Semi-Supervised Text Classification Using Partitioned EM

Gao Cong[1], Wee Sun Lee[1], Haoran Wu[1], Bing Liu[2]

[1]Department of Computer Science, National University of Singapore, Singapore 117543
{conggao, leews, wuhaoran}@comp.nus.edu.sg
[2]Department of Computer Science, University of Illinois at Chicago, Chicago, IL, United States 60607-7053
liub@cs.uic.edu

Abstract. Text classification using a small labeled set and a large unlabeled data is seen as a promising technique to reduce the labor-intensive and time consuming effort of labeling training data in order to build accurate classifiers since unlabeled data is easy to get from the Web. In [16] it has been demonstrated that an unlabeled set improves classification accuracy significantly with only a small labeled training set. However, the Bayesian method used in [16] 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. This may not be the case in many applications. In many real-life applications, a class may cover documents from many different topics, which violates the one-to-one correspondence assumption. In such cases, the resulting classifiers can be quite poor. In this paper, we propose a clustering based partitioning technique to solve the problem. This method first partitions the training documents in a hierarchical fashion using hard clustering. After running the expectation maximization (EM) algorithm in each partition, it prunes the tree using the labeled data. The remaining tree nodes or partitions are likely to satisfy the one-to-one correspondence condition. Extensive experiments demonstrate that this method is able to achieve a dramatic gain in classification performance.

Keywords. Text mining, text classification, semi-supervised learning, labeled and unlabeled data

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 categories that are manageable and easy to understand. 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 used 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., naïve Bayesian classifier [16], *k*-nearest neighbor [18], and support vector machines [10]. These existing techniques have been used to automatically catalog news articles [13], classify Web pages [5] and learn the reading interests of users [12]. 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 documents.

However, the main drawback of these classic techniques is that a large number of labeled training documents are needed in order to build accurate classification systems. The labeling is often done manually, which is very labor intensive and time consuming. Recently, Nigam et al. [16] proposed to use a small set of labeled data and a large set of unlabeled data to help deal with the problem. They showed that the unlabeled set is able to improve the classification accuracy substantially, and the number of labeled set can be extremely small. This approach thus saves labor, as the unlabeled set is often easy to obtain from sources such as the Web and UseNet.

Nigam et al. [16] proposed a technique that utilizes the EM (Expectation Maximization) algorithm [6] to learn a classifier from both labeled and unlabeled data. The EM algorithm is a class of iterative algorithms for maximum likelihood estimation in problems with missing data. Its objective is to estimate the values of the missing data (or a distribution over the missing values) using existing values. In the context of learning from labeled and unlabeled sets, the EM algorithm tries to estimate the class labels of the unlabeled data. In each iteration, the EM algorithm performs value estimation by making use of the naïve Bayesian classification method [16]. The method works quite well when the data conforms to its generative assumptions (see next paragraph).

However, since the algorithm in [16] makes use of the naïve Bayesian method, it suffers for the shortcomings of the naïve Bayesian method. In particular, 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 despite the 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. Several examples are given as follows:

– Given one's bookmarks, one wants to find all the interesting documents rom a document collection. We can build a binary classifier for the purpose. However, the set of documents in the bookmarks are unlikely to belong to a single topic because one is often interested in many things, e.g., those related to one's work and those related to one's hobbies. The labeled negative documents are also unlikely to have come from a single topic.
– Junk e-mail filtering. Filtering junk e-mails is a typical binary classification problem ("junk" and "normal" e-mails) in which each class contains multiple sub-topics.

In these cases, the classification accuracy using EM algorithm suffers badly. In [16], attempts are made to relax this restriction. That is, it allows many mixture components to one class (each component represents a particular sub-topic). To determine the number of mixture components for a particular class, it uses cross-validation on the training data. However, experimental results on Reuters data in [16] show that it is unsatisfactory. It has almost the same performance as the naïve Bayesian method without using unlabeled data. Our experimental results on Reuters data and 20 Newsgroup (in section 3) also show that the method in [16] of choosing the number of mixture component through cross-validation hardly improve the performance of naïve Bayesian.

