ACIT'2017
The International Arab Conference on Information Technology
Yassmine Hammamet, Tunisia
December 22-24, 2017

Large-Scale Arabic Text Classification Using MapReduce

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

Abstract:*Text classification on large-scale real documents has become one of the most core problems in text mining. For English and other languages many text classification works have been done with high performance. However, Arabic language still needs more attention and research since it is highly rich and requires special processing. Existing Arabic text classification approaches use techniques such as feature selection, data representation, feature extraction and sequential algorithms. Few attempts were done to classify large-scale Arabic text document in a parallel manner.This paper presents a parallel classification approach based on the Naïve Bayes algorithm for large volume of Arabic text using MapReduce with enhanced speedup, and preserved accuracy.The experiments show that the parallel classification approach can process large volume of Arabic text efficiently on a MapReduce and can significantly improve the speedup. Also, classification results show that the parallel classifier has achieved accuracy close to 97%.*

Keywords:*Text Classification, Naïve Bayes Algorithm, Parallel Classifier, MapReduce, and Hadoop*.

1. Introduction

Text classification (also known as text categorization) is the task of assigning text documents automatically into one or more predefined categories.This task, that falls at the crossroads of information retrieval (*IR*) and machine learning (*ML*), has witnessed increasing interest in the recent years from researchers and developers alike [17]. Automatic text classification has several useful applications such as classifying text documents in electronic format, spam filtering, improving search results of search engines, web-page content filtering, and opinion mining [12].

Several methods have been used for text classification [16] such as: Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Artificial Neural Networks, Naïve Bayes (NB) Probabilistic Classifier, Random Forest, Distance Detection and Decision Trees.

NB classifier is a popular machine learning method for text classification and is widely applied by many researchers to classify Arabic text documents [2]. It is fast and easy to implement, but it consumes much time and has low accuracy when used in classifying large volume of text documents.

NB classifier assumes that each feature word is independent from other feature words in a document and makes higher efficiency possible but also adversely affects the quality of its results because some of feature words are interrelated [18].

The large amount of text documents with high dimensionality (i.e. the features or attributes are the words that occur in documents) and particularly in Arabic language which has a rich nature and complex morphology requires a large amount of computational power for classification. To be more accurate, we mean by large-scale Arabic text 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) [14].

So, in order to increase accuracy and decrease execution timeto implement and execute classifications of large volume of Arabic textdocuments, we need to resort to parallel programming model such as MapReduce.

MapReduce is a parallel programming model [6] for processing and generating large data sets. It is used to solve many problems, such as data distribution, job scheduling, fault tolerance, machine to machine communication. MapReduce allows developers to write programs that process large-scale of unstructured data in parallel across a distributed cluster of processors or stand-alone computers. It works by breaking the processing into two phases: map phase and reduce phase. Each phase has key-value pairs as input and output, and is specified by two functions: the map function and reduce function [19].

This paper presents a MapReduce-based parallel classification approach for large scale Arabic text based on Naïve Bayes algorithm that reduces time and achieves enhanced accuracy. The proposed approach consists of: First, collecting Arabic text documents and applying text preprocessing. Second, splitting and distributing the documents of the collected corpus as MapReduce tasks. Third, performing term-specific calculations to reduce the dimensionality of feature space, namely, Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF)

using MapReduce model. Finally, performing Naïve Bayes text classification using MapReduce model.

Results show that the proposed approach efficiently and accurately classifies a large scale Arabic text with high dimensionality and outperforms the sequential NB algorithm in terms of efficiency and accuracy.

The rest of the paper is organized as follows: Section 2 reviews related works. Section 3 presents Naïve Bayes (NB) classifier. Section 4 introduces MapReduce and Hadoop. Section 5 describes the proposed parallel classifier approach. Section 6 presents the experimental results and evaluation. Finally, Section 7 presents the conclusion and future work.

2. Related Works

Al-Thubaity et al. [4] study the effect of combining five feature selection methods, on Arabic text classification accuracy, two approaches of combination were used, intersection (AND) and union (OR). They collected a corpus from the website of The Saudi Press Agency (SPA). The SPA consists of 6,300 texts comprises six classes of news. They apply Naïve Bayes (NB) classification algorithm on the SPA dataset. Also, they used three feature representation schemas, namely Boolean, Term Frequency Inverse Document Frequency (TF-IDF) and LTC as a weighting scheme for feature selection. Results show that using CHI feature selection method and LTC for feature representation increase the classification accuracy.

