Text Categorization Using Unlabeled Data

Advisor: Prof. Jungyun Seo

by

Youngjoong Ko

Department of Computer Science

(Natural Language Processing)

DISSERTATION

submitted to the faculty of the Graduate School of the Sogang University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science

Seoul, Korea

2003. 7

Dedicated to my parents

Contents

List of Figures

List of Tables

Abstract

In recent years, automatic text categorization has gained a prominent status in the information systems field, due to the widespread and continuously increasing availability of documents in digital form. Many machine learning algorithms have been applied to text categorization tasks. In the machine learning paradigm, a general inductive process automatically builds an automatic text classifier by learning, generally known as "*supervised learning*". The supervised learning approach finds a representation or decision rules from an example set of labeled documents for each category.

However, the supervised learning approach has some difficulties. One key difficulty is that it requires a large, often prohibitive, number of labeled training documents for accurate learning. Since labeling tasks must be done manually, this is a painfully time-consuming process. In addition, since the application area of text categorization has diversified from newswire articles and web pages to E-mails and newsgroup postings, it is also a difficult task to create training data for each application area.

In this thesis, we propose a new automatic text categorization method based on unsupervised or semi-unsupervised learning. Our proposed method uses only unlabeled documents and the title word of each category, and it automatically constructs machine-labeled training documents from them by a bootstrapping method. Then a text classifier learns with them and classifies text documents. Here, since our method exploits the machine-labeled documents as training data (they include many incorrectly labeled documents), we need a robust text classifier from noisy data. Therefore, we propose a new classifier, the TCFP classifier, which showed the best performance in our experiments.

The experiment results showed that our method achieved the significant performance in comparison with that of a supervised Naive Bayes classifier. In general, since unlabeled documents are much less expensive and easier to collect than labeled documents, our method is useful for text categorization tasks including online data sources such as web pages, E-mails and newsgroup postings. If one uses our method, building text categorization systems will be significantly faster and less expensive than by the supervised learning approach.

Chapter 1

Introduction

In recent years, automatic content-based document management tasks have gained a prominent status in the information systems field, due to the widespread and continuously increasing availability of documents in digital form. As a result, automatic text categorization (TC – also known as *text classification*, or *topic spotting*) has witnessed an increased and renewed interest.

A generally accepted definition of text categorization is

> "Text Categorization is the task of deciding whether a piece of text belongs to any of a set of prescribed categories. It is a generic text processing task useful in indexing for later retrieval, as a stage in natural language processing systems, for content analysis, and in many other roles."
>
> *Lewis*

Namely, text categorization systems categorize documents into one (or several) of pre-defined topics of interest. More recently, there has been an explosion of electronic text from the World Wide Web, electronic mail, corporate databases, chat rooms, and digital libraries. One way of organizing this overwhelming amount of data is to classify them into descriptive or topical taxonomies. By automatically creating and maintaining these taxonomies, we can aid people to search information and knowledge.

Until the late ‘80s, the most effective approaches to text categorization were based on the manual construction of rule sets [1]. In this approach, a person must define a detailed set of rules for automatic classification. Highly accurate text classifiers are built with this approach, but it takes significant costs. Constructing a complete rule set requires a lot of domain knowledge and a large amount of human time to create and tune the rules correctly. Therefore, this is not useful approach for text categorization in many cases.

In the ‘90s, a more efficient approach, the *machine learning* paradigm, emerged, and it definitely took the place of the rule based approach [1]. In the machine learning paradigm, a general inductive process automatically builds an automatic text classifier by “*supervised learning*”. The supervised learning approach finds a representation or decision rules from an example set of labeled documents for each category. This approach has shown high performance, and it is significantly less expensive than the rule based approach because it automatically composes the decision rules. A wide range of supervised learning algorithms has been applied to this area using a training data set of labeled documents. For examples, there are Naive Bayes [2][3], Rocchio [4], Nearest Neighbor (*k*-NN) [5], TCFP [6], and Support Vector Machines (SVMs) [7].

However, the supervised learning approach has some difficulties. One key difficulty is that it requires a large, often prohibitive, number of labeled training documents for accurate learning. Since labeling tasks must be done manually, it is a painfully time-consuming process. In addition, since the application area of automatic text categorization has diversified from newswire articles and web pages to E-mails and newsgroup postings, it is also a difficult task to create training data for each application area [8]. Lang (1995) found that after a person read and manually labeled about 1000 articles, a learned classifier achieved a precision of about 50% when making predictions for only the top 10% of documents about which it was most confident [9]. Most users of a practical system, however, would not do the labeling task for a thousand articles to obtain only this level of precision. They would obviously prefer algorithms that can have a high accuracy, but do not require a manual labeling task.

In this thesis, we propose a new automatic text categorization method based on unsupervised or semi-unsupervised learning. Our proposed method uses only unlabeled documents and the title word of each category, and it automatically constructs labeled training data from them by a bootstrapping method. Then a text classifier learns with them and classifies text documents. Our method makes use of data which are automatically labeled by machine and which include many incorrectly labeled data for learning. Hence, we develop a new classifier, *Text Categorization using Feature Projections (TCFP)*, with robustness from noisy data and fast execution speed. TCFP showed the best performance in our experiments. The results of our experiments showed that our method could produce a significant performance compared to the supervised Naive Bayes classifier. In general, since unlabeled documents are much less expensive and easier to collect than labeled documents, our method is useful for text categorization task including online data sources such as web pages, E-mails, and newsgroup postings. If one uses our method, building text categorization systems will be significantly faster and less expensive than by the supervised learning approach.

1.1 How Can an Automatic Text Classifier Be Built from Unlabeled Data?

Do you think it is possible to build a text classifier with only unlabeled data? Perhaps one can think that we cannot gain any information from unlabeled data for building the text classifier, because they do not contain the most important information - their categories. In general, the existing supervised learning approaches cannot construct any decision rules without the labeled data. Thus, we need to label our collected documents in order to use the existing supervised learning approaches. First of all, you can think of clustering techniques. Many clustering techniques were applied in this area so far, but they have not achieved a useful accuracy and they contained several problems for using clustering results as training data.

In this thesis, we explain how unlabeled data can be used as training data in text categorization. Above all, we look at the definition of text categorization again: "Text Categorization is the task of deciding whether a piece of text belongs to any of a set of *prescribed categories*." In other words, text categorization is a task based on the prescribed categories. Hence, we must know the categories for classifying documents before starting a text categorization task. Knowing the categories means that we can at least choose the representative title word of each category. This is the starting point of our proposed method. We carry out a bootstrapping task from title words for creating labeled training data. For the bootstrapping task, we first estimate the degree of semantic similarity between the title word and the other words using co-occurrence information in the collected documents. Words with high semantic similarity are chosen as keywords of each category. We next use contextual information in order to extend the vocabulary size of each category using the title word and keywords of each category. We define a context that contains a fixed number of words, and we extract the centroid-contexts that include the keywords or the title word within them. Using a similarity measure method and the centroid-contexts, we can group the contexts into clusters for prescribed categories, which consist of contextually similar occurrences. Each cluster is assumed to correspond to the training data of a category. Now we can do a learning task for text classifiers using these context-clusters as training data.

Here, we give an intuitive example of how to extend the vocabulary of each category. Suppose that we are interested in classifying web pages specially about '*Autos*' category. Above all, we can select 'automobile' as a title word. Then we can choose words ('car', 'gear', 'transmission', 'sedan', and so on) as having the high semantic similarity to 'automobile' as keywords. We segment the collected documents into contexts and we extract all contexts using the title word or the keywords. We call these contexts the centroid-contexts. Through the centroid-contexts, we can gain many contextually co-occurred words: 'driver', 'clutch', 'trunk', and so on. They are words in first-order co-occurrence with the title word and the keywords. But we cannot get enough vocabulary for recognizing the '*Autos*' category yet. We therefore extract contexts that are similar to centroid-contexts by a similarity measure method. These contexts contain words in second-order co-occurrence. Finally, we define the context-cluster of a category as centroid-contexts of the category and contexts clustered by a similarity measure method. Now we can gain enough vocabulary from context-clusters for learning a classifier.

For our classification task, we select a generative model, a Naive Bayes classifier, and set the model's parameter values for each category from the context-clusters [3]. After the training task, we classify and label the original collected documents. Finally, we can get a labeled data of a document unit. In order to use these as training data, we develop a new text classifier, TCFP, with robustness from noisy data and fast execution speed and we learn it from the machine-labeled documents in a supervised learning manner. This method is similar to the statistical technique of Expectation-Maximization (EM) [8][10][11]. Nigam however verified that EM can hurt accuracy when labeled data are sparse: the WebKB and Reuters data sets [12]. Thus, we do not apply EM algorithm to our method and use the machine-labeled data from our method as only training data for supervised-manner classifiers; they require labeled data of a document unit.

Our hope is that the proposed method gives us the faster and less expensive classifier than a classifier by the supervised learning approach, and it has the comparable accuracy.

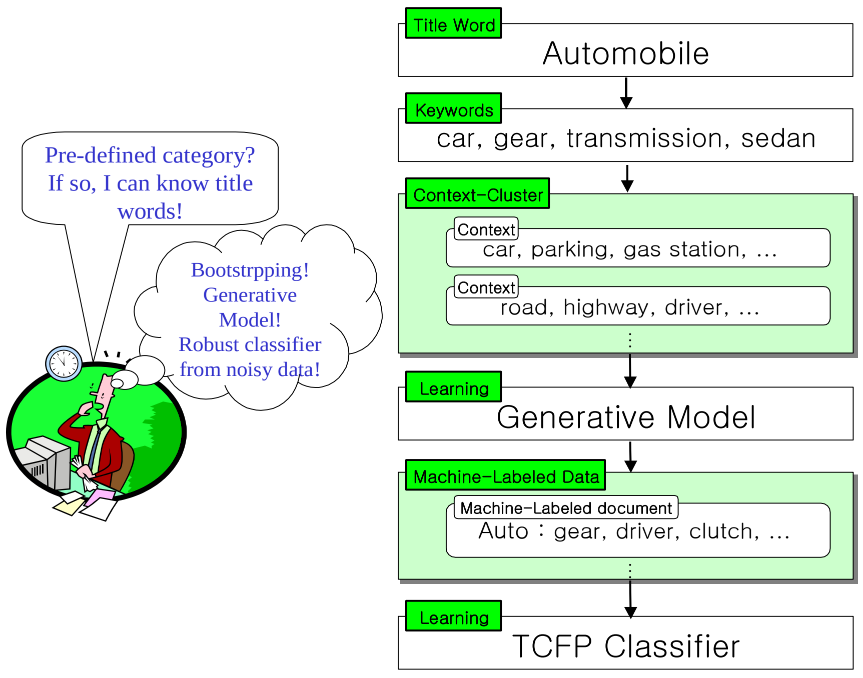

Pre-defined category? If so, I can know title words!
Bootstrpping! Generative Model! Robust classifier from noisy data!
Title Word
Automobile
Keywords
car, gear, transmission, sedan
Context-Cluster
Context
car, parking, gas station, ...
Context
road, highway, driver, ...
Learning
Generative Model
Machine-Labeled Data
Machine-Labeled document
Auto : gear, driver, clutch, ...
Learning
TCFP Classifier

Figure 1. 1 An example of the bootstrapping task in our method

1.2 Outline of This Thesis

The outline of this thesis is as follows. Chapter 2 surveys the related work. It also relates different approaches to use unlabeled data for text categorization. We compare our proposed method with them in this chapter.

Chapter 3 presents how to automatically build the basic training data from unlabeled documents using the title word of each category. It derives the bootstrapping method that creates the context-clusters from title words of categories. Here, we evaluate our method using the Newsgroups, WebKB, and Reuters data sets. We obtained a meaningful performance from our proposed method in these data sets.

In Chapter 4, we learn the supervised-manner classifiers (Naive Bayes, Rocchio, *k*-NN, SVMs, and TCFP) using data labeled by our method (the machine-labeled data) and classify documents again by these classifiers. Here, we present a new classifier, TCFP, which has robustness from noisy data and fast execution speed. Our method requires a robust classifier from noisy data because it exploits the machine-labeled data as training data; they generally contain many incorrectly labeled documents. TCFP showed good performance in respect with its accuracy and running time. Especially, TCFP showed better performance in the application area with a lot of noisy data like our method. Through this procedure, we obtained advanced performance. The final accuracy of our method is comparable with that of the supervised learning approach.

Chapter 5 compares our method with a clustering method called the **sIB** algorithm. This clustering algorithm is applied to text classification with an unlabeled data set. We compare the performance of our method with that of **sIB**. As a result, we achieved better accuracy than the clustering method, **sIB**.

Chapter 6 presents a new keyword extraction method to overcome a problem of our system, that its performance depends on the input title words and keywords. Then we observe the effect of learning from small training data through three data sets.

In Chapter 7, we discuss the conclusions of this dissertation. We also describe several future researches about more efficient methods for learning from unlabeled data.

Chapter 2

Related Work

Text categorization is an abundant research field with existing and ongoing scientific research. Related approaches for using unlabeled data in text categorization generally have two directions; One learns classifiers from a combination of labeled and unlabeled data [8][11][12][13][14][15][16], and the other employs clustering algorithms for text classification [17]. This chapter surveys the current state of text categorization and the related approaches for using unlabeled data.

2.1 Text Categorization

Text categorization has been applied to a wide variety of practical applications: cataloging news articles [18], cataloging postings of UseNet discussion groups [12][14], classifying web pages into a symbolic ontology [19]; finding a person’s homepage [20], automatically threading and filtering email by content [21][22], and book recommendation [23].

Recently, a variety of machine learning techniques have been applied to this literature: Naive Bayes [2][3], several rule learning algorithms [24][25][26], Rocchio algorithm [4], neural net [27], instance-based methods such as k-nearest neighbor [28], TCFP [6], Support Vector Machines (SVMs) [7][29], and a variety of boosting approaches [30][31]. Among these techniques, SVMs have recently shown much promise. However, no single technique has emerged as clearly better than the others, though some recent evidence suggests that *k*-NN and SVMs perform at least better than other algorithms when there is a lot of labeled data for each category of interest [32].

Most studies of text categorization simply represent a document as bags-of-words, which uses the number of times each word occurs in a document, or even just whether or not it occurred. On the other side, there are efforts to include more linguistic or semantic information for improvements of classification accuracy. Furnkranz uses shallow syntactic phrase patterns and finds some improvements to Naive Bayes and rule learning algorithms [33]. Two studies use WordNet, a semantic network of the English language, for text categorization [34][35]. There is a study to adjust the weight of words according to the importance of the sentence [36]. Results from this research reported an improved accuracy for text categorization.

2.2 Learning with Unlabeled Data

We here present a survey of using unlabeled data for learning classifiers. There are two approaches: the combination of labeled and unlabeled data [8][11][12][13][14][15][16] and the clustering algorithm for text categorization [17].

2.2.1 Learning with Combining Labeled and Unlabeled Data

Learning using labeled and unlabeled data can be referred to as semi-supervised learning. This allows taking advantage of the strengths of both supervised learning and unsupervised learning to learn accurate classifiers and to exploit unlabeled data, while getting rid of their common drawbacks; the enormous need for labeled data in supervised learning, and the inability to identify known categories and the difficulty in choosing a suitable number of clusters in unsupervised learning.

A. The Expectation Maximization (EM) Approach

The methods of learning classifiers from a combination of labeled and unlabeled data have been studied in the statistics community. Dempster presents the theory of the EM framework, an iterative technique for likelihood maximization [10]. It is applied to estimating maximum likelihood parameters for mixture models from labeled and unlabeled data [37]. Then it is used for classification [38].

Using likelihood maximization of mixture models for combining labeled and unlabeled data for classification has recently been developed by the machine learning community [12][39] [40].

Nigam et al. study an EM technique for combining labeled and unlabeled data for text categorization [8][11][12]. They show that the accuracy of learned text classifiers can be improved by augmenting a small number of labeled training documents with a large pool of unlabeled documents. They introduce an algorithm for learning from labeled and unlabeled documents based on the combination of EM and a Naive Bayes classifier. The algorithm first trains a classifier using the available labeled documents, and probabilistically labels the unlabeled documents. It then trains a new classifier using the labels for all the documents, and iterates to convergence. This basic EM procedure works well when the data confirms to the generative assumptions of the model. However, these assumptions are often violated in practice and poor performance can result; similar results are observed in their experiments with the Reuters and the WebKB data sets. Experimental results show that the use of unlabeled data reduces classification error by up to 30%. Furthermore, their algorithm is applied to the construction of a domain-specific search engine, *Cora* [12][13]. In a domain-specific search engine, more specific information such as category hierarchy, keywords, and phrases can be useful. With a category hierarchy and human-provided keywords, a rule-list classifier can be built and it preliminarily can label unlabeled documents. An improved classifier is bootstrapped using the preliminary labeled documents and EM. The bootstrapping iterations are EM steps that use unlabeled data and hierarchical shrinkage to estimate parameters of a Naive Bayes classifier. However, since this method requires specific preliminary information such as a category hierarchy and human-provided keywords to build a rule-list classifier, its application can be restricted.

B. The Semi-supervised Text Learning Approach Using an EM-like Scheme

Lanquillon presents another approach for learning from labeled and unlabeled data [14][15]. Since the most straightforward way to make use of unlabeled data is through unsupervised learning, he exploits partitional clustering methods. Partitional clustering methods generate a single partition of the data in an attempt to recover natural groups present in the data. With the guidance of labeled data, the hope is that these groups will better match the underlying category structure. The semi-supervised partitional clustering algorithm allows the category label of each unlabeled document to change after each iteration like EM. For the semi-supervised partitional learning task, he permits the use of any type of supervised learning algorithm such as single-prototype classifier (SPC) and SVMs, whereas Nigam depends on the Naive Bayes classifier [8][11][12].

C. Discriminative Approaches

A support vector machine discriminatively finds parameters for a linear separator when given labeled data [41]. Generally, this approach is equally applicable to scenarios with labeled and unlabeled data. They work to find the linear separator between the labeled examples of each category, which maximizes the margin over both the labeled and unlabeled examples. Joachims demonstrates the efficiency of this approach for several text categorization tasks [42]. Bennet and Emiriz achieve small improvements on some UCI data sets [16]. It seems that SVMs assume decision boundaries lie between categories in low-density regions of example space, and the unlabeled examples help to find these areas. However, Zhang and Oles argue both theoretically and experimentally that SVMs are unlikely to be helpful for classification in general [43].

2.2.2 The Clustering Algorithm for Text Categorization

Techniques for unsupervised document clustering have developed in the Information Retrieval (IR) community [44][45]. Slonim suggests to use clustering techniques for unsupervised document classification [17]. In this task, when a collection of unlabeled documents is given, he attempts to find clusters that are highly correlated with the true topics of the documents. Since no labeled examples are provided for the topics, he employs unsupervised clustering methods. In his paper, he suggests a new clustering method, the sequential Information Bottleneck (**sIB**) algorithm. It has the performance and time & space complexity better than those of the agglomerative clustering algorithms. On various data sets, he verified that the performance of **sIB** is superior to the other unsupervised methods used in his experiments. Additionally, results are even competitive with a standard supervised Naive Bayes classifier.

Chapter 3

Learning with Unlabeled Data Using the Title Word of Each Category

3.1 Introduction

The goal of Text Categorization is to classify documents into a certain number of pre-defined categories. Text Categorization is an active research area in information retrieval and machine learning. A wide range of supervised learning algorithms has been applied to this area using a training data set of labeled documents. However, the previous supervised learning approaches have some problems. One of them is that they require a large, often prohibitive number of labeled documents for accurate learning. To overcome this, we introduce a new algorithm for learning from only unlabeled documents based on the bootstrapping method and the generative model such as a Naive Bayes classifier. Actually, unlabeled documents cannot give us any information for text classification because they do not have the most important information (their categories) for supervised learning approach. But our hope is that we know the categories which documents are classified into. Hence, we can select title words that stand for categories, and make use of them as the seed words of our bootstrapping method.

We then extract keywords of each category by calculating semantic similarity between the title word and the other words. Since a generative model such as the Naive Bayes classifier needs more information to learn, we use contextual information. Context here is defined as 60 words. The title word and keywords are then used for choosing representative or centroid contexts of each category; they contain at least one of the title word and keywords. We call them centroid-contexts. As a result, we obtain a set of words in first-order co-occurrence with the title word and keywords. In addition, second-order co-occurrence information is gathered by assigning remaining contexts to context-cluster of each category; the remaining contexts do not contain any title word or keyword. For the assigning criterion, we use two kinds of methods. One method measures similarity between the centroid-contexts and the remaining contexts. It is a famous algorithm in the word sense disambiguation literature; the algorithm by Karov and Edelman [46]. The next method clusters contexts by the K-means algorithm: a well-known algorithm among clustering algorithms [44]. Generally, the second-order co-occurrence information is less sparse and more robust than the first-order co-occurrence information. Finally we can construct the basic training data from the collected context-clusters of categories for the Naive Bayes generative model.

From experimental results, we obtained 79.36% accuracy in the Newsgroups data set, 73.63% accuracy in the WebKB data set, and 88.62% precision-recall breakeven point in the Reuters data set. The results are comparable with that of supervised method.

The outline of this chapter is as follows. Section 3.2 describes the feature selection method and the Naive Bayes classifier. They are used as the baseline through this thesis. Section 3.3 presents how to create the training data in our proposed method in detail. In Section 3.4, we discuss the experimental results showing that we get meaningful accuracy even though using only unlabeled data. Section 3.5 discusses and evaluates our method and results.

