# Text Classification by Labeling Words

**Bing Liu[1], Xiaoli Li[2], Wee Sun Lee[2] and Philip S. Yu[3]**

[1] Department of Computer Science, University of Illinois at Chicago. liub@cs.uic.edu
[2] Department of Computer Science, National University of Singapore. {lixl, leews}@comp.nus.edu.sg
[3] IBM T. J. Watson Research Center. psyu@us.ibm.com

**Abstract**

Traditionally, text classifiers are built from labeled training examples. Labeling is usually done manually by human experts (or the users), which is a labor intensive and time consuming process. In the past few years, researchers investigated various forms of semi-supervised learning to reduce the burden of manual labeling. In this paper, we propose a different approach. Instead of labeling a set of documents, the proposed method labels a set of representative words for each class. It then uses these words to extract a set of documents for each class from a set of unlabeled documents to form the initial training set. The EM algorithm is then applied to build the classifier. The key issue of the approach is how to obtain a set of representative words for each class. One way is to ask the user to provide them, which is difficult because the user usually can only give a few words (which are insufficient for accurate learning). We propose a method to solve the problem. It combines clustering and feature selection. The technique can effectively rank the words in the unlabeled set according to their importance. The user then selects/labels some words from the ranked list for each class. This process requires less effort than providing words with no help or manual labeling of documents. Our results show that the new method is highly effective and promising.

## Introduction

The classic approach to building a text classifier is to first (often manually) label a set of training documents, and then apply a learning algorithm to build the classifier. This method is called *supervised learning*.

Manual labeling of a large set of training documents is a bottleneck of this approach as it is a time consuming process. To deal with this problem, (Nigam *et al.*, 2000; Blum & Mitchell, 1998) propose the idea of using a small labeled set of every class and a large unlabeled set for classifier building. (Denis, 1998; Liu *et al*., 2002) also propose to learn from positive and unlabeled examples (without labeled negative examples). These research efforts aim to reduce the burden of manual labeling.

In this paper, we explore an alternative approach. Instead of asking the user to label a set of training documents, we ask him/her to provide some representative words for each class. These words and an unlabeled document set are then used to build a classifier. The main idea of the proposed approach is as follows: The user first provides a set of representative words for each class. These words are employed to identify a set of reliable documents for the class from the unlabeled set. This set of documents acts as the initial set of labeled training documents for the class. A classification algorithm is then applied to build a classifier. In this paper, we use the naïve Bayesian classification method (Lewis & Gale, 1994; McCallum & Nigam 1998a) and the Expectation Maximization (EM) algorithm (Dempster, Laird & Rubin, 1977) for classifier building[1].

The reason that this technique works is because the class of a text document essentially depends on the words that it contains. Most classification techniques use only words as features. Hence, if we can obtain a set of representative keywords for each class, we can extract an initial set of documents for the class by comparing the similarity of a document with the set of representative words of the class.

The key problem of the proposed approach is how to obtain a set of representative words for each class. One method is to ask the user to provide them. We tried this method with some success. However, a problem also revealed itself, i.e., it is hard for the user to provide a big number of keywords that are required for accurate learning. Thus, the issue is how to help users to provide a sufficient number of representative words for each class.

Clearly, it is not appropriate to list all the distinctive words in the unlabeled document set and to ask the user to select from them because the number of words is often too large (tens of thousand or more). It will be much better if the words can be ranked according to their importance or discriminative power. With this ranked list, it becomes a much easier task to select those important words for each class. The question is how to identify the set of important words automatically for an unlabeled document set (with no class information). We propose a novel approach to tackle the problem, which consists of three steps:

1. Cluster the unlabeled documents. This step aims at finding inherent clusters in the data. Each resulting cluster is treated as a category or class.



[1] The focus of this paper is to demonstrate the potential of learning without any labeled training documents. We believe that other classification techniques may also be used in this step.