In this paper, we propose an alternative method for handling the difficulty in the case of two-class classification problems. We perform hierarchical clustering of the data to obtain a tree-based partition of the input space (training documents). The idea is that if we partition the input space into small enough portions, a simple two-component mixture will be a good enough approximation in each partition. We then utilize the labeled data in order to prune the tree so that for each node of the pruned tree (partition of the input space), there is enough training data for satisfactory performance. The advantage of using hierarchical clustering is that we do not need to know a suitable number of components in advance. Instead, we only need to partition the input space into small enough components such that pruning can be used to effectively find the correct sized components. Each test document to be classified is first clustered into a leaf node of the pruned hierarchical tree from top to bottom, then we use the classifier built in the leaf node for classification. We also introduce another innovation in the use of early stopping for the EM algorithm. We found that when the two-component mixture model is a poor model for the data, running EM gives much poorer result than simply using the naïve Bayesian classifier (that does not take advantage of the unlabeled data). We use a simple test based on cross-validation to decide when to stop the iterations of the EM algorithm for each partition of the input space. In this way, we gain performance when the model is reasonable for the partition but do not suffer if the model is poor. So far, extensive experiments have been conducted showing that the proposed method is able to improve the classification dramatically with a varied number of unlabeled data.

The rest of the paper is organized as follows. After reviewing related work, we present our clustering-based method to solve the multiple mixture component problems in section 2. In section 3, we empirically evaluate the proposed technique using two document text collections, i.e., the Reuters data and the 20 newsgroup data. Finally, we give the conclusions in section 4.

1.1 More Related Work

The most closely related work to ours is [16] as discussed above. Another popular technique using unlabeled data is co-training [1, 7, 8, 15, 17], which use two complementary predictors to iteratively label the unlabeled data. Clearly, these co-training based approaches are different from our proposed technique and are not specially designed to handle classification problem with multiple component in each class.

Other methods for using unlabeled data to improve classification include work on transduction using support vector machines [11], the maximum entropy discrimination method [9], and transformation of the input feature spaces using information of the unlabeled data [19]. The unlabeled data is also applied in hierarchical text classification [2], and large multi-class text classification [7].

2 Classification Techniques

We first describe the naïve Bayesian classifier and EM algorithm followed by our partitioning, pruning and early stopping techniques. We assume throughout this paper that we are dealing with a two-class classification problem.

2.1 Naïve Bayesian Classifier

Naive Bayesian method is one of the most popular techniques for text classification. It has been shown to perform extremely well in practice by many researchers [14].

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 $V = < w_1, w_2, \ldots , w_{|v|} >$. The vocabulary is the set of all words we consider for classification. We also have a set of pre-defined classes, $C=\{c_1, c_2, \ldots , c_{|C|}\}$ (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, $\mathrm{P}(c_j|d_i)$, where c_j is a class and d_i is a document. Based on the Bayesian probability and the multinomial model, we have

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

and with Laplacian smoothing,

$$\mathrm{P}(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i)\mathrm{P}(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N(w_s, d_i)\mathrm{P}(c_j \mid d_i)} \qquad (2)$$

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

Finally, assuming that the probabilities of the words are independent given the class, we obtain

$$\mathrm{P}(c_j \mid d_i) = \frac{\mathrm{P}(c_j)\prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_i,k} \mid c_j)}{\sum_{r=1}^{|C|} \mathrm{P}(c_r)\prod_{k=1}^{|d_i|} \mathrm{P}(w_{d_i,k} \mid c_r)} \qquad (3)$$

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

2.2 The EM Algorithm

The subsection describes briefly the basic EM and its extension within the probabilistic framework of naive Bayesians text classification. We refer interested readers to [16] for details.

2.2.1 Basic EM

The Expectation-Maximization (EM) algorithm [6] is a popular class of iterative algorithms for maximum likelihood estimation in problems with incomplete data. It can be used to fill the missing value in the data using existing values in the data by computing the expected value for each missing value. The EM algorithm consists of two steps, the Expectation step, and the Maximization step. The Expectation step basically fills in the missing data. The parameters are estimated in the Maximization step after the missing data are filled or reconstructed. This leads to the next iteration of the algorithm. We use the naïve Bayesian classifier that is trained using only the labeled examples as the starting point for EM.

For the naive Bayesian classifier, the steps used by EM are identical to that used to build the classifier, equation (3) for the Expectation step, and equations (1) and (2) for the Maximization step. Note that given the document, the probability of the class now takes the value in [0,1] for the unlabeled data and in {0,1} for the labeled data.