Al-Salemi et al. [2] implement three classifiers based on Bayesian theorem; Simple Naïve Bayes (NB), Multi-variant Bernoulli Naïve Bayes (MBNB) and Multinomial Naïve Bayes (MNB) models on Arabic Text. They collected 3172 documents belonging to one of four categories (Arts, Economic, Politics and Sport). They applied text reprocessing, stemming, and several feature selection methods. Results show that feature selection and reduction strategies can decrease the computation complexity, reduce the dimensionality of feature space, and improve the performance of classification.

Ding et al. [7] propose a parallel learning algorithm for text classification. It is based on the combined naïve Bayes text classifier (PC-NB) that relaxes the independence assumption without efficient reduction. They evaluated the parallel implementation on a cluster that consists of 6 nods, and MPI library as parallel programming environment. They evaluated the performance on Reuter's dataset with 9603 training documents and3299 test documents. Results show that the proposed classifier is accurate and powerful while the attributes of an instance are strongly correlated.

Viegas et al. [17] propose a parallel learning algorithm called GPU-NB. GPU-NB is based on Naïve Bayes algorithm that uses graphics processing units (GPUs). They evaluated GPU-based implementation using Compute Unified Device Architecture (CUDA), the great advantage of this technique is in the simplicity and compactness of the data structures used to represent the document. They evaluated the performance of GPU-NB using six real digital libraries. Results show that GPU-NB can speedup the classification process in up 34 x when compared to a sequential CPU-based implementation, also GPU-NB is up to 11 x faster than a CPU-based parallel implementation of Naïve Bayes running with 4 threads.

Chu et al. [5] Propose a parallel learning algorithm. The parallel algorithm based on Naïve Bayes using MapReduce model on Shared-memory system. They specify different sets of mappers to calculate them, and then the reducer sum up intermediate result to get the final result for the parameters. Their experiment was on a 16 way Sun Enterprise 6000 running Solaris 10. They evaluated the average speedup on ten datasets from the UCI Machine Learning repository with different size (from 30000 to 2500000), which makes their report more convincing. The results showed that the speedup was [4 nodes, 4x], [8 nodes, 7.8x], [16 nodes, 13x].

Zhou et al. [20] propose a model of parallel classification algorithm based on Naïve Bayes algorithm with MapReduce. They build a small cluster with 3 business machines (1 master and 2 slaves) on Linux. They test efficiency and scalability of parallel Naïve Bayes algorithm on seven datasets from the UCI Machine Learning repository with different sizes (from 178 KB to 1 MB). Results show that the performance of the algorithm is higher than the general methods with large data set; moreover the parallel algorithms can not only process large datasets, but also enhance the efficiency of the algorithm.

AbuTair and Baraka [1] propose a parallel learning algorithm based on the k-NN algorithm. They evaluated the parallel implementation using MPI library an Arabic corpus of 22,428 text documents that constitute 250 MB and divided into 10 categories (Economics, History, Entertainments, Education and Family, Religious and Fatwas, Sports, Heath, Astronomy, Low, Stories, and Cooking Recipes). They usedlight stemming, stemmingand pruning methods as feature reduction techniquesand text representations as feature vectors.Their results show an improved speedup andaccuracyaround 95%. However the used corpus is considered relatively small to give complete judgment on the performance. Also using MPI with

large-scale data is limited and it complicated when partitioning data.

3. Naïve Bayes (NB) Classifier

NB classifier is a simple probabilistic classifier which works by applying the Bayes' theorem along with naïve assumptions about feature independence. It assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence [10].

Depending on the precise nature of the probability model, Naïve Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for Naïve Bayes models uses the method of maximum likelihood [2]. Naïve Bayes classifier is described as follows [8]:

- Let D be training set of tuples and their associated class labels. Each tuple is represented by a n-dimensional attribute vector, $X= (x_1, x_2, \ldots, x_n)$, n measurements made on the tuple from n attribute, respectively, $A1, A2, \ldots, An$.
- Suppose that there are m classes, $C1, C2, \ldots, Cm$. Given a tuple, X, the classifier will predict that X belongs to the class having the highest probability, conditioned on X. That is, the NB classifier predicts that tuple X belongs to the class C_i. If and only if

$$P(C_i|X) > P(C_j|X) \; for \; 1 \leq j \leq m, \; j \neq i. (1)$$

Thus we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called the maximumposteriori hypothesis by Bayes theorem (Equation2).

