Document Classification Using a Finite Mixture Model

Hang Li **Kenji Yamanishi**
C&C Res. Labs., NEC
4-1-1 Miyazaki Miyamae-ku Kawasaki, 216, Japan
Email: {lihang,yamanisi}@sbl.cl.nec.co.jp

Abstract

We propose a new method of classifying documents into categories. We define for each category a *finite mixture model* based on *soft clustering* of words. We treat the problem of classifying documents as that of conducting statistical hypothesis testing over finite mixture models, and employ the EM algorithm to efficiently estimate parameters in a finite mixture model. Experimental results indicate that our method outperforms existing methods.

1 Introduction

We are concerned here with the issue of classifying documents into categories. More precisely, we begin with a number of categories (e.g., 'tennis, soccer, skiing'), each already containing certain documents. Our goal is to determine into which categories newly given documents ought to be assigned, and to do so on the basis of the distribution of each document's words.[1]

Many methods have been proposed to address this issue, and a number of them have proved to be quite effective (e.g.,(Apte, Damerau, and Weiss, 1994; Cohen and Singer, 1996; Lewis, 1992; Lewis and Ringuette, 1994; Lewis et al., 1996; Schutze, Hull, and Pedersen, 1995; Yang and Chute, 1994)). The simple method of conducting hypothesis testing over word-based distributions in categories (defined in Section 2) is not efficient in storage and suffers from the *data sparseness problem*, i.e., the number of parameters in the distributions is large and the data size is not sufficiently large for accurately estimating them. In order to address this difficulty, (Guthrie, Walker, and Guthrie, 1994) have proposed using distributions based on what we refer to as *hard clustering* of words, i.e., in which a word is assigned to a single cluster and words in the same cluster are treated uniformly. The use of hard clustering might, however, degrade classification results, since the distributions it employs are not always precise enough for representing the differences between categories.

We propose here to employ *soft clustering*[2], i.e., a word can be assigned to several different clusters and each cluster is characterized by a specific word probability distribution. We define for each category a *finite mixture model*, which is a linear combination of the word probability distributions of the clusters. We thereby treat the problem of classifying documents as that of conducting statistical hypothesis testing over finite mixture models. In order to accomplish hypothesis testing, we employ the EM algorithm to efficiently and approximately calculate from training data the maximum likelihood estimates of parameters in a finite mixture model.

Our method overcomes the major drawbacks of the method using word-based distributions and the method based on hard clustering, while retaining their merits; it in fact includes those two methods as special cases. Experimental results indicate that our method outperforms them.

Although the finite mixture model has already been used elsewhere in natural language processing (e.g. (Jelinek and Mercer, 1980; Pereira, Tishby, and Lee, 1993)), this is the first work, to the best of knowledge, that uses it in the context of document classification.

2 Previous Work

Word-based method

A simple approach to document classification is to view this problem as that of conducting hypothesis testing over word-based distributions. In this paper, we refer to this approach as the *word-based method* (hereafter, referred to as WBM).

[1] A related issue is the retrieval, from a data base, of documents which are relevant to a given query (pseudo-document) (e.g.,(Deerwester et al., 1990; Fuhr, 1989; Robertson and Jones, 1976; Salton and McGill, 1983; Wong and Yao, 1989)).

[2] We borrow from (Pereira, Tishby, and Lee, 1993) the terms hard clustering and soft clustering, which were used there in a different task.

Letting W denote a vocabulary (a set of words), and w denote a random variable representing any word in it, for each category c_i ($i = 1, \cdots, n$), we define its *word-based distribution* $P(w|c_i)$ as a histogram type of distribution over W. (The number of free parameters of such a distribution is thus $|W|-1$). WBM then views a document as a sequence of words:

$$d = w_1, \cdots, w_N \quad (1)$$

and assumes that each word is generated independently according to a probability distribution of a category. It then calculates the probability of a document with respect to a category as

$$P(d|c_i) = P(w_1, \cdots, w_N|c_i) = \prod_{t=1}^{N} P(w_t|c_i), \quad (2)$$

and classifies the document into that category for which the calculated probability is the largest. We should note here that a document's probability with respect to each category is equivalent to the *likelihood* of each category with respect to the document, and to classify the document into the category for which it has the largest probability is equivalent to classifying it into the category having the largest likelihood with respect to it. Hereafter, we will use only the term likelihood and denote it as $L(d|c_i)$.

Notice that in practice the parameters in a distribution must be estimated from training data. In the case of WBM, the number of parameters is large; the training data size, however, is usually not sufficiently large for accurately estimating them. This is the *data sparseness problem* that so often stands in the way of reliable statistical language processing (e.g.(Gale and Church, 1990)). Moreover, the number of parameters in word-based distributions is too large to be efficiently stored.

