Arabic Stemming Without A Root Dictionary

Kazem Taghva, Rania Elkhoury, Jeffrey Coombs
Information Science Research Institute
University of Nevada, Las Vegas
Las Vegas, NV 89154-4021
taghva@isri.unlv.edu

Abstract

We have implemented a root-extraction stemmer for Arabic which is similar to the Khoja stemmer but without a root dictionary. Our stemmer was found to perform equivalently to the Khoja stemmer as well as so-called "light" stemmers in monolingual document retrieval tasks performed on the Arabic Trec-2001 collection. A root dictionary, therefore, does not improve Arabic monolingual document retrieval.

1 Introduction

The two most successful approaches to Arabic stemming have been a root-extraction stemmer developed by Khoja[9] and a "light" affix removing stemmer developed by Larkey et al.[10, 11]. Larkey et al. have shown that the Khoja and Light stemmers, as well as co-occurrence analysis-aided stemmers, perform information retrieval tasks with statistically equivalent precision.[10]

We show that a root-extraction stemmer similar to Khoja's can be constructed without some of the shortcomings of that stemmer. Most obviously our stemmer does not require a root dictionary which can be difficult to maintain. Our stemmer was shown to be just as effective as the Khoja stemmer for information retrieval tasks.

Section 2 presents basic concepts concerning stemming in general and the Arabic stemming problem in particular. Previous work on Arabic Stemming is discussed in section 3 along with a brief presentation of the algorithm used in the Khoja stemmer. Our Arabic stemmer is presented in section 4. (The complete algorithm is listed in Appendix A.) We describe and evaluate our experiments in section 5. Finally, we present our conclusions and prospects for future work in section 6.

2 Background

A stemmer for the context of this paper is an automatic process in which morphological variants of terms are mapped to a single representative string called a *stem*. More specifically we will be focusing on the stemming approach called *affix removal* as opposed to the n-gram, successor variety, and table lookup stemmers which were offered as alternatives in Frakes' seminal work.[6]

Arabic belongs to the Semitic family of languages which also includes Hebrew and Aramaic. Since morphological change in Arabic results from the addition of prefixes and infixes as well as suffixes, simple removal of suffixes is not as effective for Arabic as it is for English. Language independent stemmers which focus on suffix removal are therefore unlikely to prove useful for Arabic.[10]

3 Previous Work

Based on the work of Al-Fedaghi[1], Al-Shalabi[2], and Bellaachia[3], Khoja and Garside produced an effective root-extracting stemmer.[9]

The Khoja stemmer follows this procedure:

1. Remove diacritics representing vowelization.
2. Remove stopwords, punctuation, and numbers.
3. Remove definite article (ال).
4. Remove inseparable conjunction (و).
5. Remove suffixes.
6. Remove prefixes.
7. Match result against a list of patterns. If a match is found, extract the characters in the pattern representing the root.
8. Match the extracted root against a list of known "valid" roots.

9. Replace weak letters اوي with و

10. Replace all occurrences of *hamza* ؤ،ئ،ء with أ

11. Two letter roots are checked to see if they should contain a double character. If so, the character is added to the root.

However, the Khoja stemmer has several weaknesses. First, the root dictionary requires maintenance to guarantee newly discovered words are correctly stemmed.

Second, the Khoja stemmer replaces a weak letter (in step 9 above) with و which occasionally produces a root that is not related to the original word. For example, the word منظمات (containers) is stemmed to ظمآ (he was thirsty) instead of نظم. Here the Khoja stemmer removed part of the root when it removed the prefix and then added a *hamza* at the end.

Third, by following a certain order of affixes, the Khoja stemmer will in some cases fail to remove all of them. For example, the terms تستغرق and ركبتيه are not stemmed although they are respectively derived from the two regular roots غرق and ركب.

4 ISRI Arabic Stemmer Algorithm

The Information Science Research Institute's (ISRI) Arabic stemmer shares many features with the Khoja stemmer. However, the main difference is that no root dictionary is used.

To begin our description of our stemmer, we define sets of diacritical marks and affix classes in figure 1. These are sets of marks which are removed by the stemmer. (Note, however, that the س is of course not a diacritical mark and is only listed in set D as an example consonant to which diacritics are applied.) We must also define some pattern sets in figure 2.

