Arabic Document Classification Using Multiword Features

Diab Abuaiadah

Abstract—**Weinvestigate the use of multiword features to improve Arabic document classification. The Arabic language is both morphologically rich and highly inflected. Accordingly it presents more challenges when enhancing Arabic information retrieval to a level comparable to English. The multiword features are modeled as a combination of words appearing within windows of varying sizes. Our experiments show multiword features combined with dice similarity distance outperform the cosine similarity function and produce results that are comparable to TF-IDF representation. Multiword features are under-explored and we believe they have the potential to improve Arabic information retrieval and, in particular, Arabic document classification.**

Index Terms—**Information retrieval, TF-IDF, arabic document classification, multiword features, dice similarity function**

I. Introduction

With the explosive growth of documentation on the web, information retrieval plays a crucial role for many users and vendors dealing with large datasets. In recent years there has been rapid growth in the creation of Arabic documents. Unlike English, not much research has been done regarding information retrieval of Arabic documents [16], [12]. Arabic is a morphologically rich and highly inflected language and consequently the algorithms that were developed for English perform poorly for Arabic.

Document classification is an important dimension of information retrieval. It assigns each document to a category from a predefined list of categories. Several well-known machine learning algorithms have been used to classify documents [4], [15]. The popular TF-IDF (term frequency-inverse document frequency) representation is used in several algorithms. In this method the input document is converted to a bag-of-words where the frequencies of terms are considered and the relative positions of terms in the text are ignored. This dramatically simplifies the computational complexities of the accompanying algorithms and is widely used by researchers at academic institutions and incorporated in several industry products [4], [18], [19], [20].

As opposed to the TF-IDF weighting scheme, a multiword features approach considers the appearance of combinations of words within windows of varying sizes. Proximity is also another form of using multiword features which is a weighting scheme used to factor in the nearness of the terms in the text. The closer the terms are to each other in the text, the higher the score is. Several published papers suggested that proximity improves English information retrieval [22], [17].

Arabic is a morphologically rich and a highly inflected language, and because of this, Arabic stemmers suffer from high stemming error ratios [3]. We predict the use of multiword features could mitigate the negative impact presented by high stemming error ratios and consequently improve Arabic information retrieval. For example الاوسط الشرق (Middle East) is composed of two terms that together refer to a geographical place whereas each word taken separately refers to another unrelated concept. After stemming, we have two words: وسط (Middle) and شرق (East). In Arabic, there are more terms unrelated to "Middle East" whose stem is شرق or وسط than terms in English whose stem is "Middle" or "East". To illustrate this, consider the concept "sunrise" (شروق الشمس). The stem of شروق (rising) is شرق (East). Consequently, for Arabic, the TF-IDF approach may link the query "sunrise" to documents that include the "Middle East" where, for English, this is unlikely to happen. Furthermore, separate documents that contain "sunrise" and "Middle East" respectively may appear more similar in Arabic than in English. This example suggests that it is reasonable to assume that the potential of multiword features for improving Arabic information retrieval is greater than that for English.

In this work we conduct several experiments to explore the impact of multiword features on Arabic document classification. Our results show that for multiword features, the dice similarity function outperforms the cosine similarity function and is comparable to the TF-IDF representation joined with the cosine function. Unfortunately, we were unable to compare our findings with the published results of the best well-known algorithms as all published papers used different in-house datasets which are not publicly available. For our experiments we used the dataset appearing in [1] which contains nine categories, each of which contains 300 documents, which is freely available for downloading.

This rest of this paper is organized as follows: In Section II we present a summary of related work, Section III covers testing methodology and the result of our experiments, and in Section IV we summarize our findings and present opportunities for future work.

II. Related Work

Several studies and experiments have used proximity-based functions to enhance information retrieval. For example, they have been embodied in document ranking, passage retrieval and other information retrieval models [4], [18]. Nevertheless, Tao and Zhai [22] indicate that the use of

Manuscript received March 7, 2013; revised July 12, 2013.

