Clustering Arabic Tweets for Sentiment Analysis

Diab Abuaiadah
Centre for Business, Information Technology and Enterprise
Waikato Institute of Technology
New Zealand
Diab.Abuaiadah@Wintec.ac.nz

Dileep Rajendran
Centre for Business, Information Technology and Enterprise
Waikato Institute of Technology
New Zealand
Dileep.Rajendran@Wintec.ac.nz

Mustafa Jarrar
Computer Science Department
Birzeit University
Palestine
mjarrar@birzeit.edu

Abstract

The focus of this study is to evaluate the impact of linguistic preprocessing and similarity functions for clustering Arabic Twitter tweets. The experiments apply an optimized version of the standard K-Means algorithm to assign tweets into positive and negative categories. The results show that root-based stemming has a significant advantage over light stemming in all settings. The Averaged Kullback-Leibler Divergence similarity function clearly outperforms the Cosine, Pearson Correlation, Jaccard Coefficient and Euclidean functions. The combination of the Averaged Kullback-Leibler Divergence and root-based stemming achieved the highest purity of 0.764 while the second-best purity was 0.719. These results are of importance as it is contrary to normal-sized documents where, in many information retrieval applications, light stemming performs better than root-based stemming and the Cosine function is commonly used.

1. Introduction

Microblogging and short text messages, such as Facebook posts and Twitter tweets, have been growing in popularity in recent years. Sentiment analysis of this data can be used to extract useful knowledge such as public views about social issues including political opinions, products, services and events. A few examples of sentiment analysis categories include; positive, negative, neutral, sarcastic and non-sarcastic are discussed in [13,15,16].

Document clustering and classification algorithms are essential components in many information retrieval applications [18]. Both algorithms assign a category from a predefined set of categories to each document. Classification requires a training set which could be a challenge for the rapidly changing nature of social media microblogs. Alternatively, clustering algorithms have a higher computational overhead and produce inferior results compared to classification but does not require the effort of building a training set.

Mining short texts is considered to be more challenging than mining normal sized documents as they are likely to be ambiguous and vague [15,16].

As seen in the related works section below, there has been recent efforts to explore how linguistic and metadata preprocessing can improve the performance of clustering and classification algorithms [19,20].

The choice of a similarity function is important and has a crucial impact on the performance of clustering algorithms [21]. The similarity functions evaluated in this study are; the Averaged Kullback-Leibler Divergence (KLD), Cosine, Pearson Correlation (Pearson), Jaccard Coefficient (Jaccard) and Euclidean functions.

Arabic is a Semitic language spoken by more than 300 million people. Preprocessing Arabic text is more challenging than English, as Arabic is morphologically complex and highly inflected [25,26]. In addition to this, Arabic has many dialects corresponding to different geographical regions across the Middle East and north Africa [31].

The motivation of this paper is to evaluate which combinations of linguistic preprocessing and similarity functions are more effective for clustering Arabic tweets. The tweets are clustered into positive and negative categories for sentiment analysis.

The K-Means clustering algorithm is the most widely used algorithm for clustering as it has less computational complexity compared to other hierarchical clustering algorithms [22].

The results in this study clearly show that the impact of selecting a stemmer and the choice of a similarity measure noticeably differ from that of normal-sized documents.

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

The root-based stemmer and the KLD similarity function clearly outperform other selections. An optimized version of the K-means algorithm (OSKM) is used for these experiments.

The paper is organized as follows: Section 2 reviews aspects of the literature that are relevant to this study. Section 3 provides details of the preprocessing techniques and the dataset. Section 4 describes the clustering algorithm used for the experiments. Section 5 provides a detailed discussion of the results. The final section provides a summary and suggestions for future work.

2. Related Work

There are numerous studies that explore the impact of Arabic preprocessing on the performance of information retrieval algorithms. As indicated in the introduction, many highlight the challenge presented by the Arabic language due to its complex morphological structure.

Many papers investigated the impact of stemming and removing stop words when classifying normal-sized Arabic documents and there are relatively few papers that discuss clustering. At the time of writing this paper, the authors could not find any published papers that discuss clustering Arabic short-text documents such as tweets. In this section, we present a brief survey of papers that closely relate to our study.

Many information retrieval applications [9,10,26] found that preprocessing with light stemming produced better results than Khoja root-based stemming [23].