A complete description of the algorithm appears in Appendix A. Here we provide a more general overview of the stemmer. Stemming proceeds in the following steps:

1. Remove diacritics representing vowels.

2. Normalize the *hamza* which appears in several distinct forms in combination with various letters to one form (أ). This step is necessary in order to ensure terms such as تعَكل (he is eating) and أؤكل (it is eaten) conflate to the same root after their prefixes are removed.

3. Remove length three and length two prefixes in that order.

4. Remove connector و if it precedes a word beginning with و.

5. Normalize آ ، أ ، إ to ا. Removing the *hamza* in this case does not affect the root.

6. Return stem if less than or equal to three. Attempting to shorten stems further results in ambiguous stems.

7. Consider four cases depending on length of the word:

   (a) Length = 4: If the word matches one of the patterns from $PR4$ (figure 2), extract the relevant stem and return. Otherwise, attempt to remove length-one suffixes and prefixes from $S1$ and $P1$ in that order provided the word is not less than length three.

   (b) Length = 5: Extract stems with three characters for words that match patterns from $PR53$. If none are matched, attempt to remove suffixes and prefixes, otherwise the relevant length-three stem is returned. If the word is still five characters in length, the word is matched against $PR54$ to determine if it contains any stems of length 4. The relevant stem is returned if found.

   (c) Length = 6. Extract stems of length three if the word matches a pattern from $PR63$. Otherwise, attempt to remove suffixes. If a suffix is removed and a resulting term of length five results, send the word back through step 7b. Otherwise, attempt to remove one character prefixes, and if successful, send the resulting length-five term to step 7b.

   (d) Length = 7. Attempt to remove one-character suffixes and prefixes. If successful, send the resulting length-six term to step 7c.

Step 7 essentially takes longer words and successively attempts to trim single-character affixes. If successful, it compares the resulting shorter term with various patterns at different levels until it either matches a pattern and extracts the relevant term, or becomes too short to be a viable stem.

5 Experiments

We evaluated our stemmer in terms of its effectiveness for document retrieval. Specifically we applied four stemmers to the Arabic Trec 2001 collection.[7]

The TREC collection consists of 383,872 Arabic news stories. Of these 18,759 have ground truth for 25 arabic queries. The ground truth set was split into two equal sets: a training and test set.

The retrieval algorithm used belongs to the family of retrieval approaches called *language modeling* which have shown promising results compared to traditional retrieval engines.[12, 4] We used the following formula which derives from the work of Hiemstra and which has proven

set	description	examples
$PR4$	length four patterns	فاعل فعول فعلة فعال فعيل مفعل
$PR53$	length five patterns and length three roots	تفاعل افتعل افعال افاعل فعالة فعلان فعولة تفعلة تفعيل مفعلة مفعول فاعول فواعل مفعال مفعيل افعلة فعائل منفعل مفتعل فاعلة مفاعل فملاع يفتعل تفتعل فعالي انفعل
$PR54$	length five patterns and length four roots	تفعلل افعلل مفعلل فعللة فعلان فعالل
$PR63$	length six patterns and length three roots	استفعل مفعالة افتعال افعوعل انفعل مستفعل
$PR64$	length six patterns and length four roots	افنلل افعلال متفعلل

Figure 2. Arabic Patterns and Roots

set	description	examples
D	diacritics-vowelizations	سَ سِ سُ سْ سّ سً سٍ سٌ
$P3$	prefixes of length three	ولل، وال، كال، بال
$P2$	length two prefixes	ال، لل
$P1$	length one prefixes	ل، ب، ف، س، و ي، ت، ن، ا
$S3$	length three suffixes	تمل، همل، تان، تين، گمل
$S2$	length two suffixes	ون، ات، ان، ين تن، گم، هن، نا يا، ها، تم، گن ني، وا، ما، هم
$S1$	length one suffixes	ة،ه،ي،گ،ت،ا،ن

Figure 1. Affix Sets

effective to us for the retrieval of Farsi documents.[8, 13] The rank of a each document for a given query is given by the formula:

$$rank(d) = \log\left(\sum_t tf(t,d)\right) + \sum_{i=1}^{n} \log\left(1 + \frac{\lambda_i tf(t_i,d)(\sum_t df(t))}{(1-\lambda_i)df(t_i)(\sum_t tf(t,d))}\right) \quad (1)$$

The *term frequency* $tf(t, T_i)$ is the number of unique terms t in a document d. The *document frequency* $df(t)$ is the number of training documents in which a term t appears. The weight λ, which represents the likelihood that a term will be relevant to a query as opposed to being merely a background term, must be determined experimentally.