3.2 Feature Selection and a Naive Bayes Generative Model

In this thesis, we use the χ^2 statistics method for feature selection [47] and the Naive Bayes generative model as a text classifier [2][3][48]. They are the methods basically used throughout this thesis.

3.2.1 Feature Selection

There are many feature selection methods in this literature. Yang and Pederson introduce and compare the feature selection methods in their paper [47]. As a result, the χ^2 statistics method shows the best performance among them. The size of vocabulary in our experiment is selected by ranking words according to their χ^2 statistics with respect to the category. Using the two-way contingency table of a word *t* and a category *c*, the word-goodness measure is defined as follows:

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C)\times(B+D)\times(A+B)\times(C+D)} \tag{3.1}$$

where i) *A* is the number of times *t* and *c* co-occur, ii) *B* is the number of times *t* occurs without *c*, iii) *C* is the number of times *c* occurs without *t*, iv) *D* is the number of times neither *c* nor *t* occurs, and vi) *N* is the total number of documents.

To measure the goodness of a word in a global feature selection, we combine the category-specific scores of a word as follows:

$$\chi^2(t) = \max_{j=1}^{m} \left\{ \chi^2(t, c_j) \right\} \tag{3.2}$$

Where *m* denotes the number of categories.

3.2.2 A Naive Bayes Generative Model

This section describes a probabilistic framework to characterize the nature of documents and classifiers. The framework defines a probabilistic generative model for the data, and embodies three assumptions about the generative process: i) the data is produced by a mixture model, ii) there is a one-to-one correspondence between mixture components and categories, and iii) the mixture components are multinomial distributions of individual words; the words of a document are produced independently of each category [12]. From these assumptions, we can derive the Naive Bayes classifier, a simple and commonly-used tool for text categorization, by finding the most probable parameters for the model.

Documents are generated by a mixture of multinomial models, where each mixture component corresponds to a category. Formally, every document is generated according to a probability distribution defined by the parameters for the mixture model, denoted θ. Let there be $|C|$ categories and a vocabulary of size $|V|$; each document d has $|d|$ words in it. The probability distribution consists of a mixture of components $c_j \in C = \{c_1, ..., c_{|C|}\}$. Each component is parameterized by a disjoint subset of θ. A document, d_i, is created by first selecting a mixture component according to the mixture weights (or category probabilities), P($c_j|\theta$), and having this selected mixture component generate a document according to its own parameters, with distribution P($d_i|c_j;\theta$). Thus, we can characterize the likelihood of document d_i with a sum of total probability over all mixture components [12]:

$$P(d_i \mid \theta) = \sum_{j=1}^{|C|} P(c_j \mid \theta) P(d_i \mid c_j; \theta) \tag{3.3}$$

We here assume that there is a one-to-one correspondence between mixture model components and categories, and thus use c_j to indicate the j-th mixture component as well as the j-th category.

Here, we focus on the second term of Formula 3.3, and express the probability of a document given a mixture component in terms of its constituent features: the document length and the words in the document. Note that, in general, we assume document length is independent of category and the words of a document are generated independently of context: the standard Naive Bayes assumption. We further assume that the probability of a word is independent of its position within the document. Thus, the Naive Bayes expression for the probability of a document given its category is as follows:

$$P(d_i \mid c_j; \theta) = P(|d_i|) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}} \mid c_j; \theta) \tag{3.4}$$

Therefore, the parameters of an individual mixture component define a multinomial distribution over words; the collection of word probabilities, each written $\theta_{w_t|c_j}$, such that $\theta_{w_t|c_j} \equiv P(w_t \mid c_j; \theta)$, where t={1,…,|V|} and $\sum_t P(w_t \mid c_j; \theta) = 1$. Since we assume that, for all categories, document length is identically distributed, it does not need to be parameterized for classification. The remaining parameters of the model are category probabilities, written θ_{c_j}. Thus, the final parameters of model consist of multinomials and category probabilities: $\theta = \{\theta_{w_t|c_j} : w_t \in V, c_j \in C; \theta_{c_j} : c_j \in C\}$.

Learning a Naive Bayes text classifier is to estimate the parameters of the generative model by using a set of labeled training data, *D*. The estimate of the parameters θ is written as $\hat{\theta}$. Since Naive Bayes uses the maximum a posteriori (MAP) estimate, it finds the value of θ that is most probable given the evidence of the training data and a prior, that is, $\arg\max_\theta P(\theta \mid D)$.

The parameter estimation formulae that result from maximization with the data and our prior are the familiar ratios of empirical counts. The estimated probability of a word given a category, $\hat{\theta}_{w_t|c_j}$, is simply the number of times word w_t occurs in the training data for category c_j, divided by the total number of word occurrences in the training data for that category. Here, the Laplace smoothing is used for the estimate of a word given a category, $\hat{\theta}_{w_t|c_j}$. Smoothing is necessary to prevent zero probabilities for infrequently occurring words.

The word probability estimates are as follows:

$$\hat{\theta}_{w_t|c_j} \equiv P(w_t \mid c_j; \hat{\theta}) = \frac{1 + N(w_t, c_j)}{|V| + \sum_{t=1}^{|V|} N(w_t, c_j)} \tag{3.5}$$

where $N(w_t,c_j)$ is the count of the number of times word w_t occurs in category c_j. The category probabilities, $\hat{\theta}_{c_j}$, are estimated in the same manner, and also use the smoothing technique:

$$\hat{\theta}_{c_j} \equiv P(c_j \mid \hat{\theta}) = \frac{1 + |c_j|}{|C| + |D|} \tag{3.6}$$

where $|c_j|$ is the number of examples in category c_j, $|D|$ is the number of all example in training data, and $|C|$ is the number of categories. In this thesis, the example is used as different meanings in each learning method: a labeled document in a supervised approach and a labeled context in our method.

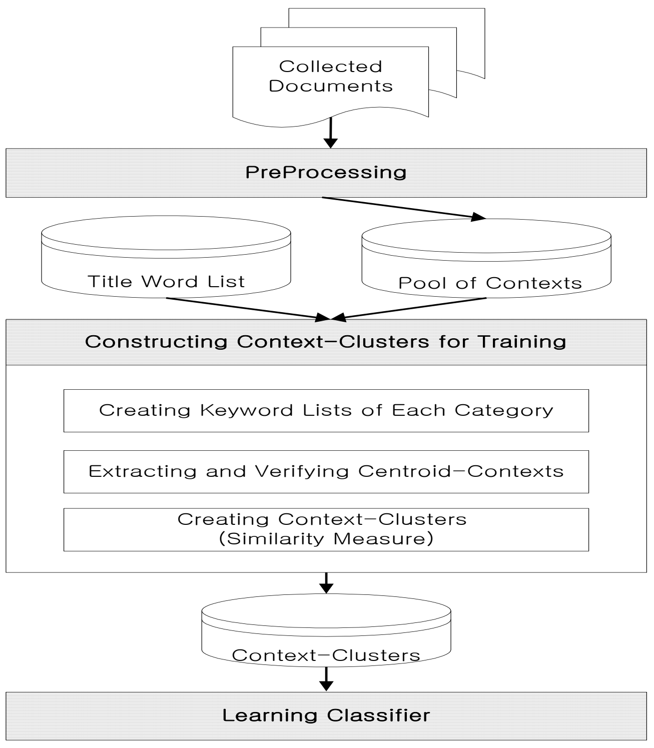
By Formulae 3.5 and 3.6, the parameters are estimated from the training data. It is then possible to turn the generative model and calculate the probability that a particular mixture component (or category) generated a given document. We formulate this by an application of Bayes's rule and substations using Formulae 3.3 and 3.4.

$$P(c_j \mid d_i; \hat{\theta}) = \frac{P(c_j \mid \hat{\theta}) P(d_i \mid c_j; \hat{\theta})}{P(d_i \mid \hat{\theta})} = \frac{P(c_j \mid \hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}} \mid c_j; \theta)}{\sum_{r=1}^{|C|} P(c_r \mid \hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}} \mid c_r; \theta)} \tag{3.7}$$

We give many assumptions for this generation of text documents; one-to-one correspondence between mixture components and categories, word independence, and document length distribution. But note that they are not all accepted in practical real-world text data. Especially, words in a document are not independent of each other due to grammar and topicality. However, the Naive Bayes is one of the most commonly-used classifiers for simplicity and high performance [2][3][6] [19][49][50].

3.3 Learning a Naive Bayes from Unlabeled Data

In the previous section, we presented how to learn a Naive Bayes classifier using labeled data. Here, we explain how to create labeled training data from unlabeled data and how to learn a Naive Bayes classifier with them.



Collected Documents
PreProcessing
Title Word List
Pool of Contexts
Constructing Context-Clusters for Training
Creating Keyword Lists of Each Category
Extracting and Verifying Centroid-Contexts
Creating Context-Clusters (Similarity Measure)
Context-Clusters
Learning Classifier

Figure 3. 1 Overview of the proposed method

The proposed method consists of three modules as shown in Figure 3.1: a module to preprocess collected documents, a module to construct context-clusters for training, and a module to learn a classifier using context-clusters. In the preprocess module, we first segment the collected documents into sentences and do a POS tagging task to extract content words from each sentence. Next, we define contexts and make the collected documents into the unit of contexts. We then create labeled context-clusters using only the title word of each category. In the last module, we estimate the parameters from the labeled context-clusters for the generative model, the Naive Bayes classifier.

We describe each module in the following subsections in detail.

3.3.1 Preprocessing

The preprocessing module has two main roles, extracting content words and reconstructing the collected documents into contexts. First of all, the contents of documents are segmented into sentences. We then extract content words from each sentence. To extract the content words, we use the Brill POS tagger [51]. Words with noun or verb POS tags are considered as content words.

Generally, the supervised learning approach with labeled data regards a document as a unit of meaning. But since we do not have labeled documents, we define a new unit of meaning, *context*. Context refers to the part of a text that surrounds the particular word or a passage and determines its meaning. It has been studied in various areas of *Natural Language Processing* such as *Word Sense Disambiguation* [52][53]. Since we can use only the title word of each category, we need to bootstrap the meaning of each category using contextual information of the title word. Our final hope is to obtain the labeled data through this bootstrapping process. In this chapter, we define *context* as a sequence of 60 words within a document, and consider only contexts disregarding the unit of document. To extract contexts, we use sliding window techniques [54][55]. The window is slid from the first word of the document to the last in the size of the window (60 words) and the interval of each window (30 words). Therefore, the final output of preprocessing is the set of context vectors that are represented as content words of each context.

3.3.2 Constructing Context-clusters for Training

This section presents how to create labeled data. Using title words, we automatically construct keywords for each category, which reflect the features of each category sufficiently. These are then used to extract centroid-contexts of each category that include at least one of the title word and keywords. We consider the centroid-contexts as core contexts of each category. To obtain more information of each category, we employ two kinds of methods. On the first method, we calculate the similarity between centroid-contexts and remaining contexts (the remaining contexts do not contain any title word or keyword) and assign each remaining context to category with the highest similarity value. As the other method, we cluster contexts using a clustering algorithm such as the K-means algorithm. By the above two methods, we can finally obtain the labeled context-cluster for each category. These context-clusters can be used as training data of a generative Naive Bayes classifier. As a result, we can do text classification task with a Naive Bayes classifier.

A. Creating the Keyword List of Each Category

The starting point of our method is that we have title words and collected unlabeled documents for classification. A title word can present the main meaning of each category but could be insufficient in representing any category for text categorization. Thus, we need to find words that are semantically related to a title word and we define them as keywords of each category.

There are several methods to calculate semantic relatedness between words: i) using a thesaurus such as WordNet [56], that records the synonym and hyponym relationships between words, or ii) estimating the degree of semantic similarity between words using co-occurrence information between words in a corpus [57]. We adopt the latter method in our method.

The score of semantic similarity between a title word, *T*, and a word, *W*, is calculated by the cosine distance, as in the following formula:

$$\cos(T,W) = \frac{\sum_{i=1}^{n} t_i \times w_i}{\sqrt{\sum_{i=1}^{n} t_i^2 \times \sum_{i=1}^{n} w_i^2}} \tag{3.8}$$

where t_i and w_i represent the occurrence (binary value: 0 or 1) of words *T* and *W* in document *i* respectively, and *n* is the total number of documents in the corpus. This method can be considered to calculate the similarity score between words, based on the degree of their co-occurrence in the same document. We use a set of documents collected for training in order to calculate the similarity score. We here consider only words with noun POS tags as candidate words for keywords.

Using this similarity score, we can construct the sorted list of words in each category. But the keywords for text categorization must have the power to discriminate categories as well as similarity with the title word. Thus we assign a word to the keyword list of a category with the maximum similarity score, and we recalculate the score of words in the category using the following formula:

$$Score(W, c_{\max}) = sim(T_{\max}, W) + (sim(T_{\max}, W) - sim(T_{\sec ond\max}, W)) \tag{3.9}$$

where $T_{\max}$ is the title word with the maximum similarity score of a word *W*, $c_{\max}$ is the category of the title word $T_{\max}$, and $T_{second\max}$ is the title word with the second high similarity score of a word *W*.

This formula means that words with high ranking in a category have a high similarity score with the title word of the category and distinct difference in similarity score with the other title words. We sort words assigned to each category according to the score calculated by Formula 3.9 in descending powers and choose top *m* words as a set of keywords in a category. In the WebKB data set, top five words are selected as keywords empirically. Table 3.1 shows the list of keywords for each category in the WebKB data set.

Table 3. 1 The list of keywords for each category in the WebKB data set

Category	***Title Word***	***Keywords***
course	course	assignments, hours, instructor, class, fall
faculty	professor	associate, ph.d, fax, interests, publications
project	project	system, systems, research, software, information
student	student	graduate, computer, science, page, university

The keywords of Table 3.1 are shown as a form without stemming. The lists of keywords for other data sets are shown in the Appendix in detail.

B. Extracting and Verifying Centroid-contexts

Now we obtain the set of keywords for each category. Using them and title words, we can extend the vocabulary size of each category. For this, we use contextual information; we define a unit of the context as a sequence of 60 words within a document and choose contexts that contain the keywords or the title word of a category. We call these contexts *centroid-contexts* of the category. A context with keywords or title words of two or more categories is excluded from centroid-contexts to prevent the ambiguousness of categories. Among centroid-contexts, some contexts could not have good features of a category even though they include the keywords or the title word of the category. To rank the importance of centroid-contexts, we can compute the importance score of each centroid-context as follows:

1) Word weights are calculated using Term Frequency (TF) and Inverse Category Frequency (ICF) [58].

① The within-category word frequency (TF_{ij})

$$TF_{ij} = \text{the number of times words } t_i \text{ occurs in the } j\text{-th category} \tag{3.10}$$

② In information retrieval, Inverse Document Frequency (IDF) is generally used. But, since a context is a processing unit in our method, the document frequency cannot be counted. Moreover, ICF is defined by Cho and Kim, and its efficiency is verified in [58]. Thus, we use ICF in our method. ICF is calculated as follows:

$$ICF_i = \log(M) - \log(CF_i) \tag{3.11}$$

where CF_i is the number of categories that contain t_i and M is the total number of categories.

③ The combination (TF-ICF) of the above Formulae 3.10 and 3.11, that is, weight w_{ij} of word t_i in j-th category is calculated as follows:

$$w_{ij} = TF_{ij} \times ICF = TF_{ij} \times (\log(M) - \log(CF_i)) \tag{3.12}$$

2) Using word weights w_{ij} calculated by Formula 3.12, the score of a centroid-context S_k in j-th category c_j is computed as follows:

$$Score(S_k, c_j) = \frac{w_{1j} + w_{2j} + \ldots + w_{Nj}}{N} \tag{3.13}$$

where N is the total number of words in a centroid-context.

3) The centroid-contexts of each category are sorted in a decreasing order according to the importance score. This order of centroid-contexts is used in the following process: *Creating Context-clusters*.

As a result, we obtain a set of words in first-order co-occurrence with the title word and keywords from centroid-contexts of each category.

C. Creating Context-clusters

Here, we gather the second-order co-occurrence information by assigning remaining contexts to the context-cluster of each category. For the assigning criterion, we employ two kinds of algorithms. The algorithms for gathering the contextual information have been studied and developed in *Word Sense Disambiguation* literature [46][59]. We here focus on a famous algorithm for measuring similarity between contexts: the algorithm by Karov and Edelman [46]. Karov and Edelman define the relation between word similarity and context similarity. By this definition, a word similarity matrix and a context similarity matrix are built and the similarities are computed by an iterative process using these matrices [46]. We then measure similarity between centroid-contexts of each category and the remaining contexts using this algorithm. In the next method, we cluster contexts into context-clusters by the K-means algorithm: the well-known algorithm among clustering algorithms [44][52]. Here, centroid-contexts of each category are used for constructing the initial centroid vector of the K-means algorithm. In this thesis, we reform a part of these algorithms on purpose to adjust them to our method. The two revised algorithms are described in detail in the following parts.

(1) Measuring similarity based on word similarity and context similarity

We here explain how to measure similarity between centroid-contexts and the remaining contexts and how to assign remaining contexts into the context-cluster of each category.

1) Measurement of word and context similarity

As similar words tend to appear in similar contexts, we compute the similarity by using contextual information [46][54][60]. In this thesis, words and contexts play complementary roles. That is, a context is represented by the set of words that it contains, and a word by the set of context in which it appears. Contexts are similar to the extent that they contain similar words, and words are similar to the extent that they appear in similar contexts. This definition is circular. Thus, it is applied iteratively using two matrices as shown in Figure 3.2. In this thesis, we set the number of iterations as 3, as is recommended by Karov and Edelman [46].

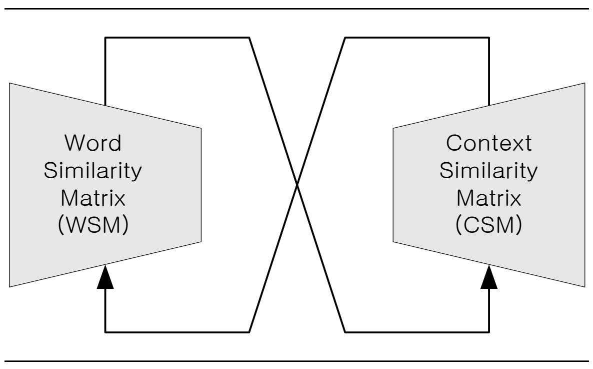


Figure 3. 2 Iterative computation of word and context similarity

In Figure 3.2, each category has a word similarity matrix WSM_n and a context similarity matrix CSM_n. In each iteration *n*, we update WSM_n, whose rows and columns are labeled by all content words encountered in the centroid-contexts of each category and input remaining contexts. In that matrix, the cell (*i*,*j*) holds a value between 0 and 1, indicating the extent to which the *i*-th word is contextually similar to the *j*-th word. Also, we keep and update a CSM_n, which holds similarity among contexts. The rows of CSM_n correspond to the remaining contexts and the columns to the centroid-contexts. In this thesis, the number of input contexts of row and column in *CSM* is limited to 200, considering execution speed and memory allocation.

To compute the similarity, we initialize WSM_n to the identity matrix. That is, each word is fully similar (1) to itself and completely dissimilar (0) to other words. The following steps are iterated until the changes in the similarity values are small enough.

1. Update the context similarity matrix CSM_n, using the word similarity matrix WSM_n.
2. Update the word similarity matrix WSM_n, using the context similarity matrix CSM_n.

2) Affinity formulae

To simplify the symmetric iterative treatment of similarity between words and contexts, we define an auxiliary relation between words and contexts as affinity. A word *W* is assumed to have a certain affinity to every context, which is a real number between 0 and 1. It reflects the contextual relationships between *W* and the words of the context. If *W* belongs to a context *X,* its affinity to *X* is 1. If *W* is totally unrelated to *X,* the affinity is close to 0. If *W* is contextually similar to the words of *X,* its affinity to *X* is between 0 and 1. In a similar manner, a context *X* has some affinity to every word, reflecting the similarity of *X* to the contexts involving that word.

Affinity formulae are defined as follows [46]. In these formulae, $W \in X$ means that a word belongs to a context:

$$aff_n(W, X) = \max_{W_i \in X} sim_n(W, W_i) \tag{3.14}$$

$$aff_n(X, W) = \max_{W \in X_j} sim_n(X, X_j) \tag{3.15}$$

In the above formulae, *n* denotes the iteration number, and the similarity values are defined by WSM_n and CSM_n. Every word has some affinity to the context, and the context can be represented by a vector indicating the affinity of each word to it.

3) Similarity formulae

The similarity of W_1 to W_2 is the average affinity of the contexts that include W_1 to W_2, and the similarity of a context X_1 to X_2 is a weighted average of the affinity of the words in X_1 to X_2. Similarity formulae are defined as follows:

$$sim_{n+1}(X_1, X_2) = \sum_{W \in X_1} weight(W, X_1) \cdot aff_n(W, X_2) \tag{3.16}$$

$$\begin{aligned} & if \quad W_1 = W_2 \\ & \quad sim_{n+1}(W_1, W_2) = 1 \\ & else \\ & \quad sim_{n+1}(W_1, W_2) = \sum_{W_1 \in X} weight(X, W_1) \cdot aff_n(X, W_2) \end{aligned} \tag{3.17}$$

The weights in Formula 3.16 are computed as the following methodology in the next section. The sum of weights in Formula 3.17, which is a reciprocal number of contexts that contain *W₁*, is 1. These values are used to update the corresponding entries of *WSMₙ* and C*SMₙ*.

