Arabic Text Classification Using Decision Trees

Motaz K. Saad
Computer Engineering Department
Islamic University of Gaza
Gaza, Palestine
e-mail: msaad@iugaza.edu.ps

Wesam Ashour
Computer Engineering Department
Islamic University of Gaza
Gaza, Palestine
e-mail: washour@iugaza.edu.ps

Abstract[1]

Text mining draw more and more attention recently, it has been applied on different domains including web mining, opinion mining, and sentiment analysis. Text pre-processing is an important stage in text mining. The major obstacle in text mining is the very high dimensionality and the large size of text data. Natural language processing and morphological tools can be employed to reduce dimensionality and size of text data. In addition, there are many term weighting schemes available in the literature that may be used to enhance text representation as feature vector. In this paper, we study the impact of text pre-processing and different term weighting schemes on Arabic text classification. In addition, develop new combinations of term weighting schemes to be applied on Arabic text for classification purposes.

1. Introduction

Text Mining is a vital process due to huge availability of information in text documents which exists in various format. However, the task is not trivial to make the text at human level understanding to machines. The process includes derive linguistic features from text to be at human like interpretation to be mined, particularly, for Arabic language.

Text mining is well motivated, due to the fact that much of the world's data can be found in text form (newspaper articles, emails, literature, web pages, etc). Mining text has the same goals as data mining including, text categorization, clustering, document summarization, and extracting useful knowledge/trends. Text mining must overcome a major difficulty that there is no explicit structure [4]. Machines can reason relational data well since schemas are explicitly available. Text, however, encodes all semantic information within natural language. Text mining algorithms, then, must make some sense out of this natural language representation. Humans are great at doing this, but this has proved to be a problem for machines [4]. Text mining usually involves the process of structuring the input text (parsing, along with the addition of some derived linguistic features and the removal of others), deriving patterns within the structured data, and finally evaluation and interpretation of the output. High quality in text mining usually refers to some combination of relevance, novelty, and interestingness.

Arabic is one of the most widely used languages in the world. It is spoken by more than 280 million people as a first language and by 250 million as a second language. Despite Arabic is wide language, there are relatively few studies on the retrieval/mining of Arabic text documents in the literature. This is due to the unique nature of Arabic language morphological principles. Arabic is a challenging language for a number of reasons [1, 2, 3, 5, 12]:

1. Orthographic with diacritics is less ambiguous and more phonetic in Arabic, certain combinations of characters can be written in different ways.
2. Arabic has a very complex morphology recording as compare to English language.
3. Broken plurals are common. Broken plurals are somewhat like irregular English plurals except that they often do not resemble the singular form as closely as irregular plurals resemble the singular in English. Because broken plurals do not obey normal morphological rules, they are not handled by existing stemmers.
4. In Arabic we have short vowels which give different pronunciation. Grammatically they are required but omitted in written Arabic texts.
5. Arabic synonyms are widespread.

The impact of text pre-processing and different term weighting schemes combinations on Arabic text classification has not been studied in the literature. In this

Proceedings of the 12th international workshop on computer science and information technologies CSIT'2010, Moscow – Saint-Petersburg, Russia, 2010

paper, we study this impact on Arabic corpus collected manually from Aljazeera news web site. In addition, develop new combinations of term weighting schemes to be applied on Arabic text for classification purposes.

The rest paper is organized as follows: section 2 describes text pre-processing steps and stages. Experimental results are presented in section 3, and finally, we draw the conclusion.

2. Text Pre-processing

One of widely used methods for text mining presentations is viewing text as a bag-of-tokens (words, n-grams). Under that model we can already summarize, classify, cluster, and compute co-occurrence stats over text. These are quite useful for mining and managing large volumes of text. However, there is a potential to do much more. The Bag-Of-Tokens (*BOT*) approach loses a lot of information contained in text, such as word order, sentence structure, and context. These are precisely the features that humans use to interpret text. Natural Language Processing (*NLP*) attempts to understand document completely (at the level of a human reader). General *NLP* has proven to be too difficult. The reason that *NLP* in general is so difficult is that text is highly ambiguous. Natural Language is meant for human consumption and often contains ambiguities under the assumption that humans will be able to develop context and interpret the intended meaning.