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{p(X)}. (2)$$

- As P(X)is constant for all classes, only $P(X|C_i)P(C_i)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equal.
- Based on the assumption that attributes are conditionally independent (no dependency relation between attributes), $P(X|C_i)$ is computed using Equation 3.

$$P(X|C_i) = \prod_{k=1}^{n} P(X_k|C_i). \quad (3)$$

- Theprobabilities $P(X_1|C_i), P(X_2|C_i), \ldots, P(X_n|C_i)$ can be estimated from the training sample, where:
  - **a.** If A_k is categorical, then $P(X_k|C_i)$ is the number of tuples C_i in D having value X_k for A_k divided by $|C_i, D|$, (number of tuples of C_i in D).
  - **b.** If A_k is continuous-valued, $P(X_k|C_i)$ is usually computed based on a Gaussian distribution with a mean μ and standard deviation σ and, $P(X|C_i)$ is:

$$P(X|C_i) = g(X_k, \mu_{C_i}, \sigma_{C_i}) \quad (4)$$

$$g(X_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

Where μ is the mean and σ^2 is the variance.

If an attribute value doesn't occur with every class value, the probability will be zero, and a posteriori probability will also be zero.

- In order to classify an unknown sample X, $P(X|C_i)P(C_i)$ is evaluated for each class C_i. Sample X is then assigned to class C_i if and only if

$$P(C_i|X) > P(C_j|X) \text{for } 1 \leq j \leq m, j \neq i. (6)$$

Where $P(X|C_i) = \prod_{k=1}^{n} P(X_k|C_i)$ (7)

4. MapReduce Model

MapReduce (MR) is a parallel programming model introduced by Google in 2004, and is used in processing and generating large data sets implementation[10].The basic idea of MapReduce comes from divide and conquer algorithms which are used to partition a large problem into smaller subproblems [15]. Key-value pairs form the basic data structure in MapReduce and are imposedon arbitrary datasets. The programmer defines a map and a reduce with the following signature:

$$map : (k_1, v_1) \rightarrow list[(k_2, v_2)]$$
$$reduce : (k_2, list[v_2]) \rightarrow [(k_3, v_3)]$$

The map is applied to every input key-value pair (split across an arbitrary number of files) to generate an arbitrary number of intermediate key-value pairs. The reduce is applied to all values associated with the same intermediate key to generate output key-value pairs. Implicit between the map and reduce phase is a distribute "group by" operation on intermediate (shuffle phase). Intermediate data arrive at each reducer in order, sorted by the key. Output key-value pair from each reduce is written in r files on the distributed file system, where r is the number of reduces[21].

Hadoop [19, 20] is an open source MapReduce framework for writing and running parallel programson large-scale data sets. We used it in the experiments to realize our MapReduce model. A Hadoop cluster runs jobs controlled by the master node, which is known as the NameNode and it is responsible for chunking the data, cloning it, sending the data to the distributed computing nodes as DataNode, monitoring the cluster status, and collecting the results.

Hadoop depends much on its distributed file system. HDFS [11, 13] is a distributed file system designed for storing and supporting very large files with streaming data access pattern (write-once and read-many times) running on a MapReduce model.It uses replication of data stored on DataNode to provide reliability.Files in HDFS are divided into block size chunks (default size is 64MB), which lead to minimizing the time necessary for seeks[15].

5. The Proposed Parallel Classifier

This section describes the proposed parallel classifier. It uses MapReduce model to solve the problem of processing a large scale Arabic text. Figure 1 illustrates the workflow of the classification process. It is roughly divided into four kinds of activities:

1. **Corpus collection and cleaning activities.** The corpus is collected and divided into text documents. Then text preprocessing is applied to remove non-Arabic text, perform tokenization, remove Arabic stop word and perform light stemming.