2. Perform feature selection on the resulting clusters to identify those important words of each cluster. The result is a list of all words ranked according to their discriminative power of all the clusters/classes.
3. The user (or human expert) then inspects the ranked list of words to select a small set of representative words for each class of documents that he/she is interested in. We call this process *word labeling*.

Our experimental results show that the proposed technique is highly effective. The time taken to select the set of representative words is small.

## Related Work

Existing text classification techniques can be grouped into three types, *supervised learning*, *semi-supervised learning*, and *unsupervised learning* (or *clustering*). The proposed technique is related to but significantly different from all these existing approaches. We compare them below.

In *supervised learning*, a set of (often manually) labeled training documents of every class is used by a learning algorithm to build a classifier. Existing text classification techniques include the naive Bayesian method (Lewis & Gale 1994; McCallum & Nigam 1998a), support vector machines (SVM) (Vapnik, 95) and many others. As discussed in the Introduction, our technique is different. We only ask the user to select a set of important words for each class from a ranked list, rather than reading and labeling a set of documents, which requires more effort.

Due to the problem of manual labeling, *semi-supervised learning* is studied, which includes two main paradigms: (1) learning from a small set of labeled examples and a large set of unlabeled examples; and (2) learning from positive and unlabeled examples (with no labeled negative examples). Many researchers have studied learning in the first paradigm (e.g., Nigam *et al.*, 2000; Blum & Mitchell 1998; Bockhorst & Craven, 2002; Ghani, 2002; Goldman & Zhou, 2000). These are different from our work as we use no manually labeled examples.

In learning from positive and unlabeled examples, some theoretical studies and practical algorithms are reported in (Denis 1998, Liu *et al*, 2002; Yu, Han & Chang, 2002). Again, our proposed technique is different as it does not use any manually labeled documents.

Active learning is another method for reducing the number of labeled documents (McCallum and Nigam, 1998b). In active learning, the algorithm sequentially selects an example to be labeled next after receiving the labels for the previously selected examples. As in the other methods above, active learning requires the user to label documents, which is different from our approach.

The proposed work is also related to *unsupervised learning* or *clustering* (Jain & Dubes, 1988) and *constraint-based clustering* (e.g., Wagstaff *et al*, 2001). Although clustering is not commonly used for classification, its results can be employed as a classifier as follows: Given a new document, it is classified to the cluster (which represents a class) that is closest to it. This method, however, has two major shortcomings: (1) the accuracy is often low, and (2) the resulting clusters may not be what the user wants.

Recent work (e.g., (Basu, Banerjee & Monney, 2002)) tries to bring clustering closer to classification by using a small number of labeled documents as seeds to initialize *k*-means clustering. The original *k*-means algorithm selects initial seeds randomly. Again, our work is different. We only use clustering (and also feature selection) to help us to identify a set of important words in a text collection so that the user can select a set of representative words for each class.

In (McCallum & Nigam, 1999), a method is proposed to utilize user-specified keywords to help build classifiers. However, as we found in our experiments, it was very hard for the user to provide such keywords. This is similar to the well known problem of *knowledge acquisition* in building expert systems. Our technique can effectively assist the user to select a set of representative words.

## The Proposed Technique

This section presents the proposed technique. Given a set $U$ of unlabeled documents, our technique has five steps.

Step 1: Cluster the documents in $U$.

Step 2: Perform feature selection on the clusters.

Step 3: Identify (by the user) a set of representative words for each class of documents that he/she is interested in.

Step 4: Automatically label an initial set of documents.

Step 5: Build the final classifier.

Below, we present these steps in turn.

### Clustering the Documents in the Unlabeled Set

This step finds inherent clusters in the unlabeled set. We use the popular *k*-means algorithm due to its efficiency, although other methods can also be applied. The inputs to *k*-means are the set $U$ and the number of clusters $k$. Note that the value of $k$ does not matter much in our case as long as it is not too small because we do not use the resulting clusters as the classifier. We only use them to help us identify some important words in $U$. In clustering, the cosine similarity metric from information retrieval (Salton & McGill, 1983) is used as the distance measure.

