Correctness, Strength and Similarity Evaluation of Stemming Algorithms for Arabic

Daoud Daoud*[1], Christian Boitet**[2]

[*]*Princess Sumaya University for Technology, Jordan*
[1]d.daoud@psut.edu.jo
[**]*GETALP, LIG, Université Joseph Fourier, France*
[2]Christian.boitet@imag.fr

Abstract: ***In this paper, we present a comprehensive evaluation of four Arabic stemmers, based on metrics for correctness, strength and similarity. Two data sets were used in this study. For correctness evaluation, we used a list of 8697 Arabic words grouped into 1606 conceptual classes. For similarity and strength evaluation, we used a list of 72,000 unique Arabic words. Conclusions about correctness, strength and similarity of the four Arabic stemming algorithms are reported.***

Key Words: ***Arabic Stemmer, Direct Evaluation, Information retrieval, Stemmer Correctness, Stemmer strength, Inter-stemmer similarity***

1 INTRODUCTION

Detecting the surface variations of the same word is one of the main challenges of any type of natural language processing system. Specifically, the effectiveness for information retrieval depends on its ability to map all those variations to the same form.

Stemming is the process of automatically revealing a word's stem. In other words, stemming a word is actually the removal of all the inflectional morphemes from the word's surface-form. Lemmatization goes a step further in identifying the *citation form* of the word, also often called its *lemma*, typically used to access dictionaries. In many languages, the inflected or derived wordforms of a lemma have several stems.

Most researchers in the field of Arabic information retrieval evaluated their systems on IR performance, using a testing system and a 'test collection' of documents, queries and relevance judgments. This involves substituting different stemmers to see which gives the best results in terms of performance metrics such as Precision, Recall, and F-measure [1]. Such task-specific evaluation makes it impossible to identify typical errors a stemmer would commit. Consequently, this type of evaluation hinders the efforts to devise appropriate solutions and enhancements.

To address this, we use an intrinsic, task-independent evaluation based on correctness, strength and similarity, and apply it to four Arabic stemmers.

This is the first step in tackling current challenges facing Arabic search engines and developing effective search tools that could suit the non-concatenative character of the morphology of Arabic.

2 STEMMER CORRECTNESS EVALUATION

The concept of *stemmer correctness* refers to the capacity of a stemmer to actually merge term variants into a single stem [2]. Because merging processes are prone to error, diverse studies have been carried out to identify the sources of error. In stemming procedures, the inaccuracies appear in the form of under-stemming errors, which occur when words that refer to the same variants are not reduced to the same stem; and over-stemming errors, which occur when words are stemmed incorrectly because they are not actual variants. An assessment approach for stemming algorithms was developed by Paice [3], who evaluates the accuracy of a stemmer by counting the under-stemming and over-stemming errors it commits. His measure provides insights which might help in stemmer and optimization. He introduces three performance evaluation indices: under-stemming index, over-stemming index, and stemming weight. The under-stemming index UI is computed as the proportion of pairs from the sample that are not merged even though they belong to the same group, whereas the over-stemming index OI is computed as the proportion of pairs that belong to different groups among those that are merged to the same stem.

Given a sample of W different words (wordforms) divided into concept groups, he computes the following for each group:

- Desired Merge Total (DMT), given by the following formula:

$$DMT = 0.5n(n-1)$$

- Desired Non-Merge Total (DNT), given by the following formula:

$$DNT = 0.5n(W-n)$$

where n is the number of words in the group.
The sum of the DMT over all groups produces the Global Desired Merge Total (GDMT) and, likewise, the sum of DNT's over all groups yields the Global Desired Non-merge Total (GDNT).
The Unachieved Merge Total (UMT) counts the number of under-stemming errors for each group and is given by the following formula:

$$UMT = 0.5\sum_{i=1}^{s} u_i(n-u_i)$$

where s is the number of distinct stems, u_i is the number of instances of each stem. The sum of UMT for all groups yields the Global Unachieved Merge Total (GUMT). The under-stemming index (UI) is given by:

