An Improved Random Forest Classifier for Text Categorization

Baoxun Xu
Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China
Email: amusing002@gmail.com

Xiufeng Guo
Department of Computer Science, Henan Business College, Zhengzhou 450045, China
guoxiufeng@gmail.com

Yunming Ye
Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China
Email: yeyunming@hitsz.edu.cn

Jiefeng Cheng
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
Email: jf.cheng@siat.ac.cn

***Abstract* — This paper proposes an improved random forest algorithm for classifying text data. This algorithm is particularly designed for analyzing very high dimensional data with multiple classes whose well-known representative data is text corpus. A novel feature weighting method and tree selection method are developed and synergistically served for making random forest framework well suited to categorize text documents with dozens of topics. With the new feature weighting method for subspace sampling and tree selection method, we can effectively reduce subspace size and improve classification performance without increasing error bound. We apply the proposed method on six text data sets with diverse characteristics. The results have demonstrated that this improved random forests outperformed the popular text classification methods in terms of classification performance.**

***Index Terms* — random forest, text categorization, random subspace, decision tree**

I. Introduction

With the ever-increasing volume of text data from Internet, databases, and archives, text categorization or classification poses unique challenges due to the very high dimensionality of text data, sparsity, multi-class labels and unbalanced classes. Many classification approaches have been developed for categorizing text documents, such as random forests [1, 2, 3], support vector machines (SVM) [4, 5], naïve Bayesian (NB) [6, 7], k-nearest neighbor (KNN) [8, 9], decision tree.

Due to its algorithmic simplicity and prominent classification performance for high dimensional data, random forest has become a promising method for text categorization. Random forest is a popular classification method which is an ensemble of a set of classification trees. One of the most popular forest construction procedures, proposed by Breiman, is to randomly select a subspace of features at each node to grow branches of a decision trees, then to use bagging method to generate training data subsets for building individual trees, finally to combine all individual trees to form random forests model [1]. Text data has many terms or features which are uninformative to a specific topic (i.e., a class). During this forest building process, topic-related or informative features would have the large chance to be missed, if we randomly select a small subspace from high dimensional text data [10]. As a result, weak trees will be created from these subspaces, the average discriminative strength of those trees in reduced and the error bound of the random forest is enlarged. Therefore, when a large proportion of such “weak” trees are generated in a random forest, the forest has a large likelihood to make a wrong decision which mainly results from those “weak” trees’ classification power.

To solve this problem, we aim to optimize decision trees of a random forest by two strategies. One straightforward strategy is to enhance the classification power of individual trees by a feature weighted method proposed by Amaratunga [10]. This method computes feature weights with respect to the correlations of features to the class feature. A feature weight can be regarded as the probability of the feature to be selected in subspaces. Its underlying principle is similar to Adaboost method [11] which selects training samples according to the sample weights computed from the result of the previous classifier. This method obviously increases the classification performance of individual trees because the subspaces of decision trees will be biased to contain more informative features. However, Amaratunga's method is only applicable to solve two class problems, because it

doi:10.4304/jcp.7.12.2913-2920

uses t-test of variance analysis to calculate the feature weights. The second strategy is more straightforward: detect and exclude bad trees for preventing their negative impacts from the performance of the random forest. In our previous work [12], we proposed to use the out-of-bag accuracy to evaluate the importance of a tree, which is then used to guide the tree selection process by choosing "important" trees into a random forest. However, this method is not applicable to the classification of text data.

In this paper, we propose an improved random forest algorithm by simultaneously taking into account of a new feature weighting method and the tree selection method to categorize text documents. Instead of using t-test's method by Amaratunga, we employ chi-square statistic as feature weights for weighting random subspace sampling which can generate random forest to solve multi-class text categorization problem. We have conducted a series of experiments on six text data sets and compared our method with the popular random forest method [1]. The results show that our method can generate better random forests with higher classification performance and lower error bound than the competing random forest method. We also make extensive experimental comparisons against other popular text classification methods, such as SVM, NB and KNN. The results indicate that our method outperforms all of them.

The rest of this paper is organized as follows. Section II introduces the feature weighting method, the tree selection method and the novel random forest algorithm. Using the evaluation methods presented in Section III, we show experimental results on the six text data sets in Section IV. Conclusion and future work are made in Section V.

II. Improved Random Forests for Text Classification

In this section, we first introduce a feature weighting method for subspace sampling, then a tree selection method is presented. By integrating these two methods, a novel improved random forest algorithm is proposed.

A. Feature Weighting Method

In this subsection, we will give details of the feature weighting method for subspace sampling in random forests. Consider an M-dimensional feature space $\{A_1, A_2, \ldots, A_M\}$. We present how to compute the weights $\{w_1, w_2, \ldots, w_M\}$ for every feature in the space. These weights are then used in improved algorithm to grow each decision tree in random forest.