The basic EM approach depends on the assumption that there is a one-to-one correspondence between mixture components and classes. When the assumption does not hold, the unlabeled data often harm the accuracy of a classifier [16]. [16] proposes the *multiple mixture components per class* technique, in which the above assumption is replaced with a many-to-one correspondence between mixture components and classes.

2.2.2 Multiple Mixture Components per Class (M-EM)

We call EM with multiple mixture components per class M-EM. M-EM assumes that a class may be comprised of several different sub-topics, and thus using multiple mixture components might capture some dependencies between words for the classes. Due to space limitation, interested readers refer to [16] or our full paper for M-EM.

One crucial problem of M-EM is how to determine the number of mixture components. [16] used leave-one-out (see Section 2.3 for the meaning) cross-validation for the purpose. But in practice such a cross-validation approach usually cannot find the optimal number of mixture components. Moreover, M-EM is sensitive to the number of mixture components. As a result, M-EM showed hardly any improvement over NB in experiments of [16].

2.3 Classification based on Hidden Clusters (CHC)

Our classification technique has three important components: hierarchical clustering (partitioning), pruning and early stopping.

2.3.1 Hierarchical Clustering

Running EM with the naïve Bayesian classifier with two mixture components fails badly when the actual number of components is larger than two. Our main idea is to recursively partition the input space until the number of components in each partition is no more than two. We choose to do each stage of the partitioning using "hard" clustering with a two-component naïve Bayesian model. The motivation for using hierarchical clustering for partitioning comes from the observation that smaller components in a mixture can often be clustered into larger components. The large components can hence be recursively divided until each partition contains no more than two components so that EM with a two-component mixture works well. The following example illustrates a situation where EM with a two-component mixture fails but hierarchical clustering works well.

Consider a situation where we have four newsgroups “baseball”, “IBM hardware”, “hockey” and “Macintosh hardware”. Assume that a user is interested in “baseball” and “IBM hardware” (positive class) but not the other classes (negative class). In this case, because both the positive and negative classes contain two components, the two-component mixture model is not suitable for the data. However, suppose we first cluster the training documents into two clusters (partitions); if the clustering method is good enough, most documents about “IBM hardware” and “Macintosh hardware” should be clustered into one cluster and most documents about “baseball” and “hockey” should be in another cluster. Now within each cluster, both the positive and negative classes mainly contain documents only from one category. Thus in each

Original training data:

Positive	Negative
rec.sport.baseball comp.sys.ibm.pc.hardware	rec.sport.hockey comp.sys.mac.hardware

Partition data into two clusters:

Cluster 1	Cluster 2
rec.sport.baseball rec.sport.hockey	comp.sys.mac.hardware comp.sys.ibm.pc.hardware

Build text classifier in each cluster

Classifier 1		Classifier 2	
Positive	Negative	Positive	Negative
rec.sport.baseball	rec.sport.hockey	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware

Fig. 1. An example of using clustering method to help improve data model for classification

partition, the assumption of two-component mixture model is basically satisfied and we can use EM combining the labeled and unlabeled data of this cluster to build a text classifier.

Fig. 1 shows the idea for the above example. For the newsgroups and classes used in this example, we find that on a training set of 40 labeled examples, 2360 unlabeled examples and a test set of 1600 examples, naïve Bayesian with a two-component mixtures achieves an accuracy of 68% while running EM with the help of the unlabeled examples reduces the accuracy to 59.6%. On the other hand, if we partition the space into two before running EM, we achieve an accuracy of 83.1%.

We use “hard” clustering (also called vector quantization) with the two-component naïve Bayesian model to recursively partition the input space. The labeled data is included in the partitioning algorithm but the labels are not used. This is because the labels can sometimes be unhelpful for partitioning the data into two components as in the case in Fig. 1, where addition of the labels will only confuse the algorithm. The algorithm is shown in Fig. 2. Line 4) uses the Equation (1) and (2) and line 5) uses the Equation (3). The hard clustering method in each node is guaranteed to converge to the maximum of the log likelihood function in a finite number of steps.

```
Cluster-Partition (Node A)
1)  If node A contains no more than 2 labeled documents, then stop
2)  Randomly partition A into two sets L and R
3)  Repeat until convergence
4)  Build a naïve Bayesian model assuming that L and R contain data for two classes.
5)  Reassign each document to L and R based on
    the classification of the naïve Bayesian model.
6)  Cluster-Partition (L)
7)  Cluster-Partition (R)
```

Fig. 2. Algorithm for recursive partitioning based on hard clustering

2.3.2 Pruning

We run the recursive partitioning algorithm until the nodes contain no more than 2 labeled examples and then perform pruning to merge partitions. The pruning step is required because of the following two reasons: first, some partitions are too small; Second, we notice that when a partition satisfies the two-component mixture model further partitioning usually deteriorates the accuracy of classifier.

