Unsupervised Stemmer for Arabic Tweets

Fahad Albogamy
School of Computer Science,
University of Manchester,
Manchester, M13 9PL, UK
albogamf@cs.man.ac.uk

Allan Ramsay
School of Computer Science,
University of Manchester,
Manchester, M13 9PL, UK
allan.ramsay@cs.man.ac.uk

Abstract

Stemming is an essential processing step in a wide range of high level text processing applications such as information extraction, machine translation and sentiment analysis. It is used to reduce words to their stems. Many stemming algorithms have been developed for Modern Standard Arabic (MSA). Although Arabic tweets and MSA are closely related and share many characteristics, there are substantial differences between them in lexicon and syntax. In this paper, we introduce a light Arabic stemmer for Arabic tweets. Our results show improvements over the performance of a number of well-known stemmers for Arabic.

1 Introduction

The last few years have seen an enormous growth in the use of social networking platforms such as Twitter in the Arab World. A study prepared and published by Semiocast in 2012 has revealed that Arabic was the fastest growing language on Twitter in 2011. People post about their lives, share opinions on a variety of topics and discuss current issues. There are millions of tweets daily, yielding a corpus which is noisy and informal, but which is sometimes informative. As a result, Twitter has become one of the most important social information mutual platforms. The nature of the text content of microblogs differs from traditional blogs. In Twitter, for example, a tweet is short and contains a maximum of 140 characters. Tweets also are not always written maintaining formal grammar and proper spelling. Slang and abbreviations are often used to overcome their restricted lengths (Java et al., 2007).

Stemming is an essential processing step in a wide range of high level text processing applications such as information extraction, machine translation and sentiment analysis. It is a non trivial problem especially with languages that have rich and complex morphology such as Arabic. The function of a stemmer is to reduce words to their stems by stripping off all affixes (Frakes, 1992). Affixes are syntactic units that do not have free forms, but are instead attached to other words. They use the same alphabet as that of words and are concatenated one after the other with no demarcating marking such as the English apostrophe. Therefore, they are not easily recognisable. Although there have been several studies on developing morphological and stemming tools for Modern Standard Arabic (MSA), a stemmer that can work on Arabic tweets or similar text styles is yet to be developed.

In this study, an unsupervised stemmer for Arabic tweets is proposed, implemented and tested. It applies a light stemming technique on Arabic words to extract their stems. Our Arabic stemming approach is not dictionary based, which is crucial for stemming Arabic tweets, since they have a very open lexicon, and we are able to reach around 78% stemming accuracy. The results are compared with an Arabic stemmer which uses similar approach described in the literature.

The rest of this paper is organised as follows: In Section 2, we give an overview of the related work, followed by the representation of Arabic words in 3. Our approach is presented in Section 4. In Section 5, we discuss the evaluation, results and their analysis. In Section 6, we talk about conclusion and future work.

Proceedings of the 2nd Workshop on Noisy User-generated Text,
pages 31–37, Osaka, Japan, December 11 2016.

2 Related Work

There have been several studies on developing morphological and stemming tools for MSA. These tools can be classified into three categories: manually constructed dictionaries (heavy stemming), light stemming and statistical stemming.

Heavy stemming approaches try to find the stems and roots by using dictionaries. There are many examples of recent research work under this category such as (Buckwalter, 2004), (Khoja and Garside., 1999) and (Algarni et al., 2014). The Buckwalter Arabic Morphological Analyzer (BAMA) is the best known example of such an approach. It is an open-source software package morphological analyser developed by Tim Buckwalter. He developed a set of lexicons of Arabic stems, prefixes, suffixes and a table containing valid morphological combinations in order to produce all possible stems for each word. Khoja stemmer is another example of this class. It based on one dictionary for Arabic roots. It removes the longest suffix and the longest prefix. Then, it matches the remaining word with roots in the dictionary and a list of patterns. This category produces well-formed stems and roots which are potentially correct, but it generates multiple analyses, therefore it has to have some downstream mechanisms for choosing between them and the dictionaries are extremely difficult to maintain.

