A Refinement Approach to Handling Model Misfit in Text Categorization

Haoran Wu, Tong Heng Phang, Bing Liu[*], Xiaoli Li

School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543
{wuhaoran, phangth, liub, lixl}@comp.nus.edu.sg

ABSTRACT

Text categorization or classification is the automated assigning of text documents to pre-defined classes based on their contents. This problem has been studied in information retrieval, machine learning and data mining. So far, many effective techniques have been proposed. However, most techniques are based on some underlying models and/or assumptions. When the data fits the model well, the classification accuracy will be high. However, when the data does not fit the model well, the classification accuracy can be very low. In this paper, we propose a refinement approach to dealing with this problem of model misfit. We show that we do not need to change the classification technique itself (or its underlying model) to make it more flexible. Instead, we propose to use successive refinements of classification on the training data to correct the model misfit. We apply the proposed technique to improve the classification performance of two simple and efficient text classifiers, the Rocchio classifier and the naïve Bayesian classifier. These techniques are suitable for very large text collections because they allow the data to reside on disk and need only one scan of the data to build a text classifier. Extensive experiments on two benchmark document corpora show that the proposed technique is able to improve text categorization accuracy of the two techniques dramatically. In particular, our refined model is able to improve the naïve Bayesian or Rocchio classifier's prediction performance by 45% on average.

Keywords

Text categorization, naïve Bayesian classifier, Rocchio algorithm

1. INTRODUCTION

With the ever-increasing volume of text data from various online sources, it is an important task to categorize or classify these text documents into manageable and easy to understand categories. Text categorization or classification aims to automatically assign categories or classes to unseen text documents. The task is commonly described as follows: Given a set of labeled training documents of *n* classes, the system uses this training set to build a classifier, which is then employed to classify new documents into the *n* classes. The problem has been studied extensively in information retrieval, machine learning and natural language processing. Past research has produced many text classification techniques, e.g., the naïve Bayesian classifier [27, 29], the Rocchio algorithm [19], and support vector machines [18]. These existing techniques have been used to automatically catalog news articles [26], classify Web pages [8] and learn the reading interests of users [24]. An automatic text classifier can save considerable time and human effort, particularly when aiding human indexers who have already produced a large database of categorized document collection.

However, most classification techniques are based on some underlying models and/or assumptions. When the data fits the model well, the classification accuracy can be very high. However, when the underlying model does not fit the data well, the performance of the resulting classifiers can be quite poor. For instance, the naïve Bayesian classifier assumes that text documents are generated from a mixture model and there is a one-to-one correspondence between the mixture components and the classes. However, this assumption can be seriously violated in many real world applications, e.g., the prevention of junk mails, where junk mails generally contain multiple sub-topics such as adult content and various unrelated business letters. Thus the one-to-one correspondence assumption does not hold. In such cases, the naïve Bayesian classifier suffers, which is the problem of model misfit. On the other hand, we may explain that the model misfit is due to *feature-space heterogeneity*. Feature-space heterogeneity [2] occurs when the best features to base the classification on are different in different regions of the feature space. Since most widely used classification techniques, such as the naïve Bayesian algorithm, rely on measures computed over features of the entire training data to build classifiers, their performances are inevitably affected by an averaging effect over the entire training space. The resulting classifiers are likely to be sub-optimal due to the complex nature of the training data.

Serious model misfit often leads to poor classification performance. For unstable classifiers such as decision trees and neural networks, boosting algorithms [12, 31, 35] can be used to enhance the basic weak learners. However, it is generally believed that boosting is not so effective on stable classifiers such as linear classifiers (e.g., the Rocchio algorithm). The naïve Bayesian classifier is also relatively stable with respect to small changes in training data [37]. Other techniques such as meta-learning can be

[*] Bing Liu is now with Department of Computer Science, University of Illinois at Chicago, liub@cs.uic.edu.

used to complement and improve individual classifiers by combining multiple classification techniques with relatively complicated mechanisms [5, 6, 23].

In this paper, we present a novel and yet simple method to deal with the problem of model misfit. In the proposed technique, we make use of training errors to successively refine the classification model on the training data. In learning, model misfit generally leads to high training error of classifiers such as the naïve Bayesian and the Rocchio algorithms. Based on the prediction errors on the training data, we retrain a sub-classifier using the training examples of each predicted class with the same learning method. In this way, we force the classifiers to learn from refined regions in the training data, making the model stronger and fit the training data better.

Our technique is very flexible, which only needs one classification method and there is no change to the method in any way. The original classification model is improved by successively splitting the training data and re-learning sub-models. Moreover, when applying our technique to enhance the probabilistic classifiers such as the naïve Bayesian classifier and linear classifiers such as Rocchio, overfitting is generally not a problem. When the training data is basically consistent with the test data, as the training classification model becomes better, the corresponding classification performance on test instances also improves.

We apply the proposed technique to improve the classification performance of two simple and efficient text classifiers, the Rocchio classifier and the naïve Bayesian classifier. Extensive experiments on two benchmark document corpora show that the proposed technique is able to improve text categorization accuracy of the two techniques dramatically. These techniques are suitable for very large text collections because they allow the data to reside on disk and need only one scan of the data to build a classifier. The resulting classifiers of our technique are very efficient, much faster than many state-of-the-art approaches, e.g., SVM and Adaboost, while also gaining in prediction performance. Our results show that the refined naïve Bayesian classifier outperforms the proven superior classifier SVM [41], which has very high memory requirements and needs the data in memory. Furthermore, SVM converges slowly for large data sets [30].

The rest of this paper is organized as follows. Section 2 discusses related work and Section 3 reviews some existing text categorization techniques related to this work. Section 4 describes the proposed technique in detail. We present the experimental results in Section 5, which is followed by some concluding remarks in the last section.