(1) Feature Weight Computation

To compute the feature weight, we measure the informativeness of each input feature A as its correlation to the class feature Y. A large weight indicates that the class labels of objects in training data are correlated with the values of feature A. Therefore, A is informative to the class label of objects and has a strong power in prediction of class labels of new objects. Amaratunga used a two-sample t-test as the feature weight so this method can only be used in two class data [10]. To solve multi-class problems, we propose to use chi-square methods to compute feature weights.

Given the class feature Y which has q distinct values or classes, denoted as y_j (for $j=1,\ldots,q$). The feature A can take p values, denoted by a_i (for $i=1,\ldots,p$). If A is numerical, it is discretized with a supervised discretization method. Let D be a data set consisting of $\sum_{j=1}^{q}\sum_{i=1}^{p}\lambda_{ij}$ data samples. The number of samples in D where $A=a_i$ and $Y=y_j$ is denoted by λ_{ij}. All these λ_{ij}'s form a *contingence table* [13] of A and Y as shown in Table I.

TABLE I.
THE CONTINGENCE TABLE OF *A* AND *Y*

	$Y = y_1$	...	$Y = y_j$	...	$Y = y_q$	Total
$A = a_1$	λ_{11}	…	λ_{1j}	...	λ_{1q}	$\sum_{j=1}^{q}\lambda_{1j}$
…	…	…	…	…	…	…
$A = a_i$	λ_{i1}	…	λ_{ij}	…	λ_{iq}	$\sum_{j=1}^{q}\lambda_{ij}$
…	…	…	…	…	…	…
$A = a_p$	λ_{p1}	…	λ_{pj}	…	λ_{pq}	$\sum_{j=1}^{q}\lambda_{pj}$
Total	$\sum_{i=1}^{p}\lambda_{i1}$	…	$\sum_{i=1}^{p}\lambda_{ij}$	…	$\sum_{i=1}^{p}\lambda_{iq}$	$\sum_{j=1}^{q}\sum_{i=1}^{p}\lambda_{ij}$

Given the contingency table of feature A and the class feature Y of a data set D, the chi-square statistic based correlation is computed as

$$corr(A,Y) = \sum_{i=1}^{p}\sum_{j=1}^{q}\frac{(\lambda_{ij} - t_{ij})^2}{t_{ij}}. \qquad (1)$$

Where λ_{ij} is the observed frequency given in the contingency table and t_{ij} is the expected frequency that can be computed as

$$t_{ij} = \frac{\sum_{j=1}^{q}\lambda_{ij} \times \sum_{i=1}^{p}\lambda_{ij}}{\sum_{j=1}^{q}\sum_{i=1}^{p}\lambda_{ij}} \qquad (2)$$

The larger the *corr(A,Y)* measure is, the more informative the feature *A* is with respect to the class feature *Y*, and the higher weight is assigned to feature A.

(2) The Normalized Weights

In practice, feature weights are normalized for feature subspace sampling. Supposing the correlation between a feature A_i and the class label feature Y is $corr(A_i,Y)$ for $i=1,\ldots,N$. We define

$$w_i = \frac{\sqrt{corr(A_i,Y)}}{\sum_{i=1}^{N}\sqrt{corr(A_i,Y)}} \qquad (3)$$

The extraction of square root of the correlation is a common technique for smoothing. It can be easily seen that the normalized weight w_i measures the relative informativeness of feature A_i. This weight information will be used in feature subspace sampling when designing our algorithm.

```
Algorithm 1. Improved Random Forest Algorithm
Input:
- $D$: the training data set,
- $A$: the feature space $\{A_1, A_2,...,A_M\}$,
- $Y$: the feature space $\{y_1, y_2,...,y_q\}$,
- $K$: the number of trees,
- $m$: the size of subspaces.
Output: A random forest µ
Method:
 1: for $i=1$ to $K$ do
 2:   draw a bootstrap sample in-of-bag data subset $IOB_i$ and
      out-of-bag data subset $OOB_i$ from the training data set $D$;
 3:   $h_i(IOB_i)$ = createTree($IOB_i$);
 4:   use out-of-bag data subset $OOB_i$ to calculate the out-of-
        bag accuracy $OOBAcc_i$ of the tree classifier $h_i(IOB_i)$ by
        Equation (4);
 5: end for
 6: sort all $K$ trees classifiers in their $OOBAcc$ descending
      order;
 7: select the top 70% trees with high $OOBAcc$ values and
      combine the 70% tree classifiers into an improved
      random forest µ ;

  Function createTree()
 1: create a new node η ;
 2: if stopping criteria is met then
 3:   return η as a leaf node;
 4: else
 5:   for $j=1$ to $j=M$ do
 6:     compute the informativeness measure corr$(A_j,Y)$ by
          Equation (1);
 7:   end for
 8: compute feature weights $\{w_1, w_2,...,w_M\}$ by Equation (3);
 9: use the feature weighting method to randomly select $m$
      features;
10: use these m feature as candidates to generate the best
      split for the node to be partitioned;
11: call createTree() for each split;
12: end if
13: return η ;
```