Al-Shammari and Lin [27] and Froud et. al. [5] indicated that Arabic stemmers suffer from high error ratio. Al-Shammari and Lin [28] suggested a better alternative to stemming called lemmatization.

There is a lot of literature on the classification of Arabic short text [2,4,12] and normal sized documents [14]. Many authors focus on the effect of pre-processing such as; stemming, N-gram, prior polarity, part-of- speech (POS) tagging, tokenization, normalization and the removal of URLs, hashtags, twitter targets, repeated words, emoticons, special Twitter words and stop words. A popular algorithm used for document classification is Support Vector Machine (SVM) as it was applied by Shoukry and Rafea [4] and Duwari [2].

For English tweets, Zangerle et al. [3] compares several similarity functions on hashtag recommendations. The Cosine, Dice, Jaccard and Levenshtein functions were included. The Cosine function was found to be superior in this study.

As mentioned above, there are a relatively few papers that investigate the clustering of normal-sized Arabic documents. Ghanem and Ashour [9] focused more on the effect of stemming in this context and found light stemming to be superior to root-based stemming and no preprocessing (stemming).

The work of Froud et al. [5]; Bsoul and Mohd [8] and Froud et al. [6] assess the performance of different similarity functions used for clustering normal-sized Arabic documents. Froud et al. [5] and Froud et al. [6] found that the Cosine, Jaccard and Euclidean functions were more effective than KLD and Pearson functions. Bsoul and Mohd [8] used a different method of root-based stemming (Information Science Research Institute) which was observed to be better than no stemming. They found that the Cosine and Jaccard functions superior to Pearson. It appears that the KLD function is comparable to the Cosine and Jaccard functions but decreases in accuracy as the number of categories increase.

For the English language, Sandhya et al. [11] studied the impact of stemming on four different similarity functions as well. It is interesting to note that these authors found that the Jaccard and Pearson functions were more effective than the Cosine function and that the Euclidean function had a poor performance.

The focus of Abuaiadah [7] was the performance of Standard K-Means and Bisect K- Means. Preprocessing and similarity functions were also investigated for normal-sized Arabic documents. It was found that stemming produced minor improvements, but deteriorated with the combination of KLD and the root-based stemmer. Without preprocessing (stemming and removing stop words), KLD outperformed other similarity functions.

This study compares the effectiveness of preprocessing and different similarity functions in clustering Arabic short text documents.

3. Dataset and Preprocessing

The dataset used is discussed in Abdulla et al. [24] and is freely available for downloading. This dataset contains 2000 tweets where each one is manually labeled as positive or negative. Each tweet is put into a separate text file. Positive and negative tweets are placed in separate directories. This dataset without any processing is referred to as Raw. Figure 1 highlights the main steps for creating an additional three versions of the dataset *NoSW*, *Light10* and *Root*.

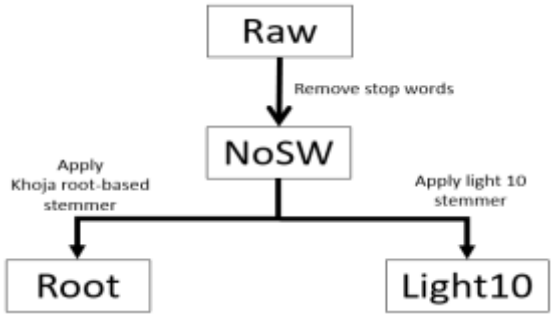

Figure 1: Creating the new three versions of the dataset

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

NoSW is created by removing stop words from the Raw. Stop words are commonly used terms and likely to be spread evenly between all documents in a collection. Consequently, removing stop words speeds up the running time as less memory is required to store documents. Furthermore, many papers argue that removing stop words improve the quality of results for many Arabic information retrieval algorithms for normal-sized documents. Certain stop words could be very useful for sentiment analysis as they explicitly point to a positive or negative sentiment. For example the words 'no' and 'not' indicates a negative sentiment. The list of stop words in this study were built manually.

As detailed in the related works section, the light10 stemmer [10] is considered by many as the best Arabic stemmer for information retrieval applications. It has a predefined set of prefixes and suffixes. This set is used to strip off leading and trailing characters from each term. The aim of light stemming is to apply minimal changes to the original term. Other light stemmers aim to improve the quality of the associated algorithms by modifying the predefined sets of prefixes and suffixes.

