Design and Evaluation of a Parallel Classifier for Large-Scale Arabic Text

Mohammed M. Abu Tair
College of Science and Technology
KhanYounis, Palestine

Rebhi S. Baraka
Faculty of Information Technology
Islamic University of Gaza
Gaza, Palestine

ABSTRACT

Text classification has become one of the most important techniques in text mining. A number of machine learning algorithms have been introduced to deal with automatic text classification. One of the common classification algorithms is the k-NN algorithm which is known to be one of the best classifiers applied for different languages including Arabic language. However, the k-NN algorithm is of low efficiency because it requires a large amount of computational power. Such a drawback makes it unsuitable to handle a large volume of text documents with high dimensionality and in particular in the Arabic language. This paper introduces a high performance parallel classifier for large-scale Arabic text that achieves the enhanced level of speedup, scalability, and accuracy. The parallel classifier is based on the sequential k-NN algorithm. The classifier has been tested using the OSAC corpus. The performance of the parallel classifier has been studied on a multicomputer cluster. The results indicate that the parallel classifier has very good speedup and scalability and is capable of handling large documents collections with higher classification results.

General Terms

Data Mining, Parallel and Distributed Computing.

Keywords

Arabic text classification, k-NN algorithm, parallel classifier, multicomputer cluster.

1. INTRODUCTION

Automatic text classification (also known as text categorization) is the task of assigning documents to one or more predefined categories based on their content. It has witnessed a growing attention in the last few years [1, 2]. Automatic text classification has been used in many applications such as topic identifications, automatic meta-data organization, documents' organization for databases and web pages [3, 4, 5].

Many algorithms have been used for text classification for different languages including Arabic language such as k-NN [6, 7, 8], Naïve Bayes (NB) [7, 9, 10], Support Vector Machines (SVM) [11, 12], and Decision Tree [11, 13, 14].

Most serial text classification methods, like the k-NN algorithm, take a large amount of running times especially when the volume of text documents available for analysis is big. The huge amount of text documents with high dimensionality (i.e. the features or attributes and in this case they are the words that occur in documents) and in particular in the Arabic language which has a rich nature and very complex morphology requires a large amount of computational power for classification.

To be more precise, the large-scale Arabic text means; the large number of text documents that are represented as records (thousands of documents) and the large number of words that are represented as features or attributes in the vector space model after preprocessing the text (thousands of features) [15].

The k-NN algorithm becomes a standard within the field of text classification for different languages and is included in numerous experiments as a basis for comparison. It has been in use since the early stages of text classification research, and is one of the best classifiers within the field [4, 16]. Furthermore, it is a simple classification algorithm and very easy to implement since it does not require a training phase that most classification algorithms must have. However, the k-NN algorithm is of low efficiency because it requires a large amount of computational power for evaluating a measure of the similarity between a test document and every training document and for sorting the similarities. Such a drawback makes it unsuitable to handle a large volume of text documents with high dimensionality and in particular in the Arabic language which has a rich nature and very complex morphology and for some applications where classification efficiency is crucial such as online text classification, in which the classifier has to respond to a lot of documents arriving simultaneously in stream format. Since text data rapidly increase on the Internet, the scalability of the algorithm is required to handle such massive data.

Parallel and distributed computing is an interesting technique for scaling up the algorithms. It presents a natural and promising method to deal with the problem of efficient classification in large-scale Arabic text collection. The current trend in parallel and distributed computing is clustering. In clustering, powerful low cost workstations are linked through fast communication interfaces to achieve high performance computing. Recent increases in communication speeds, microprocessor clock speeds, and availability of message passing libraries make cluster based computing appealing in terms of both high performance computing and cost effectiveness. Parallel and distributed computing on clustered systems is a viable and attractive proposition due to the high communication speeds of modern networks [17].

This paper presents the development of a parallel classifier for large-scale Arabic text that achieves the enhanced level of speedup, scalability, and accuracy. The proposed classifier is based on the sequential k-NN algorithm. The platform comprises a set of processors and their own exclusive memory (multicomputer cluster) which is a viable and attractive method due to the high communication speeds of modern networks. This platform is programmed using send and receive primitives; Libraries such MPI provide such primitives.

The rest of this paper is organized as follows: Section 2 reviews related works. Section 3 presents the sequential k-NN algorithm. Section 4 describes the text pre-processing steps. Section 5 describes the proposed parallel classifier. Section 6 presents the experiments and the results. Finally, Section 7 presents the conclusion and future directions.

2. RELATED WORKS

In order to improve the efficiency of sequential classification algorithms for text classification, some researches have been conducted in this area.