### Feature Selection

After clustering, $k$ document clusters are produced. We treat each cluster as a distinctive class. This gives us a set of labeled documents of $k$ classes. A feature selection technique from supervised learning is then applied to identify important words in each cluster.

There are many feature selection techniques for supervised learning (Yang & Pedersen, 1997). In this research, we utilize the entropy-based method. This method computes

the information gain of each word and then ranks all the words in $U$ according to their gain values. Those words with higher gain values have higher discriminative power and are ranked higher. These top-ranked words are regarded as the important words in the unlabeled set $U$.

### Selecting Representatives Words for Each Class

This step is performed manually by the user. We assume that the user is familiar with the topics of the documents and also knows the classes that he/she is interested in.

Since the words are ranked according to their importance, the user simply inspects the top ranked words and decides which class each word belongs to. Note that a word can be selected as a representative word for more than one class. Although a word may be more important to one class than another, we tried to assign a weight to each class, but it was not effective. The user can also add additional words for any class that are not found at the top of the rank.

### Identifying Initial Documents for Each Class

Once a set of representative words is selected for each class $c_j \in C$ (= $\{c_1, c_2, \ldots, c_n\}$), the set of words is regarded as the *representative document* ($rd_j$) of class $c_j$. We then compare the similarity of each document $d_i$ in $U$ with each $rd_j$ using the popular cosine similarity metric (Salton & McGill, 1983) to automatically produce a set $L_j$ of *probabilistically* labeled documents for each class $c_j$.

The algorithm for this step is given in Figure 1. It works as follows. In lines 1-3, each unlabeled document $d_i$ in $U$ is compared with the representative document $rd_j$ of each class $c_j$ using the function *sim* (the cosine metric). $d_i$ is assigned to the class whose representative document is most similar to $d_i$ if the maximum similarity $s_r$ is greater than 0 (lines 4 and 5). That is, $d_i$ is included in $L_r$. Otherwise, $d_i$ is included in $RU$ (the set of *remaining unlabeled documents*) in line 6. Each document in $RU$ has 0 similarity with every $rd_j$ and is thus left unlabeled. In lines 7-15, we assign a probabilistic class label, $P(c_j|d_i) \leq 1$, to each document $d_i$ in $L_j$. We first rank the documents in $L_j$ based on their similarities. Each of the top $t$ percent of the documents is given $P(c_j|d_i) = 1$. Each of the remaining documents is given a class probability proportionally (lines 12-15) as they may not be as reliable as those top-ranked documents. $t$ is a parameter here. In our experiments, we tried $t$ with 16%, 50%, 84% and 100% (to be explained later) and found that its value does not matter much.

### Building the Classifier Using NB or EM

We build the final classifier using the naïve Bayesian (NB) technique and the EM algorithm. EM also uses NB as the base classifier. Below, we introduce them in turn.

**Naïve Bayesian (NB) Classification:** The basic idea of NB is to use the joint probabilities of words and classes to estimate the probabilities of classes given a document (McCallum & Nigan, 1998a; Lewis & Gale, 1994).

Inputs: $RD = \{rd_1, rd_2, \ldots, rd_n\}$, the set of representative documents of the classes, $c_1, c_2, \ldots, c_n$.
$U = \{d_1, d_2, \ldots d_m\}$, the set of unlabeled documents.
Output: $L = \{L_1, L_2, \ldots, L_n\}$, where $L_j$ is a set of labeled documents for class $c_j$. $L_j$ is initialized to {}.
$RU$, the set of remaining documents in $U$ that are not labeled with any class. It is initialized to {}.