The main function of term weighting is to enhance text document representation as feature vector. Popular term weighting schemes are Boolean model, Term Frequency (*TF*), Inverse Document Frequency (*IDF*), and Term Frequency-Inverse Document Frequency (*TFIDF*). Boolean model indicates absence or presence of a word with Booleans 0 or 1 respectively. Term frequency *TF(t,d)* is the number that the term t occurred in the document d. Document frequency *DF(t)* is number of documents in which the term *t* occur at least once. The inverse document frequency can be calculated from document frequency using the formula *log(num of Docs/num of Docs with word i*). The inverse document frequency of a term is low if it occurs in many documents and high if the term occurs in only few documents. Term discrimination consideration suggests that the best terms for document content identification are those able to distinguish certain individual documents from the collection. This implies that the best terms should have high term frequencies but low overall collection frequencies *(num of Docs with word i)*. A reasonable measure of term importance may then be obtained by using the product of the term frequency and the inverse document frequency (*TF * IDF)* [7, 10, 11].

In many situations, short documents tend to be represented by short-term vectors, whereas much larger-term sets are assigned to the longer documents. Normally, all text documents should have the same importance for text mining purposes. This suggests that a normalization factor to be incorporated into the term-weighting to equalize the length of the document vectors [7, 10, 11].

Terms have many morphological variants that will not be recognized by term matching algorithm without additional text processing. In most cases, these variants have similar semantic interpretation and can be treated as equivalence in text mining. Stemming algorithm can be employed to perform term reduction to a root form. Stemming algorithm by Khoja [9] one is of well know Arabic Stemmers.

Weka (Waikato Environment for Knowledge Analysis) [6] is a popular suite of machine learning software written in Java, developed at the University of Waikato. It is free software available under the *GNU* General Public License. Weka provides a large collection of machine learning algorithms for data pre-processing, classification, clustering, association rules, and visualization, which can be invoked through a common Graphical User Interface. Using Weka StringToWordVector tool options with different combinations, we setup the term weighting combinations presented in table 1 to be passed to *C4.5* decision tree [9] to classify text documents. We combine *TF*, *IDF*, *TFIDF*, pruning, and normalization. These combinations have not been applied on Arabic text before in the literature. The resulting combinations (described in table 2) are *Boolean, wc, wc-tf , wc-idf, wc-tf-idf, wc-norm, wc-minFreq3, wc-norm-minFreq3, and wc-all-minFreq3.*

Two major combinations are used; Bag of Tokens (*BOT*) (without stemming), and term Stemming. Symbols used in experiment setup preprocessing combinations for Stem and *BOT* are shown in table 2.

Workshop's Proceedings are published by CSIT Organizing Committee both in a hard copy and electronically.(style CSIT-Plane Text)

This means that instead of receiving printed copies of the papers customers will select an electronic format (WinWord).

- Papers from Workshops will be printed out individually. This means each page of each paper should have enough context information to identify the paper and the Workshop it came from.
- The print area of the **page** is defined to be the largest area which fits on **A4** paper with **2 cm** margins (everywhere as this template).**(style CSIT-List N)**

3. Experimental results and analysis

We perform Experiments on Arabic text dataset collected manually from *Aljazeera* news website (http://www.aljazeera.net). The dataset contains 119 text documents belonging to one of the three categories (sport, health, computer & communications). For text classification, we use *C4.5* decision tree [9] with 10 folds cross-validation.

Table 1: Weka String to Word Vector options

TF Transform	*log(1+fij),* where *fij* is the frequency of word *i* in document *dj*.
IDF Transform	*fij*log(num of Docs/num of Docs with word i),* where *fij* is the frequency of word *i* in document *dj*.
TFIDF Transformation	*log(1 + fij) * log (num of Docs/num of Docs with word i),* where *fij* is the frequency of word *i* in document *dj*.
minTermFreq	Sets the minimum term frequency (apply term pruning)
normalizeDocLength	Sets whether if the word frequencies for a document should be normalized or not.
outputWordCounts	Output word counts rather than Boolean 0 or 1(indicating absence or presence of a word).
Stemmer	The stemming algorithm to be use on the words (Khoja Arabic Stemmer Algorithm).