Retrieval effectiveness is measured by the usual precision and recall formulas. *Recall* is the number of relevant documents retrieved divided by the total number of relevant documents. *Precision* is the number of relevant documents retrieved at a given point divided by the total number of retrieved documents.[14]

We compared three approaches to stemming Arabic words, Khoja, ISRI, and Light, with not stemming (No Stem). The Light stemmer removed affixes from the sets in figure 1 without attempting to match patterns from figure 2. In each case stopwords from a list included with the Khoja stemmer were removed.

The TREC collection contained 25 queries, and each query had three formats: *title*, *description*, and *narrative*. These differed in terms of length. The *title* queries were short with an average length of 5 words and ranged from 2 to 9 words. *Description* queries were longer at about 12 words per query ranging from 6 to 24 words. Finally, *narrative* queries averaged 30 words and ranged from a minimum of 13 words to a maximum of 85 words.

The documents with ground truth were divided into test and training sets with 50% of the documents placed in each. The training set was necessary to determine the optimal value for λ in formula (1). For each stemming type, including the case of no stemming, a greedy search algorithm was used to determine at which λ value the highest average precision was obtained using an index built from the training documents. The *description* query set was used for this task.

Figure 3 gives precision values on the eleven point recall scale. Results for each of the four stemming methods are shown for each of the query types: where t stands for the short, title queries, d for the description queries, and n for the longer, narrative queries. A summary of the averages appears in figure 4. Graphs of the eleven-point averages for the title, description, and narrative queries appear respectively in figures 5, 6, and 7.

Figure 4 seems to indicate that the ISRI stemmer performs better than the other approaches on the shorter *title* queries. A closer look at figure 5 shows that the light stemmer actually outpaces the ISRI stemmer at the lowest recall values. Since the more highly ranked documents are represented at the lower recall values, this means that the light

Recall	Khoja			ISRI			Light			NoStem		
	t	d	n	t	d	n	t	d	n	t	d	n
0.00	1.000	1.000	0.837	1.000	0.993	0.909	1.000	0.973	0.912	1.000	0.828	0.583
0.10	0.903	0.866	0.548	0.918	0.807	0.541	0.940	0.789	0.603	0.795	0.502	0.229
0.20	0.805	0.758	0.426	0.827	0.705	0.433	0.823	0.685	0.452	0.693	0.366	0.173
0.30	0.694	0.659	0.352	0.728	0.610	0.341	0.714	0.591	0.353	0.534	0.282	0.140
0.40	0.585	0.544	0.280	0.600	0.508	0.269	0.590	0.492	0.270	0.383	0.214	0.114
0.50	0.468	0.419	0.218	0.472	0.391	0.206	0.460	0.371	0.207	0.248	0.164	0.093
0.60	0.338	0.302	0.167	0.344	0.283	0.153	0.327	0.262	0.155	0.159	0.115	0.079
0.70	0.181	0.185	0.116	0.211	0.186	0.109	0.201	0.175	0.113	0.103	0.088	0.067
0.80	0.090	0.113	0.080	0.124	0.113	0.081	0.113	0.107	0.080	0.073	0.067	0.058
0.90	0.034	0.049	0.056	0.051	0.066	0.059	0.047	0.063	0.058	0.056	0.055	0.053
1.00	0.000	0.000	0.000	0.002	0.004	0.004	0.002	0.004	0.004	0.007	0.007	0.007
Average	0.463	0.445	0.280	0.480	0.424	0.282	0.474	0.410	0.291	0.368	0.244	0.145

Figure 3. Eleven Point Precision Results

Stemmer	title	description	narrative
No Stem	0.368	0.244	0.145
Light	0.474	0.410	0.291
Khoja	0.463	0.445	0.280
ISRI	0.480	0.424	0.282

Figure 4. Average Precision for Stemming Types

stemmer has a higher precision for the higher ranked documents, which are the documents most likely to be viewed by users.

Statistical analysis of the results, however, reveals in the six lower recall values (0.00-0.50) that the Khoja, ISRI, and Light stemmer essentially are equally effective on all three query types. A Willcoxon non-parametric rank sum test of the equality of the population medians over the eleven point precision scores showed that the Khoja, ISRI, and Light stemmers do not significantly outperform one another for any of the queries.[5] For the *description* and *narrative* queries, stemming made a difference: the Khoja, ISRI, and Light stemmers were significantly better than not stemming. Rather surprising was the result that for the shortest, *title* queries, the Willcoxon test showed that no stemmer was significantly better than not stemming (at $p = 0.05$). However, at recall values 0.10-0.50, all three stemmers are better than not stemming with an $p = 0.10$.