Lianga et. al [15], proposed a parallel learning algorithm. The parallel algorithm is based on the k-NN algorithm. They evaluated the parallel implementation on Compute Unified Device Architecture (CUDA) enabled Graphics Processing Unit (GPU). The advantage of this method is the highly parallelizable architecture of the GPU. Recent development in GPUs has enabled inexpensive high performance computing for general-purpose applications. Due to GPU's tremendous computing capability, it has emerged as the co-processor of the Central Processing Unit (CPU) to achieve a high overall throughput. CUDA programming model provides the programmers adequate C language like APIs to better exploit the parallel power of the GPU and manipulate it. At the hardware level, CUDA-enabled GPU is a set of Single Instruction Stream, Multiple Data Stream (SIMD) processors with 8 stream processors. They used synthetic data generated by MATLAB for the purpose of evaluation where the number of data objects is 262144 records. Their experiment showed good scalability on data objects. The result shows that CUk-NN is suitable for large scale dataset. However, since SIMD processors are specially designed, they tend to be expensive and have long design cycles and the scalability of the processors is limited.

Duwairi et. al [18], compared three dimensionality reduction techniques; stemming, light stemming, and word cluster. The purpose of employing the previous methods is to reduce the size of documents vectors without affecting the accuracy of the classifiers. They used k-NN to perform the comparison. The comparison metric includes size of documents vectors, classification time, and accuracy (in terms of precision and recall). They used Term Frequency (TF) as a weighting scheme for feature selection. They collected 15,000 documents belonging to one of three categories (sport, economic, education). Each category has 5,000 documents. They split the corpus; 9,000 documents for training and 6,000 documents for testing. In terms of vector sizes and classification time, the stemmed vectors consumed the smallest size and the least time necessary to classify a testing dataset that consists of 6,000 documents. The light stemmed vectors superseded the other three representations in terms of classification accuracy. The feature selection and reduction strategies can decrease the computation complexity, reduce the dimensionality, and improve the accuracy rate of classification. However, this approach could not do well in the case of reducing computation complexity for text documents with high number of distinct words and in particular in the Arabic language which has a rich nature and very complex morphology. Also, this approach reduces the features but what is the solution in the case of large volume of text documents which increase the computation complexity.

Guan and Zhou [19], proposed a training-corpus pruning based approach to speedup the k-NN algorithm. It depends on the removal of the noisy and superfluous documents in training corpuses, which leads to substantial classification efficiency improvement. They used clustering-based feature selection method that treating each training class as a distinctive cluster, then using a genetic algorithm to select a subset of documents features. They used Apte corpus; the number of documents sample is 5773 in ten categories, 2447 documents prepared for testing. The pruning strategy can reduce the size of training corpus significantly, decrease the computation complexity, but it can damage the classification quality of k-NN for text classification, any removal of training documents may aggravate the sparseness of the text corpus, which leads to a degradation of the k-NN classifier.

Buana et. al [20], proposed a method that combine traditional k-NN algorithm and k-Means clustering algorithm. They used TF-IDF as the weighting scheme for feature selection. They group all the training samples of each category by k-Means algorithm, and take all the cluster centres as the new training samples, the modified training samples are used for classification with the k-NN algorithm. The results show that the combination of the proposed algorithm in this study has a percentage accuracy reached 87%, an average value of f-measure evaluation= 0.8029 with the best k-values= 5 and the computation takes 55 second for one document. They collected corpus from news website www.detik.com and www.kompas.com. The number of documents sample is 802 with 5915 terms and 6 categories that are, General News, Business Economics, Education and Science, Health, Sports, and Technology. 60 documents prepared for testing, each category of 10 documents. The combination of traditional k-NN algorithm and clustering algorithm can reduce the time complexity of traditional k-NN algorithm. However, the clustering algorithm can take a large amount of time for clustering the training samples especially in the case of the large volume of text documents.

Ruoming et. al [21], proposed a parallel learning algorithm. The parallel algorithm is based on the k-NN algorithm. They evaluated the parallel implementation on a multiprocessor with shared memory that connect multiple processors to a single memory system. They experimented with a 800 MB main memory resident dataset. The reduction object in this algorithm’s parallel implementation is the list of k-nearest neighbors. The speedup results was suitable up to four processors. However, sharing memory in this way can easily lead to a performance bottleneck and the scalability of the processors is limited.

Tekiner et. al [22], proposed a parallel learning algorithm for part of speech tagging. The parallel algorithm is based on the Maximum Entropy algorithm. They used Genia which is a sequential POS tagger as a baseline for comparison. Genia is built with maximum entropy and it is specifically tuned for biomedical text. They implemented a parallel version of Genia tagger application and performance has been compared. The focus has been particularly on scalability of the application. Scaling up to 96 processors has been achieved and a hundred thousand abstracts have been processed in less than 5 minutes, whereas serial processing would take around 8 hours. The parallel implementation of Genia tagger is done using MPI library. They used two datasets; the first dataset is Medline which is a collection of Medline abstracts contain around 1.7 billion words, another dataset contains 1 Million abstracts. This work supports our approach in terms of using multicomputer cluster which is a viable and attractive method due to the high communication speeds of modern networks.