D. Abuaiadah (Abuaiadh) is with the Centre for Business, Information Technology & Enterprise, Wintec, New Zealand (e-mail: Diab.Abuaiadah@Wintec.ac.nz).

proximity and multiword features in information retrieval is underexplored. They used five TREC test collections and showed one of the proximity measures to be highly correlated with document relevance and to significantly improve retrieval effectiveness. In addition, they suggested that performance is sensitive to the parameter used in the engaged proximity function. However, finding such a function could be a challenge. Yuanhua and el. [24] proposed a positioning language model where this model incorporated several proximity-based functions. Their experiments using TREC test collection suggest the Gaussian density function performs the best.

For the English language, only a handful of published papers discussed the use of multiword features for document classification [17], [23] and [27]. For Arabic, to the best of our knowledge, no published work has investigated the use of multiword features for classifying documents. The closest work on the use of proximity functions in Arabic document classification is [26]. In this work Zaki, Mammas, Ennaji and Nouboudused fuzzy entropy and taxonomy to improve the accuracies of document classification.

Several published papers used variations of well-known algorithms to classify Arabic documents. Elkourdi, Bensaid, andRachidi [6] implemented Naïve Bayes algorithms in classifying Arabic documents and reported 68.8% accuracy. The algorithm was applied after the words were stemmed to their rootSyiam, Fayed, and Habib [21] used TF-IDF weighting with several feature selections and achieved 98% accuracy. Khreisat [10] used N-gram (N=3) frequency to illustrate that Dice measure outperforms Manhattan measure. In her work, Khreisat removed stop words, punctuation and diacritics; in some categories, the accuracy (recall) was below 50% for both Dice and Manhattan measures. Using maximum entropy, El-Halees [7] evidenced accuracy of 74.48%. Prior to application of the algorithm, the data was preprocessed using natural language techniques. The results were then compared to other existing systems. Mesleh [14] used the support vector machine algorithm combined with six commonly used feature selection methods. He reported that stemming is of no benefit to classification. Zahran and Kanaan [25] investigated feature selection using a Particle Swarm Optimization algorithm. They showed that this feature selection used with TF-IDF outperforms TF-IDF with the chi-square statistic algorithm. Al-Saleem [2] showed that the Associative Classification algorithm outperformed the Support Vector Machine and the Naïve Bayes algorithms. The average accuracy for Associative Classification was 80.7%, while the accuracy for SVM and Naïve Bayes were 77.8% and 74% respectively. Hattab and Hussein [8] used Syntactical approach to improve the performance of several well-known classification algorithms.

For the English language, the Reuters-21578 text categorization test collection [13] is used to compare results between the different algorithms. For Arabic, all the above published algorithms for classification used different in-house datasets with different numbers and types of categories and other varied characteristics. Consequently, any comparison between the published results is difficult.

III. Testing Methodology, Experiments And Results

Giving the lack of a standard dataset that is publicly and freely available, we used an in-house dataset composed of nine categories, each of which contains 300 documents. The dataset was built manually by collecting documents from well-known Arabic websites, and documents were manually assigned to an appropriate category. This dataset is freely available for downloading. More details are provided in [1].

For implementing the several variations of the multiword features, and for implementing the classification algorithm, we employed an in-house implementation using the Java programming language within the Netbeans environment, running under the Microsoft Windows operating system with two gigabytes memory.

The results are presented as a measurement of recall (accuracy). Recall is the percentage of documents successfully classified. For precision, we look at all documents assigned to a given category and take the percentage of documents that have been correctly classified as belonging to this category. Precision enables us to understand which category attracts the misclassified documents. To explain these measurements, suppose that there are 100 documents included in the testing belonging to a given category. During the classification, 120 documents were assigned to this category, 80 of which are correctly assigned to this category and the remaining 40 documents are incorrectly assigned to this category. In this case the accuracy (recall) is 80% (ratio between 80 and 100) and precision is 66.6% (ratio between 80 and 120). The number of documents in the training set is equal for each category and consequently the number of documents in the testing set is equal for all categories. In our calculations, accuracy is the average accuracy of all categories. Under these conditions, the average precision would be equal to the average recall and therefore was ignored in the experiments for this paper.

The training set for each category is chosen randomly. Each document that is not in the training set is used in the testing set. As with other supervised machine learning algorithms, the training set is used to profile each category and the testing set is used to measure the performance. In this paper, the training set size refers to the training set size for each category.

