An Incremental Spam Detection Algorithm

Elham Ghanbari
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
e_ghanbari@ce.sharif.edu

Hamid Beigy
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
beigy@sharif.edu

Abstract— The voluminous of the e-mails are spam. Several algorithms are represented for spam detection based on batch learning. In this paper, a new algorithm based on incremental learning is introduced. The algorithm composes new knowledge from new training data with previous knowledge by combining classifiers based on weighted majority voting. The experiment results show that the proposed algorithm outperforms other related incremental algorithms and non-incremental algorithms.

Keywords- Spam Detection; incremental learning; ensemble learning

I. INTRODUCTION

Recently certain methods are represented to overcome spam, one of the best methods is detecting spam based on their contents. Separating legitimate e-mails and spam within their contents can be categorized as text classification. Machine learning approaches are extremely applied in text classification; moreover, they have reasonable results. As a result, machine learning algorithms can be used for spam classification [1]. Machine Learning algorithms have brought many new techniques to the problem of spam detection. They are capable of extracting knowledge from a set of e-mails supplied, and using the obtained information in the classification of newly received e-mails. In [1], Guzella presents a comprehensive review of recent developments in the application of machine learning algorithms to Spam filtering such as Naïve Bayes (NB), vector machine (SVM) method and etc. The majority of these approaches, which have the capacity of learning use machine learning algorithms to detect the spam [1]. Although the most of the algorithm applied to filter spam are based on batch learning algorithms, in the paper the incremental learning algorithms are emphasized. Using incremental learning algorithms is preferred in many applications, especially spam detections due to not having access to the whole batch training data or limitation of memory used for training. These algorithms let agents learn to extract new knowledge from new training data and add it to the previous knowledge [2]. As a result, the intention of the paper is that spam detections using incremental learning.

In many practical applications, there is some limitation on using training data due to the following reason: 1) producing training data is time consuming and expensive, 2) there is limitation on memory for using all training data at the same time, and 3) all the training data are not available at the training time. For example, intrusion detection systems, all the training data are not available at the same time and consideration of all data for training requires a large amount of memory and increase the training time. In classification of stream data, not all the data are available at the same time and we cannot store all of them for using as training data. Consequently, usually the training data becomes available is small batches over periods. In such situations, it is necessary to update an existing classifier incrementally to accommodate the new training data without forgetting the previously used training data. There are different approaches to train the previously trained classifier incrementally without forgetting the previously used training data as given below:

1) Discarding the trained classifier and adding the new training data to previously used training data. Then use these data to train the new classifier. In some case, the previously used training data are no longer available and hence this approach is not feasible. In such situations, the new training data are added to data generated from the previously trained classifier, which is called pseudo training data. Then obtained training data is used to train a new pattern classifier

2) Using lazy or instance based learning algorithms. These algorithms only store the training data in training phase and produce the classifiers at the test phase.

3) The last approach is to use incremental learning algorithms or on-line learning algorithms. These algorithms keep the previously trained classifiers and update the classifiers to accommodate the new training data without forgetting the previously training data.

In this paper, an incremental learning algorithm based on ensemble learning, for spam detection is represented. In proposed algorithm, it is supposed that the environment is stationary. Its ultimate goal is that incremental learning model be very close and similar to batch learning model. In proposed algorithm, some weak classifiers are produced for each new training dataset and these classifiers are combined with the previous classifiers based on weighted majority voting. The experiment results show that the proposed algorithm outperforms other related incremental algorithms and non-incremental algorithm.

The rest of this paper is organized as follows: In sections 2 and 3, an overview of ensemble of classifiers and incremental learning are given, respectively. In section 4, we give the proposed incremental algorithm. In sections 5 the performance evaluation of the proposed algorithm to spam filtering are given, respectively and section 6 concludes the paper.

II. INCREMENTAL LEARNING

Incremental learning is one of the important machine learning issues. Different definitions relating this kind of learning is introduced such as relearning of previous data, which are misclassified [3].