4) Word weights

In Formula 3.16, the weight of a word is a product of three factors. It excludes the words that are expected to be given unreliable similarity values. The weights are not changed in their process of iterations.

1. *Global frequency*: Frequent words in total contexts are less informative of context similarity. For example, a word like ‘do’ frequently appears in any context. The formula is as follows:

$$1 - \frac{freq(w)}{\max\ freq(w)} \tag{3.18}$$

where max*freq(w)* is the value of the highest frequency in total contexts.

2. *Log-likelihood factor*: In general, the words that are indicative of the category appear in centroid-contexts more frequently than in total contexts. The log-

likelihood factor captures this tendency. It is computed as follows [46]:

$$\log \frac{\Pr(w_i \mid w)}{\Pr(w_i)} \tag{3.19}$$

where *Pr(w$_i$)* is estimated from the frequency of w_i in the total contexts, and *Pr(w$_i$|w)* from the frequency of w_i in centroid-contexts. To avoid poor estimation for words with a low count in centroid-contexts, we multiply the log-likelihood by Formula 3.20 where *count(w$_i$)* is the number of occurrences of w_i in centroid-contexts. For the words which do not appear in centroid-contexts, we assign the weight (1.0) to them. And, for the other words, we assign the weight that adds 1.0 to the score computed by Formula 3.19:

$$\min \{1, \frac{count(w_i)}{3}\} \tag{3.20}$$

3. *Part of speech:* Each part of speech is considered as a weight. We assign the weight (1.0) to proper noun, common noun, and foreign word, and assign the weight (0.6) to verb.

The weight of a word is the product of the above factors and each weight are normalized by the sum of weights of words in a context as follows [46]:

$$weight(w_i, X) = \frac{F(w_i, X)}{\sum_{w_j \in X} F(w_j, X)} \tag{3.21}$$

where $F(w_i, X)$ is the weight before normalization.

5) Assigning remaining contexts to a category

We first compute similarity of the remaining contexts to the centroid-contexts. Then we decide a similarity value of each remaining context for each category using the following method:

$$\underset{c_i \in C}{sim}(X, c_i) = aver\{ \underset{S_j \in CC_{c_i}}{sim} (X, S_j) \} \tag{3.22}$$

In Formula 3.22, i) X is a remaining context, ii) $C = \{c_1, c_2, ..., c_m\}$ is a category set, and iii) $CC_{c_i} = \{S_1, ..., S_n\}$ is a controid-contexts set of category c_i.

Each remaining context is assigned to a category which has a maximum similarity value. But there may exist remaining contexts which do not belong to any category. To remove these remaining contexts, we set up a dropping threshold using normal distribution of similarity values as follows:

$$\max\{ \underset{c_i \in C}{sim}(X, c_i) \} \geq \mu + \theta\sigma \tag{3.23}$$

where i) X is a remaining context, ii) μ is an average of similarity values; the similarity values, $\underset{c_i \in C}{sim}(X, c_i)$, include those of all input remaining contexts in each iteration, iii) σ is a standard deviation of similarity values, and iv) θ is a numerical value corresponding to the threshold (%) in normal distribution table.

Finally, a remaining context is assigned to the context-cluster of any category, when the category has a maximum similarity above the dropping threshold value.

(2) K-means algorithm

Here, we explain how to cluster the remaining contexts into the context-cluster of each category by a K-means algorithm.

1) K-means algorithm in our method

K-means is a hard clustering algorithm that defines clusters by the center of mass of their members [52]. We go through several iterations to assign each context to the cluster whose center is closest. After all contexts have been assigned, we re-compute the center of each clusters as the mean $\vec{\mu}$ of its members as follows:

$$\vec{\mu} = \frac{1}{|c_j|} \sum_{\vec{x} \in c_j} \vec{x} \tag{3.24}$$

where c_j is a cluster at each iteration, and $\vec{x}$ is a remaining context.

In our method, we use the cosine distance metric as the distance function as follows:

$$\cos(\vec{\mu}, \vec{x}) = \frac{\sum_{i=1}^{n} u_i \times x_i}{\sqrt{\sum_{i=1}^{n} u_i^2 \times \sum_{i=1}^{n} x_i^2}} \tag{3.25}$$

We need a set of initial cluster centers in the beginning. Since the centroid-contexts of each category were constructed previously, each initial cluster center is created as the mean of its members.

2) Pseudo-code for the K-means algorithm

```
Given:
        a set of remaining contexts : $X = \{\vec{x}_1,\ldots,\vec{x}_n\}$
        a distance measure : cosine distance metric
        a function for computing the mean $\mu$ : $\vec{\mu} = (1/|c_j|)\sum_{\vec{x}\in c_j}\vec{x}$
        a set of centroid-contexts : $CC_{c_j} = \{\vec{s}_1,\ldots,\vec{s}_n\}$

Compute $k$ initial centers $\vec{f}_1,\ldots,\vec{f}_k$ using each centroid-contexts, $CC_{c_j}$
Main Loop:
While stopping criterion is not true do
    For all clusters $c_j$ do
        $c_j = \{\vec{x}_i \mid \forall\vec{f}_k, d(\vec{x}_i,\vec{f}_j) \geq d(\vec{x}_i,\vec{f}_k)\}$
    For all means $\vec{f}_j$ do
        $\vec{f}_j = u(c_j)$
End
```

3) Feature selection for the K-means algorithm

Since the K-means algorithm must calculate similarity between all contexts and the center of each cluster at all iterations, it requires too much processing time. The processing time depends on the number of features and contexts. Hence, we limit the number of features by a standard feature selection procedure. Since we do not hold labeled data, we cannot use the χ^2 statistics method in this case. We thus employ the Mutual Information (MI) method for a feature selection procedure as follows [45]:

$$I(w;D) \equiv p(w)\sum_{d\in D} p(d \mid w)\log\frac{p(d \mid w)}{p(d)} \tag{3.26}$$

where *w* is a content word, $D = \{d_1,..., d_n\}$ is a set of documents in unlabeled data.

After we sort all words by a calculated mutual information score, we select the top words with the highest contribution to the mutual information about the documents. Through an experiment using a validation set, we determined the number of features as 4,000 (see Figure 3.5).

3.3.3 Learning a Naive Bayes Classifier Using Context-clusters

Finally, we obtain labeled training data: context-clusters. Using them, we can build a Naive Bayes classifier. Since training data are labeled as the context unit, we employ a Naive Bayes classifier because it can be built by estimating the word probability in not a document but a category. That is, the Naive Bayes classifier does not require labeled data with a unit of documents unlike other classifiers.

Here, we use the Naive Bayes classifier with minor modifications based on Kullback-Leibler Divergence. This method makes the same classifications as the original Naive Bayes classifier, but produces classification scores that are less extreme. It then reflects better uncertainty than those produced by Naive Bayes [19]. More precisely, we classify a document d_i according to the following formula:

$$\begin{aligned} P(c_j \mid d_i;\hat{\theta}) &= \frac{P(c_j \mid \hat{\theta})P(d_i \mid c_j;\hat{\theta})}{P(d_i \mid \hat{\theta})} \approx P(c_j \mid \hat{\theta})\prod_{t=1}^{|V|} P(w_t \mid c_j;\hat{\theta})^{N(w,d_i)} \\ &\propto \frac{\log P(c_j;\hat{\theta})}{n} + \sum_{t=1}^{|V|} P(w_t \mid d_i;\hat{\theta}) \log\left(\frac{P(w_t \mid c_j;\hat{\theta})}{P(w_t \mid d_i;\hat{\theta})}\right) \end{aligned} \tag{3.27}$$

where i) *n* is the number of words in document d_i, ii) w_t is the *t*-th word in the vocabulary, iii) $N(w_t,d_i)$ is the frequency of word w_t in document d_i.

If the task is to classify a test document d_i into a single category, then the category with the highest posterior probability, $\arg\max_j P(c_j \mid d_i; \hat{\theta})$, is selected.

To estimate the parameter for a word given a category, $\hat{\theta}_{w_t|c_j}$, the Laplace smoothing is used as follows:

$$\hat{\theta}_{w_t|c_j} \equiv P(w_t \mid c_j; \hat{\theta}) = \frac{1 + N(w_t, G_{c_j})}{|V| + \sum_{t=1}^{|V|} N(w_t, G_{c_j})} \tag{3.28}$$

where $N(w_t, G_{c_j})$ is the count of the number of times word w_t occurs in the context-cluster (G_{c_j}) of category c_j. The category probabilities, $\hat{\theta}_{c_j}$, are estimated in the same manner as follows:

$$\hat{\theta}_{c_j} \equiv P(c_j \mid \hat{\theta}) = \frac{1 + |G_{c_j}|}{|C| + \sum_{c_i} |G_{c_i}|} \tag{3.29}$$

where $|G_{c_j}|$ is the number of contexts in the context-cluster of category c_j and $|C|$ is the number of categories.

3.4 Empirical Evaluation

This section provides empirical evidence of the proposed method. In our experiments, results are based on three different kinds of data sets: UseNet newsgroups (Newsgroups), web pages (WebKB), and newswire articles (Reuters).

3.4.1 Data Sets and Experimental Setting

The **Newsgroups** data set, collected by Ken Lang, contains about 20,000 documents evenly divided among 20 UseNet discussion groups [61][62][63]. Many of the categories fall into confusable clusters; for example, five of them are comp.* discussion groups, and three of them discuss about religion. However, we use only 16 categories after removing 4 categories: 3 miscellaneous categories (*talk.politics.misc, talk.religion.misc, and comp.os.ms-windows.misc*) and 1 duplicate meaning category (*comp.sys.ibm.pc.hardware*). We regard these as unsuitable categories to be classified in our method because their title words have duplicate meaning with other category or comprehensive meaning. In our experiments, 3,200 documents (20%) are used for test data and the remaining 12,800 documents (80%) for training data. For fair evaluation, we use the five-fold cross-validation method. That is, each data set is split into five subsets, and each subset is used once as test data in a particular run while the remaining subsets are used as training data for that run. The split into training and test sets for each run is the same for all classifiers. Therefore, all results of experiments using this data set are averages of five runs. After removing words that occur only once or on a stop word list, the average vocabulary from five training data has 43,249 words (no stemming).

The second data set comes from the **WebKB** project at CMU [19]. This data set contains web pages gathered from university computer science departments. The pages are divided into seven categories: *course, faculty, project, student, department, staff, and other*. As used in other studies [2][12], we exploit the four most populous entity-representing categories: *course, faculty, project, and student*. The resulting data set consists of 4,198 pages. It is an uneven data set; the largest category has 1,641 pages and the smallest one has 503 pages. Using the same method as in the Newsgroups data set, the five-fold cross-validation method is used and the resulting average vocabulary from five training data has 18,742 words.

The **Reuters 21578** Distribution 1.0 data set consists of 12,902 articles and 90 topic categories from the Reuters newswire. Following several other studies in [7][12][64], we build binary classifiers for each of the ten most populous categories to identify the news topic. Since the documents in this data set can have multiple category labels, each category is traditionally evaluated with a binary classifier. To split train/test data, we follow a standard ‘ModApte’ split. The standard ‘ModApte’ train/test split divides the articles by time, such that the later 3,299 documents form the test set and the earlier 9,603 are available for training. We use all the words inside the title and body, and we use a stop word list and no stemming. The vocabulary from training data has 12,001 words.

About 25% of documents from training data of each data set are selected for a validation set. After all parameter values of our experiments are set from the validation set, we evaluate our method using these parameter values.

3.4.2 Performance Measures

We follow the standard definition of recall, precision, and F_1 measure as performance measures as follows [7][50]:

$$\text{Recall}\,(R) = \frac{\#\,\text{of correct positive predictions}}{\#\,\text{of positive examples}} \tag{3.30}$$

$$\text{Precision}\,(P) = \frac{\#\,\text{of correct positive predictions}}{\#\,\text{of positive predictions}} \tag{3.31}$$

$$F_1\ measure = \frac{2RP}{R+P} \tag{3.32}$$

The F_1 measure combines recall and precision with an equal weight. The recall, precision, and F_1 measure can be first computed for individual categories, and then averaged over categories as a global measure of the average performance over all categories; this way of averaging is called *macro-averaging*. An alternative way, *micro-averaging*, is to count the decisions for all the categories in a joint pool and computes the global recall, precision, and F_1 values for that global pool [50].

Results on Reuters are reported as precision-recall breakeven points, which is a standard information retrieval measure for binary classification; Given a ranking of documents, the precision-recall breakeven point is the value at which precision and recall are equal [7][50].

Since a document in the Newsgroups and WebKB data sets has a single category label, micro-averaged accuracy, precision, recall, and F_1 scores are all equal. Hence, all of the above measures can be used interchangeably in micro-averaged results.

3.4.3 Empirical Evaluation

We test our method through the following steps. First, using the validation sets of each data set, we set the number of keywords. We then set a dropping threshold value in the similarity measure algorithm and the number of feature in K-means algorithm from the validation set of the Newsgroups data set. Using the resulting parameter values, we conduct experiments and compare our method with a supervised learning method using the Naive Bayes classifier.

A. Setting Parameters Using Validation Set

Here, we do experiments for setting parameters using the validation sets of each data set. We first show the changes of performance according to the number of keywords and observe the performance at each dropping threshold value of the similarity measure algorithm. Then we set the number of features for the K-means algorithm.

(1) Setting the number of keywords

First of all, we must determine the number of keywords in our method. The number of keywords in our experiment is limited by top *n*-th keyword from the ordered list of each category. Figure 3.3 displays the performance at different number of keywords (from 0 to 20) in each data set. Here, keywords means words from keyword lists created in Section 3.3.2 and the title word must be used. If we use zero keywords, it means that we use only the title word.

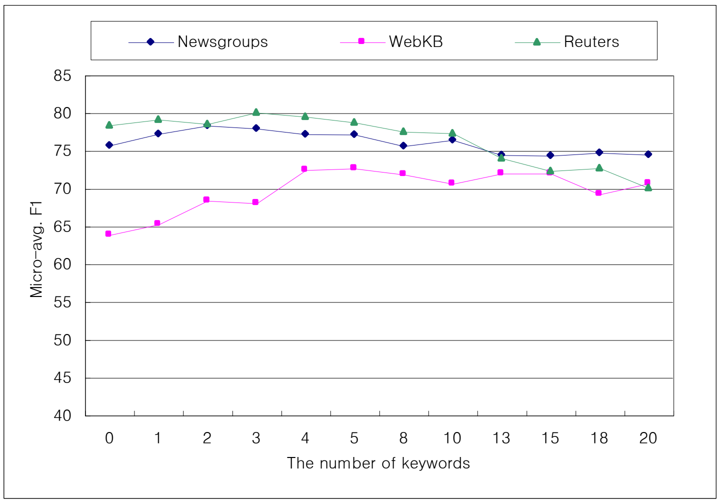


Figure 3. 3 The comparisons of the performance according to the number of keywords

As shown in Figure 3.3, we find that the number of keywords depends on each data set; we obtained the best performance at 2 keywords in the Newsgroups data set, at 5 keywords in the WebKB data set, and at 3 keywords in the Reuters data set. As a result, we set the number of keywords to 2 in the Newsgroups data set, 5 in the WebKB data set, and 3 in the Reuters data set. We generally recommend the number of keywords to be from 2 to 5.

(2) Setting the dropping threshold value in a similarity measure algorithm

For removing the meaningless remaining contexts, we set a threshold value such as Formula 3.23 and drop the remaining contexts with a maximum similarity below the dropping threshold value. The Figure 3.4 shows the performance at each dropping threshold value in the Newsgroups validation set.

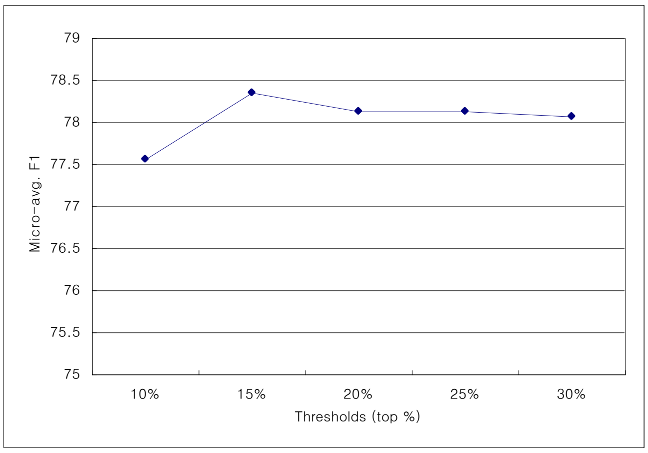


Figure 3. 4 The performance in each dropping threshold value

This graph verifies that our method obtains the best performance in top 15% as the dropping value. Therefore, we use the top 15% as the dropping value in all the data sets.

(3) Setting the number of features in the K-means algorithm

To reduce high processing time of the K-means algorithm, the number of features is limited by the mutual information method such as Formula 3.26. Our experiment is done from 1,000 to 10,000 and the results are shown as Figure 3.5.

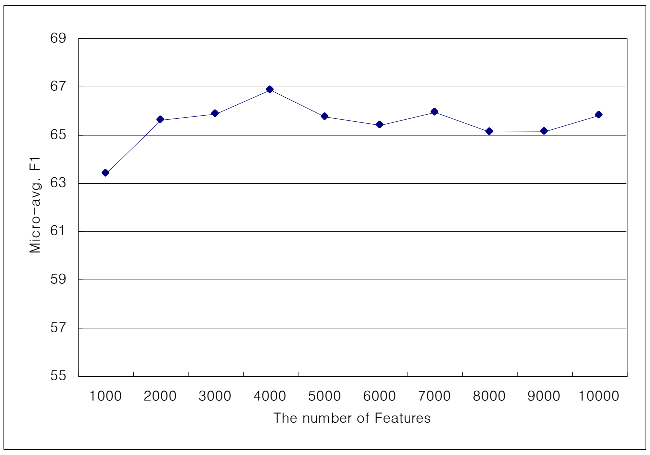


Figure 3. 5 The performance according to the number of features in the K-means algorithm

As shown in Figure 3.5, we use 4,000 features in the following experiments.

B. Comparison of the Remaining Context Assignment Algorithms

We determined all parameter values for our experiments in previous section. Now we start our main experiments. Before going into main experiments, we compare two kinds of remaining context assignment algorithms: the similarity measure algorithm and the K-means algorithm. Figure 3.6 shows the performance curve of each assignment algorithm by using the Newsgroups validation set. Since the similarity measure algorithm obtains the better performance over all intervals in Figure 3.6, our method in the following experiments is based on the similarity measure algorithm.

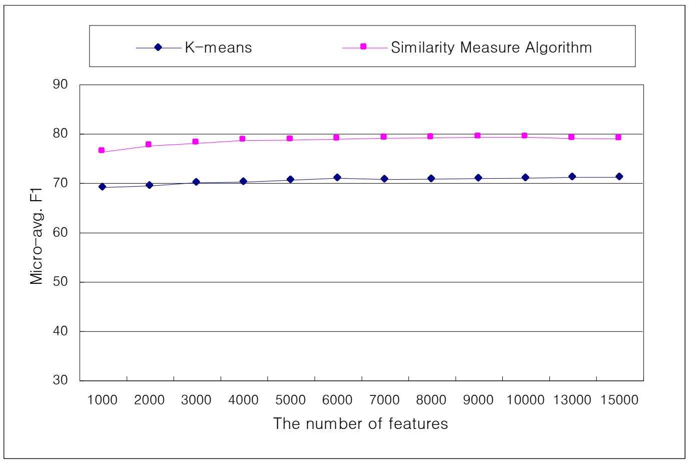
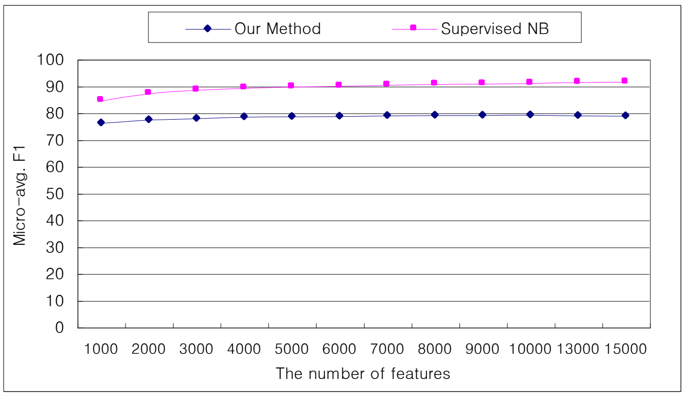


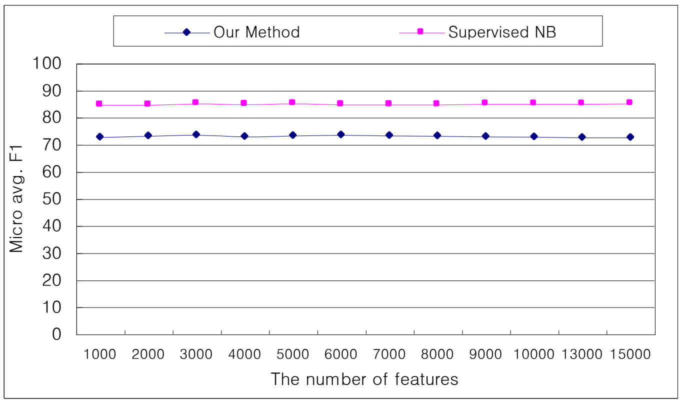
Figure 3. 6 The performance curves for comparison of context clustering methods using the Newsgroups validation set

C. Results in the Newsgroups, WebKB, and Reuters Data Sets

In main experiments, we use all three data sets (Newsgroups, WebKB, and Reuters) and employ a supervised Naive Bayes classifier for comparing our method with the supervised method; the supervised Naive Bayes classifier learns from human-labeled documents. Figure 3.7 and Table 3.2 report the results from three data sets. In Figure 3.7, the horizontal axes indicate the number of features, vocabulary size, and the vertical axes indicate the average micro-average F_1 scores on test sets in five-fold validation. Performance in the Newsgroups data set increases with the number of features but performance in the WebKB data set is similar regardless of the number of features. In the Reuters data set, we use train data and test data by standard ‘ModApte’ split for evaluation. Table 3.2 presents precision-recall breakeven points in each category showing performance of binary classifiers on the Reuters data set with our method and the supervised Naive Bayes classifier.