In our explorations we use two similarity functions: the cosine similarity function and the dice similarity function. With the dice similarity function, the frequencies of the features in the document are ignored and only their existence is acknowledged. Thus, a combination of words adjacent to each other in a window would have the same weight as if they were apart. With the cosine similarity function, we apply IDF (inverse document frequency) to prevent frequent terms from dominating the value of the function.

To assign a category to a document in the testing set, we calculate the similarities between the document and the profile of each category (viewed as the sum of all documents in the training set for the specific category) and assign the document to the category with maximum similarity. In more detail, for multiword combinations, we sum all the combinations in all the windows in the given document. A multiword features may appear several times in adjacent windows. With dice function, this has no effect on the classification.

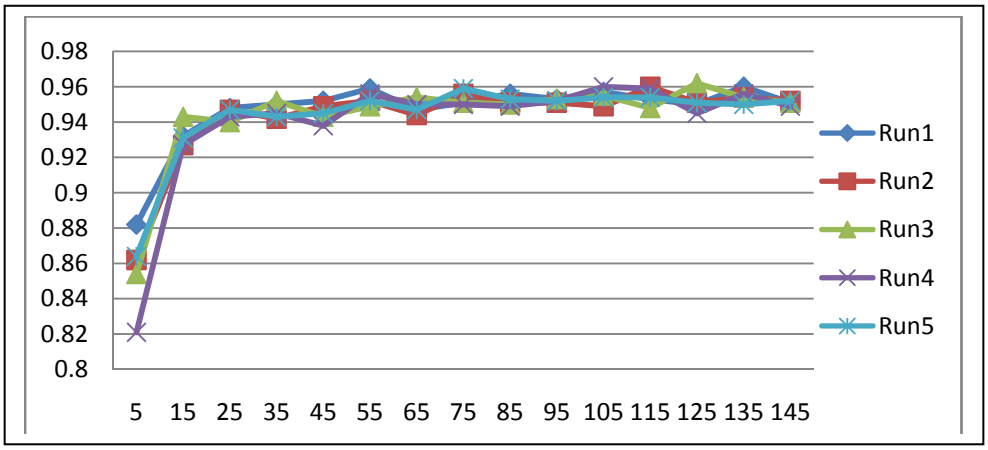

Fig. 1. Accuracies for the cosine function with TF-IDF of the single term as a function of the training set sizes.

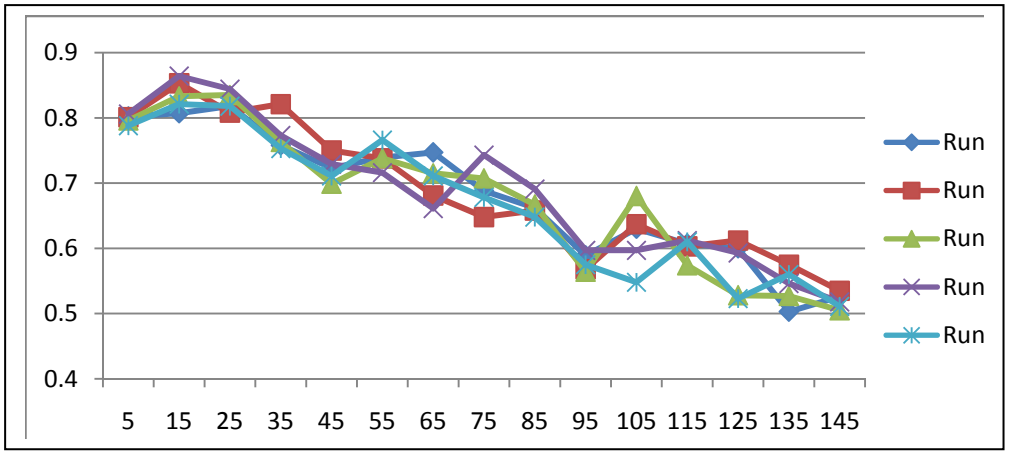

Fig. 2. Accuracies for the dice function with TF-IDF of the single term as a function of the training set sizes