B. Tree Selection Method

The key problem of tree selection method is how to evaluate the accuracy of each tree. In [12], we used the out-of-bag accuracy as a measure to evaluate the importance of a tree. In the random forest construction model proposed by Breiman, the bagging method is used to generate a series of training data subsets, which are then used these training subsets to build trees. In each tree, the training data subset used to grow the tree is called in-of-bag (IOB) data, and the data subset formed by the remaining data is called out-of-bag (OOB) data. Since OOB data is not used to build trees, it can be used to test the OOB accuracy of each tree and furthermore, this OOB accuracy can be used to evaluate the importance of the tree.

Given a tree classifier $h_k(IOB_k)$ built from the kth training data subset IOB_k and assuming there are n instances in the whole training dataset D. For each $d_i \in D$, we define the OOB accuracy of the tree $h_k(IOB_k)$ as

$$OOBAcc_k = \frac{\sum_{i=1}^{n} I(h_k(d_i) = y_i; d_i \notin IOB_k)}{\sum_{i=1}^{n} I(d_i \notin IOB_k)} \qquad (4)$$

where I(.) is an indicator function. According to formula (1), the larger the $OOBAcc_k$ is, the better a tree is.

We then sort all the trees by the descending order of their OOB accuracies, and select the top ranking 70% trees to build the random forest. Such tree selection process can generate a population of “good” trees.

C. Algorithm

In this subsection, we present our improved random forest algorithm which integrates feature weighting and tree selection methods. The framework of our methods is introduced in **Algorithm 1**.

In this algorithm, input parameters are the training data set, the feature space, the class feature, the number of trees in the random forest and the size of subspaces. The output is a random forest model. Steps 1-5 are the loop for building K decision trees. In the loop, Step 2 samples the training data with the bootstrap method to generate an in-of-bag data subset for building a tree classifier, and generate an out-of-bag data subset for testing the tree classifier on out-of-bag accuracy. Step 3 calls the recursive function createTree() to build a tree classifier. Step 4 uses out-of-bag data subset to calculate the out-of-bag accuracy of the tree classifier. After the loop, Step 6 sorts all built tree classifiers in their out-of-bag accuracies descending order. Step 7 selects the top 70% trees with high out-of-bag accuracy values and combines the 70% tree classifiers into an improved random forest model. In practice, 70% is sufficiently enough to obtain good results.

Function **createTree** first creates a new node. Then, it tests the stop criteria to decide whether to return to the upper node or to split this node. If splitting this node, it uses the feature weighting method to randomly select m features as a subspace for node splitting. These features are used as candidates to generate the best split to partition the node. For each subset of the partition, **createTree** is called again to create a new node under the current node. If a leaf node is created, it returns to the parent node. This recursive process continues until a full tree is generated.

Compared with Breiman’s method, there are two changes for building a random forest model. The first change is the way to select the feature subspace at each node. Breiman uses simple random sampling method. For very high dimensional text data, the subspace must be set large in order to contain informative feature. This will increase computation burden. With the feature weighting method, we can still use Breiman’s formula $\lfloor \log_2(M)+1 \rfloor$ to specify the subspace size. The second change is that tree selection method is added. This method is further optimizing random forest model.

III. Evaluation Methods

In this paper, we use three measures, i.e., out-of-bag estimate for error bound c/s^2, test accuracy and F-measure metric, to evaluate the performance of the improved random forest algorithm. The ratio c/s^2 is the upper bound for the generalization error of a random forest. The test accuracy measures the performance of a random forest from the test data set that is not used in growing the random forest. The F-measure metric is a commonly used measure of text classification performance.

A. Out-of-bag Estimate for Error Bound c/s^2

The reason of using out-of-bag estimate for error bound *c/s*^{2} is that the out-of-bag strength and correlation estimates give the foundation of understanding how well a random forest works. Strength gives estimates of how accurate the individual classifiers are. Correlation gives estimates of the dependence between component classifiers. The ratio of correlation over the square of strength *c/s*^{2} indicates the general error bound of the random forest model.

We use Breiman's method described in [1] to calculate the error bound *c/s*^{2}. Following Breiman's notations, we use s to denote Strength and $\overline{\rho}$ to Correlation. Let D be a training data and Y be the class labels. Let $h_k(IOB_k)$ be the kth tree classifier grown from the kth training data IOB_k sampled from D with replacement. Assume the random forest contains K trees. Given $d_i \in D$, the out-of-bag proportion of votes for d_i on class j is

$$Q(d_i, j) = \frac{\sum_{k=1}^{K} I\left(h_k(d_i) = j, d_i \notin IOB_k\right)}{\sum_{k=1}^{K} I(d_i \notin IOB_k)} \tag{5}$$

Q(d_i, j) represents the number of trees, which are trained without d_i and classify d_i into class j, divided by the number of training datasets not containing d_i.