1. **for** each $d_i \in U$ **do**
2. **for** each $rd_j \in RD$ **do**
3. $s_j = sim(d_i, rd_j)$;
4. $s_r = \max(\{s_1, s_2, \ldots, s_n\})$;
5. **if** $s_r > 0$ **then** $L_r = L_r \cup \{d_i\}$;
6. **else** $RU = RU \cup \{d_i\}$;
7. **for** each $L_j \in L$ **do**
8. Rank the documents in $L_j$ according to their similarity values in a descending order;
9. Let $T_j$ be the set of $t$% top-ranked documents in $L_j$;
10. **for** each $d_i$ in $T_j$ **do**
11. $P(c_j|d_i) = 1$;
12. Let $ms$ be the minimal similarity of the documents in $T_j$;
13. **for** each document $d_i$ in $L_j - T_j$ **do**
14. $P(c_j|d_i) = d_i.sim / ms$; // $d_i.sim$ is the similarity of $d_i$
15. 1-$P(c_j|d_i)$ is shared by other classes;

Figure 1: Labeling a set of documents automatically

In the NB formulation, each document in the labeled training set $D$ is considered as an ordered list of words. $w_{d_i,k}$ denotes 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 considered in classification. There is also a set of pre-defined classes, $C = \{c_1, c_2, \ldots, c_n\}$. In order to perform classification, we 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

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

and with Laplacian smoothing,

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

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

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

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

To build a classifier for our purpose, NB will only use the set of labeled documents in each $L_i$. Those remaining unlabeled documents in $RU$ could not be used. The EM algorithm below is able to make use them.

**The EM Algorithm:** The EM algorithm is a popular class of iterative algorithms for maximum likelihood estimation for problems involving missing data (Dempster, Laird & Rubin, 1977). It is often used to fill the missing values in

the data using existing values. Each iteration of EM consists of two steps, the E step and the M step. The E step basically fills in the missing values. The parameters are estimated in the M step after the missing data are filled. This leads to the next iteration of the algorithm. EM can be applied to our problem because the classes of the documents in *RU* (the remaining unlabeled documents) can be regarded as missing. The steps used by EM are identical to those used to build a NB classifier, equations (3) for the E step and equations (1) and (2) for the M step. Note that the probability of the class given the document takes the value in [0, 1] instead of {0, 1}.

Our algorithm, called NB-EM, is given in Figure 2. Initially, we use the labeled documents in $L$ to build a NB classifier NB-C (line 1), which is then applied to classify the documents in *RU*. In particular, NB-C computes $P(c_i|d_j)$ of each document in *RU* (using equation (3)), which is assigned to $d_j$ as its new probabilistic class label. After every $P(c_i|d_j)$ is revised, a new classifier NB-C is built based on the new $P(c_i|d_j)$ values of the documents in *RU*, and the initially labeled documents in $L$. The next iteration starts. This iterative process goes on until EM converges (when the probability parameters stabilize).

NB-EM(L, RU)
1. Build an initial naive Bayesian classifier NB-C using the document set $L$;
2. **Loop** while classifier parameters change
3. **for** each document $d_j \in RU$
4. Compute $P(c_i \mid d_j)$ using the current NB-C;
5. Update $P(w_t|c_i)$ and $P(c_i)$ with the probabilistically assigned class for $d_j$ ($P(c_i|d_j)$) and $L$ (a new NB-C is being built in the process);

Figure 2: The NB-EM algorithm

## Empirical Evaluations

**Datasets**: We used the 20 newsgroup collection[1] in our experiments, which are Usenet articles from 20 different newsgroups. The reason for using this collection is that its topics are general. We can find graduate students to act as human experts to select keywords. Each newsgroup in the collection has approximately 1000 articles. There are four (4) main categories, namely, *Science* (SCI), *Recreation* (REC), *Talk* (TALK) and *Computing* (COMP). Within each main category, there are 4 to 5 sub-categories. We build a classifier for the sub-categories within each main category. That is, each sub-category within a main category is regarded as a class. This gives us four (4) datasets, and each dataset has 4 or 5 classes.

For each dataset, we randomly selected 30% of the documents for testing, and the rest 70% for training.

**Evaluation measures**: We use accuracy as the main evaluation measure. Accuracy is adequate because every class in our four datasets has the same number (1000) of documents. We also used the F-score. However, its trends are similar to accuracy. Thus, its results are not included.