Many experiments show that light10 stemmer consistently outperforms the others [9, 26]. In this study, the light10 stemmer is applied to the terms in NoSW to create the light10 version of the dataset.

Kohja's root-based stemmer [23] extracts the root of a term using morphological analysis. Stems extracted by this stemmer are close to (but not necessary the same as) linguistic roots, and they suffers from over stemming and may conflate terms with different concepts to the same root. This stemmer is also applied to the terms in the NoSW dataset to create the KRoot dataset.

Table 1 presents the four versions of one tweet in the corresponding datasets. It is obvious that there are several stemming and preprocessing errors from an Arabic linguistic perspective. For example, in the term النار (English equivalent: 'the fire'), the word 'the' is considered to be a stop word in English but not in Arabic as it is not a separate word in Arabic. An example of a stop that has been removed is اذا (English equivalent: 'if'). When applying the light10 stemmer to the term النار the leading characters 'ال' are removed and it becomes نار (English equivalent: 'fire'). When applying the root-based stemmer, the resulting term is نور (English equivalent: 'light'). A detailed discussion of these errors is beyond the scope of this study.

An in-house Java program is used to create the new three versions of the dataset. Removing stop words is simply scanning each document term by term and removing terms (stop words) appear in the predefined list of stop words. The implementation of the light10 stemmer is also straightforward as the two sets of prefixes and suffixes are predefined. The root-based stemmer is more complex as it involves morphological analysis. We took the Java implementation from the home page of the inventor of this algorithm [1].

The in-house implementation also gathers basic information about the datasets as shown in Table 2.

Table 1: An example of a tweet in the four corresponding datasets.

Dataset	Tweet
Raw	الله يجعل ابوك وامك من اهل النار اذا بتعيد هالحركه مره ثانيه
NoSW	الله يجعل ابوك وامك اهل النار بتعيد هالحركه مره ثاني
Light10	له يجعل ابوك وامك اهل نار بتعيد الحرك مر ثاني
KRoot	له جعل بوأ وامك أهل نورعود هالحركه مرر ثن

The first row of Table 2 captures the total number of terms. The second row indicates the total number of different terms. The third row displays the average length (in characters) of a term. It is worth noting that total number of different terms and the average length of a term for the root based stemmer is conspicuously less than that of light10.

Table 2: basic information for the four versions of the datasets.

Attributes	Raw	NoSW	Light10	KRoot
# terms	18381	14702	14324	14677
# diff terms	7392	7008	5988	2486
# Av length	4.26	4.74	3.96	3.19

At the time of writing this paper, no other Arabic short text datasets were publicly available.

4. Clustering Algorithm

The standard K-means algorithm (SKM) is popular and effective for clustering documents. An overview of the algorithm is presented by Jain [29]. The algorithm receives the number of clusters, K, as an input. It starts by randomly selecting K documents as the centroids of the associated clusters. The algorithm then iterates by assigning each document to the closest centroid using a similarity function. At the end of each iteration, the centroid of each cluster is recalculated based on all the documents it

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

contains. A centroid of a cluster is a representation of all documents assigned to this cluster. It is calculated using the term frequencies (tfidf values) as described in Section 4.2. The algorithm halts when all clusters contain the same documents as the previous iteration i.e. there is no change to any document assignment.

The Bisect K-means algorithm (BKM) [22] starts with all documents in one cluster. The algorithm then uses SKM to bisect a cluster. Typically, the cluster with the maximum number of documents is selected for bisecting but other criteria could be employed. When bisecting a cluster, this bisect is repeated several times. The bisect that appears to produce better results is selected. In many published papers the number of repetitions, ITER, is set to five and the bisect that produces the maximum similarities between documents and the associated centroids is selected [30]. The algorithm continues to select and bisect a cluster until the designated number of clusters (K) is reached. A more detailed and formal description of both algorithms appear in many papers.

When clustering documents into two clusters (positive and negative), as in this paper, SKM and BKM are practically the same algorithm with one exception: BKM repeats the bisect several times and selects the bisect that appears to produce better results. Since BKM repeats the bisection several times, it is considered to be an optimization of SKM (OSKM) in this particular scenario. When OSKM is used in these experiments, ITER is set to 5 and the bisect that produces maximum similarities between documents and the associated two centroids is selected.