Incremental learning is defined where training data are not available at the beginning of the training process and they add in small batches over a period of time in [2]. A system, which works based on incremental learning, is a system that is capable of learning new information with remembering previous information. An incremental learning classification is defined as follows [4]:

- The learning algorithm must be able to learn additional information from new training data.
- The learning algorithm should not require the previously used training data, which is used to train the existing classifier.
- The learning algorithm should not forget the previously acquired knowledge.
- The learning algorithm must be able to learn new classes, that may be introduced with new data.

There are two techniques relating incremental learning, introduced as batch incremental learning and on line incremental learning [5]. Batch incremental learning is handling a subset of training dataset that are available over a period of time, while on line incremental learning is handling a single training data instance over a period of time.

There are many algorithms related to incremental learning [6,7,8]. Recently certain methods are suggested based on ensemble learning including several classifiers and the method of their combinations such as weighted majority voting [3].

III. ENSEMBLE OF CLASSIFIERS

Ensemble based learning algorithms consists of multi-classifier and a method to combine the classifiers. The efficient algorithm considers both of the parts. Nevertheless, well-known algorithms such as Boosting algorithm and Bagging algorithm only consider the first part and to combine classifiers use simple and constant functions. Ensemble learning algorithms are divided into two classes, homogenous and heterogeneous [9]. In homogenous ensemble learning, classifiers are learned by the same type of learning algorithm, for instance, all them are learned by Bayesian algorithm. In heterogeneous ensemble learning, each classifier can be learned by different learning algorithm. For example, one of them is learned by k-NN algorithm and another is learned by SVM algorithm.

Numerous incremental algorithms are designed and implemented based on ensemble learning [3]. The most important of them is Learn ++ which use multi weak learn in order to construct a system based on incremental learning [10]. When a new batch training data is entered, this algorithm generates multi classifiers for the dataset and then combines them and previous classifier based on weighted majority voting. Since ensemble learning algorithms use the information of the all classifiers, in the most cases, they provide precise and robust results. Moreover, the algorithm doesn't require to remember the previous training data in order to get new knowledge [11].

IV. PROPOSED ALGORITHM

In this section, we present an incremental version of RotBoost algorithm [14]. The RotBoost algorithm is a kind of batch learning algorithms, which constitutes combination of two algorithms AdaBoost [12] and Rotation Forest [13] Interested readers can found the details of RotBoost in [14].

In essence, both proposed incremental learning algorithm and RotBoost creates an ensemble of weak classifiers, each trained on a subset of the different distributions of the current training dataset, and later combined through weighted majority voting. Training instances for each classifier are drawn from an iteratively updated distribution. The main difference between the two algorithms is on the distribution update rule. In RotBoost, the updating rule is based on the performance of the previous classifier [14]. whereas the updating rule of proposed incremental learning algorithm is based on the performance of the entire ensemble, which focuses this algorithm on the samples containing new knowledge. As new data become available, the proposed algorithm generates additional ensemble of classifiers, until the ensemble learns the novel information. Since no classifier is discarded, the previously acquired knowledge is retained The proposed incremental algorithm is explained in detail below, and flow char appears in Fig. 1.

For each dataset φ_k (for $k=1, 2,\ldots K$) that become available, the inputs to the algorithm are: 1)training set $\varphi_k=\{(x_i,y_i)\}_{i=1}^{N_k}$ consisting of N independent instances, where x_i is the training instance and y_i is its correct label. 2) a base learning algorithm Weak Learn that contains any supervised learning algorithms used to generate classifiers. 3) the value of S_k specifies the numbers of iterations done for Rotation Forest in the proposed algorithm; in other words, S_k is the number of weak classifiers to be generated for each dataset in the Rotation Forest. 4) the value of T_k specifies the numbers of iterations of AdaBoost in the proposed algorithm; in the other words T_k is the number of weak classifiers to be generated for each dataset in the AdaBoost. 5) the value of M can be chosen to be a moderate value according to the size of the attribute set. This parameter used to divide feature sets into arbitrary subsets in order to apply PCA, which is similar to the part of the Rotation Forest algorithm used in RotBoost. All principal components are retained in order to preserve the variability information in the data. Thus, M axis rotations take place to form the new attributes for a base classifier [14].