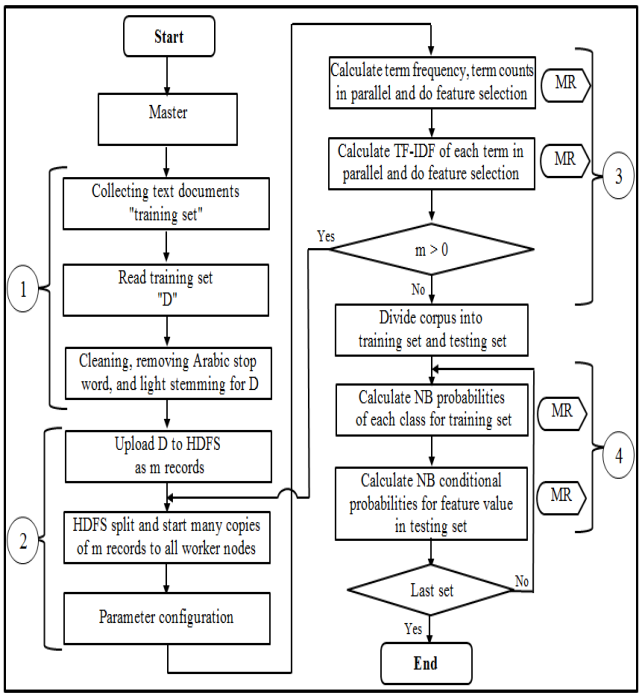

Figure1: The Workflow of the Proposed Approach

2. **HDFS document uploading, splitting and configuration activities.** An important step in developing a parallel algorithm is to split the problem into tasks that can be executed in parallel by identifying the data on which computations to be performed. Then partitioning this data across various tasks. A task performs the computations with its part of data. In our classifier, the input training data set (corpus) are transferred into a sequence of files then uploaded to HDFS in the MapReduce setting. HDFS splits corpus into 64 MB chunks each presented as a map task and then distribute them among workers with replication 3 times by default. Also, it assigns the parameter configurations such as: the document number, the classes number and the document number in each class of corpus.
3. **Term-specific MapReduce-based calculations activities.** Each MapReduce worker node receives its assigned data and calculates parameters such as: word frequency and word counts, then calculates the term frequency-inverse document frequency (TF-IDF) value to generate the vector space model.
4. **Naïve Bayes MapReduce computation activities.** The result of the last step is divided into training set and testing set. The master node assigns workers to calculate probabilities based on Equation 5for each class of the training set using Naïve Bayes MapReduce classification. Finally, the master node assigns another MapReduce worker nodes to calculate conditional probabilities based on Equation 6for each feature value in the testing set to predict the class for each new document.

5.1Classification as a MapReduce Model

The process of building the parallel Naïve Bayes classifier includes three main phases: text preprocessing phase, training phase, and testing phase. These phases are shown in Figure 2.

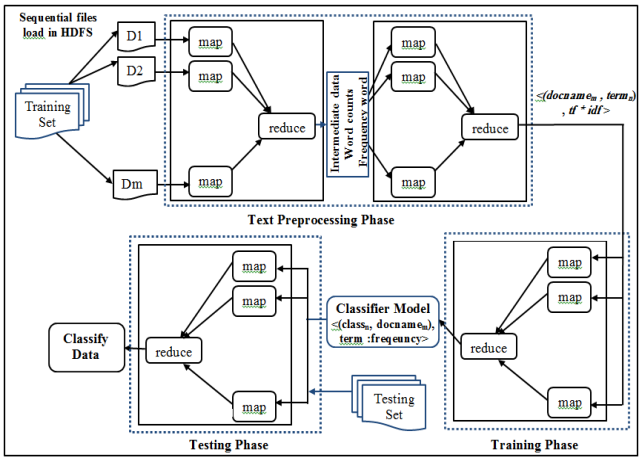

Figure 2: The Proposed Parallel Classifier Approach

Text preprocessing phaseconsists of two steps: In the first step the dataset *D* is divided into *m* subsets {D_1, D_2,..., D_m}. In the second step two MapReduce computations are performed. Onecomputation calculates the parameters required in the next computation. The outputs of thiscomputation are *<(term, docname),n)>* pair, where *n* is the word count in document and *<(term, docname),(n, N)>*pair,where *N* is the frequency of words in documents. The other computation uses the output of the first computation (which is *N*) to calculate Term Frequency-Inverse Document Frequency (TF-IDF) for each term and extracts terms to generate Vector Space Model (VSM).The output of this step is *<docname, (term, tf*idf)>*pair.