Method based on hard clustering

In order to address the above difficulty, Guthrie et.al. have proposed a method based on hard clustering of words (Guthrie, Walker, and Guthrie, 1994) (hereafter we will refer to this method as HCM). Let $c_1, \cdots, c_n$ be categories. HCM first conducts hard clustering of words. Specifically, it (a) defines a vocabulary as a set of words W and defines as clusters its subsets $k_1, \cdots, k_m$ satisfying $\cup_{j=1}^{m} k_j = W$ and $k_i \cap k_j = \emptyset$ ($i \neq j$) (i.e., each word is assigned only to a single cluster); and (b) treats uniformly all the words assigned to the same cluster. HCM then defines for each category c_i a distribution of the clusters $P(k_j|c_i)$ ($j = 1, \cdots, m$). It replaces each word w_t in the document with the cluster k_t to which it belongs ($t = 1, \cdots, N$). It assumes that a cluster k_t is distributed according to $P(k_j|c_i)$ and calculates the likelihood of each category c_i with respect to the document by

$$L(d|c_i) = L(k_1, \cdots, k_N|c_i) = \prod_{t=1}^{N} P(k_t|c_i). \quad (3)$$

Table 1: Frequencies of words

	racket	stroke	shot	goal	kick	ball
c_1	4	1	2	1	0	2
c_2	0	0	0	3	2	2

Table 2: Clusters and words ($L = 5$,$M = 5$)

k_1	racket, stroke, shot
k_2	kick
k_3	goal, ball

Table 3: Frequencies of clusters

	k_1	k_2	k_3
c_1	7	0	3
c_2	0	2	5

There are any number of ways to create clusters in hard clustering, but the method employed is crucial to the accuracy of document classification. Guthrie et. al. have devised a way suitable to documentation classification. Suppose that there are two categories c_1='tennis' and c_2='soccer,' and we obtain from the training data (previously classified documents) the frequencies of words in each category, such as those in Tab. 1. Letting L and M be given positive integers, HCM creates three clusters: k_1, k_2 and k_3, in which k_1 contains those words which are among the L most frequent words in c_1, and not among the M most frequent in c_2; k_2 contains those words which are among the L most frequent words in c_2, and not among the M most frequent in c_1; and k_3 contains all remaining words (see Tab. 2). HCM then counts the frequencies of clusters in each category (see Tab. 3) and estimates the probabilities of clusters being in each category (see Tab. 4).[3] Suppose that a newly given document, like d in Fig. 1, is to be classified. HCM calculates the likelihood values

[3] We calculate the probabilities here by using the so-called expected likelihood estimator (Gale and Church, 1990):

$$P(k_j|c_i) = \frac{f(k_j|c_i) + 0.5}{f(c_i) + 0.5 \times m}, \quad (4)$$

where $f(k_j|c_i)$ is the frequency of the cluster k_j in c_i, $f(c_i)$ is the total frequency of clusters in c_i, and m is the total number of clusters.

Table 4: Probability distributions of clusters

	k_1	k_2	k_3
c_1	0.65	0.04	0.30
c_2	0.06	0.29	0.65

$L(d|c_1)$ and $L(d|c_2)$ according to Eq. (3). (Tab. 5 shows the logarithms of the resulting likelihood values.) It then classifies d into c_2, as $\log_2 L(d|c_2)$ is larger than $\log_2 L(d|c_1)$.

d = kick, goal, goal, ball

Figure 1: Example document

Table 5: Calculating log likelihood values

$\log_2 L(d\|c_1)$ $= 1 \times \log_2 .04 + 3 \times \log_2 .30 = -9.85$
$\log_2 L(d\|c_2)$ $= 1 \times \log_2 .29 + 3 \times \log_2 .65 = -3.65$

HCM can handle the data sparseness problem quite well. By assigning words to clusters, it can drastically reduce the number of parameters to be estimated. It can also save space for storing knowledge. We argue, however, that the use of hard clustering still has the following two problems:

1. *HCM cannot assign a word to more than one cluster at a time.* Suppose that there is another category c_3 = 'skiing' in which the word 'ball' does not appear, i.e., 'ball' will be indicative of both c_1 and c_2, but not c_3. If we could assign 'ball' to both k_1 and k_2, the likelihood value for classifying a document containing that word to c_1 or c_2 would become larger, and that for classifying it into c_3 would become smaller. HCM, however, cannot do that.
2. *HCM cannot make the best use of information about the differences among the frequencies of words assigned to an individual cluster.* For example, it treats 'racket' and 'shot' uniformly because they are assigned to the same cluster k_1 (see Tab. 5). 'Racket' may, however, be more indicative of c_1 than 'shot,' because it appears more frequently in c_1 than 'shot.' HCM fails to utilize this information. This problem will become more serious when the values L and M in word clustering are large, which renders the clustering itself relatively meaningless.