The out-of-bag estimate for the average strength s is computed as

$$s = \frac{1}{n}\sum_{i=1}^{n}(Q(d_i, y_i) - \max_{j \neq y_i} Q(d_i, j)) \quad . (6)$$

where n is the number of objects in D and y_i indicates the true class of d_i.

The out-of-bag estimate for the average correlation $\overline{\rho}$ is computed as

$$\overline{\rho} = \frac{\frac{1}{n}\sum_{i=1}^{n}(Q(d_i, y_i) - \max_{j \neq y_i} Q(d_i, j))^2 - s^2}{\left(\frac{1}{K}\sum_{k=1}^{K}\sqrt{p_k + \overline{p}_k + (p_k - \overline{p}_k)^2}\right)^2} \quad ..(7)$$

where

$$p_k = \frac{\sum_{i=1}^{n} I(h_k(d_i) = y_i; d_i \notin IOB_k)}{\sum_{i=1}^{n} I(d_i \notin IOB_k)} \quad . (8)$$

and

$$\overline{p}_k = \frac{\sum_{i=1}^{n} I(h_k(d_i) = \hat{j}(d_i, Y); d_i \notin IOB_k)}{\sum_{i=1}^{n} I(d_i \notin IOB_k)} \quad . (9)$$

where

$$\hat{j}(d_i, Y) = \arg\max_{j \neq y_i} Q(d, j) \quad ..(10)$$

is estimated for every instance d in the training set with *Q(d$_i$, j)*.

Breiman defined the *c/s*^{2} ratio to measure the upper bound of the generalization error of the random forest ensemble

$$c/s^2 = \overline{\rho}/s^2 \quad . (11)$$

The *c/s*^{2} ratio gives a direction of reducing the generalization error of random forests. The smaller the *c/s*^{2} ratio is, the better a random forest is.

B. Test Accuracy Estimate

The test accuracy measures the classification performance of a random forest on the test data set. Let D_t be a test data and Y_t be the class labels. Given $d_i \in D_t$, the number of votes for d_i on class j is

$$N(d_i, j) = \sum_{k=1}^{K} I(h_k(d_i) = j). \tag{12}$$

The test accuracy is calculated as

$$Acc = \frac{1}{n}\sum_{i=1}^{n} I(N(d_i, y_i) - \max_{j \neq y_i} N(d_i, j) > 0). \tag{13}$$

where n is the number of objects in D_t and y_i indicates the true class of d_i.

C. F-measure Estimate

To evaluate the performance of text categorization methods on an unbalanced class distribution data, we use the F-measure metric introduced by Yang and Liu [14]. This measure is the harmonic mean of recall (α) and precision (β). The overall F-measure score of the entire classification result can be computed by micro-average and macro-average.

Micro-averaged F-measure is computed globally over all category decisions, and it emphasizes the performance of a classifier on common categories. α and β are defined as follows:

$$\alpha = \frac{\sum_{i=1}^{q} TP_i}{\sum_{i=1}^{q}(TP_i + FP_i)} \;,\; \beta = \frac{\sum_{i=1}^{q} TP_i}{\sum_{i=1}^{q}(TP_i + FN_i)} \tag{14}$$

where q is the number of class labels or categories. TP_i (True Positives) is the number of text documents correctly predicted as class i, FP_i (False Positives) is the number of documents that do not belong to class i but are classified into class i, and FN_i (False Negatives) is the number of documents with class label i that are not correctly predicted. Micro-averaged F-measure is computed as:

$$Micro - F = \frac{2\alpha\beta}{(\alpha + \beta)} \tag{15}$$

Macro-averaged F-measure is first computed locally over each category, and then the average over all categories is taken. It emphasizes the performance of classifier on rare categories. α and β are defined as follows:

$$\alpha_i = \frac{TP_i}{TP_i + FP_i} \;,\; \beta_i = \frac{TP_i}{TP_i + FN_i} \tag{16}$$

F-measure from each category i and the macro-averaged F-measure are computed as:

$$F_i = \frac{2\alpha_i\beta_i}{(\alpha_i + \beta_i)} \;,\; Macro - F = \frac{\sum_{i=1}^{q} F_i}{q} \tag{17}$$

The larger Micro-F and Macro-F values are, the higher classification performance of text classifier is.