3. THE SEQUENTIAL k-NN ALGORITHM

The k-NN algorithm [23]: was first described in the early 1950. It is based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it. The training tuples are described by n attributes. Each tuple represents a point in an n-dimensional space. In this way, all of the training tuples are stored in an n-dimensional pattern space. When given an unknown tuple, a k-NN classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are the k nearest neighbors of the unknown tuple. Closeness is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples, X=(x1,x2,…,xn) and Y=(y1,y2,…,yn) is:

$$d(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \qquad (1)$$

The pseudo code of the sequential k-NN algorithm is shown in Algorithm 1.

```
Input: Training set D = {( x1 , y1), . . . , (xn, yn)}.
        x′ new instance to be classified.
Output: predicted class label y′ for x′.
ALGORITHM
FOR each labeled instance (xi, yi ) calculate d(xi , x′) from (1)
Order d(xi , x′) from lowest to highest, (i = 1, . . . , n).
Select the k nearest instances to x′: Dx′.
Output y′ that is the most frequent class in Dx′.
```

Algorithm 1. The k-NN algorithm [24].

4. TEXT PRE-PROCESSSING

One of the widely used methods for text mining presentations is viewing text as a Bag Of Tokens (BOT) (words, n-grams). Under that model we can already classify text [6].

Some pre-processing in the corpus is performed. It includes tokenizing string to words, normalizing the tokenized words, applying stop words removal, applying the suitable term stemming and pruning methods as a feature reduction techniques, and finally applying the suitable term weighting scheme to enhance text document representation as feature vector. We use the open source machine learning tool Rapid Miner for text pre-processing.

In linguistics, morphology is the identification, analysis and description of the structure of morphemes and other units of meaning in a language like words, affixes, and parts of speech. For Arabic Language, there are two different morphological analysis techniques; stemming and light stemming. Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. Stemming algorithm by Khoja [25] is one of the well known Arabic stemmers. Light stemming, in contrast, removes common affixes from words without reducing them to their stems and keeps the words' meanings unaffected [1, 2]. A light stemmer [26] is a standard Arabic light stemmer.

The aim of term weighting is to enhance text document representation as feature vector. Popular term weighting schemes are Binary Term Occurrences (BTO), Term Frequency (TF), Term Occurrences (TO), and Term Frequency-Inverse Document Frequency (TF-IDF). BTO indicates absence or presence of a word with Booleans 0 or 1 respectively. TF(t,d) is the number that the term t occurred in the document d. TO is the number of occurrences of term t in the document d. TF-IDF is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. 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). 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) [1, 2, 27, 28, 29].

5. THE PROPOSED PARALLEL CLASSIFIER

This section describes the proposed parallel classifier model including the decomposition and mapping techniques and the steps of the proposed parallel classifier.

The parallel classifier model is a way of structuring a parallel classifier by selecting the most suitable decomposition and mapping techniques and applying the appropriate strategy to minimize interactions [17].

5.1 Decomposition Technique

The first step in developing a parallel algorithm is to decompose the problem into tasks that can be executed concurrently by identify the data on which computations are performed, then partition this data across various tasks.

The task performs the computations with its part of the data. In our classifier, the input training data partitioning is the natural decomposition technique because the output (the computed distances) is not clearly known a-priori.

5.2 Mapping Technique

Once a problem has been decomposed into concurrent tasks, these must be mapped to processors (that can be executed on a parallel platform). In this classifier, we use the static mapping technique that distributes the tasks among processes prior to the execution of the program. The scheme for this static mapping is mapping based on data partitioning because our data represented in a two-dimensional array. So, the most suitable scheme used for distributing the two-dimensional array among processes is the row-wise 1-D block array distribution that distributes the array and assign uniform contiguous portions of the array to different processes.

According to the previous selected decomposition and mapping techniques, the suitable parallel model is the master-slave model in which the master processor generates the work and allocates it to the worker processors.

Since the most time consuming in the k-NN algorithm taken by the calculation of the distance between the query-instance and all the training samples, and the sorting of the distances to determine nearest neighbors based on the k-th minimum distance. This classifier takes into consideration these two factors by partitioning the work of distances computation and sorting among several worker processors. The pseudo code of the proposed parallel classifier is shown in Algorithm 2.

```
     Input: Training set D = {( x1 , y1), ….. , (xn, yn)}.
            x′ new document to be classified.
     Output: predicted class label y′ for x′.
     ALGORITHM
1    The master processor divides D equally among
     worker processors and sends a one partition for
     each of them.
2    While True:
a        If processor = master:
i            Load x′.
ii           Send x′ to the worker processors.
viii         Receive Dx′ from the worker
             processors and put it in TDx′.
ix           Order TDx′ from lowest to highest.
x            Output y′ that is the most frequent
             class in TDx′.
b        Else:
iii          Receive x′ from the master
             processor.
iv           FOR each labeled instance (xi, yi )
             calculate d(xi , x′) from (1).
v            Order d(xi , x′) from lowest to
             highest, (i = 1, . . . , n).
vi           Select the k nearest instances to x′:
             Dx′.
vii          Send Dx′ to the master processor.
```

Algorithm 2. The proposed parallel classifier.

6. EXPERIMENTAL RESULTS AND EVALUATION

This section gives the experimental results to provide evidence that our parallel classifier design can improve both the computational efficiency and the quality of classification.

The sequential k-NN algorithm has been implemented using C++ programming language to serve as a baseline when it compares with the proposed parallel classifier to give a fair comparison. The proposed parallel classifier has been implemented using C++ programming language and the MPI library.