Training phasehas one MapReduce computationas described in Algorithms 1 and 2 for training the classifier.The mapper function (Algorithm 1) parses the class and the value of each term (attribute). The output of the mapper function is a combination of (*class, docname, term, tf*idf*) as the key and *1* as the value.This output is written to intermediate files which are processed by the reducer function.

Algorithm 1: Training Naïve Bayes-Mapper

```
input:                          /* training dataset
key: (class, docname);
value: (term, tf*idf);
output :
key': (class, docname, term, tf*idf);
value': the frequency           /* the frequency of  term value
for each  sample
   parse the class and the value of each term
   key': class;
   value': 1;
   output:<key', value'> pair;  /* count the frequency of each term in category
for each (term, tf*idf)  value do
   contract a string as  (class, docname , term, tf*idf);
        set key'  as sting;
   set  value'  as 1;
output:<key', value'> pair;/* write the result to an intermediate files
    end for.
end for.
```

The reduce function(Algorithm 2)counts the frequency of each key. The parameter of the Naïve Bayes classifier is calculated, including P(cj) and P(Ai|cj), where cj denotes the j-th category, Ai the i-th attribute (term). The reducer function aggregates the number of term and category values.

Algorithm 2: Training Naïve Bayes-Reducer

```
input :                         /* output by map function, respectively
key: (class, docname, term, tf*idf);
value: the frequency;
output:
 key': (class, docname, term, and  tf*idf);
value': is the result of frequency;
initialize a counter sum as 0 to record the current statistical
frequency of the key;
while(value .hasnext ())
   sum+= value. next().get();
   set key as  (class, docname, term, tf*idf);
set value' as sum;      /* no of document having the term value
output:<key', value'> pair;/* write the result to an intermediate files
end while.
```

Testing phase has one MapReduce computation for testing the classifier, the input of this process is the testing set and the classifier model resulting from training phase and the output is the result of the final classification.

6. Experimental Results and Evaluation

Experiments aim to provide evidence that the parallel classification approach enhances speedupand preserves accuracy.

The experimental environment is built on a MapReduce cluster with 16 machines. One machine acts as NameNode and the other 15 machines act as DataNodes implemented as virtual machines. All the virtual machines have the same configuration; Intel Core2Quad CPU at 2.5 GHz, 4.00 GB RAM, 320 GB hard disk drive and operating system is Ubuntu 12.4 Linux with Java JDK 1.6.0, and Hadoop version 1.2.0.

6.1 The Corpus

We used Shamela (http://shamela.ws) as the source of our corpus, where we collected 101,647 text documents that constitute 5,310 MB in size and 5,100 MB after stop words removal. Each text document belongs to 1 of 8 classes (Creed, Usual, Fiqh, Hadith, History, Seerah, Tafsir, and Trajem) shown in Table 1.

Table 1: The Shamela Corpus

Category	Number of Text Documents	Size of Text Documents (MB)
Creed	6,776	373
Usual	2,245	128
Fiqh	22,405	1180
Hadith	23,530	1200
History	9,232	488
Seerah	4,641	240
Tafsir	18,048	973
Trajem	14,722	784
Total	101,647	5,310 MB

We performed text preprocessing on the collected corpus including non-Arabic text removal, tokenizing string to word, Arabic stop word removal, term stemming. We generated all text representations for the corpus including light stemming, stemming,TF and TF-IDF.

6.2 Evaluating Speedup

The 101,647 documents corpus that is represented as records and 4096 words that are represented as attributes have been used to measurespeedup of the parallel classifier. Speedup is computed using theformula$S_n=t_s/t_p$, wheret_s is the execution time of one node and$t_p$is the execution time using$n$nodes.

For evaluation purposes, the largest generated text representation for the corpus has been split into two parts; training set and testing set.

We have executed the parallel classifier on a MapReduce system of nodes varied from 2 to 16. Also we used different number of testing documents to observe the effects of different problem (documents) sizes on the performance. Three sets were used as testing documents 20329, 25412, and 50329 documents. Figure 3 shows the execution time for the classifier. It decreases as the number of processors increases. In addition, the execution time increases when the number of documents increases.It can be observed that the sequential NB algorithm (one node) takes the longest time(10.49 minutes). While the parallel NB classifier clearly decreases the classification timeon 16 MapReduce nodes. It takes 52.9 seconds.