IV. Experiments

In this section, two experiments are conducted to demonstrate the effectiveness of the improved random forest algorithm for classifying text data. Text data sets with various sizes and characteristics are used in the experiments. The first experiment is designed to show how our proposed method can reduce the generalization error bound *c/s^2*, and improve test accuracy when the size of selected subspace was not too large. The second experiment is used to demonstrate the classification performance of our proposed method comparing with other text classification method.

A. Data Sets

In the experiments, we use six real world text data sets. These text data sets are selected due to their diversities in number of features, data volume and number of classes. Their dimensionalities vary from 2000 to 8460, the numbers of documents vary from 918 to 18772, and the minority category rates vary from 0.32% to 6.43%. In each text data set, we randomly select 70% of documents as training data, and the remaining data as test data. Detailed statistics of the six data sets is listed in Table II.

The **Fbis**, **Re0**, **Re1**, **Oh5**, and **Wap** datasets are classical text document classification benchmark data which have been carefully selected a preprocessed by Han and Karypis [15]. The data set **Fbis** is from the Foreign Broadcast Information Service data of TREC-5 [16]. The data sets **Re0** and **Re1** are from Reuters-21578 text categorization test collection Distribution 1.0 [17]. The data set **Oh5** is from OHSUMED-233445 collection [18]. The data set **Wap** is from the WebACE project (WAP) [19]. The classes of these data sets were generated from the relevance judgment provided in these collections.

TABLEII
SUMMARY STATISTIC OF THE DATA SETS

Data set	#Term	#Document	#Classes	%Minority class
Fbis	2000	2463	17	1.54
Re0	2886	1504	13	0.73
Oh5	3012	918	10	6.43
Re1	3758	1657	25	0.6
Newsgroups	5000	18772	20	3.34
Wap	8460	1560	20	0.32

Newsgroups data set is a popular text corpus for experiments in text applications of machine learning techniques. It was obtained from 20 different Usenet newsgroups and contains 18772 documents divided into 20 different classes [20]. We preprocess this data by removing stop terms, and therefore, kept 5000 most informative terms as distinct features of the dataset.

B. Performance Comparisons between Random Forest Methods

The purpose of this experiment is to compare three random forest methods, i.e., our proposed improved random forest methods with both feature weighting and tree selection methods (WTRF), Breiman's Random forest (BRF), and the random forest with only tree selection method (TRF). The comparisons are made for these three random forest methods with different sizes of subspaces on the six text data sets by using two evaluation measures: (1) error bound c/s^2 and classification accuracy. For each text data set, different

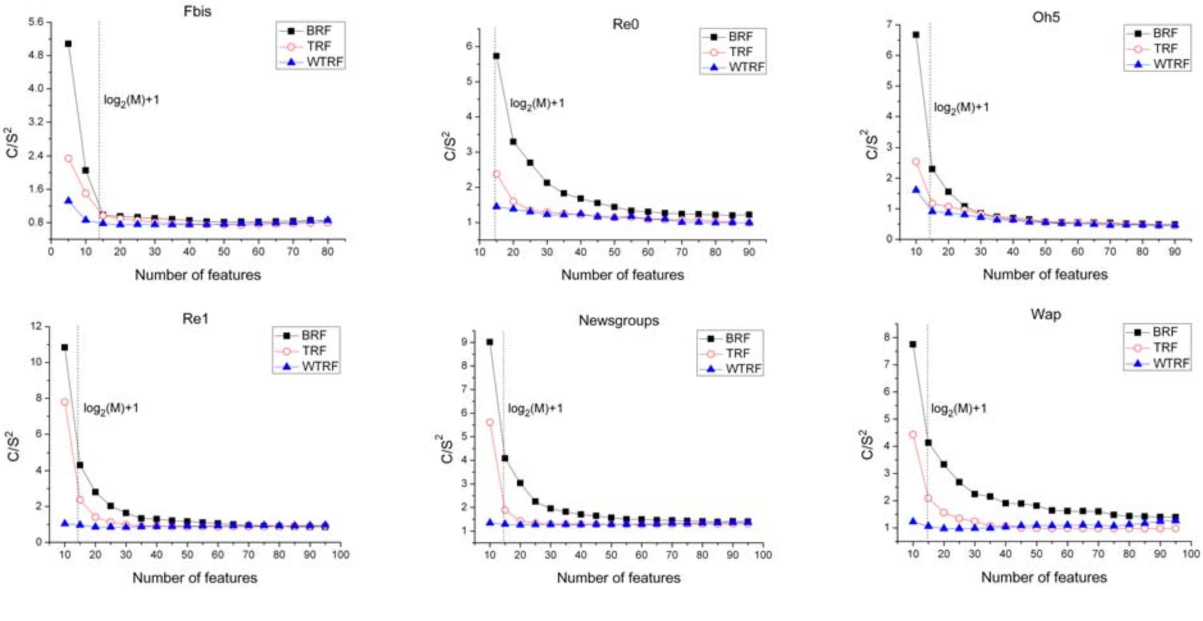

Figure 1. *c/s^2* changes against the number of features in the subspace on the six text data sets. The plot of squares is the result of Breiman's method (BRF), the plot of rounds is the results of the tree selection method (TRF), the plot of triangle is the result of the new method (WTRF).