From the perspective of number of parameters, HCM employs models having very few parameters, and thus may not sometimes represent much useful information for classification.

3 Finite Mixture Model

We propose a method of document classification based on soft clustering of words. Let $c_1, \cdots, c_n$ be categories. We first conduct the soft clustering. Specifically, we (a) define a vocabulary as a set W of words and define as clusters a number of its subsets $k_1, \cdots, k_m$ satisfying $\cup_{j=1}^{m} k_j = W$; (notice that $k_i \cap k_j = \emptyset$ $(i \neq j)$ does not necessarily hold here, i.e., a word can be assigned to several different clusters); and (b) define for each cluster k_j $(j = 1, \cdots, m)$ a distribution $Q(w|k_j)$ over its words $(\sum_{w \in k_j} Q(w|k_j) = 1)$ and a distribution $P(w|k_j)$ satisfying:

$$P(w|k_j) = \begin{cases} Q(w|k_j); & w \in k_j, \\ 0; & w \notin k_j, \end{cases} \tag{5}$$

where w denotes a random variable representing any word in the vocabulary. We then define for each category c_i $(i = 1, \cdots, n)$ a distribution of the clusters $P(k_j|c_i)$, and define for each category a linear combination of $P(w|k_j)$:

$$P(w|c_i) = \sum_{j=1}^{m} P(k_j|c_i) \times P(w|k_j) \tag{6}$$

as the distribution over its words, which is referred to as a *finite mixture model* (e.g., (Everitt and Hand, 1981)).

We treat the problem of classifying a document as that of conducting the likelihood ratio test over finite mixture models. That is, we view a document as a sequence of words,

$$d = w_1, \cdots, w_N \tag{7}$$

where $w_t (t = 1, \cdots, N)$ represents a word. We assume that each word is independently generated according to an unknown probability distribution and determine which of the finite mixture models $P(w|c_i)(i = 1, \cdots, n)$ is more likely to be the probability distribution by observing the sequence of words. Specifically, we calculate the likelihood value for each category with respect to the document by:

$$\begin{aligned} L(d|c_i) &= L(w_1, \cdots, w_N|c_i) \\ &= \prod_{t=1}^{N} P(w_t|c_i) \\ &= \prod_{t=1}^{N} \left(\sum_{j=1}^{m} P(k_j|c_i) \times P(w_t|k_j) \right). \end{aligned} \tag{8}$$

We then classify it into the category having the largest likelihood value with respect to it. Hereafter, we will refer to this method as FMM.

FMM includes WBM and HCM as its special cases. If we consider the specific case (1) in which a word is assigned to a single cluster and $P(w|k_j)$ is given by

$$P(w|k_j) = \begin{cases} \frac{1}{|k_j|}; & w \in k_j, \\ 0; & w \notin k_j, \end{cases} \tag{9}$$

where $|k_j|$ denotes the number of elements belonging to k_j, then we will get the same classification result as in HCM. In such a case, the likelihood value for each category c_i becomes:

$$\begin{aligned} L(d|c_i) &= \prod_{t=1}^{N}(P(k_t|c_i) \times P(w_t|k_t)) \\ &= \prod_{t=1}^{N} P(k_t|c_i) \times \prod_{t=1}^{N} P(w_t|k_t), \end{aligned} \tag{10}$$

where k_t is the cluster corresponding to w_t. Since the probability $P(w_t|k_t)$ does not depend on categories, we can ignore the second term $\prod_{t=1}^{N} P(w_t|k_t)$ in hypothesis testing, and thus our method essentially becomes equivalent to HCM (c.f. Eq. (3)).

Further, in the specific case (2) in which $m = n$, for each j, $P(w|k_j)$ has $|W|$ parameters: $P(w|k_j) = P(w|c_j)$, and $P(k_j|c_i)$ is given by

$$P(k_j|c_i) = \begin{cases} 1; & i = j, \\ 0; & i \neq j, \end{cases} \tag{11}$$

the likelihood used in hypothesis testing becomes the same as that in Eq.(2), and thus our method becomes equivalent to WBM.

4 Estimation and Hypothesis Testing

In this section, we describe how to implement our method.

Creating clusters

There are any number of ways to create clusters on a given set of words. As in the case of hard clustering, the way that clusters are created is crucial to the reliability of document classification. Here we give one example approach to cluster creation.