We now present the settings for each step of our technique.

**Clustering (step 1)**: We use the $k$-means algorithm. As mentioned earlier, the number of clusters $k$ does not matter much for our technique. We purposely used $k = 7$ for every dataset, which is different from the actual number of classes of each dataset. We also experimented with $k$ being the actual number of classes of each dataset. Those top-ranked words are very much the same as for $k = 7$.

**Feature selection and word ranking (step 2)**: We used the information gain (or entropy) based approach.

**Selecting representative words (step 3)**: Choosing representative words is a subjective task. Two students and one researcher were asked to serve as experts to choose representative words. Although they choose words independently, there is little difference in the words that they choose because they use the same ranking and most top keywords are obvious. Our experiments also show that slight differences in the selected words have almost no effect on the final classification results.

To test how many words are needed to produce good classifiers, we tried 5, 15, 20, and 30 words per class. We will see that additional words will not help.

**Labeling some training documents (step 4):** We experimented with different $t$ settings (percent of documents in $L_j$ for each class $c_j$ that are given the class probability of 1, $P(c_j|d_i) = 1$). The rest of the documents in $L_j$ are given $P(c_j|d_i) < 1$ according to their similarity values. Those documents with $P(c_j|d_i) = 0$ for every class are in *RU* (the set of remaining unlabeled documents). We experimented with $t$ = 16%, 50%, 84%, and 100%. $t$ = 16% covers documents whose similarities are greater than one standard deviation above the mean similarity (these documents have the highest similarities). $t$ = 84% covers documents whose similarities are greater than one standard deviation below the mean similarity.

**Classifier building (step 5)**: Two classification methods, NB and EM, were experimented. EM was run 8 iterations for each dataset. After that, the results rarely change.

EM-Hard and EM-Soft: We experimented with two EM variations. (1) In each iteration of EM, we do not allow the class probability $P(c_j|d_i)$ of each document $d_i$ in $L_j$ (the set of labeled documents of class $c_j$) to change, but only allow the class probability of each document in *RU* to change. This method is called *EM-Hard*. (2) In each iteration, we allow $P(c_j|d_i)$ of each document $d_i$ in both $L_j$ and *RU* to change. This method is called *EM-Soft*.

**Experimental results:** Figure 3 shows the accuracy results of all datasets. In each chart (each dataset), the results are grouped according to different $t$ values, 16%, 50%, 84% and 100%. For each $t$, we give the results of NB, EM-Hard and EM-Soft. The 4 bars in each group are the results of using 5, 15, 20, 30 representative words respectively.