$$UI = \frac{GUMT}{GDMT}$$

The number of over-stemming errors for each group is counted by the Wrongly-Merged Total (WMT) and is given by:

$$WMT = 0.5\sum_{i=1}^{t} v_i(n_s-v_i)$$

where t is the number of original groups that share the same stem, n_s is the number of instances of that stem, and v_t is the number of stems for group t. The sum of WMT for all groups is the Global Wrongly Merged Total (GWMT). The over-stemming index (OI) is given by:

$$OI = \frac{GWMT}{GDNT}$$

The *Stemming Weight* (SW), which is a measure of the strength of the stemmer, is calculated by dividing the Over-stemming Index OI by the Under-stemming Index UI. Low SW values indicate a weaker stemmer and higher values indicates a stronger stemmer. A strong stemmer merges a much wider variety of forms, therefore committing many over-stemming errors. A light stemmer fails to merge semantically related words, therefore committing many under-stemming errors. Under-stemming errors tend to decrease the Recall in the IR search, while over-stemming errors will deteriorate Precision. Therefore, correctness metrics facilitate specifying the type of errors made by the stemmers. Consequently, it helps devising appropriate solutions and enhancements with regard to retrieval systems.

3 STEMMER STRENGTH

The degree to which a stemmer changes words that it stems is called *stemmer strength* [4]. Stemmer strength is important because it helps to anticipate recall and precision. There are several ways to measure stemmer strength:

- Number of Words per Conflation Class (WC)—This is the average number of words that are reduced to the same stem. If the conflation of 100 different words resulted in 25 distinct stems, then the mean number of words per conflation class would be 4. Stronger stemmers will have more words per conflation class.
- The *Index Compression Factor* represents the fractional reduction in index size accomplished through the stemming process, the idea being that the heavier the stemmer, the greater the Index Compression Factor. This can be calculated by:
  IC = Index Compression Factor
  N = Number of unique words before stemming
  S = number of unique Stems after stemming
  IC = (N – S)/N
- The mean Levenshtein distance (LD) between words and their stems[1]. For example, the Levenshtein distance between "استعان" and "يستعينون" is 4. Our measure will be the average LD for every word in the original sample.

[1] The Levenshtein distance between two strings is the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character [5].

4 INTER-STEMMER SIMILARITY

It is possible to compare two separate stemmers by comparing their outputs. This provides a measure of the similarity (or conversely, the dissimilarity) between the two algorithms. The approach is to take a set of words and apply both algorithms in turn, thus producing two output lists [1]. Corresponding stems in the two output lists are then compared to give a measure of similarity between the stemmers.

Inter-stemmer similarity could provide valuable information for the designers of IR systems by helping them understand the performance of different stemmers. This type of comparison also helps in developing more efficient stemmers.

$$SSM(U,V) = 100\left[1 - \left(\frac{\sum_w \left(\frac{LD(U_w - V_w)}{MD(U_w - V_w)}\right)}{N}\right)\right]$$

where

U & V are the stemmers being compared,

N = Number of words in the sample

LD = Levenshtein distance

MD = Maximum Distance

Arabic Stemmers under Consideration

Table 1 summarizes the Khoja [5], Light10 [6, 7], Buckwalter [8, 9], and APIR [10] stemming algorithms.

TABLE 1: SUMMARIZATION OF THE FOUR STEMMING ALGORITHMS

Stemmer	Type	Algorithm	Lexical resources
Khoja	Root-based	Longest-match affix removal	Yes
Light10	Stem-based	Longest-match affix removal	No
Buck++	Stem-based	Longest-match affix removal	Yes
APIR	Stem-based	Longest-match and dynamic normalization	Yes

Khoja's stemmer removes diacritics, stop words, punctuations, and numbers. It then removes the longest suffix and the longest prefix. Finally, it matches the remaining word with verbal and noun patterns, to extract the root. It makes use of several linguistic data files, namely a list of all diacritic characters, punctuation characters, definite articles, and 168 stop words. A major problem with this type of stemmer is that many word variants are different in meaning, though they originate from one identical root [11].