Table 6: Clusters and words

k_1	racket, stroke, shot, ball
k_2	kick, goal, ball

We let the number of clusters equal that of categories (i.e., $m = n$) [4] and relate each cluster k_i to one category c_i ($i = 1, \cdots, n$). We then assign individual words to those clusters in whose related categories they most frequently appear. Letting γ ($0 \leq \gamma < 1$) be a predetermined threshold value, if the following inequality holds:

$$\frac{f(w|c_i)}{f(w)} > \gamma, \tag{12}$$

then we assign w to k_i, the cluster related to c_i, where $f(w|c_i)$ denotes the frequency of the word w in category c_i, and $f(w)$ denotes the total frequency of w. Using the data in Tab.1, we create two clusters: k_1 and k_2, and relate them to c_1 and c_2, respectively.

[4] One can certainly assume that $m \geq n$.

For example, when $\gamma = 0.4$, we assign 'goal' to k_2 only, as the relative frequency of 'goal' in c_2 is 0.75 and that in c_1 is only 0.25. We ignore in document classification those words which cannot be assigned to any cluster using this method, because they are not indicative of any specific category. (For example, when $\gamma \geq 0.5$ 'ball' will not be assigned into any cluster.) This helps to make classification efficient and accurate. Tab. 6 shows the results of creating clusters.

Estimating $P(w|k_j)$

We then consider the frequency of a word in a cluster. If a word is assigned only to one cluster, we view its total frequency as its frequency within that cluster. For example, because 'goal' is assigned only to k_2, we use as its frequency within that cluster the total count of its occurrence in all categories. If a word is assigned to several different clusters, we distribute its total frequency among those clusters in proportion to the frequency with which the word appears in each of their respective related categories. For example, because 'ball' is assigned to both k_1 and k_2, we distribute its total frequency among the two clusters in proportion to the frequency with which 'ball' appears in c_1 and c_2, respectively. After that, we obtain the frequencies of words in each cluster as shown in Tab. 7.

Table 7: Distributed frequencies of words

	racket	stroke	shot	goal	kick	ball
k_1	4	1	2	0	0	2
k_2	0	0	0	4	2	2

We then estimate the probabilities of words in each cluster, obtaining the results in Tab. 8.[5]

Table 8: Probability distributions of words

	racket	stroke	shot	goal	kick	ball
k_1	0.44	0.11	0.22	0	0	0.22
k_2	0	0	0	0.50	0.25	0.25

Estimating $P(k_j|c_i)$

Let us next consider the estimation of $P(k_j|c_i)$. There are two common methods for statistical estimation, the maximum likelihood estimation method

[5] We calculate the probabilities by employing the maximum likelihood estimator:

$$P(k_j|c_i) = \frac{f(k_j|c_i)}{f(c_i)}, \tag{13}$$

where $f(k_j|c_i)$ is the frequency of the cluster k_j in c_i, and $f(c_i)$ is the total frequency of clusters in c_i.

Table 10: Calculating log likelihood values

$\log_2 L(d\|c_1) = \log_2(.14 \times .25) + 2 \times \log_2(.14 \times .50) + \log_2(.86 \times .22 + .14 \times .25) = -14.67$
$\log_2 L(d\|c_2) = \log_2(.96 \times .25) + 2 \times \log_2(.96 \times .50) + \log_2(.04 \times .22 + .96 \times .25) = -6.18$

Table 9: Probability distributions of clusters

	k_1	k_2
c_1	0.86	0.14
c_2	0.04	0.96

and the Bayes estimation method. In their implementation for estimating $P(k_j|c_i)$, however, both of them suffer from computational intractability. The EM algorithm (Dempster, Laird, and Rubin, 1977) can be used to efficiently approximate the maximum likelihood estimator of $P(k_j|c_i)$. We employ here an extended version of the EM algorithm (Helmbold et al., 1995). (We have also devised, on the basis of the Markov chain Monte Carlo (MCMC) technique (e.g. (Tanner and Wong, 1987; Yamanishi, 1996))[6], an algorithm to efficiently approximate the Bayes estimator of $P(k_j|c_i)$.)

For the sake of notational simplicity, for a fixed i, let us write $P(k_j|c_i)$ as θ_j and $P(w|k_j)$ as $P_j(w)$. Then letting $\theta = (\theta_1, \cdots, \theta_m)$, the finite mixture model in Eq. (6) may be written as

$$P(w|\theta) = \sum_{j=1}^{m} \theta_j \times P_j(w). \tag{14}$$

For a given training sequence $w_1 \cdots w_N$, the maximum likelihood estimator of θ is defined as the value $\hat{\theta}$ which maximizes the following log likelihood function

$$L(\theta) = \frac{1}{N} \sum_{t=1}^{N} \log \left(\sum_{j=1}^{m} \theta_j P_j(w_t) \right). \tag{15}$$