We use the training error on the labeled set as the pruning criterion under the assumption that if the EM algorithm on the two-component mixture works well, the training error on the labeled set will be small; but if it does not work well, the training error will be large. First, EM with the two-component mixture model is run on every node in the tree (initialized using the labeled data). Pruning is then performed in a bottom-up fashion from the leaves on. The algorithm is given in Fig. 3:

```
Prune-Partition (Node N)
1)  Prune-Partition (the left children of N);
2)  Prune-Partition (the right children of N);
3)  If the number of errors $E_P$, on the labeled set in the node, is no more than the combined
    number of errors $E_C$, on the labeled sets in its two children;
4)  then the children are pruned away.
5)  else keep the children, and substitute $E_P$ by $E_C$.
```

Fig. 3. Algorithm for pruning partitions

2.3.3 Early Stopping

After pruning, we are left with a partitioned input space where we want to run EM with a two-component mixture model in each partition. For some data sets, even the partitioned input space is still not perfectly suitable for two-component mixture model. When the model is incorrect, we notice that the generalization performance of the algorithm gets worse as the number of iterations of EM increases. In such a case, it is often better to use the naïve Bayesian classifier or to run only a small number of iterations of EM instead of running EM to convergence. Moreover, different number of iterations of EM may be required for different partitions of the input space.

We use leave-one-out cross validation to estimate the generalization error at each iteration of EM. For the naïve Bayesian classifier, this can be efficiently calculated by subtracting the word counts of each document from the model before it is tested. For EM, we use the same approximation as used in [16] in order to have an efficient implementation. In the approximation, all the data (labeled and unlabeled) are used in the EM algorithm. In each iteration, before a labeled document is tested, the word counts of the document are subtracted from the model just like in the naïve Bayesian algorithm. We get:

$$\mathrm{P}^{'}(w_t \mid c_j) = \frac{1+\sum_{i=1}^{|D|} N(w_t, d_i)\mathrm{P}(c_j \mid d_i) - N(w_t, d_v)P(c_j \mid d_v)}{|V| + \sum_{s=1}^{|V|}(\sum_{i=1}^{|D|} N(w_s, d_i)\mathrm{P}(c_j \mid d_i) - N(w_t, d_v)P(c_j \mid d_v))} \quad (4)$$

For each labeled document d_v, we calculate P(c_j $|d_v$) using Equation (3) by replacing P(w_t | c_j) with P'(w_t | c_j) and determine its class. We then can compute the classification accuracy of the resulting classifier for each EM iteration on the set of labeled documents. EM will stop when the accuracy decreases.

2.3.4 Using the CHC classifier

For a test document to be classified, the CHC classifier built from training data using the methods given in the above is used at the following two steps:

(1) We first cluster the documents to one leaf node of the hierarchy obtained in Section 2.3.2 from top to bottom. At each node of the hierarchy (except the leaf nodes), there is a clustering model obtained from training data which determines whether the test document belongs to its left sub-cluster or right sub-cluster. In this way, each test document will **be clustered into the only** leaf node. The classifier built at the leaf node will be used to classify the test document at the next step.

(2) We then classify the document using Equation (3) with the parameters obtained in Section 2.3.3.

3 Experiments

3.1 Experimental Setup

The empirical evaluation is done on four groups of data, each group consisting of many datasets. Two data groups are generated from the 20 newsgroups collection, which consists of 19997 articles divided almost evenly among 20 different UseNet discussion topics. The other two data groups used in our evaluation are generated from the Reuters-21578 collection, which is composed of 12902 articles covering 135 potential topic categories. The four data groups are listed as follows (details are given in our full paper due to space limitation):

– 20A: It is composed of 20 datasets. Each positive set is one of 20 newsgroups and the negative set is composed of the other 19 newsgroups.