The target platform for the experiments is a cluster of computers and their own exclusive memory connected through fast local area network. The cluster consists of 14 node, all nodes have the same specifications; Intel(R) Core(TM) i3-2120 CPU @ 3.30 GHz, 4.00 GB RAM, 320 GB hard disk drive. The sequential k-NN algorithm and the proposed parallel classifier have been implemented on Windows 7 operating system, and we have used the parallel message passing software MPICHI2 that offers small latencies and high bandwidths.

6.1 The Corpus

We use the largest freely public Arabic corpus of text documents which called OSAC from [30] to perform our experimentations. The OSAC corpus is available publically at [31].

The OSAC Arabic corpus collected from multiple websites as presented in Table 1, the corpus includes 22,428 text documents. Each text document belongs to 1 of 10 categories (Economics, History, Entertainments, Education and Family, Religious and Fatwas, Sports, Heath, Astronomy, Low, Stories, and Cooking Recipes). The corpus contains about 18,183,511 (18M) words and 449,600 district keywords after stop words removal. We generate all text representations for OSAC corpus to evaluate the obtained classification results. The generated text representations for OSAC corpus are: (Light stemming, Stemming) and percentual Term pruning (min threshold = 3%, max threshold = 30%) with (TF-IDF, TF, TO, BTO).We have described these text representations in more details in section 4.

Table 1. The OSAC corpus.

Category	Number of text documents	Sources
Economic	3102	bbcarabic.com – cnnarabic.com – aljazeera.net- khaleej.com – banquecentrale.gov.sy
History	3233	تاريخ الحكام www.hkam.net – moqatel.com – التاريخ altareekh.com – تاريخ الإسلام islamichistory.net
Education and family	3608	صيد الفوائد saaid.net – نصائح للسعادة الأسرية naseh.net – المربي almurabbi.com
Religious and Fatwas	3171	CCA corpus – EASC corpus – moqatel.com – شبكة الفتاوى الشرعية islamic-fatwa.com – صيد الفوائد saaid.net
Sport	2419	bbcarabic.com – cnnarabic.com – khaleej.com
Health	2296	العيادة الالكترونية dr-ashraf.com – CCA corpus – EASC corpus – W corpus – صحة الطفل kids.jo – العلاج البديل العربي arabaltmed.com
Astronomy	557	الفلك العربي arabstronomy.com – الكون نت alkawn.net – بوابة الفلك المغربية bawabatalfalak.com – الفلك -موسوعة النابلسي nabulsi.com – www.alkoon.alnomrosi.net
Low	944	القانون الليبيlawoflibya.com – قانون كوم qnoun.com
Stories	726	CCA corpus – قصص الأطفال kids.jo – صيد الفوائد said.net
Cooking Recipes	2372	aklaat.com – fatafeat.com
Total	22,428	

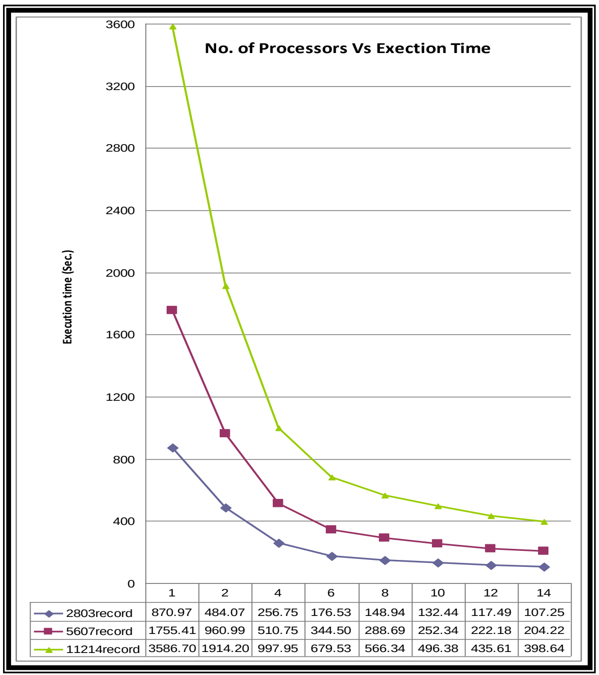
6.2 Discussion of the Parallel Classifier Results

The largest text representation for OSAC corpus which is (Light stemming + percentual term pruning (min threshold = 3%, max threshold = 30%) + TF-IDF), (22,428 documents that are represented as records and 2114 words that are represented as attributes) has been used to evaluate the proposed parallel classifier using different performance metrics for parallel systems such as execution time, parallel overhead, speedup, and efficiency which determines the scalability.

For evaluation purposes, the largest generated text representation for OSAC corpus has been splitted into two parts; 50% of the corpus for training (11214 documents) and the remaining 50% for testing (11214 documents) using stratified sampling which keep class distributions remains the same after splitting.

We have executed the parallel classifier varying the number of processors from 2 to 14; also we varied the number of tested documents to observe the effects of different problem sizes on the performance. Three sets were used with the number of tested documents 2803, 5607, and 11214 documents.