The EM algorithm first arbitrarily sets the initial value of θ, which we denote as $\theta^{(0)}$, and then successively calculates the values of θ on the basis of its most recent values. Let s be a predetermined number. At the lth iteration ($l = 1, \cdots, s$), we calculate $\theta^{(l)} = (\theta_1^{(l)}, \cdots, \theta_m^{(l)})$ by

$$\theta_j^{(l)} = \theta_j^{(l-1)} \left(\eta(\nabla L(\theta^{(l-1)})_j - 1) + 1 \right), \tag{16}$$

where $\eta > 0$ (when $\eta = 1$, Hembold et al. 's version simply becomes the standard EM algorithm), and $\nabla L(\theta)$ denotes

$$\nabla L(\theta) = \left(\frac{\partial L}{\partial \theta_1} \cdots \frac{\partial L}{\partial \theta_m} \right). \tag{17}$$

After s numbers of calculations, the EM algorithm outputs $\theta^{(s)} = (\theta_1^{(s)}, \cdots, \theta_m^{(s)})$ as an approximate of $\hat{\theta}$. It is theoretically guaranteed that the EM algorithm converges to a local minimum of the given likelihood (Dempster, Laird, and Rubin, 1977).

For the example in Tab. 1, we obtain the results as shown in Tab. 9.

Testing

For the example in Tab. 1, we can calculate according to Eq. (8) the likelihood values of the two categories with respect to the document in Fig. 1 (Tab. 10 shows the logarithms of the likelihood values). We then classify the document into category c_2, as $\log_2 L(d|c_2)$ is larger than $\log_2 L(d|c_1)$.

5 Advantages of FMM

For a probabilistic approach to document classification, the most important thing is to determine what kind of probability model (distribution) to employ as a representation of a category. It must (1) appropriately represent a category, as well as (2) have a proper preciseness in terms of number of parameters. The goodness and badness of selection of a model directly affects classification results.

The finite mixture model we propose is particularly well-suited to the representation of a category. Described in linguistic terms, a cluster corresponds to a *topic* and the words assigned to it are related to that topic. Though documents generally concentrate on a single topic, they may sometimes refer for a time to others, and while a document is discussing any one topic, it will naturally tend to use words strongly related to that topic. A document in the category of 'tennis' is more likely to discuss the topic of 'tennis,' i.e., to use words strongly related to 'tennis,' but it may sometimes briefly shift to the topic of 'soccer,' i.e., use words strongly related to 'soccer.' A human can follow the sequence of words in such a document, associate them with related topics, and use the distributions of topics to classify the document. Thus the use of the finite mixture model can be considered as a stochastic implementation of this process.

The use of FMM is also appropriate from the viewpoint of number of parameters. Tab. 11 shows the numbers of parameters in our method (FMM),

[6] We have confirmed in our preliminary experiment that MCMC performs slightly better than EM in document classification, but we omit the details here due to space limitations.

Table 11: Num. of parameters

WBM	$O(n \cdot \|W\|)$
HCM	$O(n \cdot m)$
FMM	$O(\|k\| + n \cdot m)$

HCM, and WBM, where $|W|$ is the size of a vocabulary, $|k|$ is the sum of the sizes of word clusters (i.e.,$|k| = \sum_{j=1}^{m} |k_j|$), n is the number of categories, and m is the number of clusters. The number of parameters in FMM is much smaller than that in WBM, which depends on $|W|$, a very large number in practice (notice that $|k|$ is always smaller than $|W|$ when we employ the clustering method (with $\gamma \geq 0.5$) described in Section 4. As a result, FMM requires less data for parameter estimation than WBM and thus can handle the data sparseness problem quite well. Furthermore, it can economize on the space necessary for storing knowledge. On the other hand, the number of parameters in FMM is larger than that in HCM. It is able to represent the differences between categories more precisely than HCM, and thus is able to resolve the two problems, described in Section 2, which plague HCM.

Another advantage of our method may be seen in contrast to the use of *latent semantic analysis* (Deerwester et al., 1990) in document classification and document retrieval. They claim that their method can solve the following problems:

synonymy problem how to group synonyms, like 'stroke' and 'shot,' and make each relatively strongly indicative of a category even though some may individually appear in the category only very rarely;

polysemy problem how to determine that a word like 'ball' in a document refers to a 'tennis ball' and not a 'soccer ball,' so as to classify the document more accurately;

dependence problem how to use dependent words, like 'kick' and 'goal,' to make their combined appearance in a document more indicative of a category.

As seen in Tab.6, our method also helps resolve all of these problems.

6 Preliminary Experimental Results

In this section, we describe the results of the experiments we have conducted to compare the performance of our method with that of HCM and others.