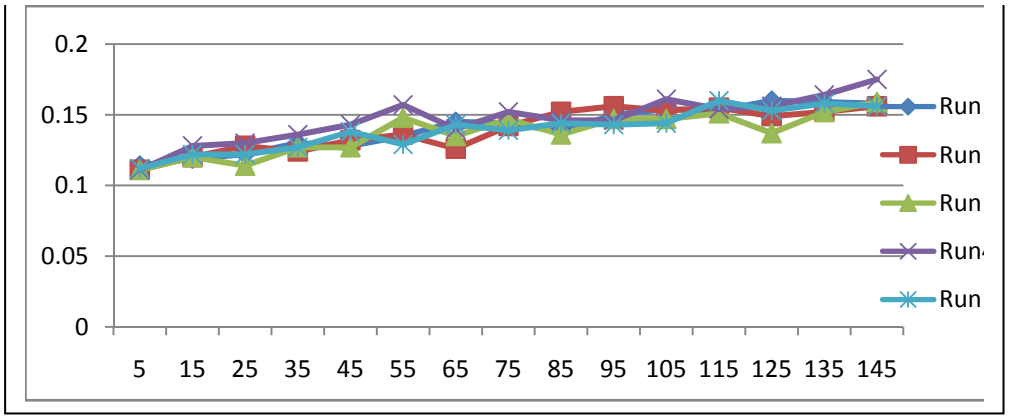

Fig. 3. Accuracies for the cosine function for ordered adjacent terms as a function of the training set sizes.

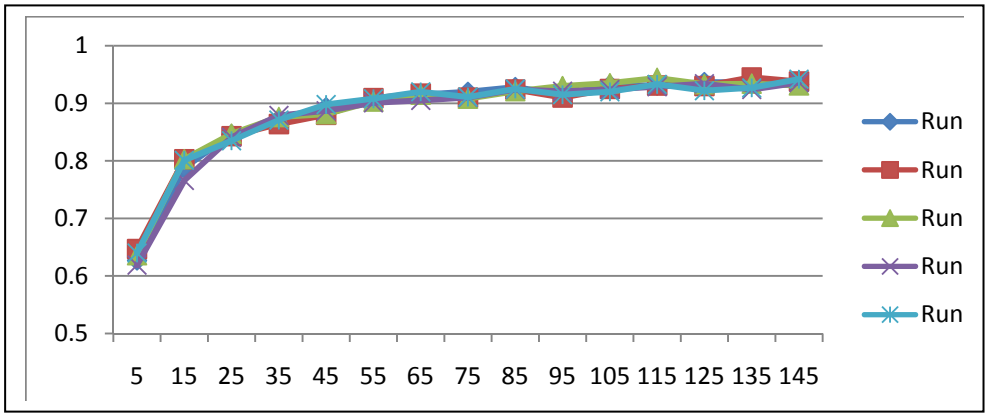

Fig. 4. Accuracies for the dice function for ordered adjacent terms as a function of the training set sizes.

If not explicitly stated otherwise, the training set size for each category is 100 documents and version 3 of the dataset is used. This version represents the original documents modified by the removal of stop words and application of the light10 stemmer [11], considered by many researchers to be the best stemmer for the Arabic language. The accuracies presented in the graphs below are always the average of five independent runs, unless the accuracies of all five runs are plotted in the associated graph. In the latter case we use the labels *Run1…Run5*.

Our experiments analyse the use of multiword features and can be grouped into three different groups: the First group (sub section *III.A*) shows the impact of the training set sizes; the second group (sub section *III.B*) the impact of different stemmers; and the third group (sub section *III.C*) ordered and unordered term combinations in different window sizes.

A. Training Set Sizes

Fig. 1 and Fig. 2 show the accuracies for the cosine similarity function and the dice similarity functions respectively where the single term frequency, TF-IDF, is applied. As mentioned, the dice function does not capture the frequency of each term and only records the existence of the term in the document. For the dice function, it is interesting to see the accuracies significantly deteriorate when the size of the training set increases. This could be explained as the appearance of more terms in more categories diminishing the uniqueness of the profile of each category. As we will see in the following graphs, dramatic improvement results when using multiword features. With the cosine similarity function (Fig. 1), the accuracy stabilizes when the training set size is above 20 documents and the deviation between the different runs is minimal.