For each dataset, the algorithm generates an ensemble of classifiers with $S_k \times T_k$ classifiers, each trained on a different subset of the available training data. This is achieved by iteratively updating a distribution D_t^k $t=0,1,\ldots,T_k$ that itself is obtained by normalizing a set of weights assigned to each instance based on the classification performance of the classifiers on that instance. With entering a new dataset φ_k, first rotation matrix R_s^a is calculated according to the procedure

given in [12] and then in each S iteration the input dataset is changed into φ_s^k, where $\varphi_s^k=[XR_s^aY]$. Then initialize weight step (I-W) is done based on Fig. 2. This step starts by initialization weight distribution vector. This distribution is initially set to be uniform. Thus, the weights for the first iteration are initialized based on (1).

$$w_1^k(i)=\frac{1}{N_k} \quad \forall i=1,2,...,N_k \tag{1}$$

where N_k is the number of samples in the dataset φ_s^k.Thus all instances have the same probability of being selected in to the first training subset. If the algorithm is being trained on its first dataset (k=1), the distribution doesn't change. For k^{th} dataset (k>1), the initial distribution vector must be reinitialized, which is described later in this section. At each iteration t (for s^{th} iteration from k^{th} dataset), the distribution D_t^k is obtained by normalizing weights w_t^k from the previous iteration as given in (2).