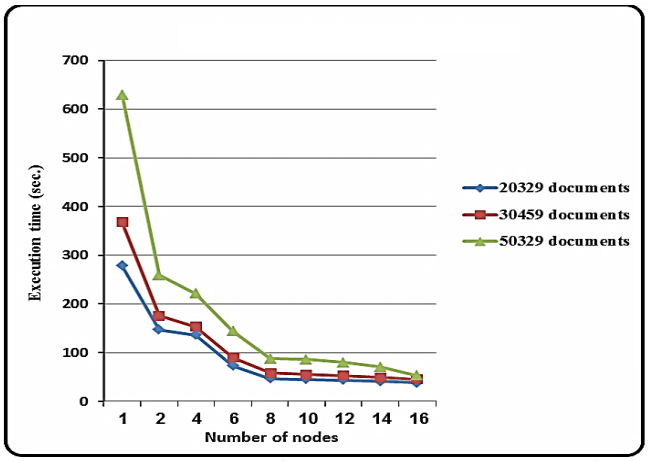

Figure 3: Execution Time for one node and 16nodes Classifiers

The time that the parallel NB classifier spends doesnot appear to have a linear relationship with the number of nodes. This is due to the fact that when running jobs, starting a cluster first take some time. So when the size of data set is small, the processing time is relatively longer.Figure 4 illustrates the relative speedup. It shows that the NB classifier has high speedup. Specifically, as the size of records increases, the speedup improves.

For example, on the largest tested set (50823 documents), it achieves the relative speedups of 2.43, 2.84, 7.12 and 11.90 on 2, 4, 8, and 16 nodes, respectively. When a small set of tested documents are used, the speedup tend to drop from the linear to sub-linear. The classifier achieves the relative speedups of 1.89, 2.04, 5.98, and 7.20 on 2, 4, 8, and 16 nodes respectively. The smallest tested documents sizes give similar results. If we increase the number of nodes further, the speedup gains tend to significantly drop. Figure 4 also shows, the speedups for three documents sets. On 4 nodes the speedup improves from 2.04 to 2.84, on 8 nodes it improve from 5.98 to 7.12, and on 16 nodes it improves from 7.20 to 11.90. It can be conducted that the proposed parallel classifier gives better performance with larger volume Arabic text documents than with smaller volume Arabic text documents.

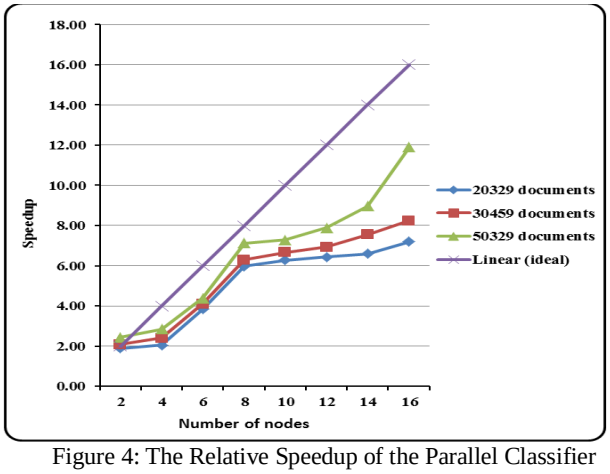

Figure 4: The Relative Speedup of the Parallel Classifier

6.3 Evaluating Accuracy

We tested the accuracy of the classification usingthe confusion matrix shown in Table 2. Accuracy is defined as: *Accuracy= (TP + TN) / (TP + TN + FP + FN).*

Table 2: Simple Confusion Matrix

		True Class	
		Positive	**Negative**
Predicted Class	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

We have conducted two experiments, one with a large number of small files and the other with a small number of large files. In the first experiment, the corpus is split into two parts; 50% of the corpus for training (50833 documents with 4K attributes for each document) and the remaining 50% for testing (50833 documents).In the second experiment, the corpus is split into two parts; 50% of the corpus for training (452 documents with 512K attributes for each document) and the remaining 50% for testing (452 documents).

We split the corpus in this way to achieve higher classification results. The average classification of two the experiments results is illustrated in Figure 5. Two main observations can be made on these results.