Fig. 3 and 4 introduce the finding of our first choice of multiword features. In these experiments, each two ordered adjacent terms are used as a feature (the size of the window is two). Fig. 3 shows that the TF-IDF with the cosine function (TF indicates the frequency of the ordered pair) performs very poorly and the accuracy is only slightly better than a random choice (one out of nine). Increasing the size of the training set brings insignificant improvement. Fig. 4 represents the accuracies for the dice function. We reach over 90% accuracy when the training set size is above 30 documents. The other interesting finding is that the deviation between the different runs is minimal given the fact the documents in the training set are chosen randomly.

B. Stemmers

In this sub section we test the behavior of different versions of the dataset (created with different stemmers) on multiword features. Version 1 is the original text documents, version 2 is version 1 after removing stop words, punctuation and diacritics, version 3 is version 2 after applying the light10 stemmer, version 4 is version 2 after applying the Chen stemmer [5] and version 5 is version 2 after applying Khoja and Garside root-based stemmer [9].

For multiword features we choose two ordered adjacent terms in window size equal to two (ordered pairs) and each three ordered adjacent terms (ordered triples) in window size equal to three. Obviously, the dice function is used and the training set size is 100 documents for each category.

The accuracies for the triples are lower than the accuracies for pairs but this could be attributed to the training set size. There is a drop in the accuracy when removing stop words, punctuation and diacritics (version 2). This is interesting as with several algorithms using TF-IDF, experiments show a gain in accuracy. For pairs, the three stemmers produce a 1% improvement in accuracy compared to the original dataset (version 1). The accuracies for triples are relatively low but this could change for larger training set sizes.

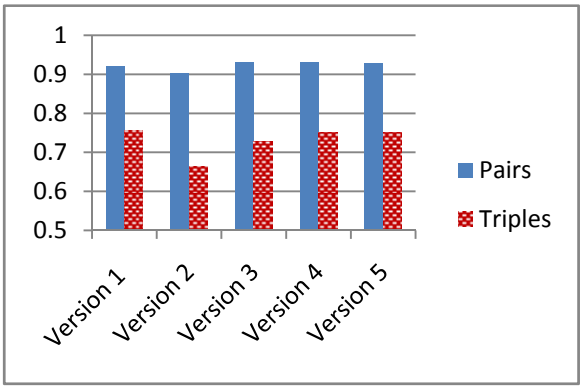

Fig. 5.Accuracies for the five versions of the dataset.

Light stemmers strip off a predefined list of suffixes and prefixes from each term while root-based stemmers attempt to extract the linguistic root of each term. Version 3 and version 4 correspond to light stemmers while version 5 corresponds to a root-based stemmer. Our exercises show minor differences in accuracy between the different stemmers. Arabic is a highly inflected language and on average the term length after a root-based stemmer has been applied, is significantly lower than after light stemmers. The average term lengths for version 1, version 2, version 3, version 4 and version 5 are 4.86, 5.49, 4.09, 3.85 and 3.23 respectively [1].

C. Ordered and Unordered Multiword Combinations in Different Window Sizes

In this subsection we conduct experiments to find the combination of multiword and window sizes that gives the best accuracy. Fig. 6 show the impact of increasing window sizes for ordered and unordered pairs. To explain this, consider the window is $\{(t_i, t_{i+1}, t_{i+2}, t_{i+3}) \mid 1 \le i \le n-3\}$, the size of this window is four and the combinations of ordered pairs are: $(t_i, t_{i+1}), (t_{i+1}, t_{i+2}), (t_{i+2}, t_{i+3}), (t_i, t_{i+2}), (t_i, t_{i+3}), (t_{i+1}, t_{i+3})$ .The number of ordered pairs in a window of size $n$is$\binom{n}{2}$. For the unordered pairs, we add the reverse of each ordered pair to the set of combinations. For each document, and for a given window size, we sum all combinations (multiword features) for all windows in the document. The experiments show unordered pairs perform better than ordered pairs and the accuracies improve when increasing the window size. However, the improvement in accuracy from window size 3 to window size 9 is less than 2%. Unfortunately, our Java implementation runs out of memory when the window size is greater than nine. If the window size is 10, each window may create 45 different combinations for ordered pairs, and 90 different combinations for unordered pairs. Fig. 7 shows the effectiveness of multiword features when using ordered triples, when using several window sizes. It is interesting to see the dramatic improvement in accuracy when increasing

the window sizes. Triplesas multiword features presentcomputational challenges for larger window sizes and our implementation of Java runs out of memory at window size seven. In this scenario, each window may create $\binom{7}{3}$ combinations which are equal to 35.