Larkey's Light10 stemmer is used not to produce the linguistic root of a given Arabic surface form, but to remove the most frequent suffixes and prefixes [11].

Buckwalter developed an Arabic morphological analyzer that returns the possible segmentations of an Arabic word. This analyzer uses three lexicons of possible Arabic prefixes, stems and suffixes, and three compatibility tables to validate the prefix-stem, stem-suffix, and prefix-suffix combinations. It accepts an Arabic word and produces its possible segmentations (transliterated into English characters). It cannot be used directly for stemming, as it provides more than one possible solution for the same word. Thus, we decided to modify it (Buck++) to return the longest stem out of all the stems that might be generated.

Arabic Parsing for Information Retrieval (APIR) was developed by the first author recently. APIR implements the longest-match and dynamic normalization approach. It implements a lexical, or dictionary-based, segmentation which utilizes a lexicon accessed by morphs of the language being analyzed. The input text is scanned (in the right-to-left writing direction) and matches are returned. The longest (or "maximal") match at any given point is returned.

The segmentation part uses the strategy of maximal match segmentation, or "best" segmentation. The maximal match segmentation attempts to minimize the number of words in a sequence of characters by finding the longest matches in the dictionary at each point in the input. APIR employs as-needed normalization to handle internal inflections and boundary distortions. In other words, if there is a mismatch at a point caused by one of the long vowels characters (ى، آ، و، ي، ا) or hamza forms (ئ، ؤ، أ، إ), it will try with another character from each group before starting again.

The lookup dictionary contains only valid Arabic stems without any grammatical or morphological features. Thus, the cost of building this lexical resource and maintaining it is kept minimal.

5 EXPERIMENTAL DATA

The word sample we used in testing the correctness of the four stemmers consisted of 8697 distinct inflectional wordforms collected from Arabic Web sites. We manually categorized them into 1606 conceptual groups. Each group contains only inflectional wordforms (not derivational variations) and has clear-cut semantic boundaries. As shown in Table 2, سقط "to fall (Verb)" is not grouped with سقوط "falling (Noun)". In the same line, سفارة "embassy" is not grouped with "traveling", although they are derived from the same root.

TABLE 2: SAMPLES OF CONCEPTUAL GROUPS

Group # n	Group # n+1	Group # n+2	Group # n+3	Group # n+4
تؤيد	السفارات	السفر	سقط	السقوط
تؤيده	السفارة	بالسفر	سقطت	بالسقوط
تؤيدها	بالسفارة	سفر	سقطوا	سقوط
ستؤيده	سفارات	سفرنا	فسقط	سقوطه
سيؤيدونه	سفاراتها	سفرها	فسقطت	سقوطهم
نؤيد	سفارة	للسفر	وسقط	لسقوط
نؤيدك	سفارتها	والسفر	وسقطت	للسقوط
وتؤيد	للسفارات	وسفر	وسقطوا	والسقوط
ونؤيد	للسفارة			وسقوط
ويؤيد	والسفارات			وسقوطهم
يؤيد	والسفارة			

Regarding the wordlist used for inter-similarity and strength evaluation, we have collected 72,000 Arabic words from the Web. This wordlist contains different categories of Arabic words, such as nouns, adjectives, verbs, proper names and transliterated names.

6 STEMMER CORRECTNESS COMPARISONS

A computer program has been written to calculate the UI, OI and SW indices. The program reads the file containing the words sample in addition to the outputs generated by the Khoja, Light10, Buck++ and APIR stemmers. The results are listed in Table 3.

TABLE 3: STEMMING PERFORMANCE INDICES FOR THE FOUR STEMMERS

	UI	OI	SW
Khoja	0.200	0.002286	0.011418
Light10	0.708	0.000236	0.000333
APIR	0.044	0.000025	0.000568
Buck++	0.161	0.000332	0.002051

Light10 has the highest under-stemming errors, followed by Khoja and Buck++. APIR has the lowest under-stemming errors at 0.044. The magnitude of differences is significant between APIR and the other three stemmers. With regard to the over-stemming index, Khoja's stemmer has the highest value, followed by the Light10 stemmer and then by Buck++. The lowest OI is recorded by the APIR stemmer, with a very significant difference compared to the other three stemmers. These results are graphically shown in Figure 1.