As a first data set, we used a subset of the Reuters newswire data prepared by Lewis, called Reuters-21578 Distribution 1.0.[7] We selected nine overlapping categories, i.e. in which a document may belong to several different categories. We adopted the Lewis Split in the corpus to obtain the training data and the test data. Tabs. 12 and 13 give the details. We did not conduct stemming, or use stop words[8]. We then applied FMM, HCM, WBM , and a method based on cosine-similarity, which we denote as COS[9], to conduct *binary* classification. In particular, we learn the distribution for each category and that for its complement category from the training data, and then determine whether or not to classify into each category the documents in the test data. When applying FMM, we used our proposed method of creating clusters in Section 4 and set γ to be $0, 0.4, 0.5, 0.7$, because these are representative values. For HCM, we classified words in the same way as in FMM and set γ to be $0.5, 0.7, 0.9, 0.95$. (Notice that in HCM, γ cannot be set less than 0.5.)

Table 12: The first data set

Num. of doc. in training data	707
Num. of doc in test data	228
Num. of (type of) words	10902
Avg. num. of words per doc.	310.6

Table 13: Categories in the first data set

wheat,corn,oilseed,sugar,coffee
soybean,cocoa,rice,cotton

Table 14: The second data set

Num. of doc. training data	13625
Num. of doc. in test data	6188
Num. of (type of) words	50301
Avg. num. of words per doc.	181.3

As a second data set, we used the entire Reuters-21578 data with the Lewis Split. Tab. 14 gives the details. Again, we did not conduct stemming, or use stop words. We then applied FMM, HCM, WBM , and COS to conduct *binary* classification. When applying FMM, we used our proposed method of creating clusters and set γ to be $0, 0.4, 0.5, 0.7$. For HCM, we classified words in the same way as in FMM and set γ to be $0.5, 0.7, 0.9, 0.95$. We have not fully completed these experiments, however, and here we only

[7] Reuters-21578 is available at http://www.research.att.com/lewis.

[8] 'Stop words' refers to a predetermined list of words containing those which are considered not useful for document classification, such as articles and prepositions.

[9] In this method, categories and documents to be classified are viewed as vectors of word frequencies, and the cosine value between the two vectors reflects similarity (Salton and McGill, 1983).

Table 15: Tested categories in the second data set

earn,acq,crude,money-fx,grain
interest,trade,ship,wheat,corn

give the results of classifying into the ten categories having the greatest numbers of documents in the test data (see Tab. 15).

For both data sets, we evaluated each method in terms of *precision* and *recall* by means of the so-called micro-averaging [10].

When applying WBM, HCM, and FMM, rather than use the standard likelihood ratio testing, we used the following heuristics. For simplicity, suppose that there are only two categories c_1 and c_2. Letting ϵ be a given number larger than or equal 0, we assign a new document d in the following way:

$$\begin{array}{ll} \frac{1}{N}(\log L(d|c_1) - \log L(d|c_2)) > \epsilon; & d \rightarrow c_1, \\ \frac{1}{N}(\log L(d|c_2) - \log L(d|c_1)) \geq \epsilon; & d \rightarrow c_2, \\ \text{otherwise;} & \text{unclassify } d, \end{array} \tag{18}$$

where N is the size of document d. (One can easily extend the method to cases with a greater number of categories.) [11] For COS, we conducted classification in a similar way.

Figs. 2 and 3 show precision-recall curves for the first data set and those for the second data set, respectively. In these graphs, values given after FMM and HCM represent γ in our clustering method (e.g. FMM0.5, HCM0.5,etc). We adopted the break-even point as a single measure for comparison, which is the one at which precision equals recall; a higher score for the break-even point indicates better performance. Tab. 16 shows the break-even point for each method for the first data set and Tab. 17 shows that for the second data set. For the first data set, FMM0 attains the highest score at break-even point; for the second data set, FMM0.5 attains the highest.

We considered the following questions:

(1) The training data used in the experimentation may be considered sparse. Will a word-clustering-based method (FMM) outperform a word-based method (WBM) here?

(2) Is it better to conduct soft clustering (FMM) than to do hard clustering (HCM)?

(3) With our current method of creating clusters, as the threshold γ approaches 0, FMM behaves much like WBM and it does not enjoy the effects of clustering at all (the number of parameters is as large

[10] In micro-averaging(Lewis and Ringuette, 1994), precision is defined as the percentage of classified documents in all categories which are correctly classified. Recall is defined as the percentage of the total documents in all categories which are correctly classified.

[11] Notice that words which are discarded in the clustering process should not to be counted in document size.