6 Conclusion and Future Work

Our testing showed that stem lists are not required in an Arabic Stemmer. Hence a stemmer with the stem lists one finds in the Khoja stemmer functions as well on document retrieval as one without.

Consistent with the findings of Larkey et al., we found that overall a light stemmer which removes affixes without pattern checking performs document retrieval as well as more complex stemmers, such as ours with its pattern checking and the Khoja stemmer with both pattern checking and stem lists.

These results seem to indicate that finding the "true" grammatical root of a term should not be the goal of a stemmer for document retrieval. Identifying a sufficiently large list of roots as well as maintaining that list as a language changes do not appear to be worth the effort and expense.

Of course, a root-finding program for Arabic, as well as for other languages in which tense, mood, etc., are marked by prefix changes, are useful. Arabic dictionaries list roots, and one must know the prefix-free root in order to find a definition.[9] Whether a root list is necessary for such purposes or not is still an open question and one we would like to pursue.

References

[1] Sabah S. Al-Fedaghi and Fawaz S. Al-Anzi. A new algorithm to generate arabic root-pattern forms. In *Proceeding of the 11th National Computer Conference and Exhibition*, pages 391–400, 1989.

[2] Riyas Al-Shalabi and Marth Evens. A computational morphology system for arabic. In *Workshop on Computational Approaches to Semitic Languages COLING-ACL98*, 1998.

[3] Abdelghani Bellaachia. Building arabic informational retrieval systems, 1998.

[4] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd ACM*

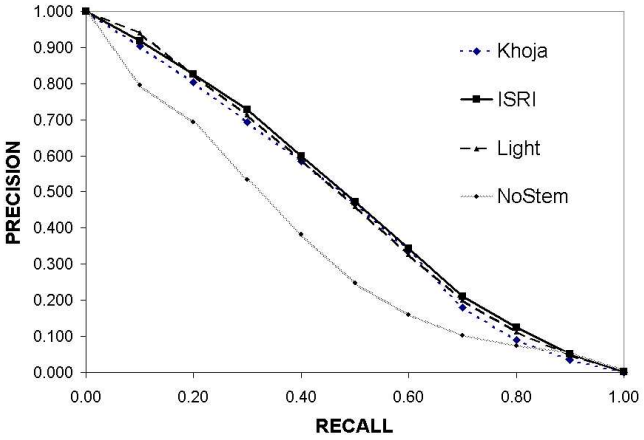

Figure 5. Test on Title Queries

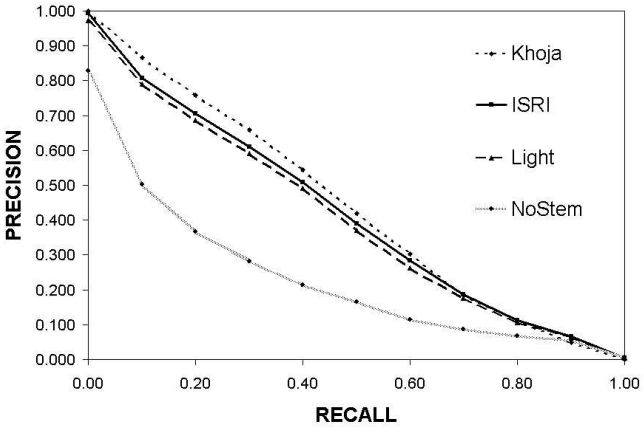

Figure 6. Test on Description Queries

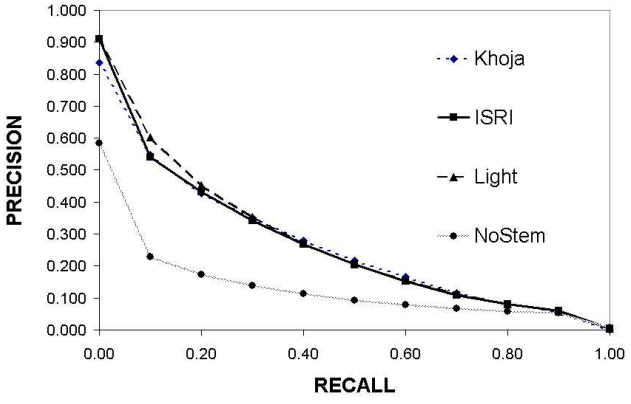

Figure 7. Test on Narrative Queries

Conference on Research and Development in Information Retrieval (SIGIR'99), pages 222–229, 1999.