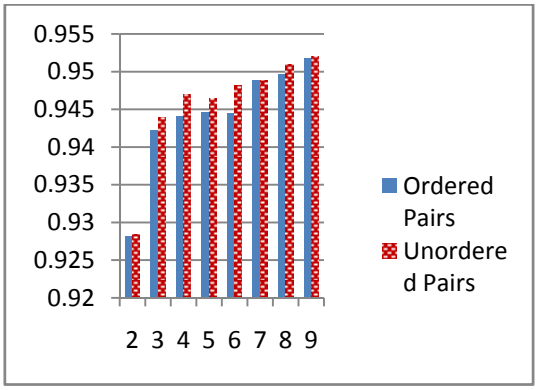

Fig. 6. Accuracies for ordered and unordered pairs as a function of window sizes.

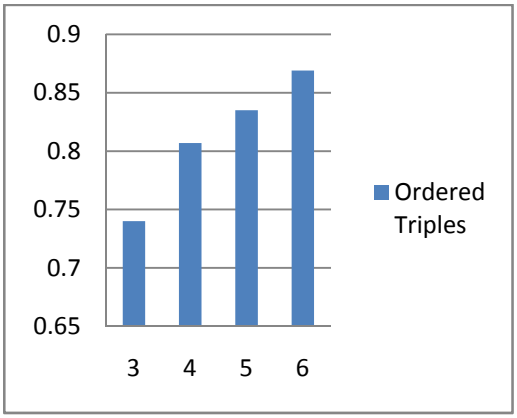

Fig. 7. Accuracies for ordered triples as a function of window sizes.

IV. Conclusion and Future Work

Several interesting conclusions can be drawn from the experiments: The dice similarity function outperforms the cosine similarity function; unordered pairs produce 2% improvement compared to ordered pairs; ordered triples produce poor results, but this could be attributed to a small training set size, and multiword features introduce computational challenges for large window sizes.

The multiword technique is under-explored, and with further research, this could improve information retrieval for Arabic and other highly inflected languages. One way of improving this technique is to employ proximity functions. Such functions measure the closeness of terms in a given window. However, in this case, the dice function does not factor in the different scores. With proximity functions, there is a need to find an appropriate similarity function. Thus, we believe there are several opportunities to investigate multiword features and proximity to enhance Arabic information retrieval and in particular Arabic document classification.

Acknowledgment

I would like to thank Professor Jeffery H. Kingston and Professor Lim Chia Sien for reviewing the paper and for their valuable comments.