$$D_t^k = w_t^k / \sum_{i=1}^{N_k} w_t^k(i). \tag{2}$$

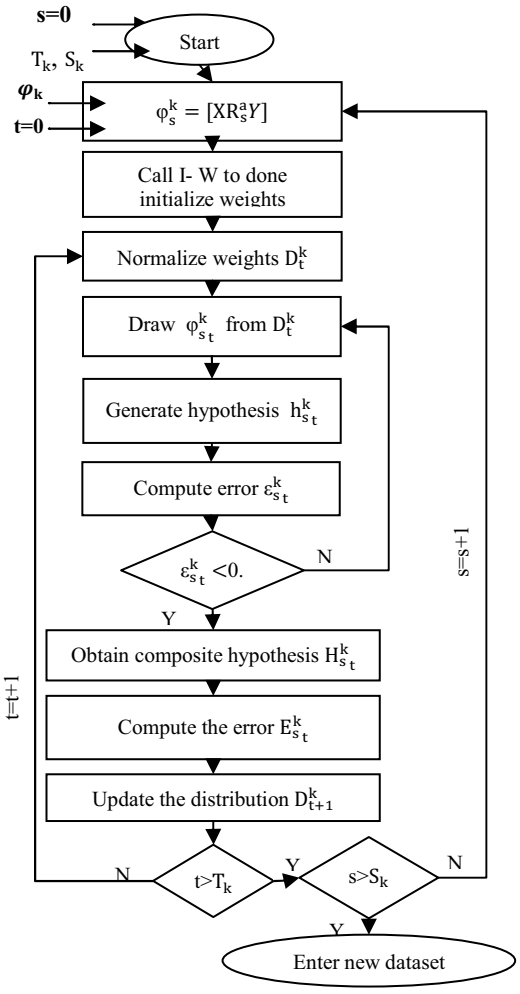

Figure 1. The flowchart of the proposed algorithm

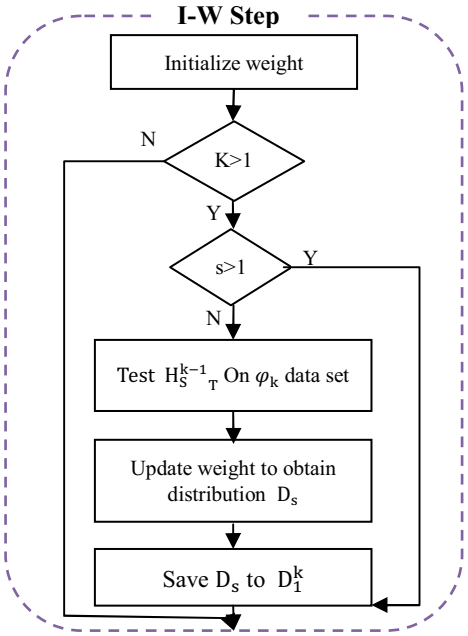

Figure 2. The flowchart of the I-W step

Then, the proposed incremental algorithm calls Weak Learn to generate a hypothesis, $h_{s_t}^k$, from a subset of $\varphi_{s_t}^k$ that is drawn from distribution D_t^k. The error $\varepsilon_{s_t}^k$ of $h_{s_t}^k$ is computed on $\varphi_{s_t}^k$ by adding the distribution weights of misclassified instances.

$$\varepsilon_{s_t}^k = \sum_i I\left[h_{s_t}^k(x_i) \neq y_i \right] D_t^k(i) \tag{3}$$

where I [•] is 1 if the predicate is true, and 0 otherwise. If the classifier correctly classifies at least one half of the dataset, we can be sure about the accuracy of the produced classifier. The error is used in order to determine whether the produced hypothesis can satisfy the desirable accuracy; i.e. if the error is less than 0.5, the hypothesis is accepted and added to the previous hypotheses and algorithm continues next steps. Otherwise, the hypothesis is not accepted and the algorithm returns to choose another arbitrary subset of φ_s^k set again and produces another hypothesis. Then if error of $h_{s_t}^k$ is acceptable (i.e. $\varepsilon_{s_t}^k$<0.5), then the algorithm calculates the normalized error $\alpha_{s_t}^k$ according to (4).

$$\alpha_{s_t}^k = \varepsilon_{s_t}^k / (1 - \varepsilon_{s_t}^k) \tag{4}$$

All hypotheses generated during the previous iterations (t, s and k) are combined using weighted majority voting to construct the combined hypothesis$H_{s_t}^k$, that the weight for each hypothesis is computed as the logarithms of the inverse of normalized errors $\alpha_{s_t}^k$. The combined hypothesis is computed based on (5).

$$H_{s_t}^k = \operatorname{argmax}_{y \in Y} \sum_{p=1}^{k} \sum_{q=1}^{s} \sum_{r=1}^{t} \log\left(1 / \alpha_{q_r}^p \right) I\left[h_{q_r}^p(x)=y \right] \tag{5}$$

The error $E^k_{s_t}$ of hypothesis $H^k_{s_t}$ stated in (6), which is the performance of the entire ensemble constructed so far, is then determined by summing up the distribution weights of all instances misclassified in φ^k_s. Then the normalized error is calculated as given in (7).

$$E^k_{s_t} = \sum_i I(H^k_{s_t}(x_i) \neq y_i) D^k_t(i) \quad (6)$$

$$A^k_{s_t} = E^k_{s_t} / (1 - E^k_{s_t}) \quad (7)$$

In the end, the normalized error is used to update the instance weights, $w^k_t(i)$, according to the performance of hypothesis $H^k_{s_t}$, which is in (8).

$$w^k_{t+1}(i) = w^k_t(i) \times \begin{cases} A^k_{s_t} & , \text{if } H^k_{s_t}(x_i) = y_i \\ 1 & , \text{if } H^k_{s_t}(x_i) \neq y_i \end{cases} \quad (8)$$

According to this rule, the weights of those instances that correctly classified by the composite hypothesis $H^k_{s_t}$ will be reduced; while the weights of the misclassified instances are kept unchanged. This updating rule causes that the proposed incremental algorithm focuses on instances that are difficult to classify by the entire ensemble generated so far. This is precisely what allows proposed incremental algorithm to learn incrementally or on instances that carry novel information, especially when new classes are introduced in the new training data.