(a) The Newsgroups data set



(b) The WebKB data set

Figure 3. 7 The comparison of our method and the supervised Naive Bayes classifier

Table 3. 2 The precision-recall breakeven points in the Reuters data set

Category	Our Method	Supervised NB
acq	94.01	96.24
corn	62.5	66.07
crude	80.42	89.41
earn	96.13	97.42
grain	74.49	92.61
interest	77.86	77.09
money-fx	79.32	78.21
ship	77.52	85.39
trade	80.5	81.35
wheat	61.97	67.6
micro-avg.	*88.62*	*91.64*

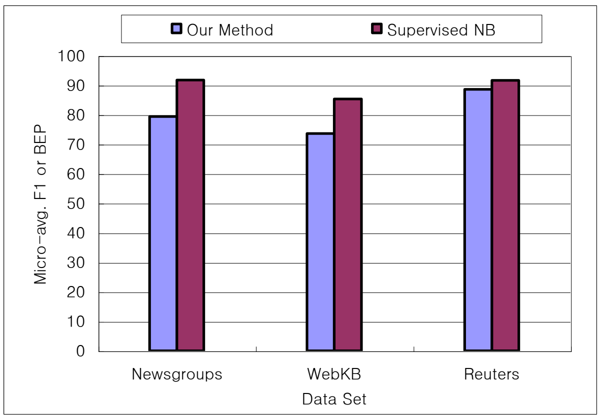


Figure 3. 8 The performance differences of the best micro-avg. F_1 scores or precision-recall break-even points in three data sets: our method vs. supervised NB

Table 3. 3 The performance differences of the best micro-avg. F_1 scores or precision-recall break- even points in three data sets: our method vs. supervised NB

Data Set	Our method	Supervised NB	Difference
Newsgroups	79.36	91.72	*-12.36*
WebKB	73.63	85.29	*-11.66*
Reuters	88.62	91.64	*-3.02*

As shown in Table 3.3, we obtained a 79.34% micro-average F_1 score in the Newsgroups data set, a 73.63% micro-average F_1 score in the WebKB data set, and an 88.62% micro-average precision-recall breakeven point in the Reuters data set. The differences between our method and the supervised Naive Bayes classifier in each data set are 12.36% in the Newsgroups data set, 11.66% in the WebKB data set, and 3.02% in the Reuters data set. Especially, the result of Reuters reached 3.02% close to that of the supervised Naive Bayes classifier. Since we use only unlabeled data and title words, the performance of our method is much more significant.

3.5 Discussion

From our experimental results, our method in the Reuters data set almost achieved comparable performance with the supervised method. As previously noted in [61], categories like wheat and corn are known for a strong correspondence between a small set of words (like our title words and keywords) and the categories, while categories like acq are known for more complex characteristics. Since the categories with narrow definitions attain best classification with small vocabularies, we can achieve good performance in the Reuters data set with our method which depends on title words.

In the Newsgroups and WebKB data sets, we could not attain comparable performance with the supervised method. In fact, the categories of these data sets are somewhat confusable. In the Newsgroups data set, many of the categories fall into confusable clusters: for example, five of them are comp.* discussion groups, and three of them discuss religion. In the WebKB data set, meaningful words of each category also have high frequency in other categories. Worst of all, even title words (course, professor, faculty, project) have a confusing usage. We think these factors contributed to a comparatively poor performance of our method.

To improve our method, we propose to use the documents automatically labeled from our method as training data. But a problem is that the machine-labeled data contains many incorrectly labeled data. Thus, we need a robust classifier from noisy data. Now we propose a new classifier, TCFP. This classifier is robust from noisy data and it has fast execution speed. The next chapter presents it in detail.

Chapter 4

The TCFP Classifier for Learning with Machine-labeled Data

4.1 Introduction

In Chapter 3, we can classify the original collected documents using the bootstrapping method and the generative model, Naive Bayes classifier, with only title words and unlabeled documents. We can finally obtain a labeled training data of a document unit: *machine-labeled document data*. Therefore, we can learn other text classifiers (TCFP, Rocchio, k-NN, and SVMs) using the machine-labeled document data in a supervised learning manner. This is an EM-like method. In a previous work, Nigam verified that EM can hurt accuracy when labeled data are sparse such as in the WebKB and Reuters data sets [12]. Moreover, in our experiments, EM also showed bad performance in the WebKB and Reuters data sets. Therefore, we do not employ the EM algorithm and we learn classifiers using the machine-labeled data generated from our method.

However, when the machine-labeled document data is used as training data, a problem comes from many incorrectly labeled data: *noisy data*. Since the machine-labeled data is created by our method, they generally include much more incorrectly labeled data than the human-labeled data. Hence, a classifier with robustness from noisy data is more useful in this application. We here introduce a new classifier with robustness from noisy data; we call this classifier *Text Categorization using Feature Projections (TCFP)*. Furthermore, TCFP is a fast classifier in execution and it is a very simple algorithm to learn and build.

The main idea of the TCFP classifier starts from the Nearest Neighbor algorithm [5][65]. In particular, the *k*-Nearest Neighbor (*k*-NN) classifier in text categorization is one of the state-of-the-art methods including the Support Vector Machines (SVMs) and Boosting algorithms [1]. Since the Nearest Neighbor algorithm is much simpler than other algorithms, the *k*-NN classifier is intuitive and easy to understand, and it learns quickly. But main weak points of *k*-NN are that its running time is too much slow and its performance degrades rapidly with the introduction of noisy data and irrelevant features. The reason is that its main computation is the on-line scoring of all training documents to find the *k* nearest neighbors of a test document. In order to reduce these problems in on-line ranking, a number of techniques have been studied. Techniques such as the instance pruning technique [66] and feature projections [67] are well known.

The instance pruning technique is one of the most straightforward ways to accelerate classification speed in a nearest neighbor system. It removes instances from the training data and thus reduces time and storage requirements. It also reduces the sensitivity of the system to noise. So far a large number of such pruning techniques have been proposed. To name a few, there are the Condensed Nearest Neighbor Rule [68], IB2 and IB3 [69], the Typical Instance Based Learning [70], and the Reduction Techniques (RT1-RT3) [71]. These pruning techniques and others are surveyed in depth by Wilson et al. [66]. They then develop several new pruning techniques such as DROP1-DROP5. Of these, DROP4 shows the best performance.

Another trial to overcome the problem exists in *feature projections*. Akkus and Guvenir present a new approach to classification based on feature projections [67]. They call their resulting algorithm *k-Nearest Neighbor on Feature Projections (k-NNFP)*. In this approach, the classification knowledge is represented as sets of projections of training data on each feature dimension. The classification of a test example is based on the voting by the *k* nearest neighbors of each feature of the test example. The resulting system allows the classification to be much faster than that of *k*-NN, and its performance is comparable with *k*-NN.

In this chapter, we present a particular implementation of text categorization using feature projections. When we apply the feature projection technique to text categorization, we can find several problems caused by the special properties of text categorization. We describe these problems in detail and then we propose a new approach to solve them. The proposed classifier, TCFP, shows better performance than *k*-NN. It is much faster than *k*-NN and it also has the advantage of robustness from noisy data.

The rest of this chapter is organized as follows. Section 4.2 simply presents *k*-NN, the DROP4 pruning algorithm, and the *k*-NNFP algorithm. Section 4.3 explains a new approach using feature projections in detail. Section 4.4 describes other classifiers used in our experiments. In Section 4.5, we discuss empirical results in our experiments and an analysis for strong points of the new proposed classifier. The final section presents conclusions.

4.2 *k*-NN, the DROP4 Pruning Algorithm, and the *k*-NNFP Algorithm

In this section, we simply describe *k*-NN, the DROP4 pruning algorithm, and the *k*-NNFP algorithm.

4.2.1 The *k*-Nearest Neighbor (*k*-NN) Algorithm

As an instance-based classification method, *k*-NN has been known as an effective approach to a broad range of pattern recognition and text classification problems [65][72]. In the *k*-NN algorithm, a new input instance (or example) should belong to the same category as its *k* nearest neighbors in the training data set. After all the training data is stored in memory, a new input instance is classified with the category of *k* nearest neighbors among all stored training instances.

For the distance measure and the document representation, we use a conventional vector space model; each document is represented as a vector of term weights and similarity between two documents is measured by the cosine value of the angle between the corresponding vectors.

Let a document *d* with *n* terms (*t*) be represented as the feature vector:

$$\vec{d} = < w(t_1, \vec{d}\,), w(t_2, \vec{d}\,), \ldots, w(t_n, \vec{d}\,) > \tag{4.1}$$

We compute the weight vectors for each document using one of conventional TF-IDF schemes [73]. The weight of term *t* in document *d* is calculated as follows:

$$w(t,\vec{d}) = \frac{(1+\log tf(t,\vec{d}))\times \log(N/n_t)}{\left\|\vec{d}\right\|} \tag{4.2}$$

where

i) $w(t,\vec{d})$ is the weight of term t in document d,

ii) $tf(t,\vec{d})$ is the within-document Term Frequency (TF),

iii) $\log(N/n_t)$ is the Inverted Document Frequency (IDF),

iv) N is the number of documents in the training set,

v) n_t is the number of training documents in which t occurs,

vi) $\left\|\vec{d}\right\| = \sqrt{\sum_{t\in\vec{d}} w(t,\vec{d})^2}$ is the 2-norm of vector $\vec{d}$.

Given an arbitrary test document d, the k-NN classifier assigns a relevance score to each candidate category c_j using the following formula:

$$s(c_j,\vec{d}) = \sum_{\vec{d}'\in R_k(\vec{d})\cap D_j} \cos(\vec{d}',\vec{d}) \tag{4.3}$$

where $R_k(\vec{d})$ denotes a set of the k nearest neighbors of document d and D_j is a set of training documents in category c_j.

4.2.2 The *DROP4* Pruning Algorithm

This section presents an instance pruning algorithm called the *Decremental Reduction Optimization Procedure 4 (DROP4)* [66]. The procedure of this algorithm is decremental, meaning that it begins with the entire training set and then removes instances that seem to be unnecessary.

Before the DROP4 algorithm is described, some notation is introduced here. A training set T consists of n instances (i). Each instance i has k nearest neighbors. Each instance *i* also has a nearest *enemy* which is the nearest instance *e* with a different category from *i*. Those instances, which have *i* as one of their *k* nearest neighbors, are called *associates* of *i*.

DROP 4 uses the following basic rule to decide if it is safe to remove an instance *i* from the instance set S (where *S*=*T* originally).

Remove instance i from S if at least as many of its associates in T
would be classified correctly without i.

The order of removal can be important to the success of a pruning algorithm. DROP4 initially sorts instances in *S* by the distance to their nearest *enemy*. And then the instances are checked for removal by beginning at the instance which is farthest from its nearest *enemy*. This tends to remove instances farthest from the decision boundary first, which increases the chance of retaining border points. However, noisy instances are also border points, so it is desirable to remove the noisy instances before any of the others so that the rest of the algorithm is not influenced heavily by the noisy instances. Therefore, DROP4 uses a noise-filtering pass before sorting the instances in *S*. For the noise-filtering pass, DROP4 removes each instance only if (1) it is misclassified by its *k* nearest neighbors and (2) it does not hurt the classification of its *associates*.

4.2.3 The *k*-Nearest Neighbor on Feature Projection (*k*-NNFP) Algorithm

The *k*-NNFP is a variant of the *k*-NN method. The main difference is that instances are projected on their features in the *n*-dimensional space (see Figure 4.1) and distance between two instances is calculated according to a single feature. The distance between two instances d_i and d_j with regard to *m*-th feature t_m is $dist_m(t_m(i), t_m(j))$ as follows:

$$dist_m(t_m(i), t_m(j)) = \left| w(t_m, \vec{d}_i) - w(t_m, \vec{d}_j) \right| \tag{4.4}$$

where $t_m(i)$ denotes the m-th feature t in an instance d_i and $w(t_m, \vec{d}_i)$ is the weight of term t_m in document d_i.

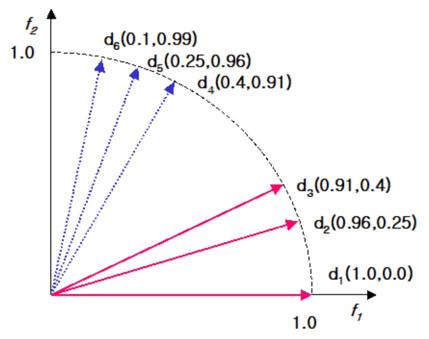
The classification on a feature is done according to votes of the k-nearest neighbors of that feature in a test instance; the category of a nearest neighbor for the vote follows that of document including the nearest neighbor. The final classification of the test instance is determined by a majority voting from individual classifications of each feature. If there are n features, this method returns $n \times k$ votes whereas the k-NN method returns k votes.

4.3 A New Approach of Text Categorization on Feature Projections

First of all, we show an example of feature projections in text categorization for easier understanding. We then enumerate the problems to be considered when the feature projection technique is applied to text categorization. Finally, we propose a new approach using feature projections to overcome these problems.

4.3.1 An Example of Feature Projections in Text Categorization

We give a simple example of the feature projections in text categorization. To simplify our description, we suppose that all documents have just two features (f_1 and f_2) and two categories (c_1 and c_2). The TF-IDF value by Formula 4.2 is used as the weight of a feautre. Each document is normalized as a unit vector and each category has three instances: $c_1 = \{d_1, d_2, d_3\}$ and $c_2 = \{d_4, d_5, d_6\}$. Figure 4.1 shows how document vectors in a conventional vector space are transformed into feature projections and stored on each feature dimension. The result of feature projections on a term (or feature) can be seen as a set of weights of documents for the term. On feature projections, the category of each element for a feature is set to the category of documents including the feature. Since a term with 0.0 weight is useless, the size of the set equals the Document Frequency (DF) value of the term.



Document representation in a conventional vector space

w: weight, **d**: document, **c**: category

w, d, c
1.0, d_1, c_1
0.96, d_2, c_1
0.91, d_3, c_1
0.4, d_4, c_2
0.25, d_5, c_2
0.1, d_6, c_2

feature projections on feature f_1

w, d, c
0.99, d_6, c_2
0.96, d_5, c_2
0.91, d_4, c_2
0.4, d_3, c_1
0.25, d_2, c_1

feature projections on feature f_2

Feature representation in feature projections

Figure 4. 1 Feature representation on feature projections in Text Categorization

4.3.2 Problems in Applying Feature Projections to Text Categorization

There are three problems: (1) the diversity of Document Frequency (DF) values of terms, (2) the property of using TF-IDF values as the weight of features, and (3) the lack of contextual information.

A. The Diversity of Document Frequency Values of Terms

Table 4.1 shows a distribution of the DF values of the terms in the Newsgroups data set. The numerical values of Table 4.1 are calculated from a training data set with 16,000 documents and 10,000 features chosen by feature selection. The *k* in the fourth column means the number of nearest neighbors selected in *k*-NNFP; the *k* in *k*-NNFP was set to 20 in our experiments.

Table 4. 1 A distribution of the DF values of the terms in the Newsgroups data set

Average DF	Maximum DF	Minimum DF	The number of features DF < *k* (20)
54.59	8,407	4	6,489

According to Table 4.1, more than a half of the features have DF values less than *k* (20). This result can be also explained by Zipf's law. The problem is that some features have DF values less than *k* while other features have DF values much greater than *k*. For features with the DF value less than *k*, all the elements of the feature projections on the feature could and should participate in voting. In this case, the number of elements chosen for voting is less than *k*. On the other hand, for features with the DF value more than *k*, only maximum *k* elements among the elements of the feature projections should be chosen for voting. Therefore, we need to normalize the voting ratio for each feature. As shown in Formula 4.5, we use a proportional voting method to normalize the voting ratio.

B. The Property of Using TF-IDF Values as the Weight of Features

The TF-IDF value of a term is its presumed value for identifying the content of a document [74]. Therefore, on feature projections, elements with a high TF-IDF value for a feature must become more useful classification criterions for the feature than any elements with low TF-IDF values. In order to apply this property to our algorithm, we use only elements with TF-IDF values above the average TF-IDF value for voting. The selected elements also participate in proportional voting with the same importance as the TF-IDF value of each element. The voting ratio of each category c_j in a feature t_m of a test document d is calculated by the following formula:

$$r(c_j, t_m) = \sum_{t_m(l) \in I_m} w(t_m, \vec{d}_l) \cdot y(c_j, t_m(l)) \Big/ \sum_{t_m(l) \in I_m} w(t_m, \vec{d}_l) \qquad (4.5)$$

In Formula 4.5, I_m denotes a set of elements selected for voting and $y(c_j, t_m(l)) \in \{0.1\}$ is a function; if the category for an element $t_m(l)$ is equal to c_j, the output value is 1. Otherwise, the output value is 0.

C. The Lack of Contextual Information

Since each feature votes separately on feature projections, contextual information is missing. Thus, we use co-occurrence frequency to apply contextual information to our algorithm.

To calculate a co-occurrence frequency value between two terms t_i and t_l, we count the number of documents that include both terms. It is separately calculated in each category of training data. Finally, the co-occurrence frequency value of the two terms is obtained by a maximum value among co-occurrence frequency values in each category as follows:

$$co(t_i, t_l) = \max_{c_j} \{co(t_i, t_l, c_j)\} \tag{4.6}$$

where $co(t_i, t_l)$ denotes a co-occurrence frequency value of t_i and t_l, and $co(t_i, t_l, c_j)$ denotes a co-occurrence frequency value of t_i and t_l in category c_j.

TF-IDF values of two terms t_i and t_j in a test document d are modified by reflecting the co-occurrence frequency value. That is, terms with a high co-occurrence frequency value and a low category frequency value have higher term weights as follows:

$$tw(t_i, \vec{d}) = w(t_i, \vec{d}) \cdot \left(1 + \left(\frac{1}{\log(cf + 1)}\right) \cdot \left(\frac{\log(co(t_i, t_j) + 1)}{\log(maxco(t_k, t_l) + 1)}\right)\right) \tag{4.7}$$

where i) *tw(t_i,d)* denotes a modified term weight assigned to term t_i, ii) *cf* denotes the category frequency that is the number of categories in which t_i and t_j co-occur, and iii) $maxco(t_k, t_l)$ is the maximum value among all co-occurrence frequency values.

D. The Final Voting Score Reflecting the Improvements and the Information of Features

Through a feature selection process, we can measure how much information each feature contributes to our knowledge for correct classification; we employ the χ^2

statistics for feature selection in this paper. Moreover, since each feature in feature projections separately participates in voting, we can use the information of features (a calculated χ^2 statistics value) as an important weight in voting. As a result, in order to apply the improvements (Formulae 4.5 and 4.7) and the information of features to our algorithm, we calculate the voting score of each category c_j in the *m*-th feature t_m of a test document *d* as the following formula:

$$vs(c_j, t_m) = tw(t_m, \vec{d}\ ) \cdot r(c_j, t_m) \cdot \log(1 + \chi^2(t_m)) \qquad (4.8)$$

where $\chi^2(t_m)$ denotes the calculated χ^2 statistics value of t_m .

So far, we have explained how to calculate the voting score of each feature in each category. The voting score of a test document can be gained by summing up the voting scores of each feature in the test document. However, when the number of training data is unevenly distributed, a problem can occur. The cause of the problem is that a larger category has more voting candidates than a smaller category. To normalize the number of voting candidates in each category, we calculate the normalization factor (*nf(c$_j$)*) and apply it to the major voting score of each category as follows:

$$nf(c_j) = \max_{c_i}\{|c_i|\} \Big/ |c_j| \qquad (4.9)$$

$$vote[c_j] = vote[c_j] \cdot nf(c_j) \qquad (4.10)$$

where $|c_j|$ denotes the number of training documents in category c_j.

In Formula 4.9, the normalization factor can be adjusted according to the degree of skewness of the number of training data.