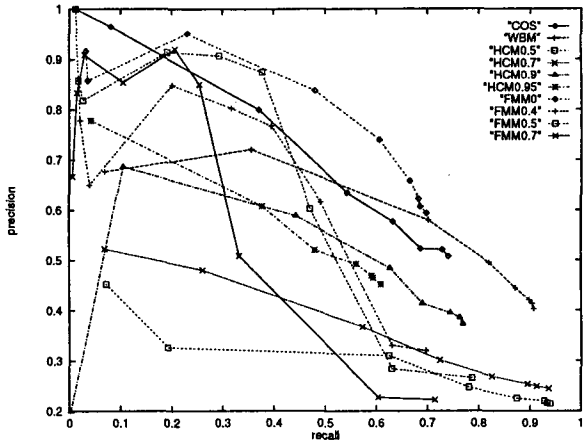

Figure 2: Precision-recall curve for the first data set

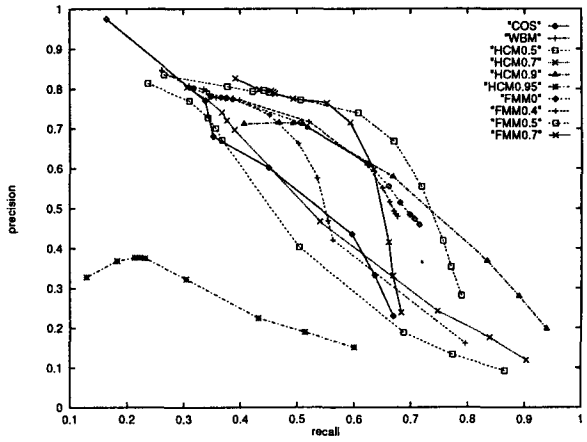

Figure 3: Precision-recall curve for the second data set

as in WBM). This is because in this case (a) a word will be assigned into all of the clusters, (b) the distribution of words in each cluster will approach that in the corresponding category in WBM, and (c) the likelihood value for each category will approach that in WBM (recall case (2) in Section 3). Since creating clusters in an optimal way is difficult, when clustering does not improve performance we can at least make FMM perform as well as WBM by choosing $\gamma = 0$. The question now is "does FMM perform better than WBM when γ is 0?"

In looking into these issues, we found the following:

(1) When $\gamma \gg 0$, i.e., when we conduct clustering, FMM does not perform better than WBM for the first data set, but it performs better than WBM for the second data set.

Evaluating classification results on the basis of each individual category, we have found that for three of the nine categories in the first data set,

Table 16: Break-even point for the first data set

COS	0.60
WBM	0.62
HCM0.5	0.32
HCM0.7	0.42
HCM0.9	0.54
HCM0.95	0.51
FMM0	0.66
FMM0.4	0.54
FMM0.5	0.52
FMM0.7	0.42

Table 17: Break-even point for the second data set

COS	0.52
WBM	0.62
HCM0.5	0.47
HCM0.7	0.51
HCM0.9	0.55
HCM0.95	0.31
FMM0	0.62
FMM0.4	0.54
FMM0.5	0.67
FMM0.7	0.62

FMM0.5 performs best, and that in two of the ten categories in the second data set FMM0.5 performs best. These results indicate that clustering sometimes does improve classification results *when we use our current way of creating clusters.* (Fig. 4 shows the best result for each method for the category 'corn' in the first data set and Fig. 5 that for 'grain' in the second data set.)

(2) When $\gamma \gg 0$, i.e., when we conduct clustering, the best of FMM almost always outperforms that of HCM.

(3) When $\gamma = 0$, FMM performs better than WBM for the first data set, and that it performs as well as WBM for the second data set.

In summary, FMM always outperforms HCM; in some cases it performs better than WBM; and in general it performs at least as well as WBM.

For both data sets, the best FMM results are superior to those of COS throughout. This indicates that the probabilistic approach is more suitable than the cosine approach for document classification based on word distributions.

Although we have not completed our experiments on the entire Reuters data set, we found that the results with FMM on the second data set are almost as good as those obtained by the other approaches reported in (Lewis and Ringuette, 1994). (The results are not directly comparable, because (a) the results in (Lewis and Ringuette, 1994) were obtained from an older version of the Reuters data; and (b) they used stop words, but we did not.)

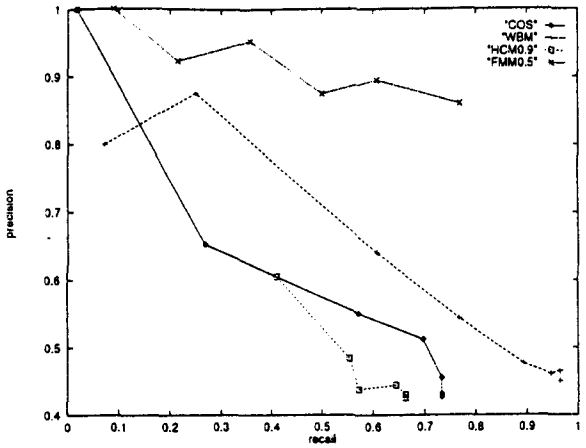

Figure 4: Precision-recall curve for category 'corn'