When a new dataset is arrived, its distribution is initially set to be uniform. But, If arrived dataset is not the first dataset, (k>1), it should not be supposed that the probability of choosing samples is uniform, then initial distribution must be updated. At first, the new arrived dataset is evaluated by combined hypothesis $H^{k-1}_{S_{k-1}T_{k-1}}$ which is produced based on (5), for previous datasets and the normalized error is calculated to update the distribution. Updating is needed to do only in the first s iteration for each dataset (but it change in t iterations) and the distribution produced in the first s iterations is used in the other s iterations. Since the distributions illustrate that the algorithm incrementally functions, the probability of choosing new sample data is increased to be applied in the next iteration, and the most important part of the algorithm is updating the distributions

V. EXPERIMENTAL RESULTS

In order to evaluate the proposed algorithm, the complement experiment are conducted and then the results obtained are analyzed

A. The Spam Datasets and data preprocessing

One of the most important parts of the experiments is that the datasets used for training and testing. The experimental results of the algorithms depend on the datasets used for training and testing. There are several known and well-defined collections of legitimate and spam messages and many researchers use them as a benchmark datasets to compare the performance of different algorithms [15]. LingSpam [16], SpamAssassin [17], trec05p-1 [18], and Spambasse [19] datasets are examples of public data sets used to evaluate spam filtering algorithms. TABLE. I. summarizes the main characteristics of the data sets utilized in our empirical study. Since trec05p-1 is a large dataset, performing experiments is very difficult. Therefore, only 10 percent of trec05p-1 is used.

The information of the e-mail consists of header and body. Its header comprises general information such as subject, receiver, sender and date. Its body embraces a natural language text along with images and html tags. In the paper, subject and body of the e-mail are used by classifier as a text. Original body of the e-mail cannot be processed and used by classifiers. As a result, the preprocessing in order to generate data structure is applied.

1)Separating tokens: The body of the e-mail consists of a sequential of words separated by space and punctuations. 2)Stop-word removal: Stop-words which repeatedly observed in many texts are removed. 3)Word stemming: converting words to their morphological original form. 4)Feature selection: After performing the steps introduced, the features of all e-mails are extracted and restored in the feature vectors. i.e. After preprocessing, the document is represented by a vector that contains a normalized weights for every word according to its importance.

Then for experimental, the first, preprocessing for all datasets is done, then each dataset is split into two subset with equal size: one of them for training and another for test. Each subset is stratified, i.e. contains approximately the same number of examples from each class. All tests were conducted with training set is randomly divided into five subsets, for incremental learning. A classifier is trained incrementally on training subsets .Each trained classifier is then tested on the test data from the set which was not seen during the training phase.

B. Experimental design

To evaluate the performance of proposed incremental algorithm as a filter for spam, we employ the measures that are widely used in email classification. It is shown in TABLE. II, where, $n_{s,s}$ is the numbers of spam mails which are classified actually as spam; $n_{l,s}$ is the numbers of legitimate mails which are classified as spam mails; $n_{s,l}$ is the numbers of spam mails which are classified as legitimate mails; $n_{l,l}$ is the numbers of legitimate mails which are classified as legitimate email; n_s is the numbers of total spam mail; n_l is the numbers of total legitimate mail.

Now, the required parameters should be defined. The first parameter is choosing base classifier or weak learn for classification. Base Classifier can be any supervised algorithm that achieves at least 50% correct classification. Since the good performance of an ensemble, the method largely depends on the instability of the used base learning algorithm. In the paper, in order to determine the base classifier, certain different algorithms such as decision tree algorithm, Naïve Bayes, support vector machine and k-NN in proposed algorithm are considered as weak learn algorithms. Based on comparing the accuracy of the results, which shown in Fig. 3, and the majorities of ensemble methods, a classification tree is adopted as a weak learn in [14]; decision tree algorithm is the best base classifier for four data sets all.