[2] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes/20_newsgroups.tar.gz

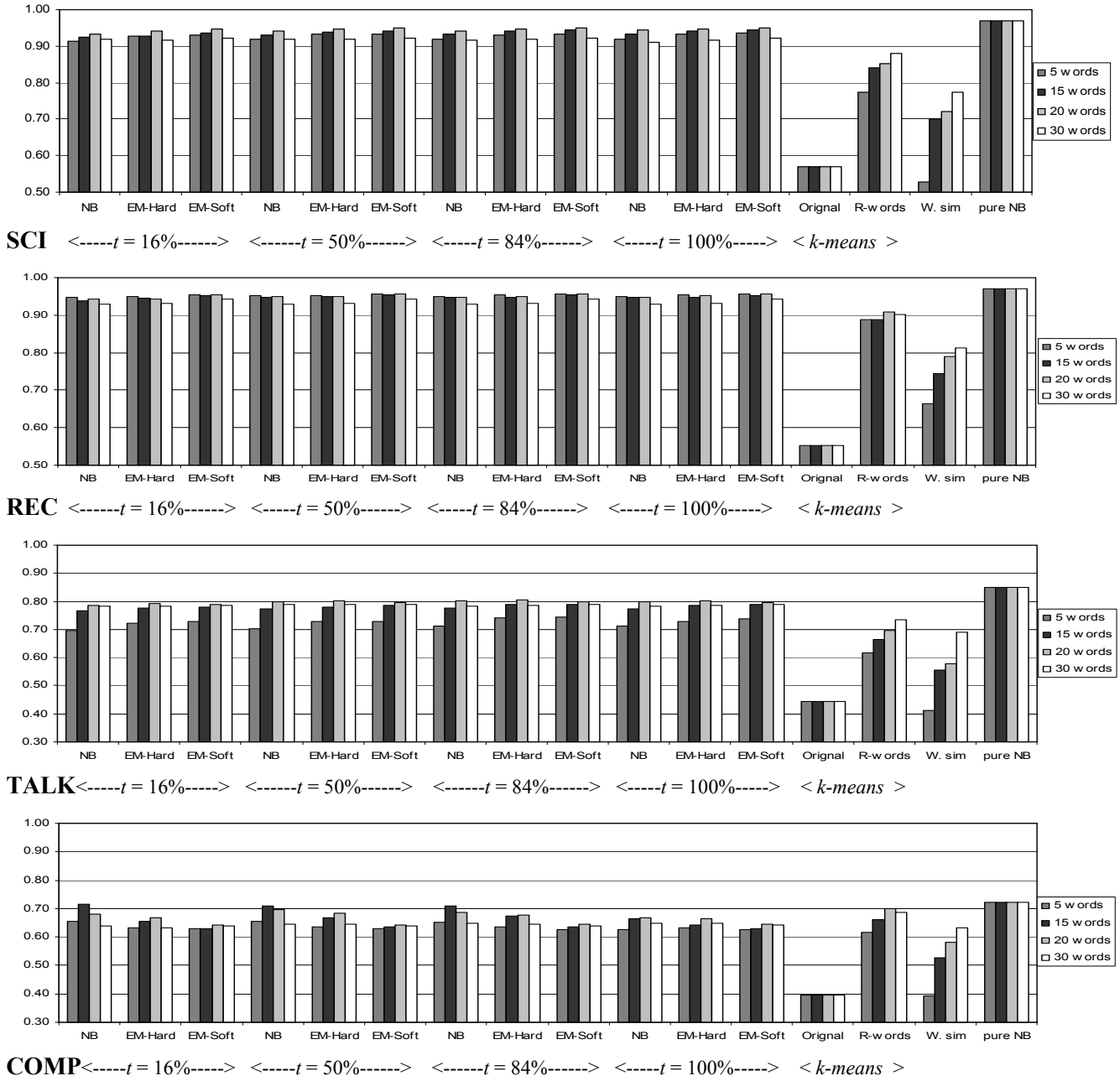


Figure 3: The accuracy results

*NB, EM-hard and EM-Soft*: We observe that these three methods all have similar results for all $t$ settings. Using 15 or 20 words for each class gives slightly better results in general with $t$ = 50%, 84% and 100%. For all $t$ values, the accuracy stabilizes or worsens after 20 words. This is because the additional words may not be as representative as the earlier words for each class.

EM based methods slightly outperform NB for the first three datasets. For dataset COMP, EM methods are slightly worse because the representative words for COMP are not as reliable as for other datasets. The representative words of COMP were the hardest to select because some classes of the dataset are similar. Thus, more EM iterations further blur the decision boundaries.

*k-means*: We also give two results for *k*-means. "*original*" standards for the original *k*-means with its initial *k* seeds selected randomly. To allow fair comparison, we use the correct number of clusters *k* for each dataset. The accuracy on the test set is obtained as follows. After *k*-means ends, for each test document *d*, we compute the similarity of *d* with each cluster center. *d* is assigned to the cluster that gives the highest similarity (using the cosine measure). After all test documents are assigned to

clusters, we match the classes of the dataset with the clusters in such a way that gives the best accuracy. Due to the random nature of *k*-means the accuracies are the averaged values of 10 runs. Note that each row has the same value because representative words are not used.

"*R-words*" gives the results of *k*-means using the set of representative words as the initial cluster centers. From these two sets of results of *k*-means, we see that R-words outperforms the original *k*-means by a large margin.