Light stemming approach is a process of stripping off a set of affixes from a word without trying to recognise patterns and find roots. It is fast, does not need roots or stems dictionaries and plays safe in order to avoid over-stemming errors, but it may produce the stem that may not even be a real Arabic word (Paice, 1994). (Al-Kabi et al., 2015) uses a light stemmer approach but does not cover all affixes in Arabic.

Statistical approach is a process of inducing a list of affixes automatically and using clustering techniques to group word variants. Most statistical approaches require annotated training data such as (Darwish and Oard, 2007) and (De Roeck and Al-Fares, 2000). This data is expensive and is not always available. Although there have been some positive results with un annotated training data for many different languages such as (Goldsmith, 2001) and (Dasgupta and Ng, 2007), it seems likely the complexity of Arabic language makes that kind of approach is infeasible.

On the other hand, there has been relatively little work on building stemmers for Arabic dialect. Most of these works targeted one specific dialect such as (Alamlahi and Ahmed, 2007) and (Al-Gaphari and Al-Yadoumi, 2010).

Our work is, to the best of our knowledge, the first step towards developing a stemmer for Arabic tweets or similar text styles which can benefit a wide range of downstream NLP applications such as information extraction and machine translation. Our approach does not rely on any root dictionary or list of patterns. We use a light stemming approach and shortest stem strategy to extract the stems of the words.

Word	وسيفعلون "wsyfElwn"	
Prefixes	Arabic	Translit.
	و	w
	س	s
	ي	y
Stem	فعل	fEl
Suffixes	ون	wn

Table 1: Example of Arabic Affixes

3 Arabic Word Form

Arabic is a morphologically rich language where the letters are attached together to form a word. A word often conveys complex meanings that can be decomposed into several morphemes (i.e. prefix, stem, suffix). Consequently, it presents significant challenges to many NLP applications such as tagging.

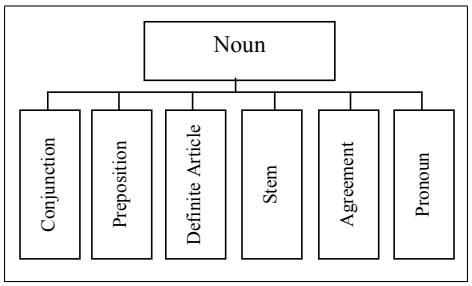

Figure 1: Possible sub-tokens in Arabic nouns

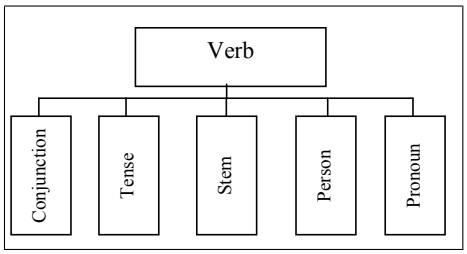

Figure 2: Possible sub-tokens in Arabic verbs

For example, the Arabic word[1] وسيفعلون "wsyfElwn" which means "and they will do", consists of five elements as shown in Table 1. We notice that Arabic affixes (prefixes and suffixes) attach to the base word in a strict order. These affixes are differ based on the word type, but in general we can represent the word as follows:

$$\text{Word} \equiv \text{Prefixes} + \text{Stem} + \text{Suffixes}$$

Arabic morphotactics allow words to have affixes. Affixes themselves can be concatenated one after the other. A noun can comprise up to six sub-tokens as illustrated by Figure1. Similarly a verb can comprise up to five sub-tokens as illustrated by Figure 2. The combination of words with affixes is governed by various rules. These rules are called grammar-lexis specifications (Dichy and Farghaly, 2003). An example of these specifications is a rule that states that the prefix "s", which denotes the future of verbs, is only combined with imperfective verb stems.