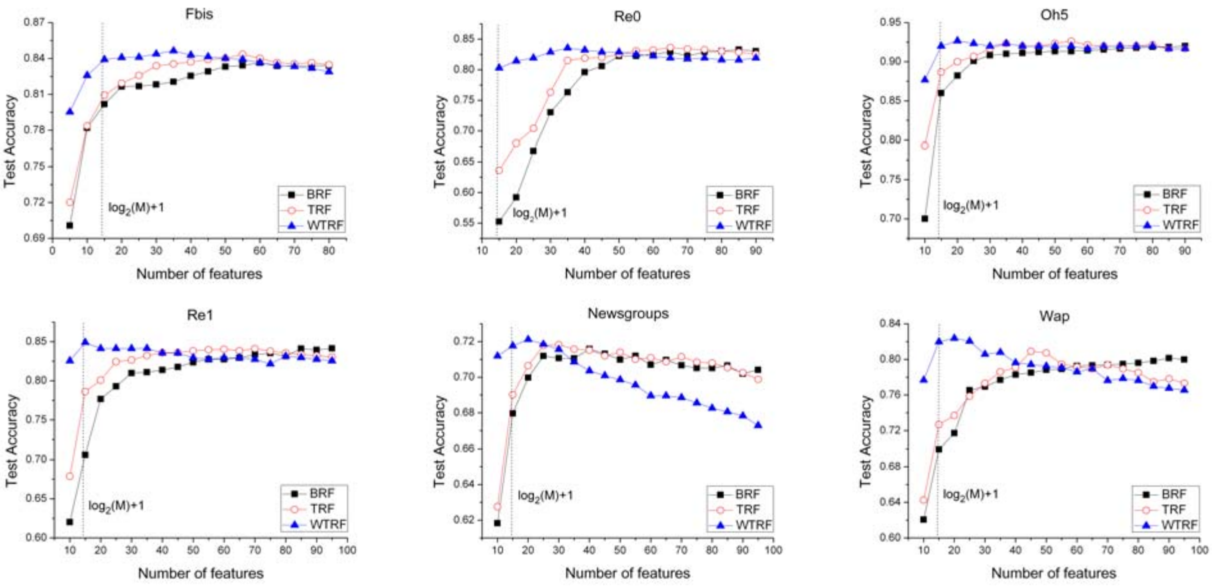

Figure2. Accuracy over the number of features in the subspace on the six text data sets.

type random forests (i.e., BRF, TRF, WTRF) are generated with different subspaces, whose sizes range from 5 to a certain percentage of features in data, with 5 features size interval. In each subspace, 80 random forests are generated, each of which consists of 300 trees. The error bound c/s^2 is the average of 80 random forests. As trees of a random forest are grown from a bagging training data, out-of-bag estimates are used to calculate the ratio c/s^2, and test data is used to compute test accuracy. Experimental results of using two measures on three random forests (i.e., BRF, TRF, WTRF) are shown in Figure 1 and Figure 2.

Figure 1 shows the ratio c/s^2 over different number of features in the selected subspace m on all six data sets. From this figure, we can observe that our proposed method (WTRF) has lowest error bounds in all the six data sets when the select subspace m is not too large. This result implies that when the number of features in the subspace is small, the proportion of the strong informative feature in the select subspace m is large in the new method. There will be a high chance that informative features are selected in the trees. So the overall performance of individual trees is increased. However, in Breiman's method and tree selection method, many randomly selected subspaces may not contain enough informative features that affect the trees' quality. Specially, when the selected subspace $m=log_2(M)+1$, the lowest c/s^2 values of the triangle plots occurred, while the lowest c/s^2 values of the square plots and the round plots occurred in uncertain subspace size. These results indicate that Breiman's proposal for $m=log_2(M)+1$ is not suitable for high dimensional text data because of its inadequate inclusion of informative features in the subspace when the random forests built by Breiman's method (BRF) and tree selection method (TRF). However, the formula $m=log_2(M)+1$ is effective when the built random forest by the new method (WTRF).

Figure 2 shows the accuracies of the three type of random forest model (i.e., BRF, TRF, WTRF) with subspaces of varying sizes. We can clearly see that the random forest with the new method (WTRF) outperforms the random forest with Breiman's method (BRF) and the random forest with tree selection method (TRF) in all the six data sets. It can be seen that the new method is more stable in classification performance than other methods. In all of these figures, it is observed that the highest test accuracy is often obtained when the select subspace $m=log_2(M)+1$. This implies that in practice, large size subspaces are not necessary to grow high-quality trees for random forest.