Comparing with the results of NB and EM, we see that NB and EM methods are superior to clustering. This is especially true for smaller numbers of representative words. For dataset COMP, the results are similar. We believe that the selected representative words for this dataset are not as reliable as for the other datasets.

*W. sim*: It gives the accuracy results of using only the cosine similarity comparison with the representative words of each class. We compare the similarity of each test document *d* with the set of representative words of each class. *d* is assigned the class that gives the highest similarity value. If *d*'s similarity with every set of representative words is 0, *d* is assigned a random class. We observe that this technique is significantly worse than *k*-means using representative words as seeds, but better than the original *k*-means. It is also dramatically worse than the NB and EM-based approaches.

*Pure NB*: These are NB's accuracies in traditional classification, i.e., all the training documents are labeled with their original classes. Since we used fully labeled training data, thus all the results for each dataset are the same as they are not affected by the number of representative words. We observe that the accuracies of our proposed methods are very close to pure NB's accuracies. For datasets SCI, REC and COMP, the proposed methods are only about 2-3% worse and for dataset TALK, it is about 5% worse. This shows that our methods are highly effective considering that no manually labeled document is used.

**Time spent to select representative words**: The process of selecting representative words is not linear in the number of words. It gets harder with increasing number of words. The first 10 representative words for each class are generally easy to select. It takes around 5-10 minutes to choose 30 representative words for each class of a dataset. The time needed also depends on how different the classes are in the dataset. For example, for the first three datasets, it was very easy, and for the last dataset it was harder.

## Conclusions

This paper proposed a new approach to constructing text classifiers. Instead of labeling a large set of training documents as in traditional learning, the new method asks the user to label or select a set of representative words for each class from a list of ranked words. The ranking technique is based on clustering and feature selection. The set of words for each class and a set of unlabeled documents are used to build a classifier. We believe that selecting these words from a ranked list requires less user effort than labeling of a large set of documents. Our results demonstrated the effectiveness of this approach.

**Acknowledgments:** The work of Bing Liu is partially supported by the National Science Foundation under the NSF grant IIS-0307239.

## References

Basu, S., Banerjee, A., and Mooney, R. (2002). Semi-supervised clustering by seeding. *ICML-02*.

Blum, A., and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT-98.*

Bockhorst, J., and Craven, M. (2002). Exploiting relations among concepts to acquire weakly labeled training data. *ICML-02*.

Dempster, A., Laird, N. M. and Rubin. D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Stat. Society*, B:39, 1-38.

Denis, F. (1998). PAC learning from positive statistical queries. *ALT-1998*.

Ghani, R. (2002). Combining labeled and unlabeled data for multiclass text categorization. *ICML-02*.

Goldman, S. and Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *ICML-00*.

Jain, A. K., and Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs, NJ: Prentice Hall.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *ECML-98*.

Lewis, D., and Gale, W. (1994). A sequential algorithm for training text classifiers. *SIGIR-94*.

Liu, B., Lee, W. S., Yu, P., and Li, X. (2002). Partially supervised classification of text documents. *ICML-02*.

McCallum, A., and Nigam, K. (1998a). A comparison of event models for naïve Bayes text classification. *AAAI-98 Workshop on Learning for Text Categorization*.

McCallum, A., and Nigam. K. (1998b). Employing EM and pool-based active learning for text classification. *ICML-98*.

McCallum, A., and Nigam. K. (1999) "Text classification by bootstrapping with keywords, EM and shrinkage." *ACL Workshop on Unsupervised Learning in Natural Language Processing*.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Vapnik, V. *The nature of statistical learning theory*, 1995.

Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. (2001). Constrained *k*-means clustering with background knowledge. *ICML-01*.

Yang, Y. and Pedersen J. P. (1997). A comparative study on feature selection in text categorization. *ICML-97*.

Yu, H., Han, J., Chang, K. (2002). PEBL: Positive example based learning for Web page classification using SVM. *KDD-02*.