4.3.3 A New Text Categorization Algorithm Using Feature Projections

A new text categorization algorithm using feature projections, **TCFP**, is described in the following:

```
Given: test document: $\vec{d}$ =<$t_1,t_2,\ldots,t_n$>, a category set: $C$={$c_1,c_2,\ldots,c_m$}

Begin
    For each category $c_j$
        vote[$c_j$] =0
    For each feature $t_i$
        $tw(t_i,d)$ is calculated by Formula 4.7

    /* majority voting*/
    For each feature $t_i$
        For each category $c_j$
            vote[$c_j$]=vote[$c_j$]+$vs(c_j,t_i)$ by Formula 4.8

    /*normalize majority voting*/
    For each category $c_j$
        vote[$c_j$]=vote[$c_j$]*$nf(c_j)$ by Formula 4.9

    prediction = $\arg\max_{c_j} vote[c_j]$
    Return prediction
End
```

In the training phase, our algorithm needs only simple process; the training documents are projected on each of their features, and numerical values for the proportional voting (Formula 4.5) and co-occurrence frequency values (Formula 4.6) are calculated.

4.4 Other Conventional Classifiers Used in Our Experiments

To compare TCFP to other conventional classifiers except *k*-NN, we implement Naive Bayes, Rocchio, and SVMs. In this section, these classifiers are briefly described: for the Naive Bayes classifier, we use the same classifier as described in Chapter 3. For Rocchio and SVMs, we use the same TF-IDF scheme as Formulae 4.1 and 4.2.

4.4.1 Rocchio

Rocchio is an effective method using relevance judgments for query expansion in information retrieval and filtering [4][73]. Applied to text categorization, it uses a vector to represent each category and document, and computes their similarity using the cosine value of these two vectors. Then a test document is assigned to a category with the highest cosine score. The vector representation for a category, called a prototype or centroid, is constructed by combining document vectors into a prototype vector $\vec{c}_j$ for each category c_j. First, both the document vectors of the positive examples and those of the negative examples are summed up. The prototype vector is then calculated as a weighted difference of each as follows:

$$\vec{c}_j = \alpha \frac{1}{|c_j|} \sum_{\vec{d} \in c_j} \vec{d} - \beta \frac{1}{|N - c_j|} \sum_{\vec{d} \in N - c_j} \vec{d} \tag{4.11}$$

where N is the number of documents in the training data set, and α, β are parameters that adjust the relative impact of positive and negative training examples.

4.4.2 Support Vector Machines

Support Vector Machines (SVMs) is proposed by Vapnik for solving two-class pattern recognition problems [41]. The SVM problem is to find the decision surface that maximizes the margin between positive examples and negative examples in a training data.

The decision surface by SVM for linearly separable space is a hyperplane as follows:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{4.12}$$

$\vec{x}$ is an arbitrary test data vector, and the vector $\vec{w}$ and the constant *b* are learned from a training data set.

The SVM problem can be solved using quadratic programming techniques [41]. The algorithms for solving linearly separable cases can be extended for solving linearly non-separable cases by mapping the original data vectors to a space of higher dimensions.

Since the SVM is a binary classifier, we must extend it to a multi-class classifier. There are several methods of multi-class classifiers for SVMs [75]. We employ the One-Against-the-Rest method among them. This method requires *k* binary classifiers $f_{c_j}(x)\ (1 \le j \le k)$ to be constructed for all categories. In training for category c_j, all data of category c_j are used as positive examples and all data of the other categories as negative examples. Given a test example *x*, its category *c* is determined by the classifier that gives the largest discriminating function value as follows:

$$c = \arg\max_{c_j} f_{c_j}(x) \tag{4.13}$$

Joachims implemented an efficient SVMs toolkit, *SVM*^{light} [7]. This toolkit is used in our experiments.

4.5 Empirical Evaluation

In this section, we provide empirical evidence that TCFP is a most useful classifier for our method in using machine-labeled training data. First of all, we verify the superiority of TCFP using the conventional human-labeled data in a supervised manner. And then we discuss the results from experiments of our method with the machine-labeled data. Results on our method show that TCFP achieved the best performance in all three data sets. For the conventional human-labeled data, we use three different test data sets as used in Chapter 3: UseNet newsgroups (Newsgroups), web pages (WebKB), and newswire articles (Reuters).

4.5.1 Data Sets and Experimental Setting

The **Newsgroups** data set, collected by Ken Lang, contains about 20,000 articles evenly divided among 20 UseNet discussion groups [61][62][63]. Contrary to experiments in Chapter 3, we use all 20 categories in this chapter. After removing words that occur only once or on a stop word list, the average vocabulary from five training data has 51,325 words (with no stemming).

The second data set comes from the **WebKB** project at CMU [19]. As used in

Chapter 3, we use the four most populous entity-representing categories: *course, faculty, project, and student*. The resulting data set consists of 4,198 pages with a vocabulary of 18,742 words. It is an uneven data set; the largest category has 1,641 pages and the smallest one has 503 pages.

The **Reuters 21578** Distribution 1.0 data set consists of 12,902 articles and 90 topic categories from the Reuters newswire. We build binary classifiers for each category to identify the news topic. Contrary to experiments in Chapter 3, we here construct all 90 binary classifiers for 90 topic categories. To split train/test data, we follow a standard 'ModApte' split. We use all the words inside the title and body. We use a stop word list and no stemming. The vocabulary from training data has 14,219 words.

For fair evaluation in the Newsgroups and WebKB data sets, we use the five-fold cross-validation method. Therefore, all results of the experiments are averages of five runs.

To compare TCFP with other algorithms for improving execution speed, we implement *k*-NNFP and *k*-NN with pruning. We use DROP4 as a pruning technique [66]. With DROP4, only about 26% of the original training documents in data sets is retained. The *k* in *k*-NNFP is set to 20 and the *k* in *k*-NN with pruning is set to 30. In addition, we implement other conventional classifiers: *k*-NN, Naive Bayes, Rocchio, and SVMs. The *k* in *k*-NN is set to 30, and α=16 and β=4 are used in the Rocchio classifier. For SVMs, we use the linear model offered by SVM^{light}.

As performance measures, we follow the standard definition of recall (r), precision (p), and F_1 measure, (2rp/(r+p)) [50]. Recall is the probability that a document belonging to any category is classified into this category and Precision is the probability that a document predicted to be in any category is classified into this category. The F_1 measure balances recall and precision in a way that gives them equal weight. Results on Reuters are reported as precision-recall breakeven points, which is a standard information retrieval measure for binary classification [50]. For evaluating performance average across categories, we use the *micro-averaging method* [50].

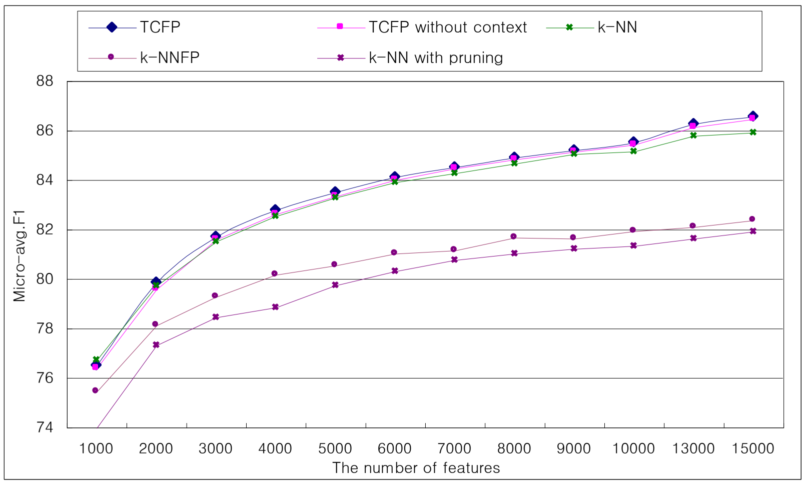
For feature selection, we employ the χ^2 statistics method [47].

4.5.2 Experimental Results in the Supervised Manner Using the Human-labeled Data Set

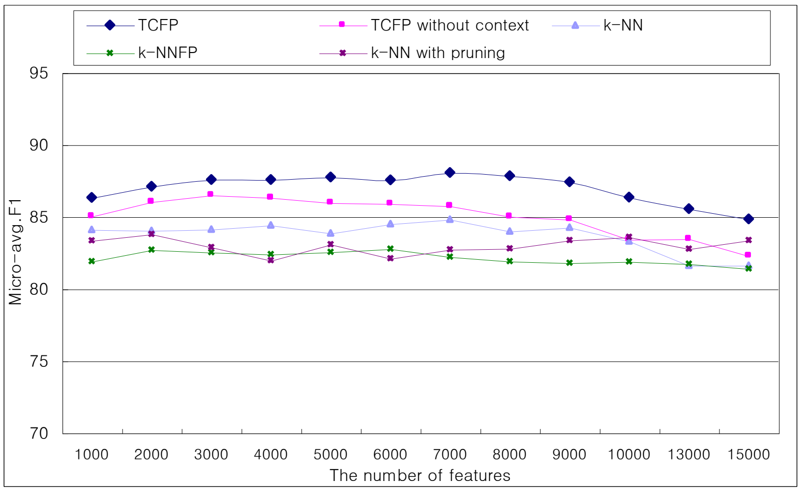
A. Comparing the Performances of TCFP, *k*-NN, *k*-NN with Pruning, and k-NNFP

Figure 4.2 and Table 4.2 show results from TCFP, *k*-NN, and other algorithms for compensating the drawbacks of the *k*-NN algorithm (*k*-NN with pruning and *k*-NNFP) in the Newsgroups and WebKB data sets. In addition, we add another type of TCFP to our experiment. This is *TCFP without contextual information*, which does not use Formula 4.7 for contextual information.

As a result, TCFP achieved the highest micro-average F_1 score on both test data sets. Also, *TCFP without contextual information* presented nearly the same performance as TCFP. Although, in all vocabulary sizes, *TCFP without contextual information* achieved a little lower performance than TCFP, it can also be a useful classifier due to its simplicity and fast execution speed (see Table 4.6).



(a) The Newsgroups data set



(b) The WebKB data set

Figure 4. 2 The classification performance according to the number of features

Table 4. 2 The best micro-average F_1 scores of each classifier

Data Set	TCFP	TCFP without context	*k*-NN	*k*-NNFP	*k*-NN with pruning
Newsgroups	***86.57***	86.46	***85.92***	82.37	81.91
WebKB	***88.07***	86.52	***84.82***	82.77	83.81

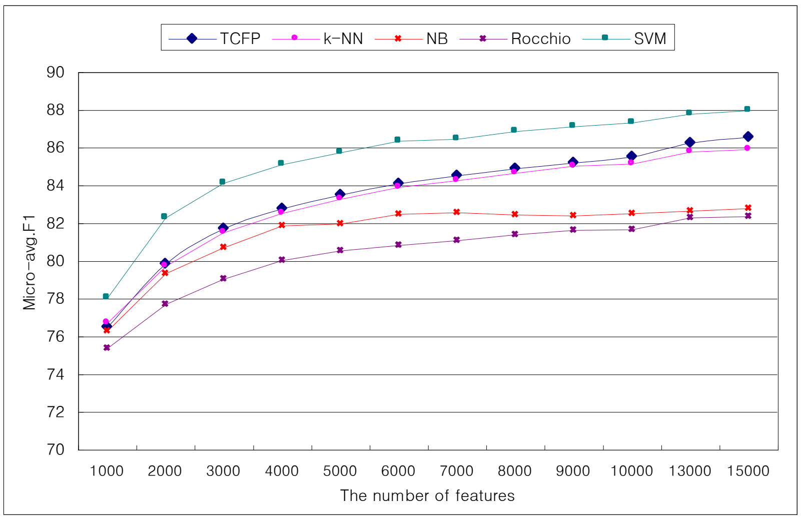
Since the Reuters data set can have documents with multiple category labels, we build binary classifiers for each category. Table 4.3 shows the precision-recall breakeven points on the ten most frequent Reuters categories and their micro-average performance. Results on the Reuters data set are also similar in performance to those of the previous two data sets. That is, TCFP also achieved the best micro-average precision-recall breakeven point in the Reuters data set. Note that, as shown in Table 4.2 and Table 4.3, *k*-NNFP and *k*-NN with pruning algorithms are inferior to TCFP although they achieved faster execution speed than *k*-NN (see Table 4.6).

Table 4. 3 Precision-recall breakeven points showing the performance of binary classifiers on the Reuters data set

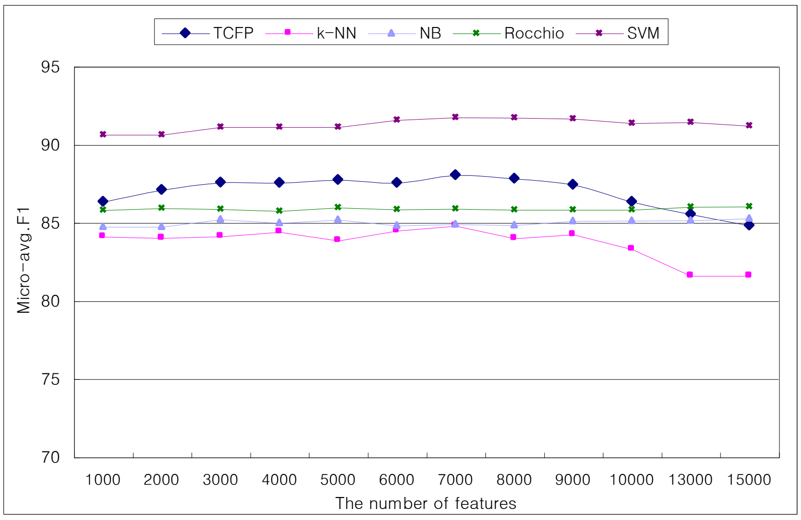
Category	TCFP	TCFP without context	*k*-NN	*k*-NNFP	*k*-NN with pruning
acq	94.29	93.6	94.57	90.54	93.32
corn	75	75	78.57	60.71	78.57
crude	84.65	79.36	82.01	82.53	80.42
earn	97.42	96.59	95.86	94.2	95.03
grain	84.56	84.56	80.53	78.52	78.52
interest	76.33	71.75	74.04	69.46	72.51
money-fx	72.06	72.06	76.53	73.18	74.86
ship	84.26	83.14	78.65	87.64	79.77
trade	72.03	71.18	79.66	63.55	78.81
wheat	77.46	76.05	64.78	67.6	70.42
micro-avg.	***90.01***	*88.8*	***88.93***	*86.26*	*88.23*

B. Comparing the Performance of TCFP and the Conventional Classifiers (*k*-NN, Naive Bayes, Rocchio, and SVMs)

For further evaluation, we implement other conventional classifiers: *k*-NN, Naive Bayes, Rocchio, and SVMs, and we compare them with TCFP. The classification performance of each classifier is shown in Figure 4.3, Table 4.4, and Table 4.5.



(a) The Newsgroups data set



(b) The WebKB data set

Figure 4. 3 The classification performance according to the number of features

Table 4. 4 The best micro-average F_1 scores of each classifier

Data Set	TCFP	*k*-NN	SVM	NB	Rocchio
Newsgroups	***86.57***	85.92	***87.97***	82.79	82.37
WebKB	***88.07***	84.82	***91.75***	85.29	86.05

Table 4. 5 Precision-recall breakeven points showing the performance of binary classifiers on the Reuters data set

Category	TCFP	*k*-NN	SVM	NB	Rocchio
acq	94.29	94.57	96.94	93.88	88.31
corn	75	78.57	91.07	64.28	66.07
crude	84.65	82.01	86.24	84.12	78.3
earn	97.42	95.86	98.62	96.41	96.13
grain	84.56	80.53	94.63	81.87	79.86
interest	76.33	74.04	78.62	74.04	73.28
money-fx	72.06	76.53	79.88	73.18	64.24
ship	84.26	78.65	87.64	82.02	80.89
trade	72.03	79.66	79.66	72.03	75.42
wheat	77.46	64.78	84.5	63.38	77.46
micro-avg.	***90.01***	*88.93*	***93.32***	*88.62*	*86.47*

The results show that TCFP is superior to *k*-NN, Naive Bayes, and Rocchio classifiers. But TCFP produced lower performance than SVMs, which has been reported as the classifier with the best performance in this literature. Although TCFP showed lower performance than SVMs, TCFP has several strong points in comparison with SVMs. Especially, TCFP is more robust from noisy data and it has faster execution speed. These are discussed in the following section in detail.

C. Analysis of Running Time Observation and Robustness from Noisy Data

TCFP overcomes the weaknesses of *k*-NN, including slow execution speed and sensitivity to noise training data, while it maintains the strong points of *k*-NN, including the simplicity of algorithm, fast learning process, and high performance. On experimental results, TCFP produced high performance even though it showed a little lower performance than SVMs. Especially, TCFP outperformed *k*-NN in all three data sets. Furthermore, since TCFP has a simple and fast learning process like *k*-NN, it can be a proper classifier for incremental learning. On the contrary, the training process of SVMs can be very complicated and time consuming, especially when dealing with noisy data. In addition, SVMs have some weaknesses such as slow running time and sensitivity to noisy data like *k*-NN. In order to verify the superiority of TCFP in execution speed and robustness from noisy data, we observed the running time in our experiments and conducted extra experiments for evaluating the robustness of each classifier from noisy data. The results are reported in the following subsections.

(1) Running time observation

Table 4.6 shows the running times in CPU seconds for each classifier on each data set. Note that we included only the testing phase for measuring the running time of each data set.

Table 4. 6 Average running time of each classifier on each data set

Data Set	TCFP without context	*k*-NNFP	Rocchio	TCFP	NB	SVM	*k*-NN with pruning	*k*-NN
Newsgroups	***0.68***	0.85	0.8	***1.25***	1.22	14.71	37.97	142.54
WebKB	***0.13***	0.23	0.14	***0.55***	0.17	2.72	4.91	15.25
Reuters	***2.65***	2.7	3.34	***2.89***	7.01	39.94	15.88	65.86

Since the computations depend on the vocabulary sizes, we calculated the above numerical values by averaging running times from 1,000 to 10,000 features. In Table 4.6, the running time of TCFP is similar to other faster classifiers: Rocchio and Naive Bayes. Especially, it is about one hundred times faster than that of *k*-NN on the Newsgroups data set, and it also has much faster execution speed than SVMs. Though the results on each data set depend on the number of training documents and categories, TCFP showed fast execution speed in all the data sets. Especially, *TCFP without contextual information* is the fastest classifier in all the data sets. The time complexity of *TCFP without contextual information* is *O(mc)*, where *m* is the number of unique words in a test document and *c* is the number of categories. That is, the classification of *TCFP without contextual information* requires a simple calculation in proportion to the number of unique terms in the test document. On the other hand, in *k*-NN, a search in the whole training space must be done for each test document; since the training documents in *k*-NN are represented as the inverted index files in practical, it does not exactly search the whole training space (more than half of the whole training documents). Furthermore, note that *TCFP without contextual information* showed higher or similar performance than *k*-NN in our experiments.

(2) Robustness from noisy data

We here analyze that TCFP is more robust from noisy data than *k*-NN and SVMs. For this experiment, we generate four data sets, increasing the number of noisy documents from 10% to 40% in the Newsgroups data set: these noisy documents are randomly chosen from each category and randomly assigned into other categories. The results of each classifier on each noisy data set are shown in Figure 4.4. These results are also obtained by a five-fold cross-validation method.

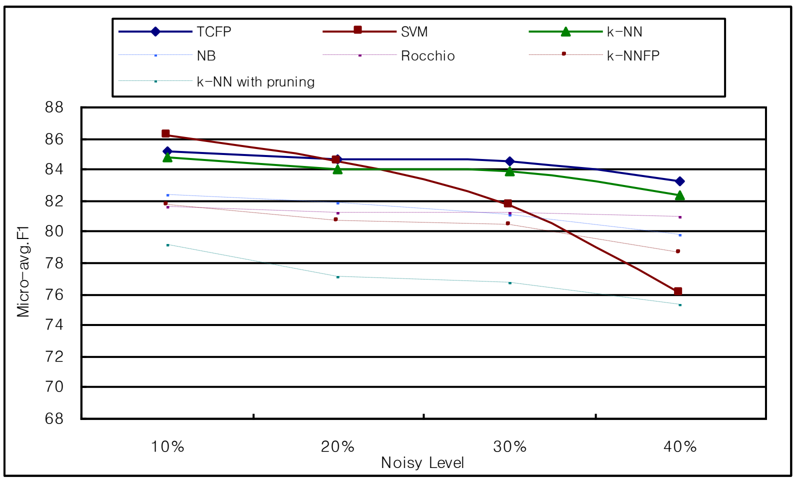


Figure 4. 4 The classification performance of each classifier on four noisy data sets (10%,20%,30%,40%)

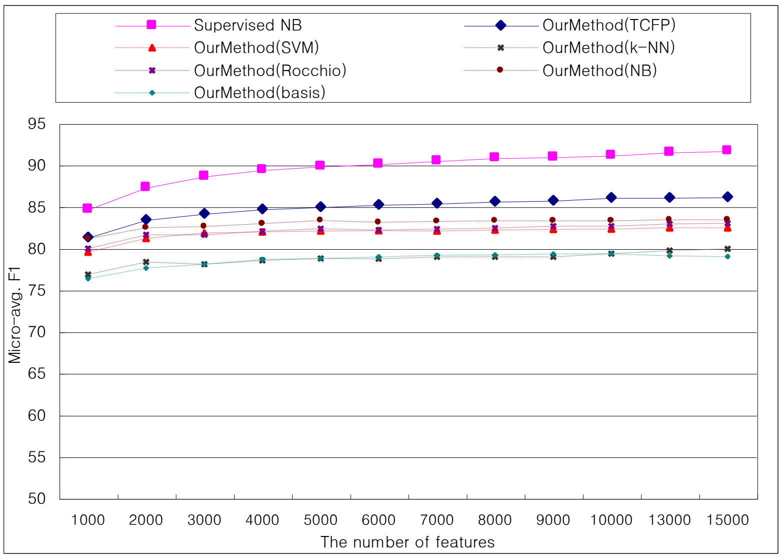
As shown in Figure 4.4, TCFP showed the best performance beginning at the 20% noisy data set, and the decreasing rate of performance of TCFP is less than that of *k*-NN and SVMs. Especially, we observed that the performance of SVMs degraded rapidly when the number of noisy documents increased. In the actual applications of text categorization, it is quite common that examples in the training collection contain some amount of noise due to various reasons such as wrong categories assigned by humans and missing appropriate features. As seen in the above results, the presence of noisy data will affect the performance of text categorization.