4 Our Approach

Although Arabic tweets and MSA are closely related and share many characteristics, there are substantial differences between them in lexicon and syntax (Albogamy and Ramsay, 2015). The lexicon is always evolving and many words in Arabic tweets are not present in MSA. Therefore, dictionary-based stemming approaches will not work for Arabic tweets. The statistical approaches that are used to induce a list of affixes automatically are also not applicable because we predefined a limited set of prefixes and suffixes for Arabic tweets and there is no annotated training data available. In Arabic tweets domain, there are millions of words that need to be stemmed, so stemming approaches that use a set of rules to identify words patterns are not suitable because they consume lots of computational resources (Al-Serhan and Ayesh, 2006).

Based on that, we decided to use a light stemming approach similar to the one that has been used in (Al-Kabi et al., 2015), but we cover all Arabic affixes. We define all possible affixes and write a set of rules of valid prefixes, suffixes and word forms and select the shortest stem of a word as the correct stem. We built our light stemmer by using Python and a basic grammar that we wrote to describe the morphological rules of Arabic noun and verb (see Figure 3). We are interested in two major word types; noun and verb. We defined them as a combination of allowable prefixes, and a word stem which is all letters between the prefix and the end of the string. In case that the word types have suffixes, the stem is defined as all letters between the prefix and the suffix. Our approach has two phases and deals with one word at a time. The first-phase is dedicated to taking the input word and trying it against the definition of its word type in the grammar. Then the stemmer will produce a list of all possible stems. The second-phase is to select the shortest stem as the correct stem. We use the word وباليد "wbalyd" which means "and by the hand" to demonstrate our approach (see Figure 4). The stemmer tried this word gainst the noun type grammar. It produced three possible stems. Then, the shortest stem (analysis 1) was selected 'yd' as the stem for this word. In this particular example, the stem chosen is the correct stem.

[1]The word is delimited by a white space

```
patterns = {
"ART": "Al|",
"CONJ": "w|b",
"NSTEM": ".{3,}?",
"VSTEM": ".{2,}?",
"PRON": "y|h|ha|hmA|hm|hn|k|kn|km|kmA|nA",
"VPRON": "PRON|NY",
"NY": "ny",
"PREP": "b|k|l",
"FUT": "s",
"TNS1": "O|n|A|y|t|",
"TENSE": "(FUT)?(TNS1)",
"PERSON": "(?(ON)AGR-ON|(?(AY)AGR-AY|
        (?(t)AGR-T|(?(past)AGR-PAST))))",
"AGR-ON": "",
"AGR-AY": "(A|An|wA|wn|n|)",
"AGR-T": "(yn|An|wn|n| )",
"AGR-PAST": "(tmA|tm|tn|t|A|wA|n|)",
"AGREE": "wn|yn|p|An|At|",
"NOUN": "(CONJ)?(PREP)?(ART)(?P<stem>NSTEM)
        (AGREE)(PRON)?",
"VERB": "(CONJ)?(TENSE)(?P<stem>VSTEM)
        (PERSON)?(VPRON)?"}
```

Figure 3: Combination rules of Arabic noun and verb"

```
Analysis 1:['w', 'b', 'al', 'yd']
           - prefix: ['w', 'b', 'al']
           - stem: yd
Analysis 2:['w', 'b', 'alyd']
           - prefix: ['w', 'b']
           - stem: alyd
Analysis 3:['w', 'balyd']
           - prefix: w
           - stem: balyd
Analysis 4:['wbalyd']
           - stem: wbalyd
```

Figure 4: Possible stems for the word وباليد "wbalyd"

5 Evaluation

As mentioned earlier, we have implemented our proposed stemming approach using Python programming language. The system accepts a text file that includes the Arabic nouns and verbs. Then, it produces the stems of those words. Example of the system's output results are shown in Table 1. Data used to test our stemmer for Arabic tweets, the results and error analysis are shown in the following subsections.