- 20B: It is composed of 20 datasets. Each of the positive sets consists of randomly selected topics from the 20 newsgroups collection with the rest of the topics becoming the negative sets.
- RA: It is composed of 10 datasets. Each positive set is one of the 10 most populous categories and each negative set consists of the rest of the documents in Reuters-21578 collection.
- RB: It is composed of 20 datasets. Each positive set consists of randomly selected categories from the 10 most populous categories from the Reuters-21578 collection. Each negative set consists of the rest of the categories.

For all four data groups, we remove stop words, but do not perform stemming. We also remove those rare words from feature set that appear in fewer than 3 documents in the whole set of documents for both the 20 newsgroups and Reuters-21578 collections.

Results on all four data groups are reported as averages over 10 random selections of labeled sets. For data group 20A and 20B, each test set contains 4000 documents, each unlabeled set contains 10000 documents, and the labeled sets with different sizes (as shown in Figures 4 & 5) are selected randomly from the remaining documents. For each dataset in data groups RA and RB, we use the “ModApte” train/test split, leading to a corpus of 9603 training documents and 3299 test documents. In each run on the Reuters collection, 8000 documents are selected randomly from the 9603 training documents as unlabeled set and labeled sets of different sizes are selected from the remaining documents.

We compare the performance of the following techniques:

- Naïve Bayesian classifier (NB)
- EM run to convergence (EM)
- EM with early stopping (E-EM) but without partitioning the space
- Classification based on Hidden Clusters (CHC), i.e., EM with partitioning using hard clustering and early stopping
- Multiple mixture components EM (M-EM).

For M-EM, [16] performed experiments by assuming one component for the positive class and multiple components for the negative class. The leave-one-out cross-validation method is used to determine the number of mixture components. We test this technique on the datasets 20A and RA whose positive classes are generated by a single mixture component while negative has multiple sub-topics. The candidate number of mixture components is set from 1 to 20. We set this range since the experiment results in [16] show that both the optimal selection and the selection via cross-validation of components for dataset RA are always less than 20 for dataset RA. For dataset 20A, we also set 20 as the upper bound for cross-validation since the dataset contains 20 sub-topics. We think that it is difficult to extend M-EM to the case that both the negative and positive classes consist of multiple mixture components since [16] does not describe how to do cross-validation to obtain the number of mixture components for the case. Thus, we do not test the technique for the other three datasets in our experiments.

We use F_1 score (see e.g. [3]) to evaluate the effectiveness of our technique on the datasets in 20A, 20B, RA and RB. F_1 score is computed as follows: $F_1 = 2pr/(p + r)$, where p is the precision and r is the recall (in our case, they are measured on the