The robustness of TCFP is due to its voting mechanism. That is, the voting mechanism of TCFP, which depends on separate voting in each feature, reduces the negative effect of possible noisy data and irrelevant features in classification.

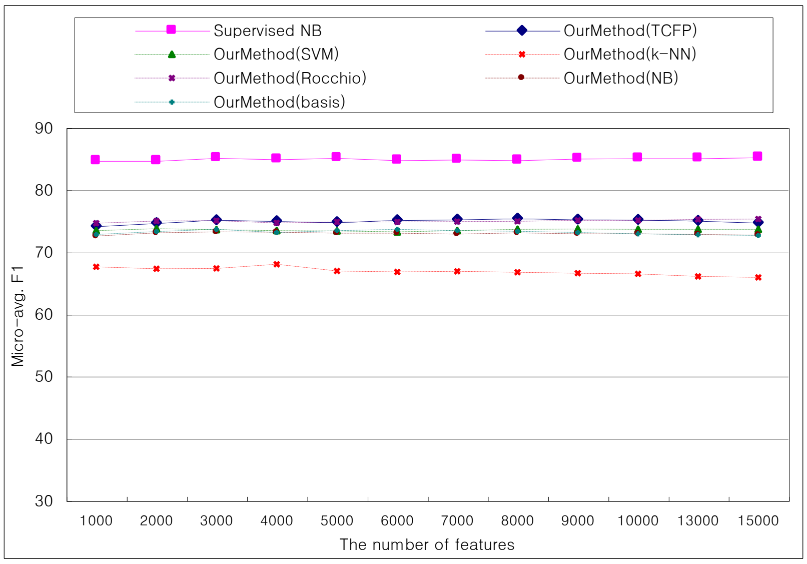
4.5.3 Experimental Results in Our Method Using the Machine-labeled Documents Data

In the above section, we verified that TCFP shows good performance on a data set with noise. This is also observed in our experiments using the machine-labeled data set. Here, we conduct experiments for improving our method by using TCFP and the machine-labeled training data from Chapter 3. All experiments of this section follow the same experimental setting and performance measures in Chapter 3. We also report the experimental results on three data sets of Chapter 3: the Newsgroups, WebKB, and Reuters data sets. These experiments employ four kinds of conventional classifiers to compare with TCFP: *k*-NN, Naive Bayes, Rocchio, and SVMs.

Here, the final results are also compared with those of the supervised Naive Bayes classifier. Figure 4.5 and Table 4.7 show the results from three data sets. For definition of notations, *OurMethod(basis)* denotes the Naive Bayes classifier using the machine-labeled contexts as training data and *OurMethod(NB)* denotes the Naive Bayes classifier using the machine-labeled documents, which are generated by *OurMethod(basis)*, as training data. The same manner is applied for the other classifiers. Results from all three data sets show that TCFP achieved the best performance in our method. This is another evidence for the superiority of TCFP, especially about robustness from noisy data. Note that, though most of the other classifiers showed lower or similar performance than that of basic method (*OurMethod(basis)*) on the WebKB and Reuters data sets, TCFP achieved the better performance in both of the data sets.



(a) The Newsgroups data set



(b) The WebKB data set

Figure 4. 5 The comparison of the performance in our methods using each classifier

Table 4. 7 The best micro-average F_1 scores of each classifier

Data Set	***OurMethod (basis)***	***OurMethod (NB)***	***OurMethod (Rocchio)***	***OurMethod (k-NN)***	***OurMethod (SVM)***	***OurMethod (TCFP)***	**Supervised NB**
Newsgroups	***79.36***	83.46	83	79.95	82.49	***86.19***	***91.72***
WebKB	***73.63***	73.22	75.28	68.04	73.74	***75.47***	***85.29***

Table 4. 8 The precision-recall breakeven points in the Reuters data set

Category	***OurMethod (basis)***	***OurMethod (NB)***	***OurMethod (Rocchio)***	***OurMethod (k-NN)***	***OurMethod (SVM)***	***OurMethod (TCFP)***	**Supervised NB**
acq	94.01	94.29	89.01	88.73	95.41	94.99	96.24
corn	62.5	55.35	64.28	51.78	53.57	64.28	66.07
crude	80.42	84.12	77.77	75.66	84.12	79.36	89.41
earn	96.13	96.5	96.13	97.33	96.87	96.96	97.42
grain	74.49	65.1	79.19	47.65	38.25	67.78	92.61
interest	77.86	77.86	68.7	77.09	82.44	80.91	77.09
money-fx	79.32	76.53	67.03	75.97	73.74	78.21	78.21
ship	77.52	80.89	77.52	79.77	78.65	82.02	85.39
trade	80.5	77.11	83.89	85.59	83.89	83.05	81.35
wheat	61.97	61.97	57.74	56.33	60.56	60.56	67.6
micro-avg.	***88.62***	*88.23*	*86.26*	*85.65*	*87.41*	***89.09***	***91.64***

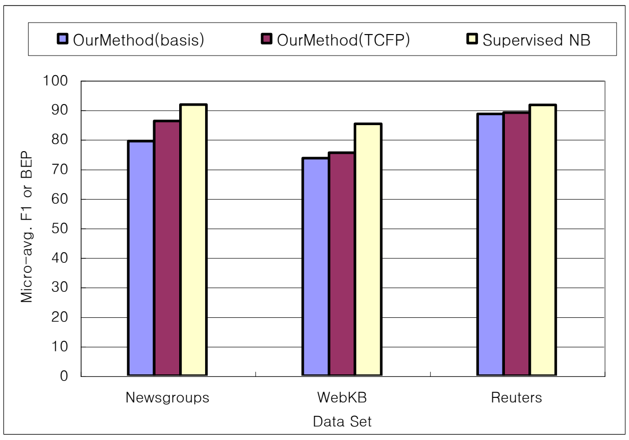


Figure 4. 6 The performance differences of the best micro-avg. F_1 scores or micro-avg. precision-recall breakeven points in three data sets: *OurMethod(basis)* vs. *OurMethod(TCFP)* vs. *Supervised NB*

Table 4. 9 The performance differences of the best micro-avg. F_1 scores or micro-avg. precision-recall breakeven points in three data sets: *OurMethod(basis)* vs. *OurMethod(TCFP)* vs. *Supervised NB*

Data Set	*OurMethod (basis)*	*OurMethod (TCFP)*	*OurMethod (basis) vs. OurMethod (TCFP)*	Supervised NB	*OurMethod (TCFP) vs.* Supervised NB
Newsgroups	79.36	***86.19***	+6.83	***91.72***	*-5.53*
WebKB	73.63	***75.47***	+1.84	***85.29***	*-9.82*
Reuters	88.62	***89.09***	+0.47	***91.64***	*-2.55*

Using TCFP, we achieved about a 6.83% advance in the Newsgroups data set, a 1.84% advance in the WebKB data set, and a 0.47% advance in the Reuters data set. In both the WebKB and Reuters data sets, our method using machine-labeled training data obtained lower improvement than in the Newsgroups data set. These results are similar to those of previous work for the EM algorithm [12]. Especially, the results of the WebKB data set indicate that the machine-labeled data from our method is not accurate enough for supervised classifiers.

As shown in Table 4.9, we finally obtained an 86.19% micro-average F_1 score in the Newsgroups data set, a 75.47% micro-average F_1 score in the WebKB data set, and an

89.09% micro-average precision-recall break-even point in the Reuters data set. The differences between our method and the supervised Naive Bayes classifier in each data set are 5.53% in the Newsgroups data set, 9.82% in the WebKB data set, and 2.55% in the Reuters data set. Especially, the result of Reuters reached 2.55% close to the supervised method.

4.4 Discussions

Through our experiments for classifiers using machine-labeled documents, we could obtain an advancement of performance from 0.47% to 6.83%. The learning technique using the machine-labeled training data can give us flexibility to use the general supervised text classifiers and higher performance. Especially, we proposed a robust and fast classifier, TCFP. TCFP is the more proper classifier in our method, which must learn from training data with a lot of noisy data. Furthermore, TCFP has several additional advantages with respect to fast execution speed, simplicity of algorithm, and high performance. Therefore, TCFP can be used in areas which require a robust, fast, and high performance text classifier. Finally, since we use only unlabeled data and title words, final results using TCFP are much significant.

Chapter 5

Comparing Our Method with a Clustering Technique

5.1 Introduction

In Chapter 2, we presented two kinds of approaches using unlabeled data in text categorization; one approach combines unlabeled data and labeled data, and the other approach uses the clustering technique for text categorization. Since our method does not use any human-labeled data, it cannot be fairly compared with the former approach. Therefore, we compare our method with a clustering technique. Slonim proposes a new clustering technique for unsupervised document classification and verifies the superiority of his algorithm in his paper [17]. He calls his clustering technique the sequential Information Bottleneck (**sIB**) algorithm. As he uses the corpora for text categorization such as the Newsgroups and Reuters data sets in his experiments, his algorithm can be applied to unsupervised document classification. In his evaluation, the **sIB** algorithm was superior to all the other clustering methods of his experiments: Agglomerative Information Bottleneck (AIB) algorithm [76], *k*-

means [44], and Iterative Double Clustering (*IDC*) algorithm [77]. Therefore, we set the same experimental setting as that of Slonim's experiments such as the same data sets and performance measures, and verify that our method outperforms his clustering algorithm: the **sIB** algorithm.

5.2 Sequential Information Bottleneck Clustering Algorithm

The **sIB** algorithm is motivated by the Information Bottleneck (**IB**) method. Since other clustering algorithms using the **IB** method are based on agglomerative procedure, they are typically computationally expensive. To solve this problem, the **sIB** algorithm casts any agglomerative procedure into a sequential clustering procedure. The **sIB** algorithm finds a partition *T(X)* which maximize some score function *F(T)* when a set of objects *X* (documents) is given. For score function *F(T)*, the following formula is used [17].

$$I(T;Y) = \sum_{t \in T, y \in Y} p(t) p(y \mid t) \log \frac{p(y \mid t)}{p(y)} \tag{5.1}$$

where *Y* denotes features (words).

The **sIB** algorithm starts with an initial random partition of $T = \{t_1, t_2, ..., t_k\}$ of *X*. At each step, it draws some $x \in X$ out of its current cluster *t(x)* and represents it as a new singleton cluster. Using a greedy agglomeration step, it can now merge *x* into t^{new} such that $t^{new} = \arg\min_{t \in T} d_F(\{x\}, t)$. By iterations of this process, it finally obtains a partition *T(X),* which constructs categories of documents. The distance metric in the **sIB** algorithm is as follows:

$$d(x,t) = (p(x) + p(t)) \cdot JS(p(y \mid x), p(y \mid t)) \tag{5.2}$$

where *JS(p,q)* is the Jensen-Shannon divergence [78] defined as follows:

$$\begin{aligned}
&JS(p,q) = \pi_1 D_{KL}(p \,\|\, \bar{p}) + \pi_2 D_{KL}(q \,\|\, \bar{p}) \\
&where\ \ \{p,q\} \equiv \{p(y \mid x), p(y \mid t)\}, \\
&\qquad \{\pi_1,\pi_2\} \equiv \{\frac{p(x)}{p(x)+p(t)}, \frac{p(t)}{p(x)+p(t)}\}, \\
&\qquad \bar{p} = \pi_1 p(y \mid x) + \pi_2 p(y \mid t)
\end{aligned} \tag{5.3}$$

The *JS* divergence is non-negative and is equal to zero if and only if both its arguments are identical.

The Pseudo-code for the **sIB** algorithm is as follows [17]:

```
Input: |X| objects to be clustered, Parameters: $K$, $n$, $maxL$, $\varepsilon$

Output: A partition $T$ of $X$ into $K$ clusters

Main Loop:

  For $i$ = 1,…,$n$
      $T_i$ is a random partition of $X$.
      $c$=0, $C$=0, $done$ = $FALSE$
      While not done
          For $j$ =1,…,$|X|$
              Draw $x_j$ out of $t(x_j)$
              $t^{new}(x_j) = \arg\min_{t'} d_F(\{x_j\}, t')$
              If   $t^{new}(x_j) \neq t(x_j)$   then c = c+1
              Merge $x_j$ into $t^{new}(x_j)$
          C=C+1
          If C $\geq$ maxL or c $\leq \varepsilon|X|$ then
               Done = TRUE

  T $\leftarrow$ argmax$_{T_i}$ $F(T_i)$
```

5.3 Empirical Evaluation

5.3.1 Data sets and Experimental Setting

For these experiments, we follow the same experimental setting as those of Slonim's paper [17]. We do experiments using two data sets in his medium-scale experiments, Newsgroups and Reuters 21578; they are the most meaningful data sets in hand. These data sets are revised in the same way according to Slonim's paper as follows:

In the **Newsgroups** data set, the categories are united with respect to 10 meta-categories: five *comp* categories, three *politics* categories, two *sports* categories, three *religions* categories, and the two *transportation* categories into 5 big meta-categories. The vocabulary from this data set has 59,130 words.

As a next test data set, we use the 10 most frequent categories in the **Reuters 21578** data set under the 'ModApte' without the train/test split. The size of vocabulary is 12,925 words.

Slonim, in his experiments, used only the documents with more than 10 features after feature selection. But it is a great difficult task to obtain the same data set due to his feature selection. Thus, we preferably use the entire documents of each data set without removing them. These experiments are conducted as a close test. That is, all the documents are used as training data and test data.

As our evaluation measures, we use the micro-average precision which is described in Chapter 3.

5.3.2 Experimental Results

In the experimental results, the performance values of the **sIB** algorithm are taken from [17]. We achieved a 6.65% improvement in a revised Newsgroups data set and a 3.2 % improvement in a revised Reuters data set.

A. Results from the Revised Newsgroups Data Set

In the revised Newsgroups data set, our method is superior to **sIB**. Our method with the TCFP classifier attained an 86.15% micro-average precision score. In this experiment, we used a title word and two keywords as input words of each category.

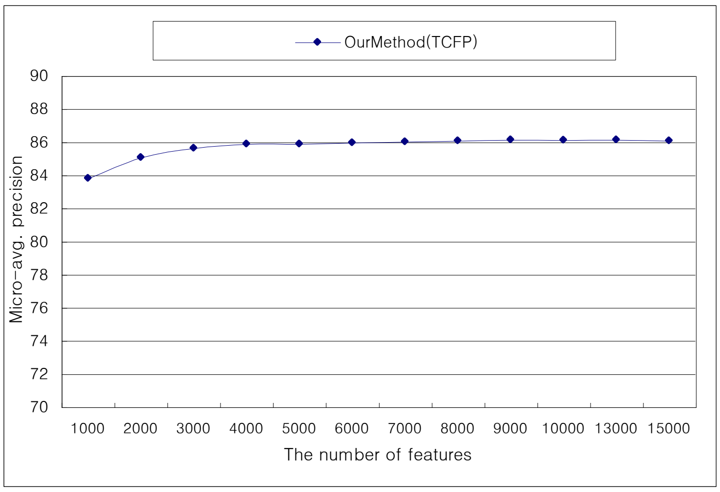


Figure 5. 1 The performance of our method according to the number of features in the revised Newsgroups data set

Table 5. 1 The best micro-avg. precision scores of aIB algorithm and our method in the revised Newsgroup data set

	sIB	*OurMethod (TCFP)*	Improvement
Best micro-avg. precision	79.5	86.15	***+6.65***

B. Results from the Revised Reuters Data Set

Our method with TCFP obtained an 89% micro-average precision score. In this experiment, we used a title word and five keywords as input words of each category.

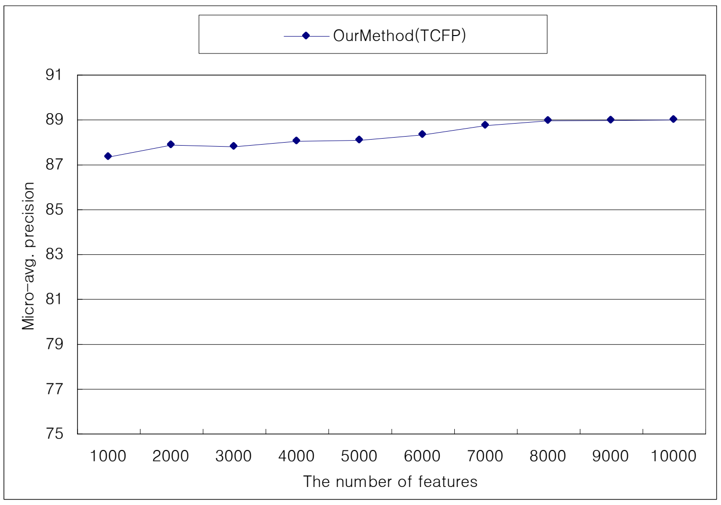


Figure 5. 2 The performance of our method according to the number of features in the revised Reuters data set

Table 5. 2 The best micro-avg. precision scores of aIB algorithm and our method in the revised Reuters data set

	sIB	*OurMethod (TCFP)*	Improvement
Best micro-avg. precision	85.8	89	***+3.2***

5.4 Discussions

Our experimental results show that our method is superior to **sIB** algorithm with regard to performance. Our improvement reported 3.2% on the revised Reuters data set and 6.65% on the revised Newsgroups data set, and it can be regarded as the significant advance. Moreover, Slonim verified that the **sIB** algorithm outperforms other clustering algorithms: various kinds of K-means algorithms and Iterative Double Clustering algorithm [17]. Therefore, we can regard that our method is superior to other clustering algorithms indirectly.

Compared to clustering algorithms, our method has additional strong points as well as high performance. Most clustering algorithms including the **sIB** algorithm have the local maximum problem. To solve this problem, they need to find a global maximum through a repetitive procedure. However, since our method does not have a local maximum problem, it does not require a repetitive procedure. Therefore, we can save processing time in our method. In addition, when a clustering algorithm is applied to text categorization for unlabeled documents, it needs to assign a category label to a cluster. For this, we must assign the most dominant category label of a cluster to all documents of that cluster. That is, we need to look into the content of documents in each cluster after all. However, results of our method are the set of automatically labeled documents.

This chapter has addressed that our method is superior to clustering methods in many ways. Especially, our method outperforms clustering methods with regard to the performance.

Chapter 6

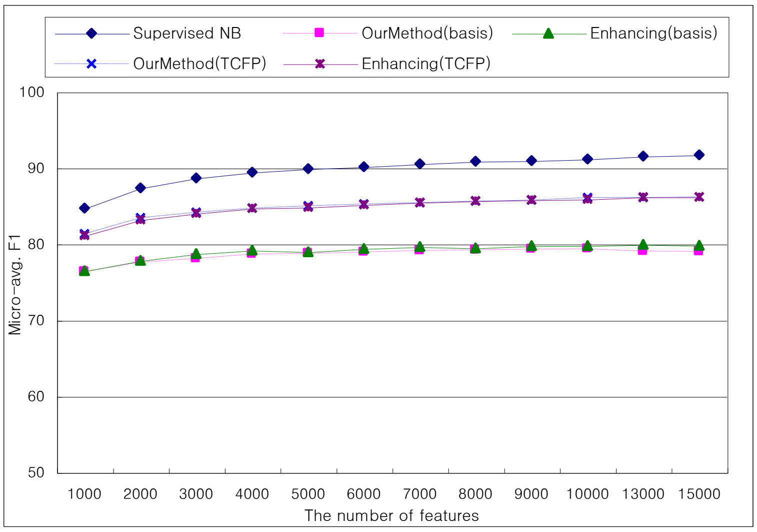
Enhancing Our Method and More Experiments

The main problem of our method is that its performance depends on the quality of the keywords and title words. As we have seen in the previous chapter, we obtained the worst performance in the WebKB data set. In fact, keywords of each category in the WebKB data set also have high frequency in other categories. Worst of all, even the title words (course, professor, student, and project) show a confused usage. We think these factors contribute to a comparatively poor performance of our method. As in the case of WebKB, the ambiguous title words and keywords can raise bad performance. If keywords as well as title words are supplied by humans, our method may achieve higher performance. However, choosing the proper keywords for each category is a much difficult task; it must be more difficult on application domains unknown to developers. Moreover, keywords from a developer, who has insufficient knowledge about an application domain, do not guarantee high performance. In order to overcome this problem, we propose a hybrid method for choosing keywords. That is, a developer obtains 10 candidate keywords from our keyword extraction method and then he can choose proper keywords from them.