From Figure 1 and Figure 2, we see that BRF and TRF get the lowest error bound c/s^2 and best test accuracy as long as the selected subspace size is enough. So we use the best accuracies of BRF and TRF as compared objects, and use the accuracy of WTRF when the selected subspace $m= log_2(M)+1$ as compared object. Moreover, we use the corresponding micro-averaged F-measure value and the corresponding macro-averaged F-measure value as compared objects.

C. Performance Comparisons with Other Popular Text Classification Methods

We conduct an extensive experimental comparison against other three widely used text categorization methods, i.e., support vector machines (SVM), Naïve Bayesian (NB), k-nearest neighbor (KNN), on the six text data sets. In this experiment, we use three measures, i.e., test accuracy, micro-averaged F-measure, and macro-averaged F-measure, to evaluate popular text classification methods, i.e., SVM, NB, KNN, BRF, TRF and WTRF on the six text data sets. The support vector machine uses linear Kernel with the regularization parameter 0.03125, which is often used in text categorization. With respect to Naïve Bayesian, we adopt

TABLEIII
COMPARISON OF BEST TEST ACCURACY AMONG SVM, NB, KNN, BRF, TRF AND WTRF.

Dataset	SVM	NB	KNN	BRF	TRF	WTRF
Fbis	0.8349	0.7794	0.7801	0.8364	0.8434	**0.8464**
Re0	0.8049	0.7849	0.7793	0.8328	0.8343	**0.8355**
Oh5	0.8763	0.8764	0.8557	0.92	0.9233	**0.9267**
Re1	0.829	0.8169	0.7889	0.8256	0.8411	**0.849**
Newsgroups	0.6807	0.7007	0.6707	0.7159	0.7183	**0.7212**
Wap	0.8547	0.797	0.7528	0.8214	0.8291	**0.8581**

TABLEIV
COMPARISON OF MICRO-AVERAGED F-MEASURE AMONG SVM, NB, KNN, BRF, TRF AND WTRF.

Dataset	SVM	NB	KNN	BRF	TRF	WTRF
Fbis	0.7996	0.7409	0.7312	0.8062	0.8067	**0.8103**
Re0	0.7956	0.7418	0.7529	0.8023	0.8089	**0.8176**
Oh5	0.8656	0.7736	0.8492	0.8981	0.9093	**0.9115**
Re1	0.8262	0.7321	0.6686	0.8117	0.8295	**0.8321**
Newsgroups	0.6783	0.6932	0.6568	0.7022	0.7106	**0.7141**
Wap	0.812	0.7429	0.6225	0.7921	0.8061	**0.8152**

TABLEV
COMPARISON OF MACRO-AVERAGED F-MEASURE AMONG SVM, NB, KNN, BRF, TRF AND WTRF.

Dataset	SVM	NB	KNN	BRF	TRF	WTRF
Fbis	0.7613	0.7012	0.7312	0.7882	0.7918	**0.7963**
Re0	0.7562	0.6189	0.7529	0.8023	0.8079	**0.8103**
Oh5	0.8442	0.7519	0.8492	0.8943	0.8985	**0.9099**
Re1	0.7066	0.58	0.6686	0.7817	0.7985	**0.8021**
Newsgroups	0.6528	0.6863	0.6398	0.6863	0.6889	**0.6942**
Wap	0.6639	0.5593	0.6225	0.7321	0.7332	**0.7337**

multinomial model that is frequently used for text classification [21]. In k-nearest neighbor (KNN) method, we set the number k of neighbor to 13. In our experiments, we use WEKA's implementation for SVM, NB and KNN [22]. Experiments results were listed in Table III, Table IV and Table V.

Table III lists the accuracy comparison of SVM, NB, KNN, BRF, TRF and WTRF for the six text data sets. Table IV and V give the classification performance comparison in terms of Micro-averaged F-measure and Macro-averaged F-measure. It can be observed that our proposed method (WTRF) outperforms all other text categorization methods.

V. CONCLUSIONS AND FUTURE WORK

We present an improved random forest algorithm by simultaneously taking into account of a new feature weighting method and the tree selection method to categorize text documents. Our algorithm can effectively reduce the upper bound of the generalization error and improve classification performance. From the results of two experiments on various text data sets, the random forest generated by our new method is superior to other text categorization methods. In the future work we will test other feature weighting methods for optimizing the random sampling subspace used in random forest.

ACKNOWLEDGMENT

This research is supported in part by Shenzhen New Industry Development Fund under Grant No.CXB201005250021A and the National Nature Science Foundation of China under Grant No. 61073195.

REFERENCES

[1] L. Breiman, "Random forests," *Machine learning,* vol. 45, no. 1, pp. 5-32, 2001.

