motaz stemmer and Light10 due to the lexicon resource verification, which increases both accuracy and execution time. Otherwise we could have had a similar reduced execution time as other stemmers. Concerning the Gs-Score metric that combines accuracy with execution time, Motaz stemmer comes in first position with 0.02, followed by SAFAR-Stemmer with 0.1, Light10 with 0.11 and finally Tashaphyne stemmer with 0.52.

Given these results, we can suggest using Light10 and Motaz stemmers in application-oriented tasks, like information retrieval, where execution time matters more than accuracy of results. Concerning SAFAR-stemmer, it will be more suitably used in tasks like Part-of-Speech tagging, parsing, etc. to enhance their performance, as the accuracy of these systems depends on the accuracy of the stemming process.

6. Conclusion

In this article, we presented a new Arabic stemmer that uses lexical resources in order to enhance the accuracy of results. Indeed, Arabic light stemming as the process of stripping off a set of prefixes and/or suffixes from a derived or inflected word, is a blinded task suffering from problems such as incorrect removal, vocalization ambiguity, single solution, etc. We used the SAFAR framework resource API in order to strip off all possible prefixes/-suffixes from a word; the remaining stems are then verified according to a lexical resource containing 181 k words with their stems and diacritization. Experiments show that this verification has improved the accuracy of results.

On the other hand, we presented an automatic solution of benchmarking light Arabic stemmers. We presented some evaluations that were made in order to compare some specific stemmers. In these evaluations, authors only provide examples of comparative assessment of some known stemmers, but they do not offer the possibility of extending their solutions to evaluate other stemmers. Our solution of benchmarking overcomes this by providing a reusable and flexible system. It should be noted that researchers

Table 1
Results of comparing four Arabic stemmers using the quranic Arabic corpus.

Metrics	Light10	Motaz	Tashaphyne	SAFAR-Stemmer
Correct stems (TP)	4777	5755	3622	9252
Incorrect stems (FP)	13,573	12595	14,728	9098
Unique stems (S)	8976	11735	5515	9500
Words not changed (U)	3513	8384	2343	6872
Removed characters	28944	11820	37,174	20,471
Accuracy	14.96%	18.59%	10.95%	**33.70%**
Words per Conflation Class (WCC)	2.04	1.56	3.32	1.93
Index compression factor (ICF)	51.08	36.04	69.94	48.22
Word change average (WCA)	80.85	54.31	87.23	62.55
Average removed characters (ARC)	1.57	0.64	2.02	1.11
Execution time (in seconds)	1.77	0.36	5.69	3.7
G_s-Score	0.11	**0.02**	0.52	0.10

Best scores of Accuracy and G_s-Score metrics are made in bold.

use two categories of metrics when evaluating stemmers: metrics related to the accuracy and those related to the strength. However, they neglect execution time, which is an important element given the huge amount of available data nowadays. To remedy this, we presented a new metric called Gs-Score (for Global Stemming Score) that combines execution time with the accuracy of stemmers. This new metric will allow researchers to make the best possible choice of the stemmer to use in their projects. We have selected three light stemmers in order to compare their results with our stemmer and give a concrete example of our system of benchmarking. The results show that our stemmer achieves the highest rate of accuracy with 33.7% and comes in the second position in terms of Gs-Score metric with 0.1.

It should be noted that most of stemmers are designed for information retrieval systems where the execution time matters more than the accuracy of results. It would be then useful to distinguish an intrinsic evaluation, measuring the accuracy at which a stemmer performs its task, from an extrinsic evaluation, measuring how useful the output of a stemmer is for a specific NLP task. Despite our stemmer is more accurate than the others, at the moment it is not intended to be used in systems such as IR. However, it will enhance results in other NLP tasks where accuracy is important, such as Part-of-Speech tagging, parsing, etc. since the accuracy of these systems depends on the accuracy of the stemming process.

It has also to be underlined that the Holy Quran is a very special text, and that we must be cautious when using and generalizing its results. That is to say, experiments done in this article could be enriched by using other evaluation corpora in order to give extended benchmarking results. Some more efforts should be put into producing these kinds of corpora.

In the future, we plan to enrich our lexicon resource with new words in order to cover more stems and enhance results. We plan also to evaluate new stemmers in order to present weaknesses and strengths of most available Arabic stemmers so that researchers can identify which ones to use in their projects. Moreover, we plan to further optimize our stemming process for it to take less time, and to publish it online on our research group website.