Obviously OSKM and SKM have the same theoretical running time (using Big O notation) as OSKM simply repeats SKM a fixed (five) number of times. However, in practical application this could be an important issue depending on the trade-off between improving the clustering results and the run time.

The subsections below presents the quality measures used to evaluate the performance of SKM and OSKM, document representation (TFIDF) and the five similarity functions. Also a brief description of the implementation details is provided.

4.1. Quality Measures

Purity and entropy measures are commonly used to calculate the quality of clustering algorithms. The assignment of documents to clusters by the algorithm is compared to the ground truth. The ground truth is where all the tweets have been labelled by a human expert.

The purity calculation used is defined in Manning et al. [31] and is commonly used to measure the quality of clustering algorithms. In our dataset, there is an equal number of positive and negative tweets and only two clusters are considered. This simplifies the calculation of purity. The outcome of a clustering algorithm will have one cluster with a majority of positive tweets (when compared to the ground truth) and the other cluster will have a majority of negative tweets. Let correct assignment indicate that the clustering algorithm assigned a positive (negative) tweet to the cluster dominated by positive (negative) tweets. In this scenario, purity is the total number of the correct assignments divided by the total number of tweets.

For example, suppose the clustering algorithm assigned 1200 tweets to the first cluster and 800 tweets to the second cluster (2000 documents in total). Suppose 750 tweets in the first cluster are positive based on the ground truth. Obviously the positive tweets are dominant in the first cluster and the negative tweets are dominant for the second cluster (the second cluster must have 800 tweets in total where 450 tweets are negative). The purity in this scenario is calculated as follow:

$\frac{750+450}{1200} = 0.6$.

It is obvious that the purity values will always be above 0.5 and a purity of 1.0 indicates perfect clustering.

The term Entropy specifies how positive and negative tweets are distributed within a given cluster. Lower values indicate better clustering where zero indicates perfect clustering and higher values indicate poor clustering and could be more than one. This measure is calculated as described in [22].

4.2. Document Representation

The bag of words, *BOW*, representation which is often referred to as *tfidf* (term frequency inverse document frequency), is used to represent documents [17] in most information retrieval applications. In this representation the positions of terms within a text are ignored. Let $tf(a,t)$ represents the frequency of the term t in document $a \in D$. The frequencies are normalized by the size of the document. The inverse document frequency aims to reduce the impact of terms which appear in most documents and are not useful to clustering and many other algorithms. It is calculated as $idf(t) = log\frac{|D|}{df(t)}$, where $|D|$ is the total number of documents in the collection and $df(t)$ is the number of documents containing the term t. Each document is represented by $tfidf(a,t) = tf(a,t) * idf(t)$ values.

Let $T = \{t_1, \dots, t_m\}$, be the set of all terms in the collection D and let $w_{t_i,a} = tfidf(a,t_i), 1 \leq i \leq m$, be the *tfidf* value of the term t_i in the document a.

For example, for a selected tweet, the tfidf values for the term السماء ("the sky") in the four versions of the dataset are as follow: 0.166, 0.214, 0.230 and 0.152 for Raw, NoSW, Light10 and Root respectively.

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

4.3. Similarity Functions

There is extensive literature that investigates the impact of similarity functions in all dimensions of information retrieval for documents [21]. However not many published papers compared these functions in the context of short text analysis.

The experiments in this paper included the five widely used similarity functions mentioned in the abstract. A centroid for a cluster is calculated by adding all t*fidf* values of all the documents belong to this cluster and normalizing (divided) by the number of documents in the cluster. Let b be a centroid of a cluster. The similarity is calculated between each document 'a' and a centroid of a cluster 'b' as follow:

1. $\text{Cosine}(a,b) = \frac{\sum_{i=1}^{m} w_{t_i,a} X w_{t_i,b}}{\sqrt{\sum_{i=1}^{m}(w_{t_i,a})^2} X \sqrt{\sum_{i=1}^{m}(w_{t_i,b})^2}}$