5.1 Experimental Setup

The corpus from which we extract our dataset contains 10 millions tokens taken from Twitter (Albogamy and Ramsay, 2015). They used Twitter Stream API to retrieve tweets from the Arabian Peninsula by using latitude and longitude coordinates of these regions since Arabic dialects in these regions share similar characteristics and they are the closest Arabic dialects to MSA.
To create our test set, we sampled 390 tweets from the above corpus to be used in our experiments. A set of correctly annotated nouns and verbs for this sample (gold standard) is required in order to be able to appraise the outputs of the stemmer. Once we have this, we can compare the outputs of the stemmer with this gold standard. Since there is no publicly available annotated corpus for Arabic tweets, we manually annotated our dataset. The dataset contains 1250 nouns and 692 verbs.

5.2 Results

In our experiments, we used two different settings of stem length: 3-character stem and 2-character stem. In the first experiment, we restricted the length of stems to be at least three characters for nouns and verbs

Word Type	Stem Length	Accuraccy
Nouns	>=3 characters	**78.16 %**
Verbs	>=3 characters	67.19 %

Table 2: Stemming accuracy if stem length is 3 characters and more

Word Type	Stem Length	Accuraccy
Nouns	>=2 characters	76.8 %
Verbs	>=2 characters	**77.45 %**

Table 3: Stemming accuracy if stem length is 2 characters and more

since most of words in MSA have a 3-letter stem. We got 78.16% and 67.19% stemming accuracy for nouns and verbs respectively as shown in Table 2. However, we noticed that for some nouns and verbs their stems were written in two letters. Therefore, they were ignored by the stemmer. In the second experiment, we restricted the length of stems to be at least two characters or more for nouns and verbs to cover all missed cases in the first experiment. The stemming accuracy for verbs was improved by about ten percent whereas the stemming accuracy for noun was decreased by around two percent as shown in Table 3.

Based on the experiments results, we decided to use two different constrains; one for nouns and the other for verbs. the length of stems has to be at least three characters or more for nouns and to be at least two characters or more for verbs. As we can see in Table 4, the best overall stemming performance is achieved when the minimum stem length is three characters for nouns whereas two characters for verbs. The results of the tests on our stemmer yield an accuracy of 77.91% of the whole collection. We have compared our stemming accuracy with (Al-Kabi et al., 2015), (Khoja and Garside., 1999) and (Ghwanmeh et al., 2009). Those three stemmers yield accuracies 75.03%, 74.03% and 67.40% respectively (Al-Kabi et al., 2015). Our results show improvements over the performance of those three well-known stemmers for Arabic. Moreover, we used a light stemming approach whereas they used heavy stemming approaches. We also experimented on Arabic tweets domain which is noisier than MSA. In addition, we are able to extract two and three characters stems length which are most stemmers for Arabic cannot do so.

Stem length	Overall accuracy (nouns and verbs)
>=2 characters	77.03 %
>=3 characters	74.25 %
Verb stem >=2 and noun stem>=3	**77.91 %**

Table 4: Overall stemming accuracy (nouns and verbs)

5.3 Error Analysis

We examined the words that were incorrectly segmented by our system. The errors can be broadly divided into three categories: under-stemming, over-stemming and orthography errors. Under-stemming errors happen when the stemmer does not remove all affixes in the words. For example, the word للمؤمنين "For the believers" the algorithm removes the first letter of the prefix لل but not the second one since it is considered as part of the stem in this case (see Table 5, the first row). Over-stemming errors happen when the stemmer considers part of the words is an affix and removes it. Consider the Arabic word واحد "One", the stemmer removes the original letter و because it looks like a conjunction (see Table 5, the second row). Orthography errors occur when the stemmer removes all affixes and finds the correct stems, but the last letter of the stems have a wrong shape. For example, the word كلمتهم "Their word" the stemmer removes the suffix هم and it considers كلمت as the stem which is correct except that the last

letter should be replaced by ة instead of ت (see Table 5, the third row)

Arabic word	Our stemmer	Gold Standard	Error Type
للمؤمنين "For the believers"	لمؤمن	مؤمن	under-stemming
واحد "One"	احد	واحد	over-stemming
كلمتهم "Their word"	كلمت	كلمة	orthography