Table 2: Symbols used in experiment setup preprocessing combinations for Stem and BOT

Boolean	Indicating presence (1) or absence (0) of a word.
wc	Output word counts
wc-tf	Apply *TF* transformation on word count
wc-idf	Apply *IDF* transformation on word count
wc-tf-idf	Apply *TFIDF* transformation on word count
wc-norm	Apply document normalization on word count
wc-minFreq3	Apply term pruning on word count that less than 3
wc-norm-minFreq3	Apply normalization and term pruning on word count that less than 3
wc-all-minFreq3	Apply *TFIDF* and normalization on word count

Table 3: Classification result for BOT and Stemmed term using different text preprocessing combinations

	Attributes		**% correctly**		**Running Time**	
	BOT	**Stemmed**	**BOT**	**Stemmed**	**BOT**	**Stemmed**
Boolean	11617	4558	88	96	106.1	37.5
wc	11617	4558	87	95	105.9	39.57
wc-tf	11617	4558	88	95	105.2	40.39
wc-idf	11617	4558	87	95	105.5	35.29
wc-tf-idf	11617	4558	88	95	105.4	39.65
wc-norm	11617	4558	85	96	102.1	37.48
wc-minFreq3	873	736	90	95	6.99	5.9
wc-norm-minFreq3	2461	1402	87	97.4	22.6	9.98
wc-all-minFreq3	873	736	87	94	6.85	5.85

Table 3 shows classification performance for *BOT* and Stemmed terms using preprocessing combinations described in table 2. Figure 1 describes dimensionality reduction for text dataset using different preprocessing combinations. Comparing *BOT* and Stemmed Term, using term stemming lead to reduce dimensionality for all preprocessing combinations because stemming reduce many terms, which have many morphological variants, to their root. Dimensionality dramatically reduced using term pruning with minimum frequency of 3 because there are many infrequent terms in the document collection. The presence of infrequent terms leads to decrease classification accuracy.

Figure 2 shows classification accuracy for different text preprocessing combinations, *wc-norm-minFreq3* give highest accuracy for Stemmed terms, while *wc-minFreq3* give the highest accuracy for *BOT*. Obviously, pruning infrequent terms enhance classification accuracy. The accuracy for stemmed terms is better than *BOT* for all preprocessing combinations. Stemming enhance term weighting and this affect classification accuracy.

Figure 3 depicts the running time for classification process. Shortest running time is achieved when use term pruning with minimum 3 occurrences. Again,

running time for stemmed terms is shorter than *BOT* for all pre-processing combinations. Term stemming and pruning dramatically reduce dimensionality and enhance classification accuracy and performance.

The empirical results also show that *wc-norm* and *wc-tfidf* give good accuracy and performance; this may vary from dataset to another. Furthermore, it is known that document normalization and *TFIDF* work well for large text dataset [7, 10, 11].