Figure 1 shows the curves of execution time for the classifiers on the OSAC corpus. The time curve decreases from 1 processor until using 14 processors.



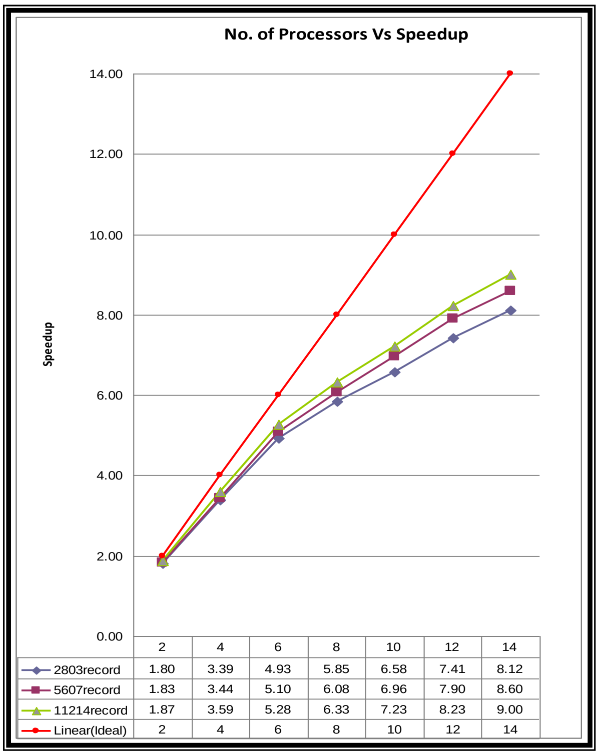
	1	2	4	6	8	10	12	14
2803record	870.97	484.07	256.75	176.53	148.94	132.44	117.49	107.25
5607record	1755.41	960.99	510.75	344.50	288.69	252.34	222.18	204.22
11214record	3586.70	1914.20	997.95	679.53	566.34	496.38	435.61	398.64

Fig 1: The curves of execution time for the two classifiers.

Several observations can be made by analyzing the results in Figure 1. First, the sequential k-NN algorithm spent a lot of time classifying the text documents. Second, the proposed parallel classifier clearly reduce the sequential time. Notice that the sequential k-NN algorithm takes about 1 hour to classify this collection, while the proposed parallel classifier reduces this time to 6 minutes on 14 processors.

Also, the speedup which gained from this parallelization is computed. Figure 2 demonstrates the relative speedup.

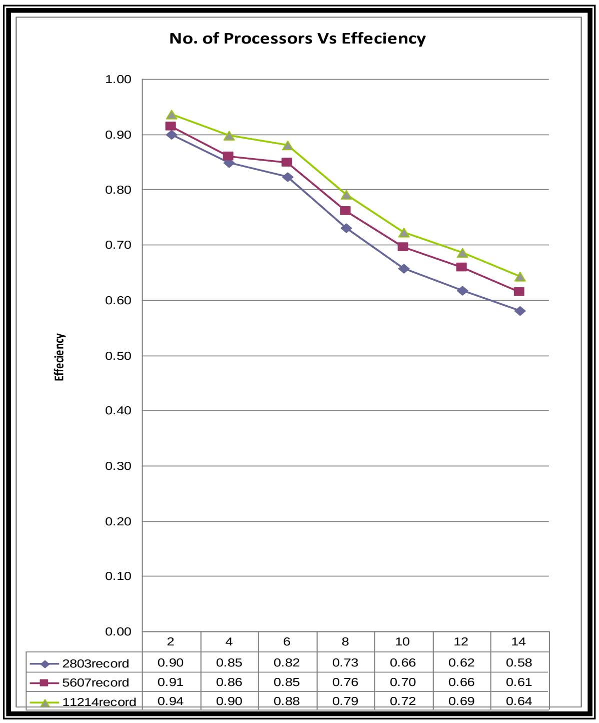
The speedup curves increase linearly in some cases. For example, on the largest tested set (11214 documents), it achieves the relative speedups of 1.87, 3.59, 6.33, and 9.00 on 2, 4, 8, and 14 processors, respectively. When it accesses to a smaller set of tested documents, the speedup curves tend to drop from the linear curve. The classifier achieves the relative speedups of 1.83, 3.44, 6.08, and 8.60 on 2, 4, 8, and 14 processors, respectively. The smallest tested documents sizes give the same trend. If we increase the number of processors further, the speedup curves tend to significantly drop from the linear curve. For a given problem instant, the relative speedups saturates as the number of processors is increased due to increased overheads. This is a normal situation when the problem size is fixed as the number of processors increases. However, it can be solved by scaling the problem size. For example, in Figure 3, the speedups for three sets on 4 processors improve from 3.39 to 3.59, on 8 processors improve from 5.85 to 6.33, and on 14 processors improve from 8.12 to 9.00. It can be seen that the parallel classifier yields better performance for the larger data sets.