First, the difference in accuracy results is based on the highest and lowest values obtained. This emphasizes that the accuracy of the classifier greatly depends on the actual representation of the text to be classified.The highest average of accuracy is achieved using light stemming and TF-IDF while the lowest average of accuracy resulted from stemming and TF.

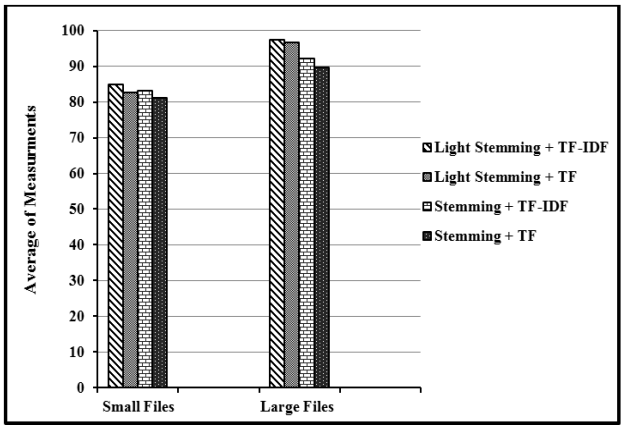

Figure5: Accuracy with Small Files and Large Files Classification

Second, there is a considerable difference in improvement of accuracy results of the two experiments. The accuracy result on large numbers of small files is 84.86%, while on small numbers of large files is 97.50% as illustrated in Figure 5.

Additionally, to further ensure to the accuracy of the classification, we measured the Precision (*Precision= TP/(TP + FP)*), Recall (*Recall=TP/(TP + FN)*) and F-

measure (*F-measure= (2 x Precision x Recall)/(Precision + Recall)*)based on the confusion Matrix (Table 2).

Figure 6 illustrates the results of applying these measures for the used domains. Tafsir domain has the highest accuracy and F-measure (98.6%) that is because Tafsir has a small size of words that are limited. Also, it shows that Seerah domain has lowest accuracy and F-measure (87.8%) that isbecause Sirah has a large space domain.

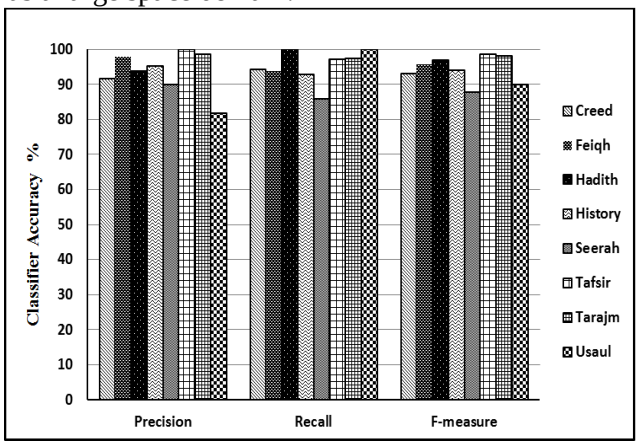

Figure 6: Classification Results for Light Stemming with TF-IDF

7. Conclusion and Future Work

A parallel Naïve Bayes classifier for large-scale Arabic text document based on MapReduce is proposed. The parallel classifier is tested using a large scale Arabic corpus. The test is performed on MapReduce cluster consisting of 16 nodes. The experimental results show that the proposed approach has high speedup when the document sizesarelarge. Also, classification results show that the proposed approach achieves accuracy around 97%.

The proposed approach can be used efficiently and accurately to classify large scale Arabic text with high dimensionality and solves the problem of low accuracy for the sequential NB algorithm.

There are several directions for improvement and future investigation. The approach can be extended to cover larger computer clusters with larger volume of Arabic text documents that constitute more than one terabytes in size. Also, other parallel classification algorithms can be applied with our approach to investigate their effectiveness and performance with large scale Arabic text. Additionally, our approach can be applied to other domains such as medical information, weather data, and social media among others to check its generalization. Finally it can also be used as online classification approach with web data.

References

[1] AbuTair M. and Baraka R., "Design and Evaluation of a Parallel Classifier for Large-Scale Arabic Text," *Int. J. Comput*. Appl., vol. 75, 2013.

[2] Alsaleem S., "Automated Arabic Text Categorization Using SVM and NB," *Int Arab J E-Technol*, vol. 2, no. 2, pp. 124–128, 2011.