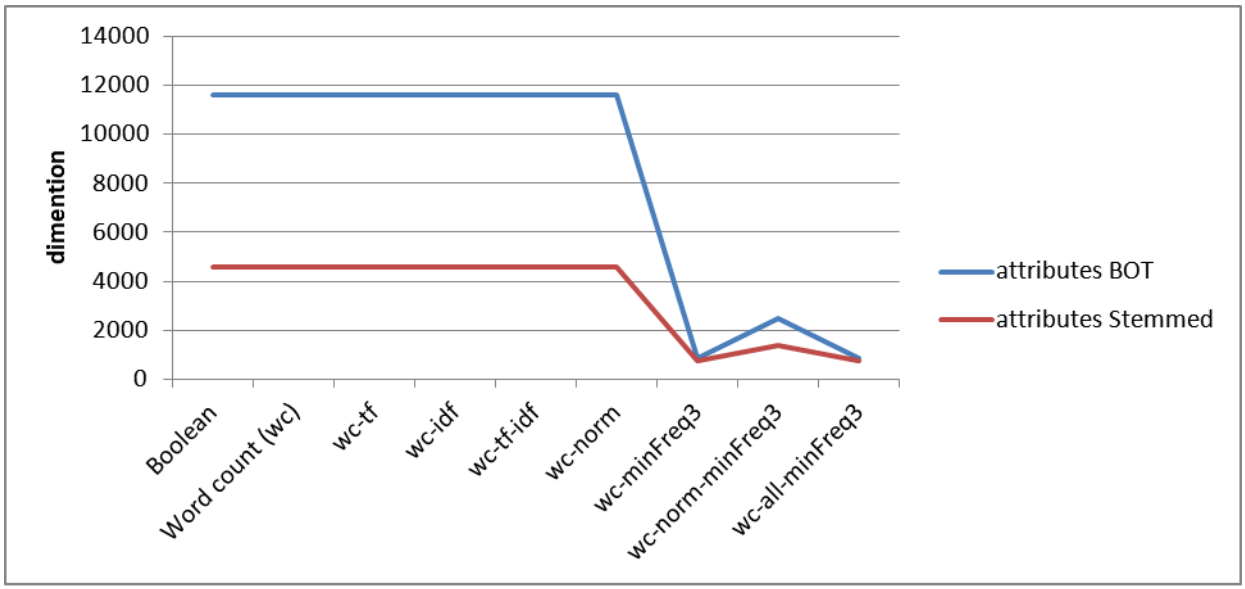

Figure 1: text dataset dimensionality for different text preprocessing combinations.

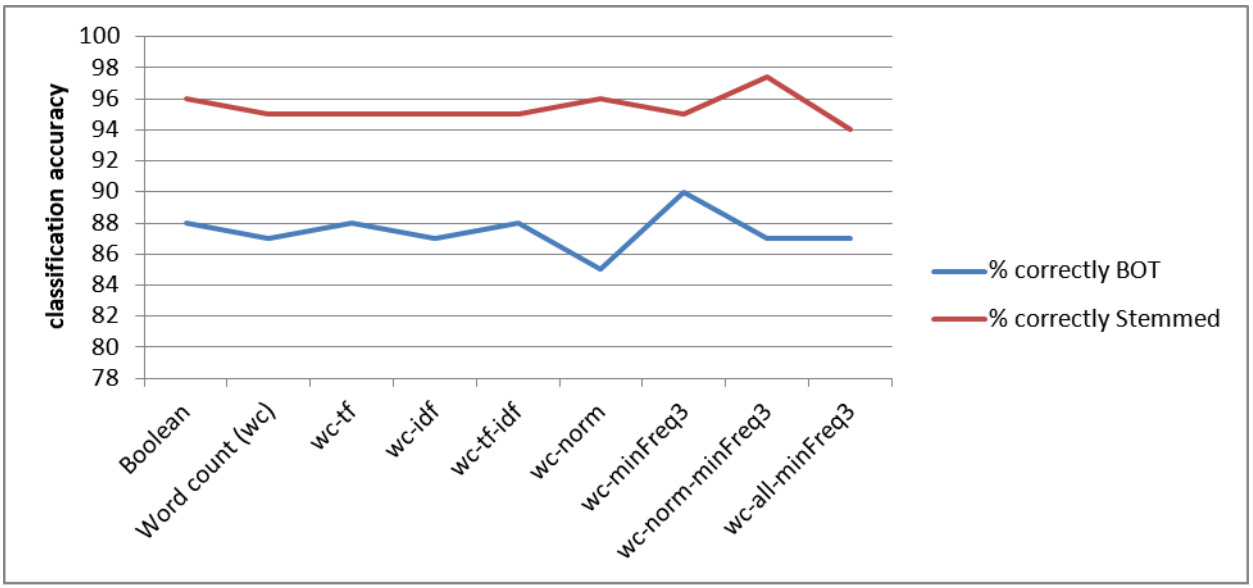

Figure 2: C4.5 classification accuracy for each text preprocessing combinations.

4. Conclusions

Text preprocessing is an important step in text mining. There are many preprocessing combination that can be used for text preprocessing, but it is very difficult to determine the best preprocessing and term weighting. In this paper we evaluated text preprocessing combinations on Arabic text mining. Empirical results showed Term stemming and pruning, document normalization, and term weighting dramatically reduce dimensionality, enhance text representation and directly impact text mining performance.

Acknowledgements

We would like to thank Dr. Alaa El-Halees for his comments.