2. $\text{Pearson(a, b)} = \frac{m\sum_{i=1}^{m} w_{t_i,a} X w_{t_i,b} - TF_a X TF_b}{\sqrt{m\sum_{i=1}^{m}(w_{t_i,a})^2 - TF_a^2} X \sqrt{m\sum_{i=1}^{m}(w_{t_i,b})^2 - TF_b^2}}$,

   where $TF_a = \sum_{i=1}^{m} w_{t_i,a}$ and $TF_b = \sum_{i=1}^{m} w_{t_i,b}$

3. $\text{Jaccard(a, b)} = \frac{\sum_{i=1}^{m} w_{t_i,a} X w_{t_i,b}}{\sum_{i=1}^{m}(w_{t_i,a})^2 + \sum_{i=1}^{m}(w_{t_i,b})^2 - \sum_{i=1}^{m} w_{t_i,a} X w_{t_i,b}}$

4. $\text{Euclidean(a, b)} = 1 - \sqrt{(\sum_{i=1}^{m} |w_{t_i,a} - w_{t_i,b}|^2)}$

5. $\text{KLD(a, b)} = \sum_{i=1}^{m}(\pi_1 XD(w_{t_i,a}||w_t) + \pi_2 XD(w_{t_i,b}||w_t))$,

   where $\pi_1 = \frac{w_{t_i,a}}{w_{t_i,a}+w_{t_i,b}}$, $\pi_2 = \frac{w_{t_i,b}}{w_{t_i,a}+w_{t_i,b}}$, $w_t = \pi_1 X w_{t_i,a} + \pi_2 X w_{t_i,b}$ and $D(a||b) = a \log\frac{a}{b}$.

4.4. Implementation Details

An in-house Java program implements SKM and OSKM. This program also converts documents from the datasets to corresponding TFIDF representations, implement the five similarity functions and calculates purities and entropies.

For testing purposes the implementation applied SKM to the well-known 20news dataset and achieved purities and entropies for the five similarity functions that are comparable to that of Huang [21]. This validates the implementation of TFIDF, SKM, the five similarity functions and the calculation of purities and entropies. Our implementation of OSKM was also applied to the eight datasets appearing in [22] and achieved comparable purity results.

5. Experiments, Results and Discussion

The performance of SKM and OSKM is affected by the initial random selection of documents (centroids). From looking at the results, two consecutive runs may produce a difference in purities with a margin exceeding 0.15. Therefore, we repeat each run 50 times and calculate the average. In addition to this, we report the standard deviation of purities as well.

Higher standard deviation values show that the algorithm has converged to a different local minimum. This also indicates that the performance is strongly dependent on the initial random selection of centroids. On the other hand, lower values show less dependency.

The results are presented in three parts. The first and second parts discuss the purities and standard deviations of SKM and OSKM respectively. The third part reports the entropies of SKM and OSKM. The Euclidean distance used as a similarity function produced inferior results in all settings and the results are omitted from the figures and discussion to improve readability.

5.1. Purities of SKM

Figure 2 shows the purities of SKM. The first noticeable result is the root-based stemmer is significantly better than the light10 stemmer, removing stop words (NoSW) and without preprocessing (Raw). The second noticeable result is that KLD and Jaccard clearly outperform Cosine and Pearson.

The third noticeable result is that the light10 stemmer does not provide any clear improvement when compared to NoSW and Raw versions of the dataset. In fact, when the light10 stemmer is applied with the Jaccard and Pearson functions, the purity results slightly deteriorated compared to that of NoSW.

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

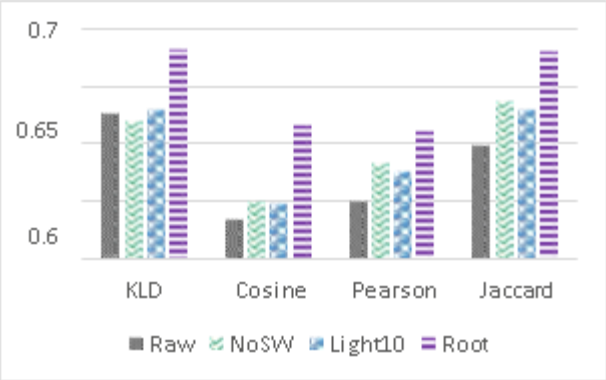

Figure 2: Purities for SKM.

Figure 3 shows the standard deviation of purities corresponding to the 50 consecutive runs. The standard deviations associated with the root based stemmer are higher than the other versions of the dataset. This is worth noting as it indicates there are more opportunities for improving the purity by selecting certain runs.