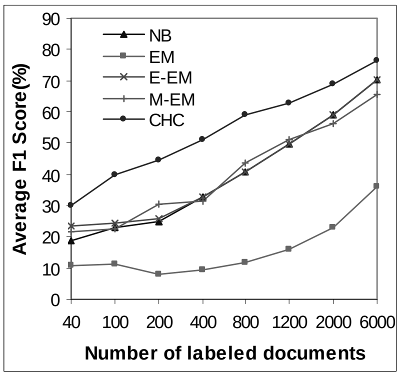

Fig. 4. Average F_1 score of datasets in 20A

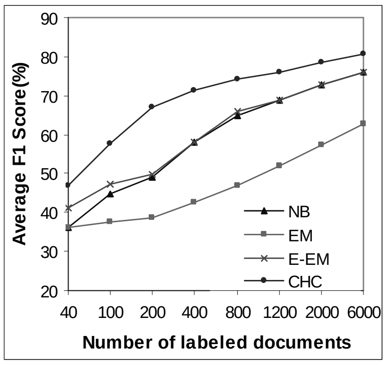

Fig. 5. Average F_1 score of datasets in 20B

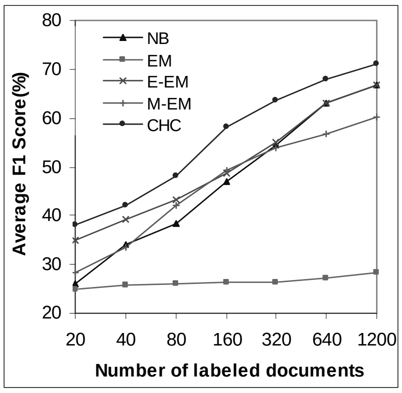

Fig. 6. Average F_1 score of datasets in RA

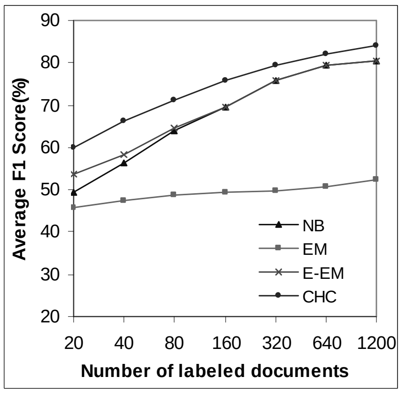

Fig. 7. Average F_1 score of datasets in RB

positive class, which is the class that we are interested in). Notice that accuracy is not a good performance metric for datasets in 20A, 20B, RA and RB since high accuracy can often be achieved by always predicting the negative class.

3.2 Experimental Results and Analyses

Figures 4, 5, 6 and 7 show the average F_1 score on datasets in data groups 20A, 20B, RA and RB. The vertical axis shows the average F_1 scores on each data group and the horizontal axis indicates the number of labeled documents. Tables 1 and 2 show the F_1 score for the positive category of each dataset with 200 labeled documents on data groups 20A and 20B respectively. Due to space limitation, we do not give the detailed results of data groups RA and RB.

CHC performs significantly better than other methods, especially when the number of labeled documents is small. For example, on data group 20A, with 200 labeled documents, on average, NB gets 0.252 F_1-score, EM gets 0.082, E-EM achieves 0.253, M-EM gets 0.302 while CHC achieves 0.442. This represents a 75.4% increase

Table 1. Results of datasets in 20A with 200 labeled documents

	F_1-score				
Set	NB	EM	E-EM	M-EM	CHC
0	26.7	6.0	26.3	35.9	**44.9**
1	15.4	9.2	15.0	11.0	**38.9**
2	12.9	4.9	12.8	9.2	**27.5**
3	14.9	1.2	14.9	5.1	**30.8**
4	12.7	2.8	12.3	13.7	**22.5**
5	26.9	7.4	26.9	7.0	**38.8**
6	18.5	1.1	18.4	14.3	**35.3**
7	14.6	0.9	14.6	6.0	**41.9**
8	34.1	3.1	34.1	40.4	**54.0**
9	34.5	20.1	34.5	41.3	**61.2**
10	51.5	37.8	50.9	63.6	**75.5**
11	50.8	10.6	46.1	51.6	**69.6**
12	6.4	0.9	6.4	6.9	**17.9**
13	21.3	5.0	21.3	36.8	**54.2**
14	26.2	2.5	26.2	60.0	**60.2**
15	36.5	11.9	41.6	49.6	**59.5**
16	22.8	3.0	22.8	**56.3**	44.8
17	46.3	20.4	50.8	**67.2**	63.4
18	16.6	8.5	15.7	14.3	**20.9**
19	14.2	6.9	14.2	13.6	**21.9**
Ave	25.2	8.2	25.3	30.2	**44.2**

Table 2. Results of datasets in 20B with 200 labeled documents

	F_1-score			
Set	NB	EM	E-EM	CHC
0	27.7	13.1	27.7	**37.3**
1	41.4	21.7	42.0	**60.1**
2	41.5	33.1	41.5	**60.6**
3	28.3	15.9	28.9	**55.5**
4	41.6	25.1	41.6	**73.1**
5	54.1	49.3	54.1	**76.1**
6	32.2	24.5	32.2	**49.3**
7	63.4	60.2	63.4	**71.1**
8	38.1	23.1	38.1	**60.7**
9	37.9	25.6	37.8	**50.6**
10	40.5	16.4	40.5	**61.6**
11	36.5	30.1	36.5	**59.8**
12	72.2	68.8	73.8	**89.9**
13	46.3	29.4	46.3	**60.9**
14	37.8	20.9	37.8	**67.2**
15	63.6	63.3	63.6	**73.8**
16	65.3	55.5	65.3	**79.4**
17	74.1	73.7	74.1	**82.1**
18	70.5	67.5	70.5	**87.3**
19	70.9	58.7	70.9	**82.8**
Ave	49.2	38.8	49.3	**67.0**

of F_1 score of CHC against NB, a 439% increase against EM, a 74.7% increase against E-EM and a 46.4% increase against M-EM.