TABLE I. SUMMARY OF THE USED DATASETS

Name	Spam	Ham	Ratio of spam	Header
LingSpam	481	2412	17%	no
Spambase	1813	2788	39.4%	no
SpamAssassin	1897	4150	31%	yes
TREC2005	5270	3940	57%	yes

T_k and S_k parameters, which were introduced in section 4, can be arbitrary for each dataset. In this study, both T_k and S_k parameters were taken to be fixed for all datasets as T and S. It is interesting, however, to consider other possible trade-offs between these two factors. In order to investigate the effect of these two factors, some more values for S and T are chosen for experiments in this subsection. Fig. 4 and Fig. 5 presents the comparison of accuracy for each considered value. From these figures, then T=14 and S=6 are chosen on the best value for experiments.

In the end, as for the parameter M (namely, the number of attributes contained in each attribute subset) included in proposed incremental algorithm, the value of it was taken to be 3 for the all data sets, based on the default value in [14].

TABLE II. EVALUATION MEASURE

name	Formulation
Accuracy	$\frac{n_{l,l}+n_{s,s}}{n_l+n_s}$
Spam Precision	$\frac{n_{s,s}}{n_{s,s}+n_{l,s}}$
Spam Recall	$\frac{n_{s,s}}{n_{s,s}+n_{s,l}}$

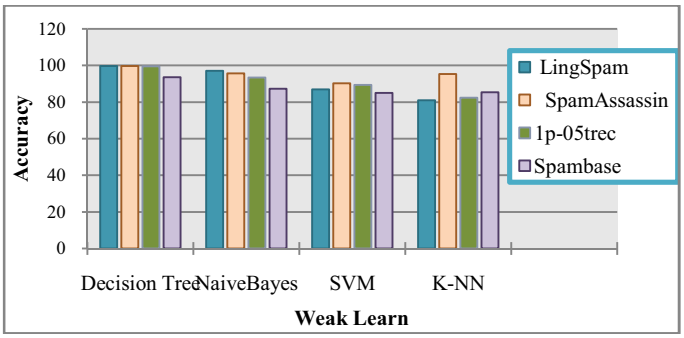

Figure 3. Weak Learn Selection

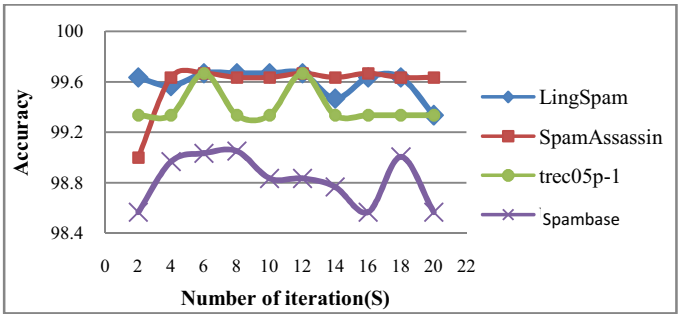

Figure 4. Number of S parameter

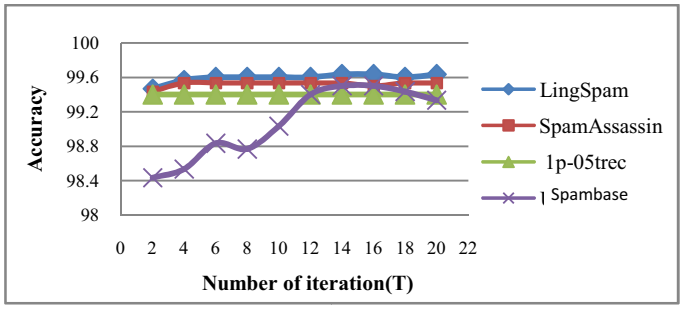

Figure 5. Number of T parameter

C. *Result*

In order to evaluate the proposed incremental algorithm, the computer experiments are conducted and then the results are compared with that of the results obtained for AdaBoost [12] and Learn++ [10] algorithms. The proposed algorithm is compared to the AdaBoost to represent the algorithm has the incremental property; i.e. if the training data enters incrementally, the proposed algorithm can generate results which are similar to non-incremental algorithms. Then, the algorithm compared with the other incremental learning algorithms in order to show that the incremental learning ability of the proposed algorithm is better than others. Hence, the proposed algorithm is compared with Learn++ algorithm.