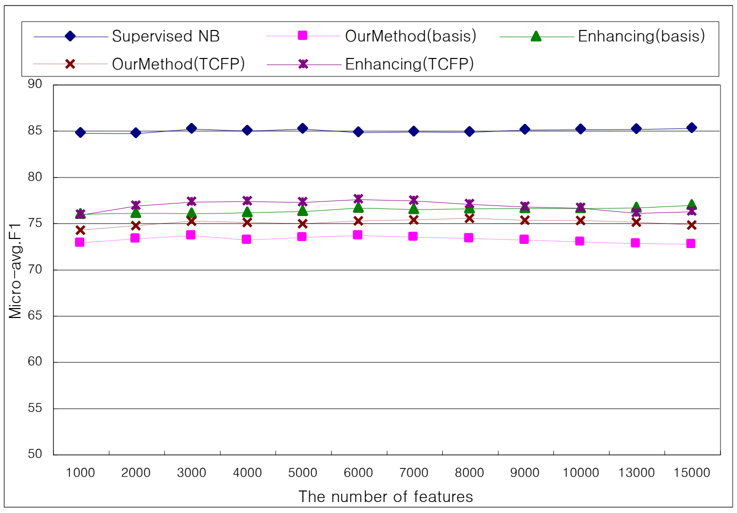
In this section, we revisit the problem associated with providing a sufficient number of labeled training examples that arises in many supervised learning tasks. In other words, for the successful application of supervised learning to solving classification tasks, it is assumed that there is a sufficient amount of labeled data. To assign a category label to an example, a human has to peruse this document first. Obviously, this human-labeling of documents is not only error prone but also a tedious and time-consuming task. Consequently, provision of labeled data requires expensive human resources, making it the bottleneck in the supervised learning setting. The severity of this problem can only be understood when we realize how large a training set must be in order to achieve reasonable results. Therefore, it is crucial to know just when a training set may be deemed large enough. It is known from computational learning theory that, usually, the number of training examples should be at least a multiple of the number of features if reasonable results are to be guaranteed [79]. This theory may give some insight as to what is learnable and can provide worst-case bounds on the number of training examples required. However, for practical problems, we generally have to find out empirically how large the training set should be. Since the high dimensional feature space is used to represent text, providing as many training examples as theoretical considerations is generally not possible in text domains. Moreover, another question is how many labeled documents are required to obtain the same performance as that of our method in each data set. Therefore, we report the classification performance of the Naive Bayes classifier when varying the number of training documents on each data set.

6.1 Results of Choosing Keywords by Hand Supported by Our Method

Here, we show the results from a new method for choosing keywords; a developer obtains 10 candidate keywords from our keywords extraction method and selects proper keywords among them. Figure 6.1 and Table 6.1 show the results from three data sets. For definition of notations in this section, *Enhancing(TCFP)* denotes the TCFP classifier using keywords selected from the new method.



(a) The Newsgroups data set



(b) The WebKB data set

Figure 6. 1 The comparison of enhancing method and previous our method

Table 6. 1 The best micro-average F_1 scores of each method

Data Set	*OurMethod (basis)*	*Enhancing (basis)*	*OurMethod (TCFP)*	*Enhancing (TCFP)*	**Supervised NB**
Newsgroups	79.36	***79.95***	86.19	***86.23***	***91.72***
WebKB	73.63	***76.97***	75.47	***77.59***	***85.29***

Table 6. 2 The precision-recall breakeven points in the Reuters data set

Category	***OurMethod (basis)***	***Enhancing (basis)***	***OurMethod (TCFP)***	***Enhancing (TCFP)***	**Supervised NB**
acq	94.01	92.62	94.99	94.15	96.24
corn	62.5	64.28	64.28	67.85	66.07
crude	80.42	81.48	79.36	76.71	89.41
earn	96.13	95.95	96.96	97.05	97.42
grain	74.49	76.51	67.78	75.83	92.61
interest	77.86	78.62	80.91	80.91	77.09
money-fx	79.32	80.44	78.21	77.65	78.21
ship	77.52	84.26	82.02	82.02	85.39
trade	80.5	82.2	83.05	85.59	81.35
wheat	61.97	63.38	60.56	69.01	67.6
micro-avg.	88.62	***88.84***	89.09	***89.52***	***91.64***

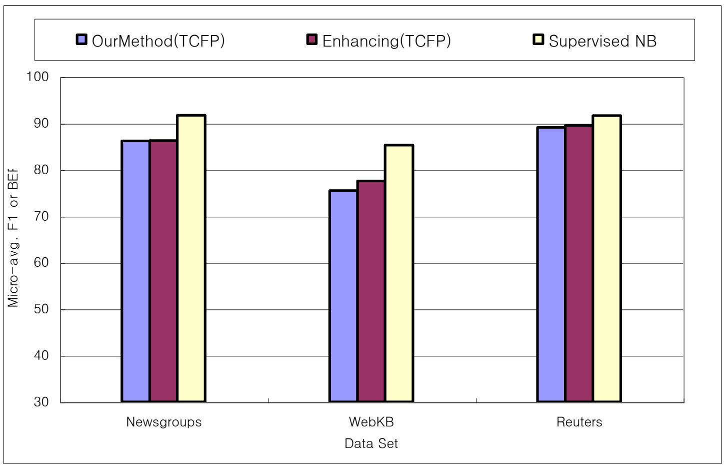


Figure 6. 2 The performance differences of the best micro-avg. F_1 scores or micro-avg. precision-recall breakeven points in three data sets

Table 6. 3 The performance differences of the best micro-avg. F_1 scores or micro-avg. precision-recall breakeven points in three data sets

Data Set	*OurMethod (TCFP)*	*Enhancing (TCFP)*	*OurMethod (TCFP) vs. Enhancing (TCFP)*	Supervised NB	*OurMethod (TCFP) vs.* Supervised NB
Newsgroups	86.19	***86.23***	+0.04	***91.72***	*-5.49*
WebKB	75.47	***77.59***	+2.12	***85.29***	*-7.7*
Reuters	89.09	***89.52***	+0.43	***91.64***	*-2.12*

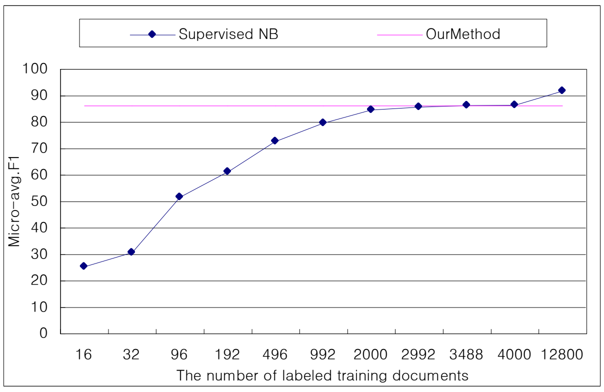
As shown in Table 6.3, we obtained about a 0.04% advance in the Newsgroups data set, a 2.12% advance in the WebKB data set, and a 0.43% advance in the Reuters data set. Especially, we could achieve high improvement in the WebKB data set, which showed the worst performance in the previous chapter. Thus, we find that the new method for choosing keywords is more useful in a domain with confused keywords between categories such as WebKB. We believe the new method is more suitable to a real application area than choosing top *n-th* keywords.

6.2 The Effect of Learning from Small Training Data

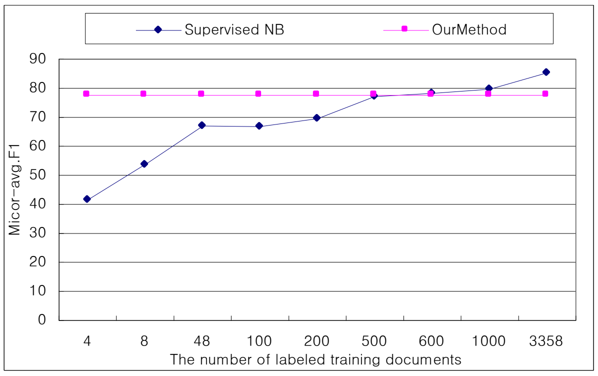
Figure 6.3 shows the classification performance of the supervised Naive Bayes classifier on the three data sets selected when the number of labeled training documents is varied. The horizontal axes indicate the number of labeled training documents. Note that, for example, a total of 16 training documents for the Newsgroups data set corresponds to one document per category and, for the Reuters data set, a total of 10 training documents correspond to one document per category. The vertical axes indicate the classification performance on the test sets.

Notice that the achieved performances vary greatly across the different data sets and different amounts of labeled data. A reason for this lies not only in the separateness of categories but also in the number of categories. Generally, the more labeled data there is, the better the performance is achieved. In particular, additional examples yield substantial improvements in classification performance when training data is scarce. In contrast, only marginal improvements are gained through providing additional training examples when many training examples are available already. For each data set examined, the learning curves begin to converge to a certain dataset-specific level when the training sets contain some hundred examples per category.

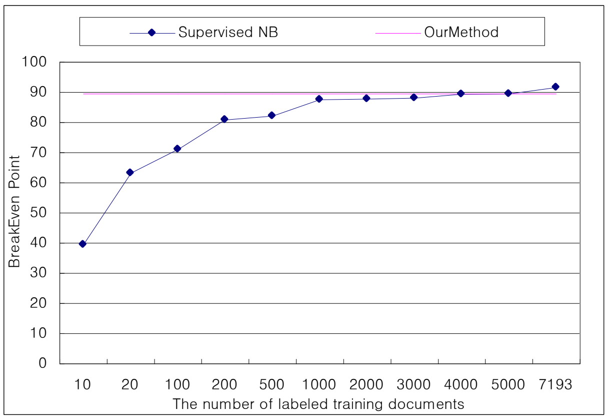
As shown in Figure 6.3, for the Newsgroups data set, we achieved similar performance to that of our method using about 3,500 labeled training documents: about 600 labeled documents for the WebKB data set and about 5,000 labeled documents for the Reuters data set. Although these training set sizes are smaller than the whole training set size, labeling some thousand documents for training is still a tedious and time-consuming task.



(a) The Newsgroups data set



(b) The WebKB data set



(C) The Reuters data set

Figure 6. 3 The effect of the training set size on the classification performance of a supervised Naive Bayes classifier in each data set and the comparison with the performance of our method. The horizontal axes indicate the number of labeled training documents.

6.3 Discussions

This section has showed that we could improve our method as choosing keywords by hand from candidate keywords provided by our keywords extraction method. Especially, in the WebKB data set with confused categories, higher improvement was gained. Although this new method needs more manpower, we think it is a better method for real application areas.

The experimental results for the effect of learning from small training data show that the performance curve is converged when using some hundred labeled training documents per each category and human-labeling tasks for some thousand documents are required to obtain the similar performance to our method. Moreover, as the previous work in [13] reports that labeling 400 documents took about six hours, human-labeling tasks for training data must be tedious and time-consuming. Therefore, our method can help text categorization tasks more tractable by solving the bottleneck of supervised approaches for them.

Chapter 7

Conclusions and Future Work

This dissertation has addressed a new unsupervised or semi-unsupervised text categorization method. Though this method uses only title words for categories and unlabeled documents, it shows significant performance in comparison with that of a supervised Naive Bayes classifier. Labeled data are expensive to collect because a person must take the time and effort to label them while unlabeled data are often inexpensive and plentiful. This is especially true for text classification tasks where almost any type of text is readily available in electronic form.

7.1 Conclusions

From studies and experiments in this thesis, we draw the following significant conclusions.

(1) An automatic text classifier can be built from unlabeled data

Although our method uses only title words for categories and unlabeled data, it can create an automatic text classifier. By a bootstrapping technique in Chapter 3, it finally creates machine-labeled training contexts for a generative model, the Naive Bayes classifier, which learns from them and classifies documents to build machine-labeled training documents. Moreover, using the machine-labeled training documents, other supervised classifiers, which require training data of a document unit, can be used in our method. Using the TCFP classifier among them, we finally achieved an 86.19% micro-average F_1 score in the Newsgroups data set, a 75.47% micro-average F_1 score in the WebKB data set, and an 89.09% micro-average precision-recall breakeven point in the Reuters data set. Especially, results of the Reuters data set reached as close as 2.55% to the supervised method. Moreover, using the new keyword selection method to enhance our method, we could achieve better performance in all the data sets: 86.23% in the Newsgroups data set, 77.59% in the WebKB data set, and 89.52% in the Reuters data set. Especially, we could obtain high improvement, 2.12%, in the WebKB data set. These results can be considered as a significant performance.

(2) The TCFP classifier has robustness from noisy data, fast execution speed, and high performance

In this thesis, a new type of text classifier, TCFP, has been proposed. The experimental results show that TCFP has high performance and fast execution speed. Above all, robustness of TCFP from noisy data makes it a more proper classifier in our method because the machine-labeled training data has many incorrectly labeled documents. Especially, TCFP achieved the best performance in our method. Furthermore, by the simplicity of the TCFP algorithm, its implementation and training process can be done very easily. Therefore, we can use TCFP in application areas which require a robust, fast, and high performance text classifier.

(3) Our method is superior to clustering methods

Compared to the **sIB** clustering algorithm in Chapter 5, our method outperforms clustering algorithms. Our method achieved a 3.2% advance on the Reuters data set and a 6.65% advance in the Newsgroups data set from our experiments and it can be regarded as significant advance. In addition, our method is also superior to clustering methods in several ways such as processing time and local maximum problems.

(4) Our method can be applied to low-cost text categorization and creation of training data

Because labeled data are expensive while unlabeled data are inexpensive and plentiful, we can readily build a text classifier using our method. Therefore, our method is useful for low-cost text categorization. However, if some text categorization tasks require high accuracy, our method is also used as an assistant tool for easily creating training data. Besides, we believe that there are many application areas for our method.

7.2 Future Work

This thesis suggests several future work that deserve attention.

- First of all, the performance of our method depends on title words and the input keywords. Thus, we need additional experiments and discussions about these factors. We need to study the characteristics of candidate words for title words and suggest the criterion for choosing title words. In addition, we have to develop an improved keyword extraction method.
- In Chapter 3, we proposed the bootstrapping method from title words to machine-labeled documents. Although our bootstrapping method achieved a significant performance in our experiments, there is still potential to improve the bootstrapping method for high classification performance.
- In Chapter 4, we presented the TCFP classifier and it achieved the best performance in our method. However, it still shows a little lower performance than SVM in using the human-labeled data. Therefore, to improve the TCFP classifier, we should study more characteristics of the feature projection technique and the feature weighting technique for TCFP.

Bibliography

[1] F. Sebastiani, “Machine Learning in Automated Text Categorization,” *Tutorial at the 18th International Conference on Computational Linguistics (COLING’00)*, Nancy, France, July 2000.

[2] A. McCallum and K. Nigam, “A Comparison of Event Models for Naive Bayes Text Classification,” *AAAI’98 workshop on Learning for Text Categorization*, pp. 41-48, 1998.

[3] Y. Ko and J. Seo, “Automatic Text Categorization by Unsupervised Learning,” *Proceedings of the 18th International Conference on Computational Linguistics (COLING’2000)*, pp. 453-459, 2000.

[4] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka, “Training Algorithms for Linear Text Classifiers,” *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR’96)*, pp. 289-297, 1996.

[5] Y. Yang, S. Slattery, and R. Ghani, “A Study of Approaches to Hypertext Categorization,” *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412-420, 2002.

[6] Y. Ko and J. Seo, “Using Feature Projection Technique Based on the Normalized Voting Method for Text Classification,” *Information Processing and Management*, 2003. (in press)

[7] T. Joachims, “Text Categorization with Support Vector Machines: Learning with Many Relevant Features,” *Proceedings of European Conference on Machine Learning (ECML)*, pp. 137-142, 1998.

[8] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, “Learning to Classify Text from Labeled and Unlabeled Documents,” *Proceedings of 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[9] K. Lang, “NewsWeeder: Learning to Filter Netnews,” *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 331-339, 1995.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society Series B*, Vol. 39, No. 1, pp. 1-38, 1977.

[11] K. Nigam, A. McCallum, S. Thrun and T. Mitchell. “Text Classification from Labeled and Unlabeled Documents Using EM,” *Machine Learning*, Vol. 39, No. 2/3, pp. 103-134. 2000.

[12] K. P. Nigam, *Using Unlabeled Data to Improve Text Classification*, Doctoral dissertation, Computer Science Department, Carnegie Mellon University, 2001.

[13] A. McCallum, K. Nigam, J. Renni, and K, Seymore, “Automating the Construction of Internet Portals with Machine Learning”, *Information Retrieval.* Vol. 3, No. 2, pp. 127-163. 2000.

[14] C. Lanquillon, “Partially Supervised Text Categorization: Combining Labeled and Unlabeled Documents Using an EM-like Scheme”, *Proceedings of the 11th Conference on Machine Learning (ECML 2000), Vol. 1810 of LNCS, Springer Verlag*, pp. 229-237, 2000.

[15] C. Lanqillon, *Enhancing Text Classification to Improve Information Filtering*, Doctoral dissertation, Otto-von-Guericke-Universität Magdeburg, 2001.

[16] K. Bennett and A. Demiriz, “Semi-supervised Support Vector Machines,” *Advances in Neural Information Processing Systems 11, MIT PRESS*, pp. 368-374,

1999.

[17] N. Slonim, N. Friedman, and N. Tishby, "Unsupervised Document Classification Using Sequential Information Maximization," *Proceedings of the 25rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 129-136, 2002.

[18] D. D. Lewis and W. A. Gale, "A Sequential Algorithm for Training Text Classifiers," *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3-12, 1994.

[19] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, "Learning to Construct Knowledge Bases from the World Wide Web," *Artificial Intelligence*, Vol. 118, No. 1-2, pp. 69-113, 2000.

[20] J. Shavlik and T. Eliassi-Rad, "Intelligent Agents for Web-based Tasks: An Advice-taking Approach," *Learning for Text Categorization: Papers from the AAAI Workshop*, Tech. Rep. WS-98-05, AAAI Press, pp. 63-70, 1998.

[21] D. D. Lewis and K. A. Knowles, "Threading Electronic Mail: A Preliminary Study," *Information Processing and Management*, Vol. 33, No. 2, pp. 209-217, 1997.

[22] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos, "An Experimental Comparison of Naive Bayesian and Keyword-based Anti-spam Filtering with Personal Email Messages," *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.160-167, 2000.

[23] R. J. Mooney and L. Roy, "Content-based Book Recommending Using Learning

for Text Categorization," *Proceedings of the Fifth ACM Conference on Digital Libraries*, pp. 195-204, 2000.

[24] C. Apte, F. Damerau, and S. M. Weiss, "Automated Learning of Decision Rules for Text Categorization," *ACM Transactions on Information Systems*, Vol. 12, No. 3, pp. 233-251, 1994.

[25] I. Moulinier, G. Raskinis, and J. G. Ganascia, "Text Categorization: a Symbolic Approach," *Proceedings of Fifth Annual Symposium on Document Analysis and Information Retrieval*, pp. 87-99, 1996.

[26] M. Craven, S. Slattery, and K. Nigam, "First-order Learning for Web Mining," *Proceedings of the 10th European Conference on Machine Learning*, pp. 250-255, 1998.

[27] E. Wiener, J. O. Pedersen, and A. S. Weigend, "A Neural Network Approach to Topic Spotting," *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pp. 317-332, 1995.

[28] Y. Yang and C. G. Chute, "An Example-based Mapping Method for Text Classification and Retrieval," *ACM Transactions on Information Systems*, Vol. 12, No. 3, pp. 252-277, 1994.

[29] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization," *Proceedings of the Seventh International Conference on Information and Knowledge Managemen*t, pp. 53-58, 1994.

[30] R. E Schapire and Y. Singer, "BoosTexter: A Boosting-based System for Text Categorization," *Machine Learning*, Vol. 39, No. 2/3, pp. 135-168, 2000.

[31] F. Sebastiani, A. Sperduti, and N. Valdambrini, "An Improved Boosting

Algorithm and its Application to Text Categorization," *Proceedings of the Ninth International Conference on Information and knowledge Management*, pp. 78-85, 2000.

[32] Y. Yang and X. Liu, "A Re-examination of Text Categorization Methods," *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 42-49, 1999.

[33] J. Furnkranz, T. Mitchell, and E. Riloff, "A Case Study in Using Linguistic Phrases for Text Categorization on the WWW," *Learning for Text Categorization: Papers from the AAAI Workshop*, pp. 5-12, Tech. Rep. WS-98-05, AAAI Press, 1998.

[34] M. Rodriguez, J. M. Gomez-Hidalgo, and B. D. Agudo, "Using WordNet to Complement Training Information in Text Categorization," *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pp. 150-157, 1997.

[35] S. Scott and S. Matwin, "Text Classification Using WordNet Hypernyms," *Usage of WordNet in Natural Language Processing Systems: Proceedings of the Workshop*, pp. 45-52, 1998.

[36] Y. Ko, J. Park, and J. Seo, "Improving Text Categorization Using the Importance of Sentences," *Information Management and Processing*, 2003. (in press)

[37] G. D. Murray and D. M. Titterington, "Estimation Problems with Data from a Mixture," *Applied Statistics*, Vol. 27, No. 3, pp. 325-334, 1978.

[38] R. J. A. Little, "Discussion on the Paper by Professor Dempster, Professor Laird and Dr. Rubin," *Journal of the Royal Statistical Society*, Series B, Vol. 39, No. 1, 25, 1977.

[39] D. J. Miller and H. Uyar, “A Generalized Gaussian Mixture Classifier with Learning based on both Labelled and Unlabelled Data,” *Proceedings of the 1996 Conference on Information Science and Systems,* 1996.

[40] S. Baluja, “Probabilistic Modeling for Face Orientation Discrimination: Learning from Labeled and Unlabeled Examples,” *Advances in Neural Information Processing Systems 11*, pp. 854-860, 1999.