	2	4	6	8	10	12	14
2803record	1.80	3.39	4.93	5.85	6.58	7.41	8.12
5607record	1.83	3.44	5.10	6.08	6.96	7.90	8.60
11214record	1.87	3.59	5.28	6.33	7.23	8.23	9.00
Linear(Ideal)	2	4	6	8	10	12	14

Fig 2: The relative speedup of the proposed parallel classifier.

From the speedup, the efficiency can be computed. Figure 3 illustrates the efficiency curves.



	2	4	6	8	10	12	14
2803record	0.90	0.85	0.82	0.73	0.66	0.62	0.58
5607record	0.91	0.86	0.85	0.76	0.70	0.66	0.61
11214record	0.94	0.90	0.88	0.79	0.72	0.69	0.64

Fig 3: The efficiency curves of the proposed parallel classifier.

As we note from Figure 3, the value of efficiency is between zero and one, the efficiency decrease as the number of processing elements is increased for a given problem size and this is common to all parallel programs due to increased overheads.

Also, we note that the efficiency of the parallel classifier increases if the problem size is increased (from 2803 documents to 11214 documents) while keeping the number of processing elements constant.

It can be seen that the parallel classifier is a scalable parallel system because the efficiency can be kept constant as the number of processing elements is increased, provided that the problem size is increased (from 2803 documents to 11214 documents).

Also, the parallel overhead can be computed. Figure 4 illustrates the parallel overhead curves.

As we note from Figure 4, the parallel overhead of the parallel classifier increases as we increase the number of processing elements for a given problem size. This is a normal situation when the problem size is fixed as the number of processors increases. However, it can be solved by scaling the problem size. we note that the parallel classifier has a parallel overhead that decreases as the data set increases (from 2803 documents to 11214 documents). It can be seen that our parallel classifier yields better performance for the larger data sets.

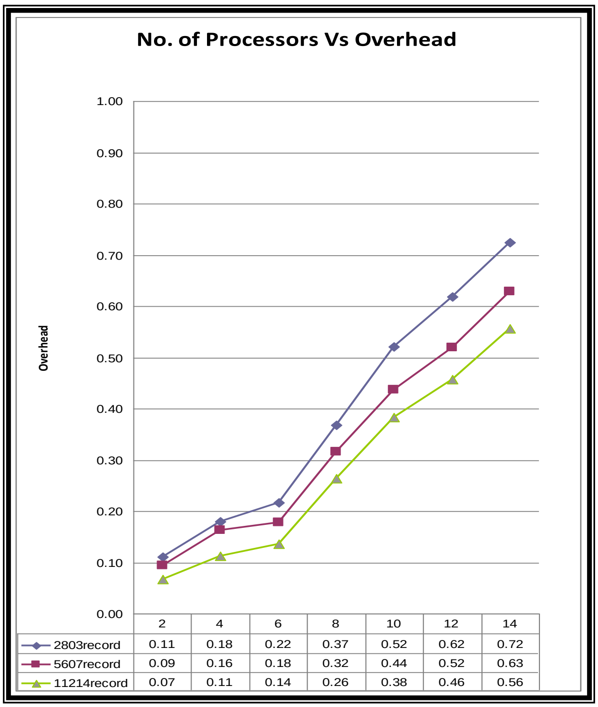


Fig 4: The parallel overhead of the proposed parallel classifier.

6.3 Discussion of the Classification Results

To ensure that the classifier works well with the tested documents, we also examined the quality of the classification. we split all generated text representations of OSAC corpus (we have described these text representations in section 6.1) into two parts; 50% of the corpus for training (11214 documents) and the remaining 50% for testing (11214 documents) using stratified sampling which keep class distributions remains the same after splitting. We split the corpus in this way to achieve higher classification results.
For the purpose of evaluating the classification results, we use confusion matrices that are the primary source of performance measurement for the classification problem. We have evaluated the obtained classification results using different classification measures such as accuracy, precision, recall, and F-measure which are generally accepted ways of measuring systems' success in this field.

The average classification results are depicted in Figure 5. The morphological analysis (stemming, light stemming), term pruning and term weighting schemes (TF-IDF, TF, TO, BTO) have obvious impact on the classifier performance as shown in Figure 5. The Figure emphasizes that light stemming and TF representation with k=10 has the best classification results, this is because light stemming is more proper than stemming from linguistics and semantic view point and keeps the words meanings unaffected. The Figure also emphasizes that the classifier is very sensitive to term weighting schemes because it depends on distance function to determine the nearest neighbors. For example, the BTO weighting scheme has the worst classification results because the text representation is 0 or 1.

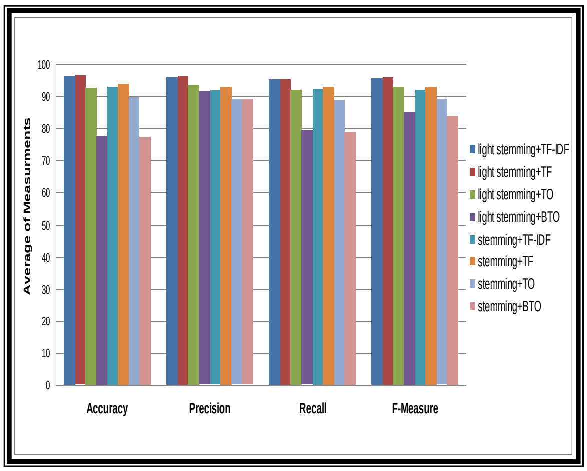


Fig 5: The classification results for OSAC text representations.