References

[1] D. Abuaiadh and W. Abusalah, "On the impact of dataset characteristics on arabic document classification," March 2013.
[2] S. M. A. Saleem, "Associative classification to categorize arabicdata sets," *The International Journal of ACM Jordan*, vol. 1, no. 3, pp. 118-127,September, 2010.
[3] E. T. A. shammari and J. Lin, "Towards an error-free Arabic stemming," in *Proc. of the 2nd ACM workshop on Improving Non English Web Searching,* iNEWS 2008, Napa Valley, California, USA, Oct 30, 2008.
[4] R. B. Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, ACM Press/Addison-Wesley.
[5] C. F. Gey, "Building an arabic stemmer for information retrieval," in *Proc. the Eleventh Text REtrieval Conf.*, National Institute of Standards and Technology, Nov, 2002.
[6] M. Elkourdi, A. Bensaid, A., and T. Rachidi, "Automatic arabicdocument categorization based on the naïve bayesalgorithm," in *Proc. Coling 20th Workshop on Computational Approaches to Arabic Script-based Languages*, Geneva, August 23-27, 2004.
[7] E. Halees, "Arabic text classification using maximum entropy,"*The Islamic University Journal of Series of Natural Studies and Engineering*, pp. 157-167, 2007.
[8] M. Hattab A.K. Hussein, "Arabic content classification system using statistical bayes classifier with words detection and correction," *World of Computer Science and Information Technology Journal*, vol. 2, no. 6, 2012.
[9] S. Khoja and R. Garside, "Stemming arabic text," Technical report, Computing Department, Lancaster University, Lancaster, September 1999.
[10] L. Khreisat, "Arabic text classification using N-Gram frequency statistics - A comparative study," in *Proc. the 2006 International Conference on Data Mining*, Las Vegas, Nevada, USA, June 26-29, 2006.
[11] L. S. Larkey, L. Ballesterosand, and M. E. Connell, "Light Stemming for Arabic Information Retrieval," *Speech and Language Technology*, Springer Netherlands, 2007, vol. 38, pp. 221-243.
[12] L. S. Larkey and M. E. Connell, "Structured queries, language modeling, and relevance modeling in cross language information retrieval,"*Information Processing and Management Special Issue on Cross Language Information Retrieval*, vol. 41, no. 3, pp. 457-473, 2005.
[13] D. Lewis. D. Reuters-21578 text categorization test collection. [Online]. Available: http://kdd.ics.uci.edu/databases/reuters21578/README.txt.
[14] A. A. Mesleh, "Chi square feature extraction based svmsarabiclanguage text categorization system,"*Journal of Computer Science*, vol. 3, no. 6, pp. 430-435.
[15] M. W. Berry, *Survey of text mining: Clustering, classification, and retrieval*, Springer, September, 2003.
[16] A. F. A. Newsri, "Effective retrieval techniques for arabictext," Ph.D. dissertation, RMIT University, Melbourne, Australia, 2008.
[17] P. J. Allan, "Document classification using multiword Features," in *Proc. ACM International Conference on Information and Knowledge Management*, 1998, pp. 124-131.
[18] Y. Rasolofo and J. Savoy, "Term proximity scoring for Keyword-Based retrieval systems," in *Proc. the 25th European Conference on IR Research*, 2003, pp. 207–218.
[19] J. Rocchio, "Relevant feedback in information retrieval"in *the SMART Retrieval System: Experiments in Automatic Document Processing, Ed., Englewood Cliffs*, NJ: Prentice-Hall, pp. 313-323.
[20] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval,"*Information Processing and Management*, vol. 24, no. 5, pp. 513-523.
[21] M. Syiam, Z. Fayed, and M. Habib, "An intelligent system forarabictext categorization," *International Journal of Intelligent* Computing *and Information Sciences*,vol. 6, no. 1, pp. 1-19.
[22] T. Tao and C. Zhai, "An exploration of proximity measures in information retrieval," in *Proc. the 30th annual international ACM SIGIR conf. on Research and development in information retrieval*, New York, USA, 2007, pp. 295–302.
[23] E. M. Voorhees, "Overview of TREC 2001," in *Proc. the Tenth Text Retrieval Conference*, Gaithersburg, Maryland, 13-16 Nov, 2001,pp 1-15.
[24] L.V. Yuanhua, H. E. Jing, V. G. Vydiswaran, K. Ganesan, and Z. X. Cheng, "A study of term proximity and document weighting normalization in pseudo relevance feedback - UIUC at TREC 2009," Million Query Track, 2009.

[25] B. M. Zahran and G. Kanaan, "Text feature selection using particle swarm optimization algorithm,"*World Applied Sciences Journal*, vol. 7, pp. 69-74.
[26] T. Zaki, D. Mammas, A. Ennaji and F. Nouboud,"Classification of Arabic Documents by a Model of Fuzzy Proximity with a Radial Basis Function," *International Journal of Future Generation, Communication and Networking*, vol. 3, no. 4, Dec., 2010.
[27] W. Zhang, T. Yoshida, and X. J. Tang, "Text classification based on Multi-Word with support vector machine," *Knowledge-Based Systems*, vol. 21, no. 8, pp. 879 – 886, 2008.

Diab Abuaiadah (Abuaiadh) holds bachelor's, master's and doctoral degrees in computer science. He earned his doctoral degree from the University of Sydney, graduating in 1996. In addition, he has 14 years of industry experience in developing software products and inventing new algorithms. Abuaiadahworked at IBM research lab for about 10 years and received the highly regarded OIA award from IBM for inventing several algorithms that improved the performance of many widely used IBM products. Since February 2011, Abuaiadah has been employed at the Waikato Institute of Technology (New Zealand) as a principal academic staff member.