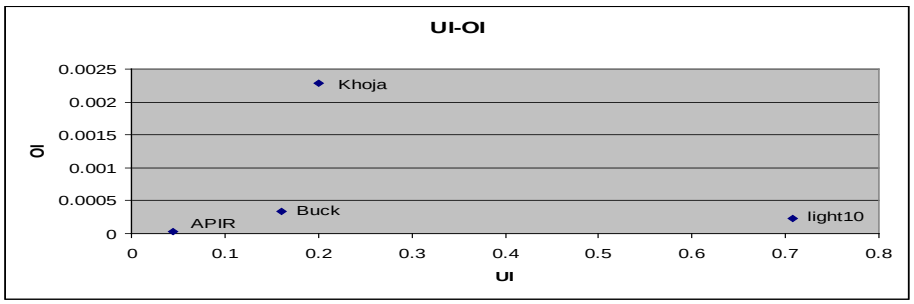

Figure 1: UI vs. OI for the four stemmers

We can say that Light10 commits fewer over-stemming errors compared to Khoja's and Buck++ stemmers, but leaving many words under-stemmed. On the other hand, Khoja's stemmer makes fewer under-stemming errors compared to light10 stemmer, but making huge over- stemming errors. This is reflected in the stemmer weight index (SW), SW index

of Khoja's stemmer is very larger compared to the other stemmers, indicating that Khoja's stemmer is the strongest one. What is interesting is the SW of APIR. Its value is less than Khoja and Buck++ but more than light10, indicating that it makes less over-stemming error and less under-stemming errors (more ideal stemmer). In summary the order of stemmer strength is:

Khoja> Buck++>APIR>light10

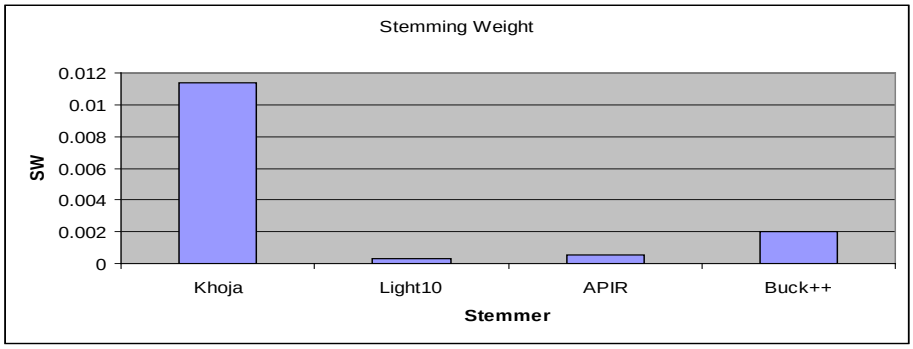

Figure 2: Stemmer strength

7 STRENGTH COMPARISONS

In this section, we analyze the strengths of the four stemmers using 3 measures: Levenshtein Distance, Words per Conflation Class, and Index Compression.

TABLE 4: RESULTS OF STEMMER STRENGTH MEASURES

Stemmer	LD	WC	IC
Light10	1.59	2.14	0.53
APIR	1.89	4.30	0.76
Buck++	1.95	4.48	0.77
Khoja	2.84	7.17	0.86

The three metrics listed in Table 4 are consistent in ordering the relative strengths of the stemmers. Based on these metrics, we found that Khoja is the strongest stemmer. We also noticed that both Buck++ and APIR are considerably weaker than Khoja's stemmer. This is because Khoja's stemmer extracts roots, while Buck++ and APIR are stem-based algorithms. Certainly, on average, the distance between a word and its stem is less than the distance between a word and its root. Light10 stemmer is a weak stemmer compared to other three stemmers.