Figures 4 and 6 show that M-EM do not consistently improve the performance of NB. In fact, M-EM basically achieved similar results as NB with small numbers of labeled documents. This is consistent with the results reported in [16], where experiments using 50 labeled documents also showed that sometimes M-EM works better than NB while at other times it does not.

One interesting finding of our experiments is that E-EM can improve the performance of NB when the number of labeled documents is small. This means that EM may improve the performance of NB at the first few iterations with small label sets. The performance can deteriorate with additional iterations as can be seen from the results of NB, (converged) EM and E-EM in Figures 4 to 7.

Experiments on the first four data groups show that converged EM does not improve the performance of NB in general. This is because that the underlying model of EM does not fit the data that covers multiple categories. The reason for CHC to achieve better performance than NB while EM is worse than NB is that CHC gets better components (satisfies simple two-component mixture) by clustering and pruning.

We also do experiments on a group of datasets whose positive set and negative set is one category of 20 newsgroups respectively, i.e. the two-component model fits the data well. We find that EM is able to perform better than NB in the case, and we also observe that our CHC method can achieve similar accuracy as EM in this case. Due to space limitation, we do not report detailed results here. In summary, we can conclude that no matter what the characteristics of the datasets are, CHC does much better than or as well as other techniques.

4 Conclusions

We proposed a new method for improving the performance of EM with the naïve Bayesian classifier for the problem of learning with labeled and unlabeled examples when each class is naturally composed of many clusters. Our method is based on recursive partitioning and pruning with the aim of partitioning the input space so that within each partition, there is a one-to-one correspondence between the classes and the clusters in the partition. Extensive experiments show that the proposed method significantly improves the performance of the EM with naïve Bayesian classifier.

References

1. Blum, A., Mitchell, T. Combining labeled and unlabeled data with co-training. Proceedings of COLT-98, (1998) 92-100
2. Boyapati,V. Improving hierarchical text classification using unlabeled data. Proceedings of SIGIR, (2002)
3. Bollmann, P., Cherniavsky, V. Measurement-theoretical investigation of the mz-metric. Information Retrieval Research (1981) 256-267
4. Cohen, W. Automatically extracting features for concept learning from the Web. Proceedings of the ICML, (2000)
5. Craven, M., DiPasquo, D., Freitag, D., MaCallum, A., Mitchell, T., Nigam, K., Slattery, S. Learning to extract symbolic knowledge from the World Wide Web. Proceedings of AAAI-98, (1998) 509-516
6. Dempster, A., Laird, N., Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B, 39(1), (1977) 1-38
7. R. Ghani, Combining Labeled and Unlabeled Data for MultiClass Text Categorization. In Proceedings of the ICML, (2002)
8. Goldman, S., Zhou, Y., enhanced supervised learning with unlabeled data. In Proceedings of the ICML, (2000)
9. Jaakkola, T., Meila, M., Jebara, T. Maximum entropy discrimination. Advances in Neural Information Pcocessing Systems 12, (2000) 470-476
10. Joachims, T. (1998). Text categorization with Support Vector Machines: learning with many relevant features. Proceedings of ECML-98, (1998) 137-142
11. Joachims, T. (1999). Transductive inference for text classification using support vector machines. Proceedings of ICML-99,(1999) 200-209
12. Lang, K. Newsweeder: Learning to filter netnews. Proceedings of ICML, (1995) 331-339
13. Lewis, D. Gale, W. A sequential algorithm for training text classifiers. Proceedings of SIGIR-94, (1994) 3-12
14. McCallum, A., Nigam, K. A comparison of event models for naïve Bayes text classification. AAAI-98 Workshop on Learning for Text Categorization. AAAI Press (1998)
15. Nigam, K., Ghani, R. Analyzing the effectiveness and applicability of co-training. Ninth International Conference on Information and Knowledge Management (2000) 86-93
16. Nigam, K., McCallum, A., Thrun, S., Mitchell, T. Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 39(2/3), (2000) 103- 134
17. Raskutti, B. Ferra, H. Kowalczyk, A. Combining Clustering and Co-training to Enhance Text Classification Using Unlabelled Data. In Proceedings of the KDD, (2002)
18. Yang, Y. An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1, 67-88, (1999)
19. Zelikovitz, S. Hirsh, H. using LSI for text classification in the presence of background text. In Proceedings of the CIKM, (2001)