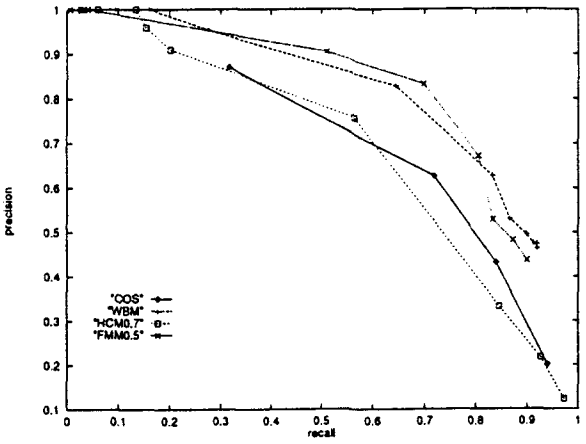

Figure 5: Precision-recall curve for category 'grain'

We have also conducted experiments on the Susanne corpus data[12] and confirmed the effectiveness of our method. We omit an explanation of this work here due to space limitations.

7 Conclusions

Let us conclude this paper with the following remarks:

1. The primary contribution of this research is that we have proposed the use of the finite mixture model in document classification.
2. Experimental results indicate that our method of using the finite mixture model outperforms the method based on hard clustering of words.
3. Experimental results also indicate that in some cases our method outperforms the word-based

[12]The Susanne corpus, which has four non-overlapping categories, is available at ftp://ota.ox.ac.uk

method *when we use our current method of creating clusters.*

Our future work is to include:

1. comparing the various methods over the entire Reuters corpus and over other data bases,
2. developing better ways of creating clusters.

Our proposed method is not limited to document classification; it can also be applied to other natural language processing tasks, like word sense disambiguation, in which we can view the context surrounding a ambiguous target word as a document and the word-senses to be resolved as categories.

Acknowledgements

We are grateful to Tomoyuki Fujita of NEC for his constant encouragement. We also thank Naoki Abe of NEC for his important suggestions, and Mark Petersen of Meiji Univ. for his help with the English of this text. We would like to express special appreciation to the six ACL anonymous reviewers who have provided many valuable comments and criticisms.

References

Apte, Chidanand, Fred Damerau, and Sholom M. Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Tran. on Information Systems*, 12(3):233–251.

Cohen, William W. and Yoram Singer. 1996. Context-sensitive learning methods for text categorization. *Proc. of SIGIR'96.*

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journ. of the American Society for Information Science*, 41(6):391–407.

Dempster, A.P., N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journ. of the Royal Statistical Society, Series B*, 39(1):1–38.

Everitt, B. and D. Hand. 1981. *Finite Mixture Distributions.* London: Chapman and Hall.

Fuhr, Norbert. 1989. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72.

Gale, Williams A. and Kenth W. Church. 1990. Poor estimates of context are worse than none. *Proc. of the DARPA Speech and Natural Language Workshop*, pages 283–287.

Guthrie, Louise, Elbert Walker, and Joe Guthrie. 1994. Document classification by machine: Theory and practice. *Proc. of COLING'94*, pages 1059–1063.

Helmbold, D., R. Schapire, Y. Siuger, and M. Warmuth. 1995. A comparison of new and old algorithm for a mixture estimation problem. *Proc. of COLT'95*, pages 61–68.

Jelinek, F. and R.I. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. *Proc. of Workshop on Pattern Recognition in Practice*, pages 381–402.

Lewis, David D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. *Proc. of SIGIR'92*, pages 37–50.

Lewis, David D. and Marc Ringuette. 1994. A comparison of two learning algorithms for test categorization. *Proc. of 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93.

Lewis, David D., Robert E. Schapire, James P. Callan, and Ron Papka. 1996. Training algorithms for linear text classifiers. *Proc. of SIGIR'96.*

Pereira, Fernando, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. *Proc. of ACL'93*, pages 183–190.

Robertson, S.E. and K. Sparck Jones. 1976. Relevance weighting of search terms. *Journ. of the American Society for Information Science*, 27:129–146.

Salton, G. and M.J. McGill. 1983. *Introduction to Modern Information Retrieval.* New York: McGraw Hill.

Schutze, Hinrich, David A. Hull, and Jan O. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. *Proc. of SIGIR'95.*

Tanner, Martin A. and Wing Hung Wong. 1987. The calculation of posterior distributions by data augmentation. *Journ. of the American Statistical Association*, 82(398):528–540.

Wong, S.K.M. and Y.Y. Yao. 1989. A probability distribution model for information retrieval. *Information Processing and Management*, 25(1):39–53.

Yamanishi, Kenji. 1996. A randomized approximation of the mdl for stochastic models with hidden variables. *Proc. of COLT'96*, pages 99–109.

Yang, Yiming and Christoper G. Chute. 1994. An example-based mapping method for text categorization and retrieval. *ACM Tran. on Information Systems*, 12(3):252–277.