2. RELATED WORK

Improving prediction accuracy of text classifiers has been an important issue. Many studies have been conducted in this area. One of the popular frameworks is *meta-learning* or *classifier committees* [6, 11, 14, 16, 22, 23, 25, 42], which is an integration of multiple learning models to achieve higher accuracy. This is because a combination of different learning approaches can represent an integration of different learning biases that could complement each other on their inefficient characteristics.

One pioneer work on applying meta-learning to data mining is done by Littlestone & Warmuth [28]. They proposed several weighted majority algorithms for combining different classifiers. Chan and Stolfo [5, 6] adapted their methods to learn the weights using a validation set. They presented techniques that learn an arbiter to arbitrate among predictions generated by different classifiers, and a combiner to merge the predictions of several classifiers. Similar work in this direction also includes the stacked generation [36, 39], and combining multiple rule sets using Bayesian utility theory [1].

Our technique is different from the meta-learning approach. We simply refine a classifier using the training data. Unlike meta-learning, no voting is involved in our process. We do not require multiple classification techniques.

Another popular framework is *adaptive resampling* [3, 17, 31], which adaptively selects instances from a labelled training set to improve classification accuracy. The selection process biases in favour of misclassified data. In particular, the boosting algorithm [12, 32, 33, 34, 35] adaptively resamples or reweights data biasing towards the misclassified examples in the training set and then combine the predictions of a set of classifiers. During the course of its execution, it assigns different importance weights to different training tuples. A weak learning algorithm takes these weights into consideration, and as the algorithm progresses, training documents that are hard to be classified correctly get incrementally higher weights while documents that are easy to classify get lower weights. This, in effect, forces the weak learning algorithm to concentrate on documents that have been misclassified most often previously.

Freund and Schapire's Adaptive Boosting (AdaBoost) algorithm [34, 35] has been reported to be a superior voting method. Several studies have been made on implementing AdaBoost with various classifiers. Elkan [10] provided a framework to apply boosting to the naïve Bayesian classifier. Recently, Kim [20] presented BayesBoost, which uses the naïve Bayes classifier as the weak learner for boosting by allowing the boosting algorithm to utilize term frequency information while maintaining probabilistic accurate confidence ratio. However, there is no clear evidence in how much BayesBoost can improve the naïve Bayes classifier. On the other hand, Ting [37] reported that boosting does not work well for the naïve Bayesian classifier. However, we will show that our proposed technique can remarkably improve the performance of the naïve Bayesian classifier. The refined NB classifier is much superior to AdaBoost (with decision stumps) [35].

In terms of using tree strategy to handle the data heterogeneity problem, Apte et. al. [2] introduced the Importance Profile Angle (IPA) to split the feature space. They compute the IPA value for each feature. If the value exceeds a suitable threshold, it is an indication of heterogeneity. Hence, training data are recursively split according to the feature that gives the largest IPA value.

Friedman [13] introduced a hybrid approach to classification by combining aspects of both the *K*-nearest-neighbor and tree-structured recursive partitioning techniques. It consists of two strategies, *machete* and *scythe*. The *machete* is a successive splitting procedure. It begins with the entire input measurement space and divides it into two regions based on one of the input variables (attributes). The splitting criterion is one that maximizes the estimated relevance. The *scythe* employs an alternative splitting strategy in which the respective variables influence each split in proportion to their estimated relevance, rather than the winner-takes-all approach of the machete. Our method is different, as we do not need sophisticated extra mechanisms to select variables to split the data.

Kohavi [21] reported a hybrid system of naïve Bayesian (NB) and decision tree. The algorithm is similar to the decision tree-building algorithm, except that the leaf nodes created are NB classifiers instead of nodes predicting a single class. It uses a validation set to determine when NB will form a leaf. Our approach differs in the aspect that we do not require a complex combination mechanism and integration of different classification techniques at the algorithm level. We only use the classification results to partition the training data, and apply only a single classification technique. We also do not need a validation set.

3. TEXT CATEGORIZATION TECHNIQUES

We review four commonly used text classification methods. We will apply our technique on them or use them as baseline algorithms for comparison in our experiments later.

3.1 Rocchio Algorithm

An early text classification technique from information retrieval is the Rocchio algorithm, which was originally designed for relevance feedback. It has also been widely used for document classification [19].

In this algorithm, documents are represented with the popular vector space representation. Building a classification is achieved by constructing document vectors into a prototype vector $\vec{c}_j$ for each class c_j. Both the normalized document vectors of the relevant examples for a class as well as those of the irrelevant examples for a class are first summed up. Next, the prototype vector is computed as a weighted difference of each summation.

$$\vec{c}_j = \lambda \frac{1}{C_j} \sum_{\vec{d} \in C_j} \frac{\vec{d}}{\| \vec{d} \|} - \mu \frac{1}{D - C_j} \sum_{\vec{d} \in D - C_j} \frac{\vec{d}}{\| \vec{d} \|}$$

λ and μ are parameters that adjust the relative impact of relevant and irrelevant training examples. Buckley et al [4] recommended $\lambda = 16$ and $\mu = 4$. In classification, for each test document $\vec{d}'$, we simply use the cosine measure to compute the similarity of $\vec{d}'$ with each prototype class vector $\vec{c}_j$. A particular class is assigned to $\vec{d}'$ when its class vector is the most similar to $\vec{d}'$.

3.2 Naive Bayesian Classifier

The naive Bayesian (NB) method is another effective technique for text classification. It has been shown to perform extremely well in practice by many researchers [9, 15, 27, 29].