[5] W. J. Conover. *Pracitical Nonparametric Statistics*. John Wiley, 1999.

[6] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval, Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[7] David Graff and Kevin Walker. Arabic newswire part 1, 2001. `http://wave.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2001T55`.

[8] Djoerd Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Universiteit Twente, 2000.

[9] Shereen Khoja and Roger Garside. Stemming arabic text. Computer Science Department, Lancaster University, Lancaster, UK, `http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps`, 1999.

[10] Leah Larkey, Lisa Ballesteros, and Margaret Connell. Improving stemming for arabic information retrieval: Light stemming and co-occurrence analysis. In *SIGIR 2002*, pages 269–274, 2002.

[11] Leah S. Larkey and Margaret E. Connell. Arabic information retrieval at UMass in TREC-10. In *Proceedings of TREC 10*, 2002.

[12] Jay Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98*, 1998.

[13] Kazem Taghva, Jeffrey Coombs, Ray Pareda, and Tom Nartker. Language model-based retieval for farsi documents. In *Proceedings of International Conference on Information Technology: Coding and Computing (ITCC'04)*, volume 2, pages 13–17.

[14] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes*. Morgan Kaufmann, 2nd edition, 1999.

A Appendix: Complete Algorithm

A.1 Helper Functions

- word-4(W):
  1. For first $p \in PR4$ such that $p \subseteq W$ return *extract(W)*.
  2. Do *short-suffix(W)*.
  3. If $len(W) = 4$ do *short-prefix(W)*.
- word-5(W)

1. For first $p \in PR53$ such that $p \subseteq W$ return *extract(W)*.
2. Do *short-suffix(W)*.
3. If $len(W) = 4$ do *word-4(W)*
4. else
   (a) do *short-prefix(W)*
   (b) if $len(W) = 4$ do *word-4(W)*
   (c) otherwise, if $len(W) = 5$, for first $p \in PR54$ s.t. $p \subseteq W$ return *extract(W)*.

- word-6(W)

1. For first $p \in PR63$ s.t. $p \subseteq W$, return *extract(W)*.
2. Do *short-suffix(W)*.
3. If $len(W) = 5$ then do *word-5(W)*
4. otherwise
   (a) do *short-prefix(W)*.
   (b) if $len(W) = 5$ do *word-5(W)*.
   (c) otherwise, if $len(W) = 6$ for any $p \in PR64$ s.t. $p \subseteq W$ return *extract(W)*.

- short-suffix(W)

1. For first $s \in S1$ s.t. $s \subseteq W$, $W = W - s$.
2. Return W.

- short-prefix(W)

1. For first $p \in P1$ s.t. $p \subseteq W$, $W = W - p$.
2. Return W.

A.2 Main algorithm

For every word W

1. Remove all $d \in D$ from W
2. Normalize ؤ،ىٔ،ء to أ
3. If $len(W) \geq 6$, for first $p \in P3$ s.t. $p \subseteq W$, $W = W - p$, else if $len(W) \geq 5$, for first $p \in P2$ s.t. $p \subseteq W$, $W = W - p$.
4. If $len(W) \geq 6$, for first $s \in S3$ s.t. $s \subseteq W$, $W = W - S3$. else if $len(W) \geq 5$, for first $s \in S2$ s.t. $s \subseteq W$, $W = W - s$.
5. If $len(W) \geq 4$ and initial characters of W are وو, remove initial و.
6. Normalize initial character ٱ ،أ ،إ to ا if necessary.
7. if $len(W) \leq 3$ return W.
8. (a) if $len(W) = 4$ then do *word-4(W)*
   (b) else if $len(W) = 5$ do *word-5(W)*
   (c) else if $len(W) = 6$ do *word-6(W)*
   (d) else if $len(W) = 7$
      i. do *short-suffix(W)*.
      ii. if $len(W) = 6$ do *word-6(W)* else
         A. Do *short-prefix(W)*.
         B. If $len(W) = 6$ do *word-6(W)*.
9. Return W.