[41] V. Vapnik, *Statistical Learning Theory*, New York: John Wiley and Sons.

[42] T. Joachims, “Transductive Inference for Text Classification Using Support Vector Machines,” Machine Learning: *Proceedings of the Sixteenth International Conference,* 1999.

[43] T. Zhang and F. J. Oles, “A Probability Analysis on the Value of Unlabeled Data for Classification Problems,” *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 1191-1198, 2000.

[44] M. Steinbach, G. Karypis, and V. Kumar, “A Comparison of Document Clustering Techniques,” *Proceedings of International Conference on Knowledge Discovery and Data Mining*, Text mining Workshop, 2000.

[45] N. Slonim and N. Tishby. “Document Clustering Using Word Clusters via the Information Bottleneck Method,” *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 208-215, 2000.

[46] Y. Karov and S. Edelman, “Similarity-based Word Sense Disambiguation,” *Computational Linguistics*, Vol. 24, No. 1, pp. 41-60, 1998.

[47] Y. Yang and J.O. Pederson, “A Comparative Study on Feature Selection in Text Categorization,” *Proceedings of the 14th International Conference on Machine*

Learning, 1997.

[48] D. D. Lewis, "Naive (bayes) at Forty: The Independence Assumption in Information Retrieval," *Proceedings of European conference on Machine Learning*, 1998.

[49] D. D. Lewis and M. Ringuette, "A Comparison of Two Learning Algorithms for Text Categorization," *Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81-93, 1994.

[50] Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," *Information Retrieval Journal*, May, 1999.

[51] E. Brill, "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging," *Computational Linguistics*, Dec, 1995.

[52] C. D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*, The MIT press, Second Edition, 1999.

[53] J. Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Inc, Second Edition, 1995.

[54] S. Park, H. Kim, Y. Ko, and J. Seo, "Implementation of an Efficient Requirements Analysis Supporting System using Similarity Measure Techniques," *Information and Software Technology*, Vol. 42, No. 6, pp. 429-438, 15 April 2000.

[55] Y. Maarek, D. Berry, and G. Kaiser, An Information Retrieval Approach for Automatically Construction Software Libraries, *IEEE Transaction On Software Engineering*, Vol. 17, No. 8, pp. 800-813, August 1991.

[56] G. Miller, "Wordnet: An Online Lexical Database," *International Journal of Lexicography*, Vol. 3, No. 4, pp. 235-312, 1990.

[57] M. Okumura, H. Mochizuki and H. Nanba, “Query-biased Summarization Based on Lexical Chaining,” *Processing of Pacific Association for Computational Linguistics*, 1999.

[58] K. Cho and J. Kim, “Automatic Text Categorization on Hierarchical Category Structure by using ICF (Inverse Category Frequency) Weighting,” *Proceedings of KISS conference*, pp. 507-510, 1997.

[59] H. Schutze, “Automatic Word Sense Discrimination,” *Computational Linguistics*, Vol. 24, No. 1, pp. 97-124, 1998.

[60] Y. Ko, S. Park, and J. Seo, “Web-based Requirements Elicitation Supporting System using Requirements Sentences Categorization,” *Proceedings of Twelfth International Conference on Software Engineering and Knowledge Engineering*, pp. 344-351, July 2000.

[61] T. Joachims, “A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization,” *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 143-151, 1997.

[62] T. M. Mitchell, *Machine Learning*, New York: McGraw-Hill, 1997.

[63] A. McCallum, R. Rosenfeld, T. D. Mitchell, “Improving Text Classification by Shrinkage in a Hierarchy of Classes,” *Machine Learning: Proceedings of the Fifteenth International Conference*, pp. 359-367, 1998.

[64] R. Liere and P. Tadepalli, “Active Learning with Committees for Text Categorization,” *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 591-596, 1997.

[65] Y. Yang, “Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval,” *Proceedings of 17th*

International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 13-22, 1994.

[66] D. R. Wilson and T. R. Martinez, “Reduction Techniques for Exemplar-Based Learning Algorithms,” *Machine Learning*, Vol. 38, No. 3, pp. 257-286, 1999.

[67] A. Akkus and H. A. Guvenir, “k Nearest Neighbor Classification on Feature Projections,” *Proceedings of ICML’96*, pp. 12-19, 1996.

[68] P. E. Hart, “The Condensed Nearest Neighbor Rule,” *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, Vol. 14, pp. 515-516.

[69] D. W. Aha, K. Dennis, and K. A. Marc, “Instance-Based Learning Algorithms,” *Machine Learning*, Vol. 6, pp. 37-66, 1991.

[70] J. Zhang, “Selecting Typical Instances in Instance-Based Learning,” *Proceedings of the Ninth International Conference on Machine Learning*, 1992.

[71] D. R. Wilson and T. R. Martinez, “Instance Pruning Techniques,” *Proceedings of the Fourteenth International Conference*, pp. 404-411, 1997.

[72] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, Second Edition, 2001.

[73] G. Salton and C. Buckley, “Improving Retrieval Performance by Relevance Feedback,” *Information Processing and Management*, Vol. 24, pp. 513-523, 1988.

[74] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc, 1983.

[75] V. Weston and C. Watkins, “Support Vector Machines for Multi-class Pattern Recognition,” *Proceedings of the Seventh European Symposium On Artificial Neural networks (ESANN-99)*, 1999.

[76] N. Slonim and N. Tishby, “Agglomerative Information Bottleneck,” *Proceedings*

of Neural Information Processing System (NIPS-12), 1999.

[77] R. El-Yaniv and O. Souroujon, “Iterative Double Clustering for Unsupervised and Semi-supervised Learing,” *Proceedings of Neural Information Processing Systems (NIPS-14)*, 2002.

[78] J. Lin, “Divergence Measures Based on the Shannon Entropy,” *IEEE Transactions on Information Theory*, Vol. 37, No. 1, 1991.

[79] D. D. Lewis, *Representation and Learning in Information Retrieval*. Doctoral dissertation, Department of Computer and Information Science, University of Massacusetts, 1992.

Appendix

Lists of Title Words and Keywords for Each Data Set

This appendix presents the entire lists of title words and keywords for each data set.

Table A. 1 The list of keywords for each category in five folds of the Newsgroups data set

Category	***Title Word***	***fold***	***Keywords***
alt.atheism	atheism	1-*th*	atheists, theism
		2-*th*	atheist, atheists
		3-*th*	theism, atheists
		4-*th*	atheists, non-existence
		5-*th*	atheist, atheists
comp.graphics	graphics	1-*th*	vesa, wate
		2-*th*	vesa, wate
		3-*th*	m13, gems
		4-*th*	herringshaw, vesa
		5-*th*	vesa, m13
comp.sys.mac.hardware	mac	1-*th*	apple, iisi
		2-*th*	apple, quadra
		3-*th*	apple, quadra
		4-*th*	apple, quadra
		5-*th*	apple, quadra
comp.windows.x	xwindows	1-*th*	postscript, x11r4
		2-*th*	script, x11r4
		3-*th*	x11r4, postscript
		4-*th*	toolkit, clone
		5-*th*	script, Athena
misc.forsale	for sale	1-*th*	offer, condition
		2-*th*	offer, sale
		3-*th*	offer, sale
		4-*th*	offer, condition
		5-*th*	offer, condition
rec.autos	auto	1-*th*	freon, cars
		2-*th*	bigots, freon
		3-*th*	cars, bigots
		4-*th*	bigots, freon
		5-*th*	cars, freon

rec.motorcycles	motorcycle	1-*th*	wheelie, bike
		2-*th*	wheelie, shaft-drives
		3-*th*	wheelie, shaft-drives
		4-*th*	wheelie, shaft-drives
		5-*th*	wheelie, shaft-drives
rec.sport.baseball	baseball	1-*th*	players, pitcher
		2-*th*	pitcher, bat
		3-*th*	players, pitcher
		4-*th*	pitcher, bat
		5-*th*	pitcher, players
rec.sport.hockey	hockey	1-*th*	nhl, cup
		2-*th*	nhl, ice
		3-*th*	nhl, cup
		4-*th*	nhl, cup
		5-*th*	nhl, cup
sci.crypt	cryptography	1-*th*	lobbying, organized
		2-*th*	encryption, lobbying
		3-*th*	lobbying, encryption
		4-*th*	lobbying, encyption
		5-*th*	encryption, lobbying
sci.electronics	electronics	1-*th*	delco, kokomo
		2-*th*	delco, kokomo
		3-*th*	stramer, lookout
		4-*th*	stramer, lookout
		5-*th*	hugo, intergraph
sci.med	medicine	1-*th*	osteopathic, gilbert
		2-*th*	osteopathic, vms.ocom.okstate.edu
		3-*th*	osteopathic, patients
		4-*th*	osteopathic, news@osuunx.ucc.okstate,edu
		5-*th*	osteopathic, patients
sci.space	space	1-*th*	nasa, shuttle
		2-*th*	nasa, shuttle
		3-*th*	nasa, shuttle
		4-*th*	nasa, shuttle
		5-*th*	nasa, shuttle
soc.religion.christian	christian	1-*th*	morality, christianity
		2-*th*	christianity, morality
		3-*th*	christianity, morality
		4-*th*	christianity, morality
		5-*th*	christianity, morality
talk.politics.guns	gun	1-*th*	firearms, weapons
		2-*th*	firearms, weapons
		3-*th*	firearms, weapons
		4-*th*	firearms, weapons
		5-*th*	firearms, weapons
talk.politics.mideast	mideast	1-*th*	political, shaw
		2-*th*	ahmad, kathleen
		3-*th*	kathleen, political
		4-*th*	ahmad, shaw
		5-*th*	political, kathleen

Table A. 2 The list of keywords for each category in five folds of the WebKB data set

Category	***Title Word***	***fold***	***Keywords***
course	course	1-*th*	assignments, hours, instructor, class, fall
		2-*th*	instructor, hours, assignments, class, fall
		3-*th*	instructor, hours, assignments, class, fall
		4-*th*	hours, assignments, instructor, class, instruction
		5-*th*	hours, instructor, assignments, class, fall
faculty	professor	1-*th*	associate, ph.d, fax, interests, publications
		2-*th*	associate, ph.d, fax, interests, publications
		3-*th*	associate, ph.d, fax, interests, publications
		4-*th*	associate, ph.d, interests, fax, publications
		5-*th*	associate, ph.d, fax, interests, publications
project	project	1-*th*	system, systems, research, software, information
		2-*th*	system, systems, software, research, information
		3-*th*	system, systems, research, software, group
		4-*th*	system, systems, research, software, group
		5-*th*	system, systems, research, software, design
student	student	1-*th*	graduate, computer, science, page, university
		2-*th*	graduate, computer, science, page, home
		3-*th*	graduate, computer, science, page, home
		4-*th*	graduate, computer, science, page, home
		5-*th*	graduate, computer, science, page, home

Table A. 3 The list of keywords for each category in the Reuters data set

Category	***Title Word***	***Keywords***
acq	Acquisition, merger	inc, shares, shareholders
corn	corn	bushel, soybeans, bushels
crude	Crude oil	bpd, barrels, barrel
earn	earnings	quarter, share, results
grain	grain	usda, harvest, crop
interest	interest rate	rates, rate, money
money-fx	foreign exchange	currency, intervention, dollar
ship	shipping	missiles, chinese-made, silkworm
trade	trade	tariffs, surplus, countries
wheat	wheat	tones, agriculture, enhancement

Table A. 4 The list of keywords for each category of the revised Newsgroups data set in Chapter 5 (10 categories)

Category	***Title Word***	***Keywords***
religion	religion, atheism, christian	god, morality
computer	graphics, mac, hardware, x-windows, ms-windows	software, apple
forsale	for sale	offer, condition
vehicle	auto, motorcycle	wheelie, cars
sport	baseball, hockey	game, players
crypt	cryptography	lobbying, encryption
electronics	electronics	delco, kokomo
medicine	medicine	osteopathic, patients
space	space	nasa, shuttle
politics	politics, gun, mideast	firearms, weapons

Table A. 5 The list of keywords for each category of the revised Reuters data set in Chapter 5

Category	***Title Word***	***Keywords***
acq	acquisition, merger	inc, shares, shareholders
corn	corn	soybeans, bushel, soybean
crude	crude oil	bpd, barrels, barrel
earn	earnings	quarter, revenues, share
grain	grain	harvest, crop, elevator
interest	interest rate	rates, rate, money
money-fx	foreign exchange	currency, dollar, intervention
ship	shipping	missiles, silkworm, ships
trade	trade	tariffs, deficit, exports
wheat	wheat	tonnes, agriculture, winter

Table A. 6 The list of keywords for each category in five folds of the Newsgroups data set in Chapter 6. These keywords were selected by the new keyword extraction method.

Category	***Title Word***	***fold***	***Keywords***
alt.atheism	atheism	1-*th*	atheists, theism, atheist, theists, mozumder, bake, timmons
		2-*th*	atheist, atheists, theism, theists, mozumder, theist
		3-*th*	theism, atheists, mozumder, atheist, theists, s.n, theist
		4-*th*	atheists, atheist, theism, theists, mozumder
		5-*th*	atheist, atheists, theism, theists, mozumder, s.n, theist
comp.graphics	graphics	1-*th*	vesa, wate, vga, animation, pixel, image
		2-*th*	vesa, wate, herringshaw, vga, pixel
		3-*th*	vesa, tmc, wate, vga, jpeg
		4-*th*	herringshaw, vesa, wate, eisa, vga, pixel
		5-*th*	vesa, vga, herringshaw, wate, animation
comp.sys.mac.hardware	mac	1-*th*	apple, iisi, quadra, nubus, scsi, powerbook
		2-*th*	apple, quadra, iisi, nubus, scsi, powerbook
		3-*th*	apple, quadra, iisi, scsi, powerbook, nubus, iifx
		4-*th*	apple, quadra, iisi, scsi, powerbook, nubus, iici
		5-*th*	apple, quadra, iisi, scsi, nubus, powerbook
comp.windows.x	xwindows	1-*th*	postscript, x11r4, x11, xhost
		2-*th*	script, x11r4, xhost, x11, postscript
		3-*th*	x11r4, postscript, x11, xview
		4-*th*	x11r4, postscropt, ncsa
		5-*th*	script, athena, x11r4, xhost, postscript, x11
misc.forsale	for sale	1-*th*	offer, postage, cod, shipping, offers
		2-*th*	offer, obo, postage, cod, offers, shipping
		3-*th*	offer, postage, obo, packaging, cod, kou
		4-*th*	offer, cod, obo, postage, shipping
		5-*th*	offer, obo, cod, offers, do-it-yourselers
rec.autos	auto	1-*th*	freon, cars, car, transmissions, sedans
		2-*th*	freon, cars, car, transmissions
		3-*th*	cars, car, lh, drivers, porsche
		4-*th*	freon, cars, car, camry
		5-*th*	cars, freon, car, transmissions
rec.motorcycles	motorcycle	1-*th*	wheelie, bike, dod, shaft-drives, wheelies, bikes, countersteering_faq, motorcyclist
		2-*th*	wheelie, shaft-drives, bike, wheelies, dod, bikes, countersteering_faq, handlebars
		3-*th*	wheeli, shaft-drives, wheelies, bike, dod, bikes, riders, suzuki
		4-*th*	wheeli, shaft-drives, wheelies, dod, bike, countersteering_faq, bikes, Suzuki, handlebars
		5-*th*	wheelie, shaft-drives, wheelies, dod, bike, bikes, countersteering_faq, motorcyclist
rec.sport.baseball	baseball	1-*th*	pitcher, bat, nl, koufax, lowenstein, braves, mets, lustig
		2-*th*	pitcher, bat, braves, hitter, reardon, inning mets, pitchers
		3-*th*	pitcher, lowenstein, koufax, stankowitz, bat, pitchers
		4-*th*	pitcher, bat, nl, hitter, lowenstein, stankowitz, koufax
		5-*th*	pitcher, nl, braves, clemens, mets, bat, lowenstein, loufax, stankowitz
rec.sport.hockey	hockey	1-*th*	nhl, cup, ahl, ice, ecac, farenell, adirondack, glens
		2-*th*	nhl, ice, cup, devils, leafs, staffan, axelsson

		3-*th*	nhl, cup, devils, ice, bruins, adirondack, farenell, ecac
		4-*th*	nhl, cup, devils, ahl, adirondack, ice, ecac, farenell
		5-*th*	nhl, cup, devils, ahl, ice, adirondack, ecac, farenell, leafs
sci.crypt	cryptography	1-*th*	encryption, cryptographic
		2-*th*	encryption, sci.crypt
		3-*th*	encryption, cryptosystem, sci.crypt
		4-*th*	encryption, sci.crypt
		5-*th*	encryption, sci.crypt, pem
sci.electronics	electronics	1-*th*	delco, kokomo, triantafyllopoulos, spiros, circuit, intergraph, gandler, hitachi, resistors
		2-*th*	delco, kokomo, spiros, triantafyllopoulos, Intergraph, circuit
		3-*th*	stramer, intergraph, circuit, waveform, hitachi, resistances
		4-*th*	intergraph, waveform, circuit, hitachi, haddy
		5-*th*	hugo, intergraph, wolfgang, circuit, gandler, resistances, connectors
sci.med	medicine	1-*th*	osteopathic, gilbert, patients, nutrition, disease, treatments, physicians
		2-*th*	osteopathic, patients, disease, physicians, patient
		3-*th*	osteopathic, patients, nutrition, gilbert, physicians, disease
		4-*th*	osteopathic, patients, disease, nutrition, gilbert, anti-fungals
		5-*th*	osteopathic, patients, nutrition, disease, gilbert, patient, physician
sci.space	space	1-*th*	nasa, shuttle, launch, orbit, moon, astronomy, telescope
		2-*th*	nasa, shuttle, launch, orbit, moon, astronomy, spacecraft
		3-*th*	nasa, shuttle, launch, telescope, moon, orbit, astronomy
		4-*th*	nasa, shuttle, launch, orbit, moon, telescope, spacecraft
		5-*th*	nasa, shuttle, launch, orbit, astronomy, moon, telescope
soc.religion.christian	christian	1-*th*	christianity, jesus, christ, bible
		2-*th*	christianity, jesus, christ, bible, scripture
		3-*th*	christianity, christ, jesus, bible, scripture
		4-*th*	christianity, christ, jesus, bible
		5-*th*	christianity, jesus, christ, bible, scripture
talk.politics.guns	gun	1-*th*	firearms, weapons, firearm, handgun, handguns, weapon
		2-*th*	firearms, weapons, handgun, handguns, firearm weapon
		3-*th*	firearms, weapons, firearm, handgun, handguns, arms, weapon
		4-*th*	firearms, weapons, handgun, handguns, firearm, weapon
		5-*th*	firearms, weapons, firearm, handgun, handguns, weapon
talk.politics.mideast	mideast	1-*th*	shaw, kathleen, ahmad, avetis, muratoff, sept, israel, bortnick
		2-*th*	ahmad, kathleen, shaw, sept, avetis, arab, Israel, arabs
		3-*th*	kathleen, avetis, ahmad, shaw, muratoff, arab, rawlinson, sept, mccarthy
		4-*th*	ahmad, shaw, kathleen, sept, brigham, davidsson, macmillan, arabs, arab
		5-*th*	kathleen, ahmad, shaw, elias, sept, davidsson, palestine, invader, yitzhak

Table A. 7 The list of keywords for each category in five folds of the WebKB data set in Chapter 6. These keywords were selected by the new keyword extraction method.

Category	***Title Word***	***fold***	***Keywords***
course	course	1-*th*	assignments, hours, instructor, class, exam, lecture, homework, syllabus
		2-*th*	instructor, hours, assignments, class, lecture, homework, exam, syllabus
		3-*th*	instructor, hours, assignments, class, lecture, exam, homework, notes
		4-*th*	hours, assignments, instructor, class, lecture, exam, notes, syllabus
		5-*th*	hours, instructor, assignments class, lecture, syllabus, notes, homework
faculty	professor	1-*th*	associate, fax, ieee, fellow, assistant
		2-*th*	associate, fax, assistant, journal, award
		3-*th*	associate, fax, fellow, assistant, journal, ieee
		4-*th*	associate, fax, assistant, journal, fellow, award
		5-*th*	associate, fax, ieee, fellow, journal, assistant
project	project	1-*th*	performance, tools
		2-*th*	performance
		3-*th*	performance
		4-*th*	performance
		5-*th*	performance
student	student	1-*th*	graduate, home, links, grad
		2-*th*	graduate, home, links, grad
		3-*th*	graduate, home, links, grad
		4-*th*	graduate, home, links, grad
		5-*th*	graduate, home, grad, links

Table A. 8 The list of keywords for each category in the Reuters data set in Chapter 6. These keywords were selected by the new keyword extraction method.

Category	***Title Word***	***Keywords***
acq	acquisition, merger	inc, shares, shareholders
corn	corn	bushel, soybeans, bushels, soybean
crude	crude oil	bpd, barrels, barrel, petroleum, opec, gasoline
earn	earnings	quarter, revenues, income, profits
grain	grain	usda, harvest, maize, buenos, aires, argentine
interest	interest rate	lending, bonds, inflation, banks, bundesbank
money-fx	foreign exchange	currency, intervention, dollar, dollars, auction, yen
ship	shipping	missiles, chinese-made, silkworm, tehran, missile, iran, ships, vessel
trade	trade	tariffs, surplus
wheat	wheat	tones, winter, barley, flour