According to the TABLE. III, IV, V, VI, it is clear that the accuracy of the incrementally trained proposed algorithm closely follows the performance of the AdaBoost trained in on batch with all data. This observation is important because it provides evidence that proposed algorithm is able to learn incrementally. Other results of proposed algorithm are too similar and even better than non-incremental algorithm (AdaBoost algorithm) especially in Lingspam dataset, because the proposed algorithm based on the Rotboost and the performance of the RotBoost is better than AdaBoost in certain cases [14]. Overally, it is shown that the proposed algorithm has the ability to learn incrementally.

Then the results of the proposed algorithm is compared to the lean++. The comparison of these two algorithms are shown in TABLE. III, IV, V, VI. Due to the characteristics of the sets trec05p-1 and SpamAssassin and the high accuracy of their spam detections both algorithms provide the same results. Thus, these two datasets are not able to compare two algorithms. However, the results of LingSpam and Spambase datasets are able to present the priority of the proposed algorithm. The proposed algorithm not only is more accurate than learn++ but also provides a model, the performance of which is the same non-incremental algorithms, whereas the learn++ doesn't have this property. The results are shown in Fig. 6.

TABLE III. RESULT FOR LINGSPAM DATASET

measure	The Proposed algorithm	Learn++ algorithm	Adaboost algorithm
Accuracy	95.6	78.64	93.68
Spam Precision	0.85	0.55	0.84
Spam Recall	0.98	1	0.95

TABLE IV. RESULT FOR SPAMASSASSIN DATASET

measure	The Proposed algorithm	Learn++ algorithm	Adaboost algorithm
Accuracy	99.97	99.80	99.98
Spam Precision	0.99	0.99	0.99
Spam Recall	1	1	1

TABLE V. RESULT FOR SPAMBASE DATASET

Measure	The Proposed algorithm	Learn++ algorithm	Adaboost algorithm
Accuracy	93.06	88.2	90.57
Spam Precision	0.93	0.87	0.89
Spam Recall	0.89	0.83	0.86

TABLE VI. RESULT FOR TREC05P-1 DATASET

Measure	The Proposed algorithm	Learn++ algorithm	Adaboost algorithm
Accuracy	99.93	99.88	99.98
Spam Precision	0.99	0.99	1
Spam Recall	1	1	0.99

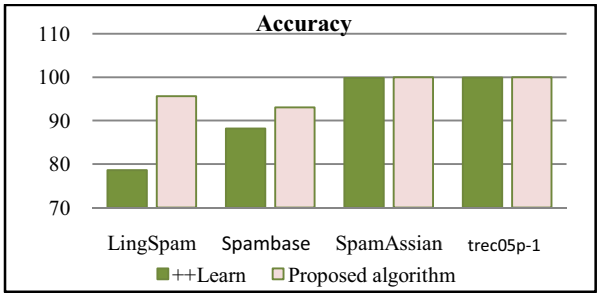

Figure 6. Comparison of accuracy for both algorithm

VI. SUMMERY AND CONCLUSIONS

The incremental algorithm for RotBoost algorithm has been illustrated. The algorithm has used the RotBoost algorithm as its body and has generated the ensemble of classifiers for each entered dataset. The classifiers have been combined to produce final classifier based on weighted majority voting. Determining distribution to select samples which have new informational knowledge has been very important to provide the ability of incremental learning in the proposed algorithm. Hence, combination of the whole classifiers have been used to produce and update the distribution. For evaluation of algorithm, spam's datasets are used. The experimental results have been shown that the proposed algorithm is able to incrementally learn to detect spam and its learning model as well as its performance is similar to nonincremental learning algorithms. The proposed algorithm outperforms learn++ in most tested datasets.