Given a set of training documents D, each document is considered an ordered list of words. We use $w_{d_i,k}$ to denote the word in position k of document d_i, where each word is from the vocabulary $V = < w_1, w_2, \dots, w_{|v|} >$. The vocabulary is the set of all words we consider for classification. We also have a set of pre-defined classes, $C = \{c_1, c_2, \dots, c_{|C|}\}$ (in this paper we only consider two class classification, so, $C = \{c_1, c_2\}$). In order to perform classification, we need to compute the posterior probability $P(c_j|d_i)$, where c_j is a class and d_i is a document. Based on the Bayesian probability and the multinomial model, we have

$$\mathrm{P}(c_j) = \frac{\sum_{i=1}^{|D|} \mathrm{P}(c_j \mid d_i)}{|D|}$$

and with Laplacian smoothing,

$$\mathrm{P}(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) \mathrm{P}(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) \mathrm{P}(c_j \mid d_i)}$$

where $N(w_t,d_i)$ is the count of the number of times the word w_t occurs in document d_i and $P(c_j|d_i) \in \{0,1\}$ depending on the class label of the document.

Finally, assuming that the probabilities of words are independent given the class, we obtain the naïve Bayesian classifier:

$$\mathrm{P}(c_j \mid d_i) = \frac{\mathrm{P}(c_j) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_i,k} \mid c_j)}{\sum_{r=1}^{|C|} \mathrm{P}(c_r) \prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_i,k} \mid c_r)}$$

In the naive Bayesian classifier, the class with the highest $P(c_j|d_i)$ is assigned as the class of the document.

3.3 Support Vector Machine

Over the recent years, Support Vector Machine (SVM) has been shown to be an accurate classification method for text documents. SVM is a relatively new approach introduced by Vapnik [7, 38] to solving two-class pattern recognition problems. It is based on the Structural Risk Minimization principle for which error-bound analysis has been theoretically motivated. This method is defined over a vector space in which the problem is to find a decision surface that "best" separates the data vectors into two classes. Joachims [18] provides both theoretical and empirical evidences that SVM is very suitable for text categorization. He compared SVM with other classification methods and showed that SVM outperformed all the other methods tested in his experiments.

3.4 AdaBoost

AdaBoost, proposed by Schapire [34, 35], is a learning algorithm that generates multiple classifiers and uses them to build an ultimate classifier. It is well suited to the text categorization problem. Given an instance x and a set of classifiers $(\theta_t(x))$ built with a weak learner, AdaBoost produces the final classifier $F(x)$ through combining the hypotheses of the weak classifiers:

$$F(x) = \sum_{i=1}^{T} \alpha_t \theta_t(x)$$

where T is the number of iterations and α_t is the weight for hypothesis $\theta_t(x)$, which can be calculated by using the equation of the original AdaBoost.

Schapire and Singer [35] introduced the BoosTexter system for text classification, which employs AdaBoost on a one-level decision tree (decision stumps) as the weak learner and reported excellent results.

4. PROPOSED TECHNIQUE

In this section, we first describe the framework of our proposed technique, we then analyse how our approach can solve the model misfit problem when using the naïve Bayesian classifier and the Rocchio classifier.

4.1 Algorithm

In this work, we consider binary text classification that assigns each document d_j either to the positive class C_+ or to its complement negative class C_-. Theoretically, binary text classification is more general than the multi-class one and a multi-class classification problem can be transformed into a set of independent binary ones.

For a text classification task, given a set D of pre-labelled training examples, we choose a classification technique Cl (e.g., the naïve Bayesian classifier) as the base classifier of the proposed technique. We begin with the entire training set D, learning an initial classifier Cl_0 from D, and then classify D into positive and negative classes respectively. We thus split D into two subsets, D_P and D_N, consisting of the predicted positive documents and negative documents respectively. The resulting predicted examples generally contain errors, e.g., the predicted positive class is made up of true positive and false positive documents. We can then learn a sub-classifier Cl_P from D_P based on the training examples in the predicted positive class. The sub-classifier Cl_P could be a refiner of the original classifier Cl_0. To combine Cl_P with Cl_0, we apply Cl_P as the classifier for the predicted positive documents produced by Cl_0. We denote their combined classifier as Cl_{0+P}. The contribution of Cl_P can be evaluated by comparing the classification performances of Cl_0 and Cl_{0+P} on the training data. If Cl_{0+P} outperforms Cl_0, we keep the sub-classifier Cl_P as the refiner of the original classifier Cl_0; otherwise, we discard Cl_P. A similar refining process is also done on the predicted negative documents produced by Cl_0 and thus can output another refined classifier Cl_N. We can recursively carry out the above learning process to build a refinement tree, which is illustrated by Figure 1.

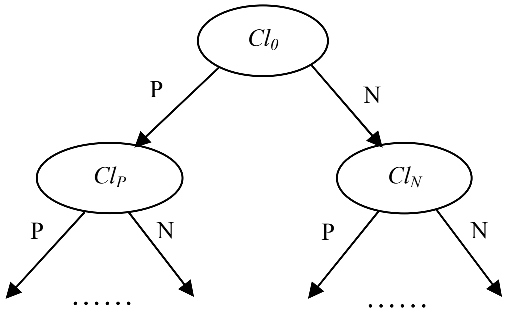

Figure 1: A tree structure of the proposed framework

The effectiveness of text classifiers is commonly evaluated by measuring the recall (r), the precision (p) and/or the F score [40] (which is computed with r and p). In our technique, we use F score, as it is often employed in text classification. We also use F score on the training data to evaluate the resulting sub-classifiers (e.g. Cl_P, and Cl_N) during the process of building a refinement tree.