Table 5: Examples of Stemming errors

6 Conclusion

We have proposed, implemented and evaluated a new stemmer for Arabic tweets. It does not rely on any root dictionary, which is crucial for stemming Arabic tweets, since they have a very open lexicon. It is a light stemming approach that uses shortest stem strategy to extract the stems of the words. It has two phases: phase 1 is dedicated to producing a list of all possible stems by using the grammar, and phase 2 is to select the shortest stem as the correct stem. We compared our stemmer with three Arabic stemmers where one of them uses a similar approach to ours. Results showed that our stemmer is better in terms of accuracy in comparison with other three Arabic stemmers.

Acknowledgments

The authors would like to thank the anonymous reviewers for their encouraging feedback and insights. Fahad would also like to thank King Saud University for their financial support. Allan Ramsay's contribution to this work was partially supported by Qatar National Research Foundation (grant NPRP-7-1334-6 -039).

References

GH Al-Gaphari and M Al-Yadoumi. 2010. A method to convert sana'ani accent to modern standard Arabic. *International Journal of Information Science & Management*, 8(1).

Mohammed N Al-Kabi, Saif A Kazakzeh, Belal M Abu Ata, Saif A Al-Rababah, and Izzat M Alsmadi. 2015. A novel root based Arabic stemmer. *Journal of King Saud University-Computer and Information Sciences*, 27(2):94–103.

Hasan Al-Serhan and Aladdin Ayesh. 2006. A triliteral word roots extraction using neural network for Arabic. In *2006 International Conference on Computer Engineering and Systems*, pages 436–440. IEEE.

Yahya Alamlahi and Fateh Ahmed. 2007. Sanaani dialect to modern standard Arabic: rule-based direct machine translation. *Computer Science Dep., Sanaa University, Sanaa, Yemen*.

Fahad Albogamy and Allan Ramsay. 2015. POS tagging for Arabic tweets. *RECENT ADVANCES IN Natural Language Processing*, page 1.

Mohammed Algarni, Brent Martin, Tim Bell, and Kourosh Neshatian. 2014. Simple Arabic stemmer. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1803–1806, New York, NY, USA. ACM.

Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0. ldc catalog number ldc2004l02. Technical report, ISBN 1-58563-324-0.

Kareem Darwish and Douglas W Oard. 2007. Adapting morphology for Arabic information retrieval. In *Arabic Computational Morphology*, pages 245–262. Springer.

Sajib Dasgupta and Vincent Ng. 2007. High-performance, language-independent morphological segmentation. In *NAACL HLT 2007: Proceedings of the Main Conference*, pages 155–163.

Anne N De Roeck and Waleed Al-Fares. 2000. A morphologically sensitive clustering algorithm for identifying Arabic roots. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 199–206. Association for Computational Linguistics.

Joseph Dichy and Ali Farghaly. 2003. Roots & patterns vs. stems plus grammar-lexis specifications: on what basis should a multilingual lexical database centred on Arabic be built. In *The MT-Summit IX workshop on Machine Translation for Semitic Languages, New Orleans*.

W. B. Frakes. 1992. Information retrieval. chapter Stemming Algorithms, pages 131–160. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Sameh Ghwanmeh, Ghassan Kanaan, Riyad Al-Shalabi, and Saif Rabab'ah. 2009. Enhanced algorithm for extracting the root of Arabic words. In *Computer Graphics, Imaging and Visualization, 2009. CGIV'09. Sixth International Conference on*, pages 388–391. IEEE.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we Twitter: Understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD*, pages 56–65, New York, NY, USA. ACM.

S. Khoja and R. Garside. 1999. Stemming Arabic text. *Computing Department, Lancaster University, Lancaster, UK*.

Chris D. Paice. 1994. An evaluation method for stemming algorithms. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 42–50, New York, NY, USA. Springer-Verlag New York, Inc.