REFERENCES

[1] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to spam filtering", Expert Systems with Applications, vol. 36, no. 7, pp. 10206-10222, 2009.

[2] H. Hamza, Y. Belaïd, A. Belaïd and B. B. Chaudhuri, "Incremental classification of invoice documents", In Proceedings of the 19th International Conference on Pattern Recognition, pp. 1-4, 2008.

[3] J.Byorick and R. Polikar, "Confidence estimation using the incremental learning algorithm, Learn++", In Proceedings of the 2003 Joint International Conference on Artificial Neural Networks and Neural Information Processing, vol. 2714, pp. 181–188, 2003.

[4] Y. M. Wen and B. L. Lu, "Incremental learning of support vector machines by classifier combining", In Proceedings of the 11th Pacific-Asia conference on Advances in knowledge discovery and Data Mining, pp. 904-911, 2007.

[5] D. Sannen, E. Lughofer and H. V. Brussel, "Towards incremental classifier fusion", Intelligent Data Analysis, vol.14, no.1, pp. 3-30, 2010.

[6] A. Bouchachia, B. Gabrys and Z. Sahel, "Overview of some incremental learning algorithms", In Proceedings of the 2007 IEEE International Conference on Fuzzy Systems, pp. 1-6, 2007.

[7] N. A. Syed, H. Liu, and K. K. Sung, "Incremental learning with support vector machines", In Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence, 1999.

[8] P. E. Utgoff, "Incremental induction of decision trees". Machine Learning, vol. 4, no. 2, pp. 161-186, 1989.

[9] G.P.C. Fung, J.X. Yu, H. Wang, D.W. Cheung, and H. Liu, "A balanced ensemble approach to weighting classifiers for text classification", In Proceedings of the 6th International Conference on Data Mining, pp. 869 - 873, 2006.

[10] R. Polikar, L. Udpa, S. S. Udpa and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks", EEE Transactions on System, Man and Cybernetics, Special Issue on Knowledge Management, vol. 31, no. 4, pp. 497-508, 2001.

[11] M. D. Muhlbaier and R. Polikar, "Multiple classifiers based incremental learning algorithm for learning in nonstationary environments", In Proceedings of the 6th International Conference on Machine Learning and Cybernetics, pp. 2953 - 2958, 2007.

[12] Y.Freund and R.E.Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119 – 139, 1997.

[13] J.J.Rodriguez, L.I. Kuncheva and C.J. Alonso, "Rotation Forest: A new classifier ensemble method", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28 , no. 10, pp. 1619 – 1630, 2006.

[14] C. X. Zhang and J.S.Zhang, "RotBoost: a technique for combining Rotation Forest and AdaBoost", Pattern Recognition Letters, vol. 29, no. 10, pp. 1524-1536, 2008.

[15] B.Yu and Z.A.Xu, "Comparative study for content-based dynamic spam classification using four machine learning algorithms", Knowledge-Based Systems, vol. 21, no. 4, pp. 355 – 362, 2008.

[16] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras and C. D. Spyropoulos, "An evaluation of naive Bayesian anti-spam filtering", In Proceedings of the Workshop on Machine Learning in the New Information Age, pp. 9 -17, 2000.

[17] Spamassassin public corpus, http://spamassassin.apache.org/publiccorpus, accessed June 2010.

[18] G. V. Cormack and T. R. Lynam, "TREC 2005 spam track overview", In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), Gaithersburg, Maryland, 2005.

[19] A. Frank, A. Asuncion, "UCI Machine Learning Repository", http://archive.ics.uci.edu/ml, Irvine, CA: University of California, School of Information and Computer Science, accessed June 2010.