Let:

a = number of true positive predictions to C_+

b = number of false positive predictions to C_+

c = number of false negative predictions to C_-

d = number of true negative predictions to C_-

Note that in this work we are only interested in the positive class. Thus, the recall and the precision are defined on the positive class as:

$$r = \frac{a}{(a + c)} \qquad p = \frac{a}{(a + b)}.$$

The F-score of the positive class is computed as follows (which gives the equal weight to recall and precision):

$$F = \frac{2pr}{p + r}$$

In the proposed technique, we recursively build a refinement tree from the root node. The root node comprises all example documents in the training set D. At any node, we first build a base classifier Cl_0 by using the selected classification technique on all the training data of this node. Then, according to the predictions of Cl_0, we split this node into positive and negative children nodes *child*(P) and *child*(N), and compute the F score of the training data of this node with the class assignments of Cl_0:

$$F_0 = 2a_0 / (2a_0 + b_0 + c_0)$$

Next, we build two sub-classifiers, Cl_P and Cl_N, using the training data of the positive and negative children nodes respectively, and then compute two new F scores by adding the contributions of Cl_P and Cl_N accordingly:

```
$F_P = 2a_P / (2a_P + b_P + c_P + c_0)$;
$F_N = 2(a_0 + a_N) / (2(a_0 + a_N) + b_0 + b_N + c_N)$;
**If** $F_P > F_0$, output child(P)**, else** prune the branch;
**If** $F_N > F_0$, output child(N) **else** prune the branch;
```

This algorithm is a heuristic one, which tries to refine the classification model by improving the F score on the training data. In the resulting refinement tree, the F score of the base classifier of a particular node is always smaller than that of the combined classifier produced by merging the node's base classifier with a child node's classifier. Since the node splitting process will stop when the F measure does not increase, our approach may reach a local maximum rather than a global maximum. The experimental results show that the proposed technique is already extremely effective. We present the algorithm in Figure 2.

```
Refiner ($D_0$)
1.  Build the base classifier $Cl_0$ using training data $D_0$;
2.  $F_0 = 2a_0 / (2a_0 + b_0 + c_0)$;
3.  Split $D_0$ into $D_P$ and $D_N$ by $Cl_0$;
4.  Build classifier $Cl_P$ using $D_P$;
5.  Build classifier $Cl_N$ using $D_N$;
6.  $F_P = 2a_P / (2a_P + b_P + c_P + c_0)$;
7.  $F_N = 2(a_0 + a_N) / (2(a_0 + a_N) + b_0 + b_N + c_N)$;
8.  If $F_P > F_0$, then Refiner($D_P$) else prune the branch;
9.  If $F_N > F_0$, then Refiner($D_N$) else prune the branch;
```

Figure 2: Constructing a refinement tree

4.2 Why Does the Technique Work?

In this sub-section, we show why the proposed approach is able to deal with the problem of model misfit in the contexts of the naïve Bayesian classifier and the Rocchio classifier.

4.2.1 Naive Bayesian (NB) Classifier

In devising the Bayesian method for text classification, two assumptions are made: (1) text documents are generated by a mixture model and there is a one-to-one mapping between mixture components and classes; (2) document features are independent given the class. Many researchers have shown that the Bayesian classifier performs surprisingly well in obvious violation of (2). However, (1) often causes difficulty when it does not hold. In many real-life situations, one-to-one correspondence of mixture components and classes does not hold. That is, a class (or category) may cover a number of sub-topics.

In the proposed technique, after building the original classifier using the entire training data, we build a sub-classifier on the predicted positive training examples that contain false positive examples (misclassified negative ones) and true positive ones. The predicted positive training examples are similar to each other according to their probabilities in the original model. When we re-train on the predicted positive training examples, we have gotten rid of most of the mixture components in the training data, which have been classified into the negative class. When we perform this process recursively, we can refine the classification and make it closer to one-to-one correspondence of mixture components and classes.

4.2.2 Rocchio Classifier

Figure 3 illustrates the problem of model misfit when using the Rocchio algorithm, where C^+ and C^- represent the positive and negative prototype vectors respectively. The Rocchio classification model is based on the assumption that a given document should be assigned to a particular class if the similarity between this document vector and the prototype vector of the class is the largest. However, when the data does not fit the model well, the Rocchio classifier suffers (see the data points in regions 1 and 2 of Figure 3). Training instances within region 2 will be misclassified as positive documents, while instances within region 1 will be misclassified as negative documents. In the proposed technique, after splitting the entire training data by the initial classifier, we obtain the situation in Figure 4 for the predicted positive class. We can see that it fits the Rocchio model very well. The original region 2 can now be classified correctly using a Rocchio sub-classifier.

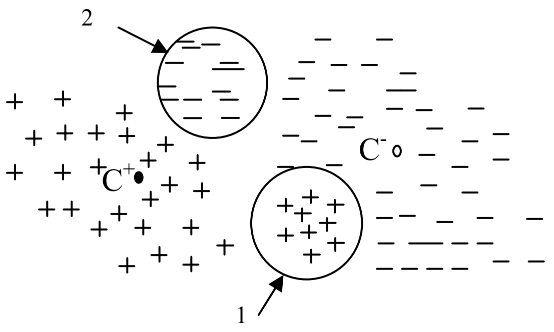

Figure 3: The model misfit of the Rocchio classifier

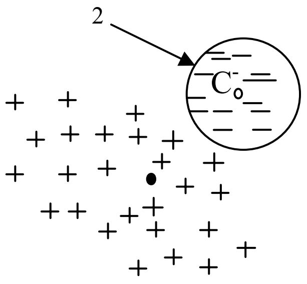

Figure 4: The refined model for the predicted positive class examples (in the training data)

5. EXPERIMENTS

The empirical evaluation is done on two data corpora. First, we will give the dataset descriptions, experimental settings and evaluation measures. Next, we present the empirical results of the experiments on the two datasets.

5.1 Experimental Setup

The first data corpus used is the Reuters-21578 dataset[1] collected from the Reuters newswire. The "ModApte" split is used, leading to a corpus of 9,603 training documents and 3,299 test documents. Of the 135 potential topic categories, only the most populous 10 are used as positive classes. Binary text classification is performed in the experiments.