[2] J. Klema and A. Almonayyes, "Automatic categorization of fanatic texts using random forests," *Kuwait journal of science and engineering,* vol. 33, no. 2, pp. 1-18, 2006.

[3] M.F. Amasyali and B. Diri, "Automatic Turkish Text Categorization in terms of Author, genre and gender," in *Proc. of the 11th International Conference on Applications*

of Natural Language to Information Systems, pp. 221-226, 2006.
[4] T. Joachims, "Text categorization with support vector machines: Learning with Many Relevant Features," in *European conference on machine learning*, pp. 137-142, 1998.
[5] N. Begum, M.A. Fattah and F.J. Ren, "Automatic text summarization using support vector machine," *international journal of innovative computing information and control*, vol. 5, no. 7, pp. 1987-1996, 2009.
[6] A. McCallum and K. Nigam, "A comparison of event models for naïve bayes text classification," in *AAAI Workshop on learning for Text Categorization*, pp. 41-48, 1998.
[7] J.N. Chen, H.K. Huang, S.F. Tian and Y.L. Qu, "Feature selection for text classification with Naïve Bayes," *Expert Systems with Applications,* vol. 36, no. 3, pp. 5432-5435, 2009.
[8] Y. Yang, "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval," in *Proc. of the 17th annual International ACM SIGIR Conference on Research and Development in information Retrieval*, pp. 13-22, 1994.
[9] S. Tan, "Neighbor-weighted K-nearest neighbor for unbalance text corpus," *Expert Systems with Applications,* vol. 28, no. 4, pp. 667-671, 2005.
[10] D. Amaratunga, J. Cabrera and Y.S. Lee, "Enriched Random Forests," *Bioinformatics,* vol. 24, no. 18, pp. 2010-2014, 2008.
[11] Y. Freund and R.E. Schapire, "Experiments with a new boosting algorithm," in *Proc. of the 13th International Conference on Machine Learning*, pp.148-156, 1996.
[12] B.X. Xu, J.J. Li, Q. Wang and X.J. Chen, "A Tree Selection Model for Improved Random Forest," in *Proc. of the International Conference on Knowledge Discovery*, pp. 382-386, 2011.
[13] K. Pearson, "On the theory of contingency and its relation to association and normal correlation," *Cambridge University Press*, London, 1904.
[14] Y. Yang and X. Liu, "A Re-examination of Text Categorization Methods," in *Proc. of the 22th annual International Conference on Research and Development in Information Retrieval,* pp. 42-49, 1999.
[15] E. Han and G. Karypis, "Centroid-based document classification: Analysis & experimental results," in *Proc. of the 4th European Conference on Principles of Data Mining and Knowledge,* pp. 424-431, 2000.
[16] TREC. Text Retrieval conference. Available at: http://trec.nist.gov.
[17] D.D. Lewis, Reuters-21578 text categorization test collection distribution 1.0, Available at: http://www.research.att.com/~lewis.
[18] W. Hersh, C. Buckley, T.J. Leone and D. Hickam, "OHSUMED: An interactive retrieval evaluation and new large test collection for research," in *Proc. of the 17th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 192-201, 1994.
[19] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar and B. Mobasher, "Web page categorization and feature selection using association rule and principal component clustering," in *Proc. of the 7th Workshop on Information Technologies and System*, 1997.
[20] J. Rennie, Available at: http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate-matlab.tgz.
[21] A. McCallum and K. Nigam, "A comparison of event models for naïve bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*, vol. 752, pp. 41-48, 1998.
[22] Ian H. Witten and Eibe Frank, "Data Mining: Practical machine learning tools and techniques," *Morgan Kaufmann*, San Francisco, second edition, 2005.

Baoxun Xu is currently a Ph.D. candidate at the Department of Computer Science, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China. He received his M.S degree in software engineering from Harbin Institute of Technology. He received the B.S degree in information management and information system from Harbin University of Science & Technology in 2004. His main research areas include data mining, classification algorithm, text categorization, parallel algorithms for data mining.

Xiufeng Guo was born in Henan Province, China, in 1978. He received the M.S degree from Northwest University for Nationalities. He is currently lecturer of Department of Computer Science at Henan Business College. His main research areas include data mining, text categorization, information security.

Yunming Ye was born in Fujian Province, China, in 1976. He received the Ph.D. degree from Shanghai Jiao Tong University. He is currently an professor of the Department of Computer Science at Harbin Institute of Technology Shenzhen Graduate School. His major research fields are data mining and web search technology.

Jiefeng Cheng was born in Sichuan, China, in 1978. He received the Ph.D degree from The Chinese University of Hong Kong in 2007. In 2011, he joined Shenzhen Institute of Advanced Technology (SIAT) as an associate professor. His research interests include query mechanism and high performance data mining algorithms.