Figure 6 shows the classification results for the best text representation of OSAC corpus (light stemming + TF) in each of the domain category. From Figure 6 we can see that the best performance is recorded in Cooking Recipes domain that because Cooking Recipes has limited space of words that are limited and cleared comparing to other domains. Also, it shows that Stories has lowest performance may be that also because Stories have a large space domain.

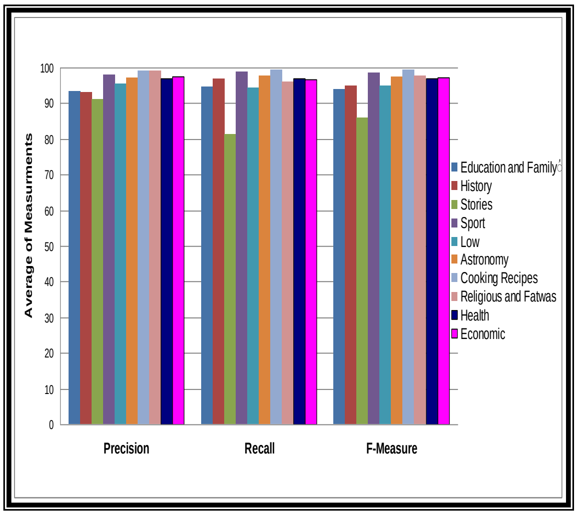


Fig 6: The classification results for light stemming + TF.

7. CONCLUSION AND FUTURE WORKS

In this paper, a parallel classifier for large-scale Arabic text has been introduced. The proposed parallel classifier is based on the sequential k-NN algorithm. The parallel classifier has been tested using the OSAC corpus. The parallel classifier has been implemented on a multicomputer cluster that consists of 14 computers. The experimental results on the performance indicate that the parallel classifier design has very good speedup characteristics when the problem sizes are scaled up. Also, classification results show that the proposed classifier has achieved accuracy, precision, recall, and F-measure with higher than 95%.

There are several directions for improvement and future investigation. This work can be extended to cover larger computer clusters and text corpora to assess the performance of our parallel implementation. Additionally, we can apply this parallel classifier to various application domains such as weather data, internet traffic, log files, medical information, among others to check its generalization. We will also extend This work to cover a popular distributed programming paradigms like MapReduce in a cloud environment. These results are encouraging and show that managed code can deliver high performance classifiers. In the future we will investigate further algorithms and apply them to interesting applications.

8. REFERENCES

[1] Feldman R., and Sanger J., The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data, Cambridge University Press, 2007.

[2] Hill T., and Lewicki P., STATISTICS Methods and Applications, 1st edition, StatSoft, Tulsa, OK, 2007.

[3] Sauban M., and Pfahringer B., “Text Categorization Using Document Profiling,” The 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003) – Conference Proceedings, Cavtat-Dubrovnik, Croatia, September 22-26, pp. 411-422, 2003.

[4] Sebastiani F.,“Machine learning in automated text categorization,” Journal of ACM Computing Surveys (CSUR), vol. 34 , no. 1, pp. 1-47, 2002.

[5] Yang Y., Slattery S., and Ghani R., “A Study of approaches to hypertext Categorization,” Journal of Intelligent Information Systems, vol. 18, no. 2-3, pp. 219-241, 2002.

[6] Al-Shalabi R., Kannan G., and Gharaibeh H., “Arabic text categorization using K-NN algorithm,” The 4th International Multiconference on Computer and Information Technology (CSIT 2006) – Conference Proceedings, Amman, Jordan, 2006.

[7] El-Halees A., “A Comparative Study on Arabic Text Classification,” Egyptian Computer Science Journal, vol. 30 , no. 2, 2008.

[8] Yang Y., “An Evaluation of Statistical Approaches to Text Categorization,” Journal of Information Retrieval, vol. 1 , no. 1-2, pp. 69-90, 1999.

[9] El-Kourdi M., Bensaid A., and Rachidi T., “Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm,” The 20th international conference on Computational Linguistics – Conference Proceedings, Geneva, August, 2004.

[10] Lewis D., “Naïve (Bayes) at forty: The Independent Assumption in Information Retrieval,” The 10th European Conference on Machine Learning (ECML 1998) – Conference Proceedings, Berlin, pp. 4–15, 1998.

[11] Feldman R., and Sanger J., The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data, Cambridge University Press, 2007.

[12] Joachims T., “Text Categorization with Support Vector Machines: Learning with Many Relevant Features,” The 10th European Conference on Machine Learning (ECML 1998) – Conference Proceedings, London, UK, pp. 137-142, 1998.

[13] Apte C., Damerau F., and Weiss S., “Text mining with decision rules and decision trees,” The Conference on Automated Learning and Discovery (CONALD 1998) – Conference Proceedings, Pittsburgh, USA, June, 1998.