References

Aljlayl, M., Frieder, O., 2002. On Arabic search: improving the retrieval effectiveness via a light stemming approach. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management, McLean, VA.

Al-Kabi, M.N., Al-Mustafa, R.S., 2006. Arabic root based stemmer. In: Proceeding of the International Arabic Conference on Information Technology. ACIT, Jordan.

Al-Kabi, M.N., Al-Radaideh, Q.A., Akkawi, K.W., 2011. Benchmarking and assessing the performance of Arabic stemmers. J. Inform. Sci. 37 (2), 111–119.

Al-Shawakfa, E., Al-Badarneh, A., Shatnawi, S., Al-Rabab'ah, K., Bani-Ismail, B., 2010. A comparison study of some Arabic root finding algorithms. J. Am. Soc. Inform. Sci. Technol. 61 (5), 1015–1024.

Chen, A., Gey, F.C., 2002. Building an Arabic stemmer for information retrieval. In: Proceedings of the 11th Text Retrieval Conference (TREC).

Dukes, K., Habash, N., 2010. Morphological annotation of quranic Arabic. In: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, Valletta, Malta, 2010.

Dukes, K., Habash, N., 2010. Morphological annotation of quranic Arabic. In: Language Resources and Evaluation Conference (LREC), Malta.

Flores, F.N., Moreira, V.P., 2016. Assessing the impact of stemming accuracy on information retrieval. Inf. Process. Manage. 52 (5), 840–854.

Frakes, W.B., Fox, C.J., 2003. Strength and similarity of affix removal stemming algorithms. ACM SIGIR Forum 37 (1), 26–30.

Galvez, C., de Moya-Anegón, F., Solana, V.H., 2005. Term conflation methods in information retrieval: non-linguistic and linguistic approaches. J. Document. 61 (4), 520–547.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: an update. ACM SIGKDD Explor. Newslett. 11 (1), 10–18.

Hofmann, M., Klinkenberg, R., 2013. RapidMiner: Data Mining Use Cases and Business Analytics Applications. CRC Press.

Jaafar, Y., Bouzoubaa, K., 2015. Arabic natural language processing from software engineering to complex pipeline. In: First International Conference on Arabic Computational Linguistics (ACLing), Egypt, Cairo, 2015.

Jaafar, Y., Bouzoubaa, K., Yousfi, A., Tajmout, R., Khamar, H., 2016. Improving Arabic morphological analyzers benchmark. Int. J. Speech Technol. 19 (2), 259–267.

Larkey, L.S., Ballesteros, L., Connell, M.E., 2002. Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland.

Larkey, L.S., Ballesteros, L., Connell, M.E., 2007. Light stemming for Arabic information retrieval. In: Arabic Computational Morphology: Knowledge-based and Empirical Methods. Springer, Netherlands, pp. 221–243.

Maabid, A.M., Elghazaly, T., Ghaith, M., 2015. An enhanced rule based Arabic morphological analyzer based on proposed assessment criteria. In: Advances in Swarm and Computational Intelligence. Springer International Publishing, pp. 393–400.

Majdi, S., Atwell, E., 2008. Comparative evaluation of Arabic language morphological analysers and stemmers. In: International Conference on Computational Linguistics – COLING, Manchester, UK, 2008.

Otair, M.A., 2013. Comparative analysis of Arabic stemming algorithms. Int. J. Manag. Inform. Technol. 5 (2).

Rogati, M., McCarley, S., Yang, Y., 2003. Unsupervised learning of Arabic stemming using a parallel corpus. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, Stroudsburg, USA, 2003.

Saad, M.K., Ashour, W., 2010. Arabic morphological tools for text mining. In: 6th International Conference on Electrical and Computer Systems (EECS'10), Lefke, North Cyprus, 2010.

Souteh, Y., Bouzoubaa, K., 2011. SAFAR platform and its morphological layer. In: Proceeding of the Eleventh Conference on Language Engineering (ESOLEC'2011), Cairo, Egypt.

Zerrouki, T., 2016. Tashaphyne 0.2 (Online). Available: <https://pypi.python.org/pypi/Tashaphyne> (Accessed 14 April 2016).