The second data collection is the Usenet articles[2] collected by Lang (1995) from 20 different newsgroups. 1000 articles were collected for each newsgroup. There are four main categories within the newsgroups, namely, *Computing* (comp)*, Recreation* (rec)*, Science* (sci)*,* and *Talk* (talk). We perform classification on the sub-categories within them. The detailed data compositions are shown in Table 11 in the appendix. For example, we take *graphics* category as the positive class and the rest of the *comp* categories (*os-ms*, *ibm-pc*, *mac* and *x-win*) as the negative class. We withhold the most recently posted articles (40% of the entire dataset) for testing.

Two classification techniques, the naïve Bayesian classifier (NB) and the Rocchio classifier, have been tested in the proposed approach as the base classifier. We also attempted to apply our algorithm on SVM. SVM has been previously proven to be a superior classifier by Yang [41] and fits training data very well in general. In most categories, SVM does not generate substantial training error, therefore our technique fails to refine it much further and the improvement is minimal. Nevertheless, we use SVM as a baseline for comparison.

Our technique works well in different feature settings. In particular, the performance of our refined classifiers is more outstanding when fewer features are used. Hence, we select the first 1000 words with the highest information gains as features for our classifiers during refinement. Experiments are conducted on AdaBoost (with decision stumps) using the BoosTexter[3] [35] package. The other experiments are conducted using the publicly available Rainbow text classification package[4].

We use F score (of the positive class) and *error* as the evaluation measures for our system. F score takes into account of both recall and precision, which makes it a more reliable and suitable measure. *Error* of classification is the fraction of all documents that are classified wrongly with respect to both classes. It is the ratio of the sum of the numbers of *false positives* and *false negatives* to the total number of documents.

With these metrics distributed over all the categories, some kind of averaging is needed to get global performance measures. There are essentially two methods to perform the averaging, *micro-averaging* and *macro-averaging*. In *micro-averaging*, the

[1] http://www.research.att.com/~lewis/reuters21578.html
[2] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes/20_newsgroups.tar.gz
[3] http://www.research.att.com/~schapire/BoosTexter/
[4] http://www-2.cs.cmu.edu/~mccallum/bow/

individual numbers for true/false positives and true/false negatives are summed up and the evaluation measures are computed on them. In *macro-averaging*, the individual measures (e.g.: F score or *error*) for every category are computed and the average is calculated over the total number of categories. Consequently, macro-averaging treats all categories equally, while micro-averaging treats all documents equally. The denominator of error rate is a constant for all the categories, resulting in the same value for macro-averaging and micro-averaging.

5.2 Reuters Dataset

Table 1 shows the experimental results on the Reuters data set. In Table 1, Columns 2 and 3 contain the error and F score of using the NB classifier on each of the top 10 categories of the Reuters-21578 collection respectively. Columns 4 and 5 show the results of the refined NB classifier with our technique. Column 6 provides the classification improvement (I+) on the F score of Column 5 over Column 3, which is the improvement percentage of our refined NB classifier over the initial NB classifier. Similarly, Table 2 shows the classification performance of the refined Rocchio classifier. For all these experiments, 1000 top features (or words) are used.

According to the results, our algorithm works very well when applied to the NB or Rocchio classifiers. For example, in datasets that NB performs badly (e.g., *ship* and *corn*), our system successfully enhances NB by an overwhelming improvement. The refined classifiers are on average 45% better than the original NB classifiers and 44% better than the original Rocchio classifiers in terms of macro-averaged F score. In terms of micro-averaged F score, the refined classifiers are also 25% and 31% better than the original NB classifiers and the original Rocchio classifiers respectively. In addition, our technique also reduces all the error rates substantially. We observe that the refined classifiers are able to improve on every dataset. Although both refined classifiers show outstanding performances the refined NB classifier gives better results than the refined Rocchio classifier. Hence we use it as our representative technique for the performance comparison in Figure 5.

Figure 5 illustrates the comparison of our techniques with several benchmarks in terms of F score. They include AdaBoost (with decision stumps) in BoosTexter, NB and linear SVM. Yang [41] reported that linear SVM provides slightly better results than non-linear models on the Reuters dataset. Thus, we use the linear version as the representative of SVM in our comparison. For this comparison, we use all features for NB and SVM. These classifiers perform better with all the features.

From the diagram, it is clear that our refined NB classifiers outperform AdaBoost greatly for all categories. In general, it also performs better than SVM. As for the Rocchio-refiner, it is slightly worse than SVM. In terms of running time, our system is much faster than SVM and AdaBoost, requiring only linear scans and thus incurring significantly less computational cost.

For most categories in the Reuters dataset, the optimal results of our algorithm are obtained after reaching 2-3 levels of the refined classification tree. As the training error decreases, the test error also decreases, apparently without overfitting. This behavior is also observed in the Usenet dataset experiments, which demonstrates the flexibility of our technique.

5.3 Usenet Dataset

As above, Tables 3, 5, 7 and 9 show the refinement results obtained from the NB classifier from the four main newsgroups. Tables 4, 6, 8 and 10 show the corresponding results using Rocchio as the base classifier. For these results, 1000 top features (or words) are used. Figures 6, 7, 8 and 9 give the comparison with the various benchmark techniques (all features are used).

From the experiments, we observe that our technique clearly outperforms all the other methods by a significant margin. BoosTexter did not perform well for the Usenet data set, which is in accordance with the results reported in [35]. SVM also does not perform well in the Usenet data set. One possible reason is that SVM overfits the training data because the training errors were very low for most datasets. However, our refined NB classifier and our refined Rocchio classifier do not seem to suffer from overfitting.

Regarding running efficiency, SVM is much slower than our system because of its quadratic optimization. AdaBoost using a large number of rounds of boosting is also time-consuming. Using NB or Rocchio as the base classifier, our refined system requires only linear scans of the data and is on average more than 1000 times faster than SVM using the Rainbow package.