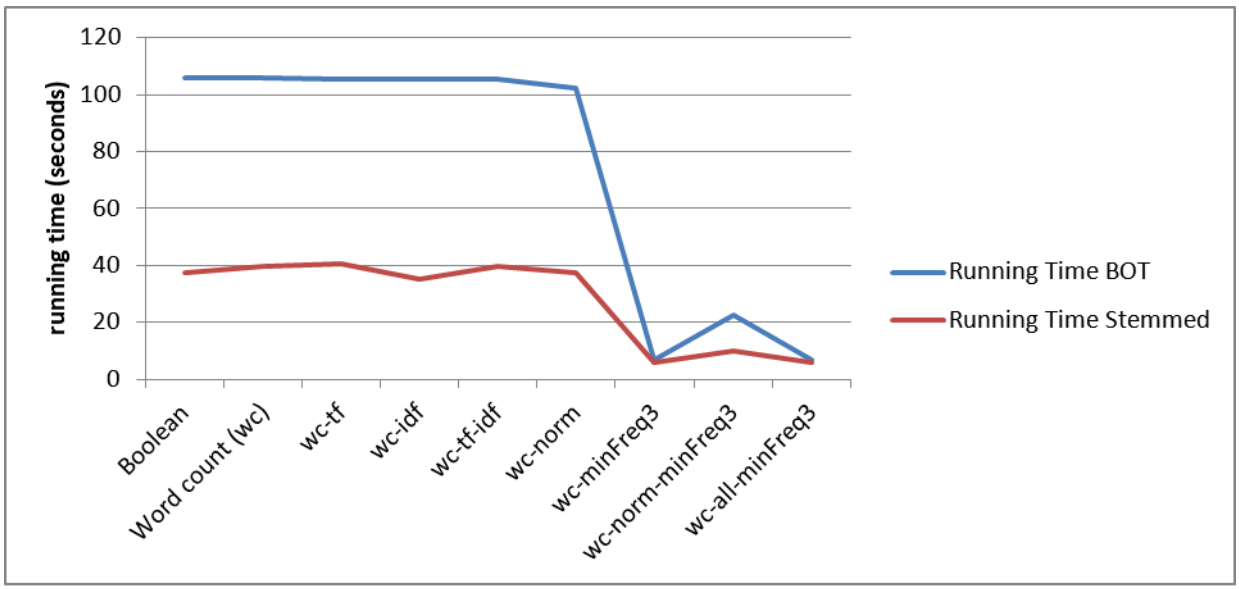

Figure 3: text classification running time for each text preprocessing combinations.

References

1. Al-Marghilani A., Zedan H., Ayesh A., *A GENERAL FRAMEWORK FOR MULTILINGUAL TEXT MINING USING SELF-ORGANIZING MAPS.* The 25th IASTED Int. Multi-Conf: Artificial Intelligence and Applications (AIA'07), Innsbruck, Austria, pp. 520-525, 2007**.**
2. Al-Marghilani A., Zedan H., Ayesh A., *Text Mining Based on the Self-Organizing Map Method for Arabic-English Documents*. Proc. of the 19th Midwest Artificial Intelligence and Cognitive Science Conf. (MAICS 2008), Cincinnati, USA, pp. 174-181 , 2008**.**
3. El-Halees A., *A Comparative Study on Arabic Text Classification*. Egyptian Computer Science Journal Vol. 20 no. 2 May, 2008.
4. Feldman R., Sanger J., The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data**.** Cambridge University Press, 2007.
5. Ghwanmeh S., Applying Clustering of Hierarchical K-means-like Algorithm on Arabic Language. Int. Journal of Information Technology Vol. 3 No 3. 2005**.**
6. Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I., *The WEKA Data Mining Software: An Update*. SIGKDD Explorations, Vol. 11, No. 1, 2009.
7. Jing L., Huang H., Shi H.: *Improved feature selection approach TFIDF in text mining*. Proc. of the 1st int. conf. of machine learning and cybernetics, Beijing, 2002.
8. Khoja S., Garside R., *Stemming Arabic text*. Computer Science Department, Lancaster University, Lancaster, UK, 1999.
9. Quinlan R., *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA. 1993.
10. Said D., Wanas N., Darwish N., Hegazy N.: *A Study of Arabic Text preprocessing methods for Text Categorization*. 2nd Int. conf. on Arabic Language Resources and Tools, Cairo, Egypt, 2009.
11. Salton G., Buckley C.: *Term weighting approaches in automatic text retrieval*. Proc. of information processing & management, Vol. 24, No. 5, pp 513-523, 1998.
12. Taghva, K., Elkhoury, R., Coombs, J.: ***Arabic stemming without a root dictionary*.** Information Technology: Coding and Computing, ITCC, Vol. 1, pp 152 – 157, 2005.