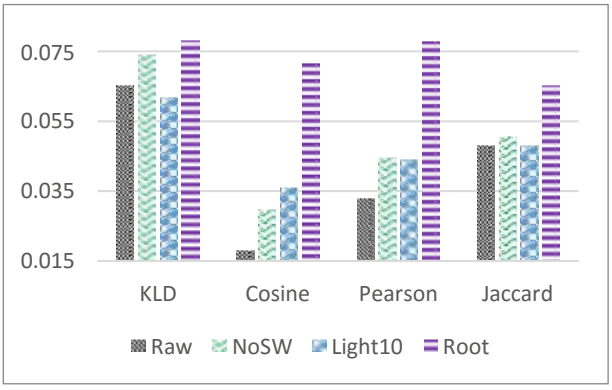

Figure 3: Standard deviations for the purities of SKM.

For Raw, NoSW and Light10, the KLD function has a higher standard deviation compared to the other three functions.

Figures 2 and 3 show that when the Cosine function is applied to the dataset without preprocessing (Raw), there is a very low standard deviation which indicates the purity results are consistently poor.

5.2. Purities of OSKM

Figure 4 presents the purities of OSKM. The root-based stemmer significantly outperformed the light10 stemmer and the other two versions of the dataset for all four functions. In particular, the root based stemmer increased the purity by more than 0.1 compared to that of the light10 stemmer and no preprocessing for all functions.

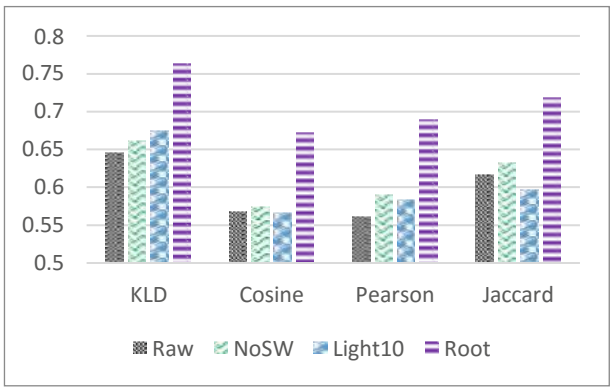

Figure 4: Purities of OSKM.

KLD marginally outperformed the other three functions. In particular, the purity of KLD was higher than the Cosine function by at least 0.07 for all four versions of the dataset.

The combination of KLD and the root-based stemmer achieved the highest purity overall (0.764). The second-best combination (Jaccard and the root- based stemmer) had a purity of 0.719.

OSKM increased the purities for KLD, Cosine and Pearson compared to that of the SKM for all versions of the dataset. For the Jaccard function, it is worth noting that OSKM increased the purity for the root-based stemmer by more than 0.11 and deteriorated the purities for the Light10 stemmer by more than 0.03. This means that repeating the runs and selecting runs with maximum similarities deteriorate the performance. The purities of the Raw and NoSW versions of the datasets were comparable.

Figure 5 shows the standard deviations of the purities for OSKM. The standard deviations of the root-based stemmer are noticeably lower than SKM. This indicates that repeating the runs and selecting the maximum similarities improve and stabilize the results.

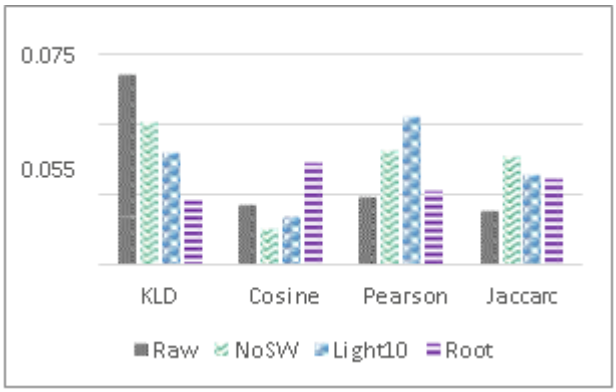

Figure 5: Standard deviations of purities for OSKM.

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

The standard deviation of Pearson and light10 stemmer changed from 0.0443 (SKM) to 0.0576 (OSKM) but the associated average purities are comparable.

5.3. Entropies of SKM and OSKM

Figures 6 and 7 show the entropies of SKM and OSKM respectively. It is shown that improvements in purities leads to an improvement in entropies i.e. lower entropies indicate better clustering. The lowest entropy achieved is 0.234 for the root-based stemmer and the KLD function. This is also the combination that lead to the best purity.