Each measure places the stemmers in the following order:

Khoja> Buck++>APIR>light10

These results correspond exactly with the Stemming Weight results obtained using correctness measures.

8 INTER-STEMMER SIMILARITY COMPARISONS

We have applied the wordlist containing 72,000 entries to the four stemmers. We then calculated the average distance for all pairs of stems. The results are listed in Table 5 for each pair.

TABLE 5: SIMILARITY MEASURES

Pairs	Inter-stemmer similarity	Percentage of same stems
APIR-Buck++	91.23	68.41
Light10-Buck++	81.6	40.43
APIR-Light10	81.47	39.17
Khoja-Buck++	69.11	20.07
APIR-Khoja	66.10	15.06
Light10-Khoja	64.01	14.63

The results suggest that the inter-similarity pairings from most similar to least similar are: APIR-Buck++, Light10-Buck++, APIR-Light10, Khoja-Buck++, APIR-Khoja and Light10-Khoja.
These results are validated by stemmer strength evaluation. We have seen that APIR and Buck++ are closer to each other in terms of strength metrics. This is also valid for the inter-similarity metric, as the APIR-Buck++ pair has the highest relatedness.
We also notice that Light10 is more similar to both Buck++ and APIR than to Khoja, which is also apparent in the strength measures. The lowest similarity is detected in the pairs involving Khoja's stemmer which has a very high strength compared to other stemmers.
Hence, the inter-stemmer similarity measure is in total agreement with the results obtained from strength measures. However, the inference of similarity pairings from the correctness indices discussed above is not straightforward. In terms of under-stemming errors, APIR is more similar to Buck++, then to Khoja and finally to Light10. With regard to over-stemming errors, APIR similarity with Light10 is higher than with Buck++, and its similarity with Khoja is the lowest.
To demonstrate this, we will try to find the Correctness Similarity metric (CSM). The correctness similarity between two stemmers can be calculated by finding the difference of UI ratio and OI ratio of the two stemmers. For identical stemmers, the CSM would be 0. The CSM is given by the following formula:

$$CSM(U,V) = \left(\frac{UI_u}{UI_v} - \frac{OI_u}{OI_v} \right)$$

where
U & V are the Stemmers being compared,
UI = Under-stemming index
OI = Over-stemming index

TABLE 6: CORRECTNESS SIMILARITY RESULTS

Pairs	**Correctness Similarity**
APIR-Buck++	9.6
Light10-Buck++	3
APIR-Light10	6.5
Khoja-Buck++	5.6
APIR-Khoja	86.9
Light10-Khoja	6.15

Table 6 summarizes the results obtained and compares them with the distance-based similarity in Table 7. We observe that there is no agreement between the two lists. For example, the APIR-Buck++ pair is very similar in terms of distance, but not similar in terms of correctness. Hence, we conclude that, as in the case of stemmer strength, inter-stemmer similarity is not directly related to correctness. Thus, one could have two stemmers which are very similar and yet which are virtually different in their ability to conflate related words [1].

TABLE 7: CORRECTNESS-BASED VS. DISTANCE-BASED SIMILARITY

Correctness Similarity (high to low)	**Distance Similarity (high to low)**
Light10-Buck++	APIR-Buck++
Khoja-Buck++	Light10-Buck++
Light10-Khoja	APIR-Light10
APIR-Light10	Khoja-Buck++
APIR-Buck++	APIR-Khoja
APIR-Khoja	Light10-Khoja

9 CONCLUSIONS

In this paper we evaluated the correctness, strength of four stemming algorithms (Khoja, Light10, Buck++ and APIR), and their mutual similarities.
Stemmer correctness, that is, the ability of the stemmer to conflate related words accurately is important, because it provides insight into the types of errors stemming algorithms commit, and helps devise solutions and enhancements with regard to retrieval experimentation. Based on the number of under- and over-stemming errors, APIR outperforms other stemmers significantly.
Stemmer strength measures the amount of alteration on wordlist a stemmer can make. Using stemmer strength is useful in predicting index size, recall and precision in IR systems. We found that all metrics are consistent in ranking the relative strength of the four stemmers. Remarkably, this ranking corresponds precisely with the Stemmer Weight (SW) results.