[14] Saad M., and Ashour W., “Arabic Text Classification Using Decision Trees,” The 12th international workshop on computer science and information technologies (CSIT 2010) – Conference Proceedings, Moscow, Saint-Petersburg, Russia, vol. 2, pp. 75-79, 2010.

[15] Lianga S., Liua Y., Wang C., and Jiana L., “CUKNN: A parallel Implementation of k-Nearest Neighbor on Cuda-Enabled GPU,” The 2009 IEEE Youth Conference on Information, Computing and Telecommunication (ICT2009) – Conference Proceedings, pp. 415-418, 2009.

[16] Manning D., Raghavan P., and Schütze H., An introduction to information retrieval, Cambridge, England: Cambridge University Press, 2006.

[17] Grama A., Gupta A., Karypis G., and Kumar V., Introduction to Parallel Computing, 2nd edition, Addison Wesley, 2003.

[18] Duwairi R., Al-Refai M., Khasawneh N., “Feature reduction techniques for Arabic text categorization,” Journal of the American Society for Information Science, vol. 60, no. 11, pp. 2347-2352, 2009.

[19] Guan J., and Zhou S., “Pruning training corpus to speed up text classification,” The 13th International Conference on Database and Expert Systems Applications (DEXA 2002) – Conference Proceedings, Aix-en-Provence, France, September, vol. 2453, pp. 831-840, 2002.

[20] Buana P., Jannet S., and Putra l., “Combination of K-Nearest Neighbor and K-Means based on Term Re-weighting for Classify Indonesian News,” International Journal of Computer Applications, vol. 50, no. 11, pp. 37-42, 2012.

[21] Ruoming J., Yang G., and Agrawal G., “Shared memory parallelization of data mining algorithms: Techniques, programming interface and performance,” IEEE Transactions on Knowledge and Data Engineering, vol. 17, no .1, pp. 71-89, 2005.

[22] Tekiner F., Tsuruoka Y., Tsujii J., and Ananiadou S., “Highly Scalable Text Mining – Parallel Tagging Application,” The 5th International Conference on Soft Computing, Computing with Words and Perceptions in

System Analysis, Decision and Control (ICSCCW 2009) – Conference Proceedings, September, pp. 1-4, 2009.

[23] Han J., and Kamber M., Data Mining: Concepts and Techniques, 2nd edition. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor, 2006.

[24] Nishida K., “Learning and Detecting Concept Drift,” Ph.D. Dissertation, Department of Information Science and Technology, Hokkaido University, 2008.

[25] Khoja S., and Garside R., “Stemming Arabic text,” Computer Science Department, Lancaster University, Lancaster, UK, 1999.

[26] Larkey L., Ballesteros L., and Connell M., “Light Stemming for Arabic Information Retrieval,” Arabic Computational Morphology, book chapter, Springer, 2007.

[27] Jing L., Huang H., and Shi H., “Improved feature selection approach TFIDF in text mining,” The 1st International Conference of machine learning and cybernetics – Conference Proceedings, Beijing, 2002.

[28] Said D., Wanas N., Darwish N., and Hegazy N., “A Study of Arabic Text preprocessing methods for Text Categorization,” The 2nd International Conference of on Arabic Language Resources and Tools – Conference Proceedings, Cairo, Egypt, 2009.

[29] Salton G., and Buckley C., “A Study of Arabic Text preprocessing methods for Text Categorization,” The Conference of information processing & management – Conference Proceedings, vol. 24, no. 5, pp. 513-523, 1998.

[30] Saad M., and Ashour W., “OSAC: Open Source Arabic Corpus,” The 6th International Conference on Electrical and Electronics Engineering and Computer Science (EEECS 2010) – Conference Proceedings, European University of Lefke, Cyprus, November 25-26, pp. 1-6, 2010.

[31] Saad M., “Open Source Arabic Language and Text Mining Tools,” (2010, August), [Online], Available: http://sourceforge.net/projects/ar-text-mining [10 August 2012], 2010.

AUTHORS PROFILE

Mohammed M. Abu Tair is a programmer at the college of Science and Technology, KhanYounis, Palestine. He holds M.Sc. degree in Information Technology in 2013 from the Islamic University of Gaza, Palestine. He received his B.Sc. degree in Information Technology Systems in 2005 from the Islamic University of Gaza, Palestine. He holds different technical certificates: MCSE, CCNA. His research activities are in the area of Data and Text Mining, Natural Language Processing, Distributed Data Mining, SOA and Web Services.

Rebhi S. Baraka is an assistant professor of computer science and vice dean of the Faculty of Information Technology at the Islamic University of Gaza, Palestine. He received his PhD degree in Computer Science in 2006 from Johannes Kepler University, Austria, He received his M.Sc. degree in Computer Science in 1996 from De La Salle University, Philippines. He received his B.Sc. degree in Electronics and Communications Engineering in 1991 from University of the East, Philippines. His research activities are in the area of Distributed Systems, Web Services Semantics, Semantic-based Discovery of Web Services, Arabic Ontology Building, and Semantic Web.