Table 1: Results after refining NB classifier for Reuters

	NB		Refined NB		
	Error	F	Error	F	I+(%)
earn	0.021	0.97	0.019	0.970	0.0
acq	0.035	0.92	0.017	0.959	4.2
money-fx	0.072	0.59	0.018	0.828	40.3
grain	0.065	0.58	0.008	0.908	56.6
crude	0.048	0.71	0.009	0.921	29.7
trade	0.100	0.41	0.013	0.826	101.5
interest	0.066	0.54	0.019	0.723	33.9
ship	0.036	0.59	0.005	0.914	54.9
wheat	0.076	0.36	0.008	0.841	133.6
corn	0.073	0.31	0.006	0.836	169.7
microavg	0.059	0.74	0.012	0.926	25.1
macroavg	0.059	0.60	0.012	0.873	45.5

Table 2: Results after refining Rocchio classifier for Reuters

	Rocchio		Refined Rocchio		
	Error	F	Error	F	I+(%)
earn	0.022	0.97	0.021	0.97	0.0
acq	0.043	0.91	0.025	0.94	3.3
money-fx	0.095	0.53	0.025	0.80	50.9
grain	0.064	0.58	0.014	0.85	46.6
crude	0.070	0.62	0.029	0.78	25.8
trade	0.140	0.34	0.025	0.72	111.8
interest	0.084	0.48	0.025	0.71	47.9
ship	0.045	0.55	0.011	0.82	49.1
wheat	0.078	0.36	0.013	0.75	108.3
corn	0.087	0.28	0.009	0.78	178.6
microavg	0.073	0.69	0.020	0.90	30.4
macroavg	0.073	0.56	0.020	0.81	44.6

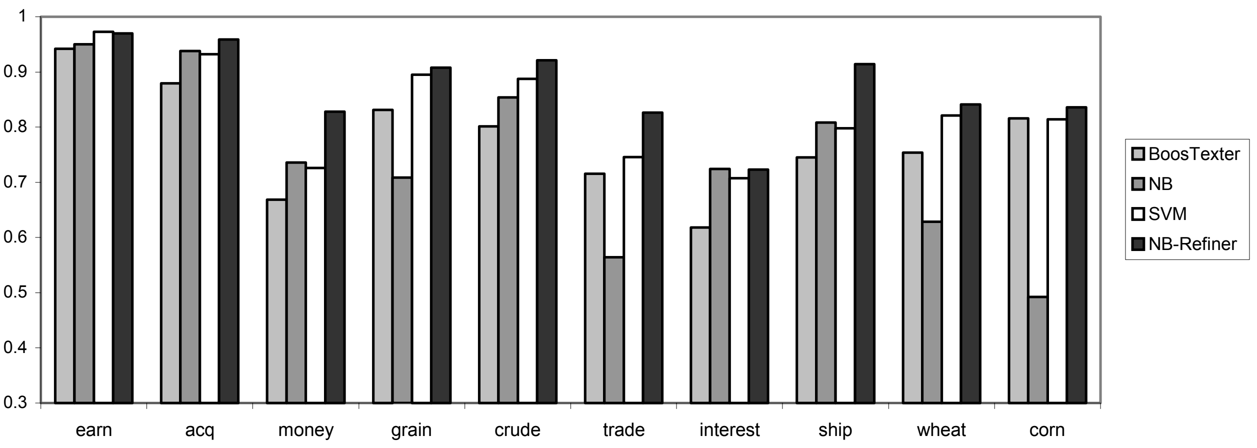

Figure 5: Comparison of F score on Reuters-21578 dataset

Table 3: Results of refining NB classifier for UseNet (comp)

Comp	NB		Refined NB		
	Error	F	Error	F	I+(%)
graphics	0.087	0.80	0.052	0.87	8.7
os-ms	0.102	0.74	0.063	0.84	13.5
ibm-pc	0.099	0.78	0.041	0.89	14.1
mac	0.084	0.81	0.049	0.88	8.6
x-win	0.098	0.76	0.058	0.85	11.8
microavg	0.094	0.78	0.052	0.87	11.5
macroavg	0.094	0.78	0.052	0.87	8.7

Table 4: Results of refining Rocchio classifier for Usenet (comp)

Comp	Rocchio		Refined Rocchio		
	Error	F	Error	F	I+(%)
graphics	0.129	0.73	0.080	0.80	9.6
os-ms	0.130	0.73	0.072	0.80	9.6
ibm-pc	0.141	0.71	0.060	0.84	18.3
mac	0.089	0.80	0.071	0.82	2.5
x-win	0.133	0.73	0.058	0.83	13.7
microavg	0.124	0.74	0.068	0.87	17.6
macroavg	0.124	0.74	0.068	0.82	10.8

Table 5: Results of refining NB classifier for UseNet (rec)

Rec	NB		Refined NB		
	Error	F	Error	F	I+(%)
autos	0.053	0.90	0.014	0.97	7.8
motorcycle	0.069	0.88	0.010	0.98	11.4
baseball	0.016	0.97	0.019	0.98	1.0
hockey	0.021	0.96	0.008	0.98	2.1
microavg	0.040	0.92	0.013	0.98	6.5
macroavg	0.040	0.93	0.013	0.98	5.4

Table 6: Results of refining Rocchio classifier for UseNet (rec)

Rec	Rocchio		Refined Rocchio		
	Error	F	Error	F	I+(%)
autos	0.080	0.86	0.019	0.96	11.6
motorcycle	0.074	0.87	0.013	0.97	11.5
baseball	0.031	0.94	0.027	0.94	0.0
hockey	0.054	0.90	0.011	0.98	8.9
microavg	0.060	0.89	0.018	0.96	7.9
macroavg	0.060	0.89	0.018	0.96	7.9

Table 7: Results of refining NB classifier for Usenet (sci)