These evaluation methods are not alternative but complementary, and the results presented provide a baseline for further enhancement and development.

REFERENCES

[1] R. Hooper and C. Paice, "Evaluation Techniques," vol. 2010: Lancaster University, 2006.

[2] C. Galvez and F. d. Moya-Anegón, "An evaluation of conflation accuracy using finite-state transducers," *Journal of Documentation*, vol. 62, pp. 328-349, 2006.

[3] D. P. Chris, "An Evaluation Method for Stemming Algorithms," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Dublin, Ireland: Springer-Verlag New York, Inc., 1994.

[4] B. F. William and J. F. Christopher, "Strength and Similarity of Affix Removal Stemming Algorithms," *SIGIR Forum*, vol. 37, pp. 26-30, 2003.

[5] S. Khoja and R. Garside, "Stemming arabic text." Lancaster, UK. Computer Science Department, Lancaster University, 1999.

[6] L. S. Larkey and L. Ballesteros, "Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis," *presented at SIGIR 2002*, 2002.

[7] L. S. Larkey, L. Ballesteros, and M. E. Connell, "Light Stemming for Arabic Information Retrieval " in *Arabic Computational Morphology*, A. Soudi, A. v. d. Bosch, and G. Neumann, Eds.: Springer Netherlands, 2007.

[8] T. Buckwalter, "Arabic lexicography," QAMUS, 2002.

[9] LDC, "Buckwalter Morphological Analyzer Version 1.0," Linguistic Data Consortium, 2002.

[10] D. Daoud and H. Qais, "Stemming Arabic Using Longest-Match and Dynamic Normalization," *presented at Arabic Language Technology International Conference (ALTIC) 2011*, Bibliotheca Alexandrina (B.A.), Alexandria, Egypt, 2011.

[11] T. Naglaa, "Stemming the Qur'an," in *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*. Geneva, Switzerland: Association for Computational Linguistics, 2004.

BIOGRAPHY

Daoud M. Daoud received his BSc degree in electrical and computer engineering from Kuwait University in 1988, his MSc in Computing Science from Glasgow University- UK and his PhD in Computing Science from Joseph Fourier University – France. Daoud is currently an assistant professor at PSUT. Recently, he co-founded Ddad IT which a specialized company for Arabic Natural Processing and Information Retrieval. He also served in Institute of Advanced Studies- United Nations University (1998-1999). He also worked as a principal investigator for the Arabic part of Universal Networking Language project (1996-1999). He also served as a director for Next Generation Services department at Paltel (1999-2001). His main research interests are Natural Language Processing, machine translation, Information Extraction, Information Retrieval and analysis of Arabic Social Media and Big Data.

تقييم خوارزميات تجذيع العربية استناداالى قياسات الصحة و القوة والتشابه

داوود داوود- [*1] كريستيان بواتيت [**2]

[*] جامعة الأميرة سمية، الأردن

[**] جامعة جوزيف فوريير، فرنسا

[1] d.daoud@psut.edu.jo

[2] Christian.boitet@imag.fr

خلاصة:

في هذه الورقة، قمنا بتقييم شامل لأربع من المجذعات "stemmers" العربية، استنادا إلى قياسات الصحة correctness والقوة strength والتشابه similarity.واستخدمنا مجموعتين من البيانات في هذه الدراسة .فلتقييم الصحة، استخدمنا قائمة من ٨٦٩٧ كلمة عربية مجمعة في ١٦٠٦ من التبويبات المفاهيمية . ولقياس التشابه والقوة،استخدمناقائمة تضم ٧٢٠٠٠من الكلمات العربية المختلفة. ونعرض في هذه الورقة الاستنتاجات حول الصحة والقوة والتشابه في أربع خوارزميات لتجذيع اللغة العربية.