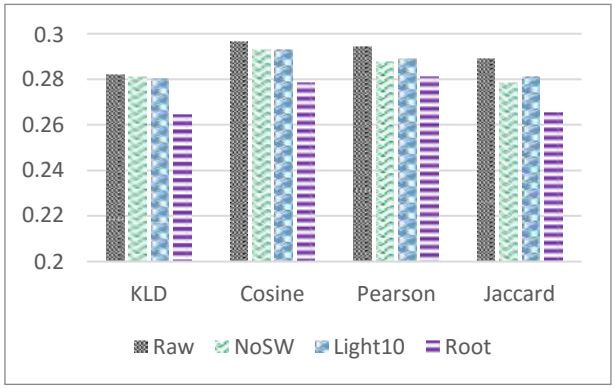

Figure 6: Entropies for SKM.

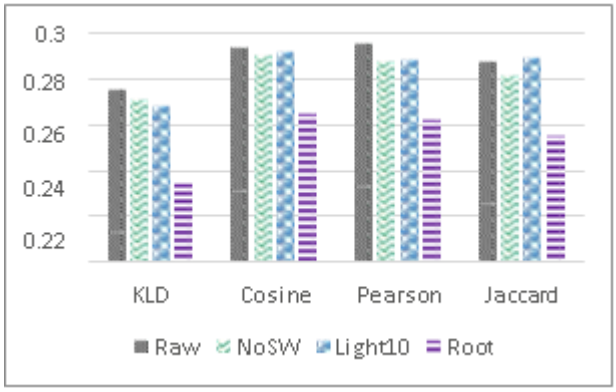

Figure 7: Entropies for OSKM.

For SKM, the entropies corresponding to the root-based stemmer are better than the entropies of the other three versions of the dataset. This is the case for all four similarity functions. This indicates that root-based stemming results have a better distribution.

OSKM does not improve the entropies when compared to SKM in all settings. However, when using the root-based stemmer there are consistent improvements for all similarity functions.

6. Conclusion and Future Work

The experiments in this study clustered Arabic short texts into positive and negative categories for sentiment analysis. Contrary to many published papers on information retrieval applications for normal-sized documents, it is shown that root-based stemming is superior to light10 stemming.

In the same context, KLD outperformed all other similarity functions. The combination of KLD and root-based stemming achieved the highest purity of 0.764 by a margin of at least 0.04.

Although Cosine similarity is used in many products and published papers, we found that Cosine produced inferior purity results to the KLD, Jaccard and Pearson functions in all settings. The best results for the Cosine function occurred when it is combined with root-based stemming but these results are at least 0.09 less than the highest purity.

The results in this paper are of further interest as root-based stemming requires less memory usage and reduces the run time of the associated applications.

It would be interesting to repeat the experiments using more datasets. Unfortunately, the authors were not able to attain access to more datasets at the time of writing this paper. Other interesting avenues for further investigations are lemmatization and distant clustering technique.

As the dataset used in this study contains some dialectal words, which might not be correctly stemmed by Light10 and Kohja's stemmers, some noise might be generated. We plan to extend our work to study how much dialectal words may affect stemmers performance in this application scenario of clustering-based Sentiment Analysis.

7. References

[1] S. Khoja, 2016. [Online]. Available: http://zeus.cs.pacificu.edu/shereen/research.htm. [Accessed 01 August 2016].

[2] R. M. Duwairi, R. Marji, N. Sha'ban and S. Rushaidat, "Sentiment Analysis in Arabic Tweets," in *5th International Conference on Information and Communication Systems (ICICS)*, 2014.

[3] E. Zangerle, G. Wolfgang and G. Specht, "On the impact of text similarity functions on hashtag recommendations in microblogging environments," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 889-898, 2013.

[4] S. R. El-Beltagy and A. Ali, "Open Issues in the Sentiment Analysis of Arabic," in *IIT*, 2013.

[5] H. Froud, R. Benslimane, A. Lachkar and S. A. & Ouatik, "Stemming and similarity measures for Arabic Documents Clustering," in *I/V IEE ISVC*, 2010.

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162

[6] H. Froud, A. Lachkar and S. Ouatik, "Arabic text summarization based on latent semantic analysis to enhance arabic documents clustering," *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 2013.