Sci	NB		Refined NB		
	Error	F	Error	F	I+(%)
crypt	0.056	0.90	0.019	0.96	6.7
electronics	0.043	0.91	0.030	0.94	3.3
medical	0.029	0.94	0.018	0.96	2.1
space	0.047	0.91	0.013	0.97	6.6
microavg	0.044	0.92	0.020	0.96	4.3
macroavg	0.044	0.92	0.020	0.96	4.3

Table 8: Results of refining Rocchio classifier for Usenet (sci)

Sci	Rocchio		Refined Rocchio		
	Error	F	Error	F	I+(%)
crypt	0.077	0.86	0.045	0.92	7.0
electronics	0.072	0.86	0.067	0.89	3.5
medical	0.032	0.94	0.031	0.94	0.0
space	0.072	0.87	0.020	0.96	10.3
microavg	0.063	0.88	0.041	0.92	4.5
macroavg	0.063	0.88	0.041	0.92	4.5

Table 9: Results of refining NB classifier for Usenet (talk)

Talk	NB		Refined NB		
	Error	F	Error	F	I+(%)
guns	0.129	0.78	0.067	0.86	10.3
mideast	0.038	0.93	0.028	0.95	2.2
misc	0.115	0.77	0.089	0.83	7.8
religion	0.086	0.83	0.058	0.87	4.8
microavg	0.092	0.83	0.060	0.88	6.0
macroavg	0.092	0.83	0.060	0.88	6.0

Table 10: Results of refining Rocchio classifier for Usenet (talk)

Talk	Rocchio		Refined Rocchio		
	Error	F	Error	F	I+(%)
guns	0.151	0.76	0.084	0.81	6.6
mideast	0.045	0.91	0.031	0.93	2.2
misc	0.154	0.72	0.104	0.76	5.6
religion	0.086	0.84	0.075	0.84	0.0
microavg	0.109	0.80	0.074	0.84	5.0
macroavg	0.109	0.81	0.074	0.84	3.7

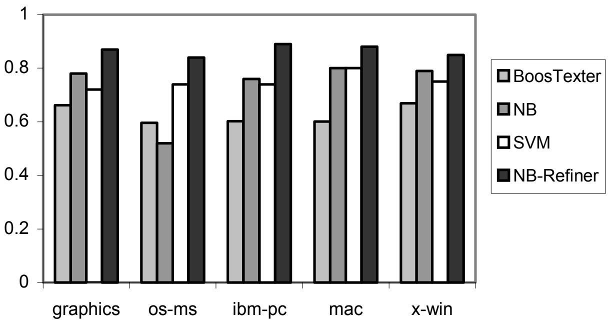

Figure 6: Comparison of F score on Usenet (Comp)

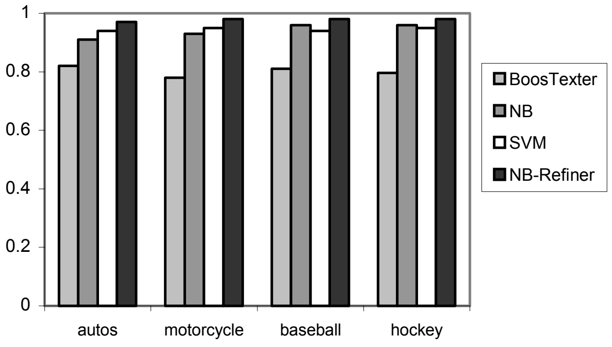

Figure 7: Comparison of F score on Usenet (Rec)

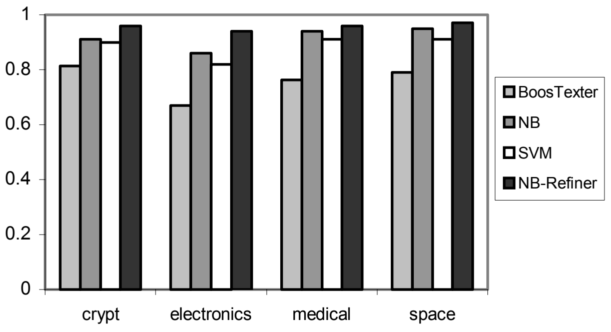

Figure 8: Comparison of F score on Usenet (Sci)

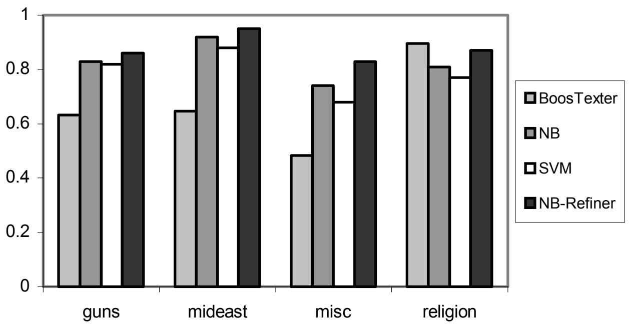

Figure 9: Comparison of F score on Usenet (Talk)

6. CONCLUSION

In this paper, we proposed a refinement approach to handle the problem of model misfit evident in some existing text categorization techniques. Our approach successively refines the classification model based on the prediction errors of the training data. Extensive experiments conducted on Reuters-21578 and Usenet newsgroups corpora confirm that our technique could make an inflexible classification model versatile, and outperform the state-of-the-art techniques like SVM and AdaBoost significantly.

The results reported here are not necessarily the best that can be achieved. There is still room for improvement by finding better ways to identify the performance characteristics of the base classifier during the initial training phase. Nevertheless, the approach described is already able to achieve outstanding prediction performance without using any complex strategies or incurring significant computational costs.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge the support of A-STAR and National University of Singapore Academic Research Fund grants R252-000-041-112 and R252-000-041-303. We would also like to thank Wee Sun Lee for many useful discussions.

8. REFERENCE

[1]. Ali, K. and Pazzani, M. Error reduction through learning multiple descriptions. *Machine Learning*, 1996.