[3] Al-Salemi A. and Ab-Aziz M., "Statistical Bayesian Learning for Automatic Arabic Text Categorization", *J. Comput. Sci*., vol. 7, no. 1, 2011.

[4] Al-Thubaity A., Abanumay N., Al-Jerayyed S., and Mannaa Z., "The Effect of Combining Different Feature Selection Methods on Arabic Text Classification," *in (SPND), 2013 14th ACSI International Conference* pp. 211–216, 2013.

[5] Chu C., Kim S., Bradski G., and Olukotun K., "Map-Reduce for Machine Learning on Multicore," *in NIPS*, vol. 6, pp. 281–288, 2006.

[6] Dean J. and Ghemawat S., "MapReduce: Simplified Data Processing on Large Clusters," *Commun ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[7] Ding W., Wang Q., and Guo Q., "A Novel Naive Bayesian Text Classifier," *in 2008 International Symposiums on Information Processing (ISIP)*, pp. 78–82, 2008.

[8] Duba R., Hart P., and Strok D., *Pattern Classification*, Burlington, MA: Johen Wiley & Sons, Inc., 2001.

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

[10] Han J. and Kamber M., *Data Mining Concepts and Techniques*, Amsterdam; Boston; San Francisco, CA: Elsevier ; Morgan Kaufmann, 2006.

[11] Joldzic O., "Applying MapReduce Algorithm to Performance Testing in Lexical Analysis on HDFS," *in Telecommunications Forum (TELFOR)*, 2013 21st, pp. 841–844, 2013.

[12] Kim S., Han K., Rim H., and Myaeng S., "Some Effective Techniques for Naïve Bayes Text Classification," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 11, pp. 1457–1466, Nov. 2006.

[13] Kuang H., Radia S., and Chansler R., "The Hadoop Distributed File System," *in Mass Storage Systems and Technologies (MSST)*, 2010 IEEE 26th Symposium on, pp. 1–10, 2010.

[14] Liang S., Liu Y. Wang, C., and Jian L., "A CUDA-Based Parallel Implementation of K-Nearest Neighbor Algorithm," *in Cyber-Enabled Distributed Computing and Knowledge Discovery*,2009. Cyber C"09. International Conference on, pp. 291–296, 2009.

[15] Lin J. and Dyer C., "Data-Intensive Text Processing with MapReduce," *Synth. Lect. Hum. Lang. Technol.*, vol. 3, no. 1, pp. 1–177, 2010.

[16] Sebastiani F., "Machine Learning in Automated Text Categorization," *ACM Comput Surv*, vol. 34, no. 1, pp. 1–47, Mar. 2002.

[17] Viegas F., Andrade G., Almeida J., and Rocha L., "GPU-NB: A Fast CUDA-Based Implementation of Naïve Bayes," *in 2013 25th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 168–175, 2013.

[18] Wang B. and Zhang S., "A Novel Text Classification Algorithm Based on Naïve Bayes and KL-Divergence," *in sixth International Conference on Parallel and Distributed Computing, Applications and Technologies*, PDCAT 2005, pp. 913–915, 2005.

[19] White T., *Hadoop: The Definitive Guide*, 3rd Edition, Storage and Analysis at Internet Scale. O"Reilly Media /Yahoo Press, 2012.

[20] Zhou L., Wang H., and Wang W., "Parallel Implementation of Classification Algorithms Based on Cloud Computing Environment," *Itkomnika Indones. J. Electr. Eng.*, vol. 10, no. 5, pp. 1087–1092, 2012.

[21] Zhou P., Lei J., and Ye W., "Large-Scale Data Sets Clustering Based on MapReduce and Hadoop," *J. Comput. Inf. Syst.*, vol. 7, no. 16, pp. 5956–5963, 2011.

Authors Profile

Maher M. Abushab is a programmerand a candidate for M.Sc. in Information Technology, the Islamic University of Gaza, Palestine. He received his B.Sc. degree in Computer Information Systems in 2000 from Al-Quds Open University, Palestine. His research activities are in the area of Data and Text Mining, Natural Language Processing, and Distributed Data Mining.

Rebhi S. Baraka is an associate professor of computer science and vice dean of the Faculty of Information Technology, 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.