[7] D. Abuaidah, "Using Bisect K-Means Clustering Technique in the Analysis of Arabic Documents," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 15, no. 3, p. 17, 2016.

[8] Q. Bsoul and M. Mohd, "Effect of ISRI stemming on similarity measure for Arabic document clustering," in *Asia Information Retrieval Symposium*, Springer Berlin Heidelberg, 2011.

[9] O. Ghanem and W. M. Ashour, "Stemming Effectiveness in Clustering of Arabic Documents," *International Journal of Computer Applications*, vol. 5, no. 49, pp. 1-6, 2012.

[10] L. Larkey, L. Ballesteros and M. Connell, "Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis," in *ACM SIGIR*, 2002.

[11] N. Sandhya, Y. S. Lalitha, V. Sowmya, A. D. K.. and D. A. Govardhan, "Analysis of Stemming Algorithm for Text Clustering," vol. 8, no. 5, 2011.

[12] A. Shoukry and A. Rafea, "Sentence-Level Arabic Sentiment Analysis," in *International Conference on Collaboration Technologies and Systems (CTS)*, 2012.

[13] B. Pang and L. Lillian, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1-135, 2008.

[14] M. Althobaiti, U. Kruschwitz and M. Poesio, "Combining Minimally-supervised Methods for Arabic Named Entity Recognition," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 243-255, 2015.

[15] M. Efron, "Information search and retrieval in microblogs," *Information search and retrieval in microblogs." Journal of the American Society for Information Science and Technology*, vol. 62, no. 6, pp. 996-1008, 2011.

[16] J. Akshay, X. Song, T. Finin and B. Tseng, "Why we twitter: understanding microblogging usage and communities.," in *In Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, 2007.

[17] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513-523, 1988.

[18] G. Salton, Automatic text processing: the transformation, analysis, and retrieval of information by computer, Boston: Addison-Wesley Longman Publishing Co., 1989.

[19] A. Go, R. Bhayani and L. Huang, "Twitter Sentiment Classification using Distant Supervision," CS224N Project Report, Stanford, 2009.

[20] A. K. Jose, N. Bhatia and S. Krishna, "TWITTER SENTIMENT ANALYSIS," National Institute of Technology, Department of Computer Science & Engineering, Calicut, Monsoon, 2010.

[21] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand, 2008.

[22] M. Steinbach, G. Karypis and V. Kumar, "A Comparison of Document Clustering Techniques," in *KDD Workshop on Text Mining*, 2000.

[23] S. Khoja and R. Garside, "Stemming Arabic text," Lancaster University, Lancaster, 1999.

[24] N. Abdulla, N. Mahyoub, M. Shehab and N. Al-Ayyoub, "Arabic sentiment analysis: Corpus-based and lexicon-based," in *In Proceedings of The IEEE conference on Applied Electrical Engineering and Computing Technologies (AEECT).*, 2013.

[25] K. Darwish, W. Magdy and A. Mourad, "Language processing for arabic microblog retrieval," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012.

[26] A. Newsri, "Effective Retrieval Techniques for Arabic Text," Melbourne, Australia, 2008.

[27] E. Al-Shammari and J. Lin, "Towards an Error-Free Arabic Stemming," in *Proceedings of the 2nd ACM workshop on Improving non english web searching (iNEWS '08)*, New York, NY, USA, 2008A.

[28] E. Al-Shammari and J. Lin, "A novel Arabic lemmatization algorithm," in *Proceedings of the second workshop on Analytics for noisy unstructured text data*, Singapore, 2008B.

[29] A. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, p. 651–666, 2010.

[30] R. Kashef and M. S. Kamel, "Enhanced bisecting k-means clustering using intermediate cooperation," *Pattern Recognition*, vol. 42, no. 11, pp. 2557-2569, 2009.

[31] C. Manning, P. Raghavan and H. Schütze, Introduction to information retrieval, vol. 1, Cambridge: Cambridge university press, 2008.

[32] M. Jarrar, N. Habash, F. Alrimawi, D. Akra, N. Zalmout: Curras: An Annotated Corpus For The Palestinian Arabic Dialect. Journal Language Resources and Evaluation. Pages(1-31) Volume(50), Issue(219). Springer. 2016

2161-5330/17 $31.00 © 2017 IEEE
DOI 10.1109/AICCSA.2017.162