[2]. Apte, C., Hong, S., Hosking, J., Lepre, J., Pednault, E. and Rosen, B. Decomposition of heterogeneous classification problems, *Intelligent Data Analysis*, 1998.

[3]. Breiman, L. Bagging predictors. *Machine Learning* 1996.

[4]. Buckley, C., Salton, G. and Allan, J. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual*

ACM SIGIR Conference, 1994.

[5]. Chan, P. and Stolfo, S. Comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[6]. Chan, P. and Stolfo, S. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proceedings of the International Conference on Knowledge Discovery and Data* Mining, 1995.

[7]. Cortes, C. and Vapnik, V. Support vector networks. *Machine learning,* 1995.

[8]. Craven, M., DiPasquo, D., Freitag, D., MaCallum, A., Mitchell, T., Nigam, K., & Slattery, S. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI,* 1998.

[9]. Duda, R. and Hart, P. *Pattern Classification and Scene Analysis*, 1973.

[10]. Elkan, C. Boosting and naive Bayesian learning. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1997.

[11]. Freitag, D. Multistrategy learning for information extraction. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[12]. Freund, Y. and Schapire, R. Experiments with a new boosting algorithm, In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[13]. Friedman, J. Flexible metric nearest neighbor classification. *Technical Report,* 1994.

[14]. Guo, Y. and Sutiwaraphun, J. Knowledge probing in distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*, 1999.

[15]. Hand, D. and Yu, K. *Idiot's Bayes - Not so Stupid After All?* 2001.

[16]. Hull, D., Pedersen, J. and Schutze, H. Method combination for document filtering. In *Proceedings of the Nineteenth International Conference on Research and Development in Information Retrieval*, 1996.

[17]. Iyengar, V., Apte C. and Zhang, T. Active learning using adaptive resampling, In *Proceedings of the Seventh International. Conference on Knowledge Discovery & Data Mining*, 2000.

[18]. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning*, 1998.

[19]. Joachims, T. A probabilistic analysis of the rochhio algorithm with TFIDF for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[20]. Kim, Y., Hahn, S. and Zhang, B. Text filtering by boosting naive Bayes classifiers. In *Proceedings of SIGIR*, 2000.

[21]. Kohavi, R. Scaling up the accuracy of naïve-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International. Conference on Knowledge Discovery & Data* Mining, 1996.

[22]. Kumar, S. A hierarchical multi-classifier system for hyperspectral data analysis. In *Proceedings of the First International Workshop on Multiple classifier systems*, 2000.

[23]. Lam, W. and Lai, K. A meta-learning approach for text categorization. In *Proceedings of SIGIR*, 2001.

[24]. Lang, K. Newsweeder: Learning to filter netnews. In *Proceedings of International Conference on Machine Learning,* 1995.

[25]. Larkey, L. and Croft, W. Combining classifiers in text categorization. In *Proceedings of the Nineteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[26]. Lewis, D. & Gale, W. A sequential algorithm for training text classifiers. *Proceedings of SIGIR,* 1994.

[27]. Lewis, D. & Ringuette, M. A comparison of two learning algorithms for text categorization. *Third Annual Symposium on Document Analysis and Information Retrieval* (pp. 81-93), 1994.

[28]. Littlestone, N and Warmuth, M. The weighted majority algorithm. Tech. report, UCSC-CRL-89-16: UC. Santa Cruz, 1989.

[29]. McCallum, A., & Nigam, K. A comparison of event models for naïve Bayes text classification. *AAAI-98 Workshop on Learning for Text Categorization.* Tech. Rep. WS-98-05, AAAI Press, 1998

[30]. Pavlov, D. and Mao, J. Scaling-up Support Vector machines using boosting algorithm. In *International Conference on Pattern Recognition*, 2000.

[31]. Quinlan, J. Bagging, boosting and C4.5. In *Proceedings AAAI*, 1996.

[32]. Schapire, R., Freund, Y. Bartlett, P. and Lee, W. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference of Machine Learning*, 1997.

[33]. Schapire, R., Singer, Y and Singhal, A. Boosting and Rocchio applied to text filtering. In *Proceedings SIGIR*, 1998.

[34]. Schapire, R. and Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 1999.

[35]. Schapire, R. and Singer, Y. BoosTexter: A boosting-based system for text categorization, *Machine* Learning, 2000.

[36]. Ting, K. and Witten, I. Stacked generalization: when does it work? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997.

[37]. Ting, K. and Zhen, Z. Improving the performance of boosting for naive Bayesian classification. In *Proceedings of PAKDD*, 1999.

[38]. Vapnik, V. *The Nature of Statistical Learning Theory,* 1995.

[39]. Wolpert, D. Stacked generalization. *Neural Networks*, 1992.

[40]. Yang, Y. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval,* 1999.

[41]. Yang, Y. and Liu, X. A re-examination of text categorization methods. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[42]. Yang, Y., Ault, T. and Pierce, T. Combining multiple learning strategies for effective cross validation. In *Proceedings of the International Conference on Machine Learning*, 2000.

APPENDIX

Table 11: Usenet newsgroups used as positive categories

Index	Group	Name of Category
graphics	comp	comp.graphics
os-ms	comp	comp.os.ms-windows.misc
ibm-pc	comp	comp.sys.ibm.pc.hardware
mac	comp	comp.sys.mac.hardware
x-win	comp	comp.windows.x
autos	rec	rec.autos
motorcycle	rec	rec.motorcycles
baseball	rec	rec.sport.baseball
hockey	rec	rec.sport.hockey
crypt	sci	sci.crypt
electronics	sci	sci.electronics
medical	sci	sci.med
space	sci	sci.space
guns	talk	talk.politics.guns
mideast	talk	talk.politics.mideast
misc	talk	talk.politics.misc
religion	talk	talk.religion.misc