Text Classification from Unlabeled Documents with Bootstrapping and Feature Projection Techniques

Youngjoong Ko[1] *and Jungyun Seo*[2]

[1]Department of Computer Engineering,
Dong-A University,
840, Hadan 2-dong, Saha-gu,
Busan, 604-714, Korea
yjko@dau.ac.kr,

[2]Department of Computer Science and Program of Integrated Biotechnology,
Sogang University
Sinsu-dong 1, Mapo-gu
Seoul, 121-742, Korea
seojy@sogang.ac.kr

1. ***Corresponding author :*** Jungyun Seo

2. ***Corresponding address :*** Department of Computer Science and Interdisciplinary Program of Integrated Biotechnology, Sogang University, Sinsu-dong 1, Mapo-gu, Seoul 121-742, Republic of Korea.

3. ***Corresponding telephone number :*** 82-2-705-8488

4. ***Corresponding fax number :*** 82-2-704-8273

5. ***Corresponding Email address :*** seojy@sogang.ac.kr

Text Classification from Unlabeled Documents with Bootstrapping and Feature Projection Techniques

ABSTRACT

Many machine learning algorithms have been applied to text classification tasks. In the machine learning paradigm, a general inductive process automatically builds a text classifier by learning, generally known as *supervised learning*. However, the supervised learning approaches have some problems. The most notable problem is that they require a large number of labeled training documents for accurate learning. While unlabeled documents are easily collected and plentiful, labeled documents are difficultly generated because a labeling task must be done by human developers. In this paper, we propose a new text classification method based on unsupervised or semi-supervised learning. The proposed method launches text classification tasks with only unlabeled documents and the title word of each category for learning, and then it automatically learns text classifier by using bootstrapping and feature projection techniques. The results of experiments showed that the proposed method achieved reasonably useful performance compared to a supervised method. If the proposed method is used in a text classification task, building text classification systems will become significantly faster and less expensive.

KEYWORDS

Text Classification, Bootstrapping, Feature Projection, Unlabeled Data, Text Classifier

1. INTRODUCTION

With the rapid growth of the World Wide Web, the task of classifying natural language documents into a *pre-defined* set of semantic categories has become one of the key methods for organizing online information. This task is commonly referred to as text classification. Since there has been an explosion of electronic texts from not only the World Wide Web but also various online sources (electronic mail, corporate databases, chat rooms, digital libraries, and so on) recently, one way of organizing this overwhelming amount of data is to classify them into topical categories.

Since the machine learning paradigm emerged in the 90's, many machine learning algorithms have been applied to text classification by *supervised learning*. The supervised learning algorithm finds a representation or decision rule from an example set of labeled documents for each class. A wide range of the supervised learning algorithms has been applied to this area using a training data set of labeled documents. For example, there are Naive Bayes (Ko and Seo, 2000; McCallum and Nigam, 1998), Rocchio (Lewis et al., 1996), Nearest Neighbor (*k*-NN) (Yang et al., 2002), and Support Vector Machine (SVM) (Joachims, 2001).

However, the major bottleneck of the supervised learning algorithms is that they require a large number of labeled training documents for accurate learning. Since a labeling task must be done manually, it is a painfully time-consuming process. Furthermore, since the application area of automatic text classification 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 (Nigam et al., 1998). McCallum et al. (1999) found that only 100 documents could be hand-labeled in the 90 minutes and the result of a classifier learned from this small training set achieved just 30% accuracy in their experiments. Most users of a practical system, however, do not want to do the labeling task for a long time only to obtain this level of accuracy. They obviously prefer algorithms that have high accuracy, but do not require a large amount of manually labeling task.

In this paper, we propose a new text classification method based on unsupervised or semi-supervised learning. The proposed method uses only unlabeled documents and the title word of each category as initial data for learning of text classification. While labeled data is difficultly obtained, unlabeled data is readily available

and plentiful. Therefore, this paper advocates an automatic labeling task using *a bootstrapping technique* and a robust text classifier using *a feature projection technique*. The input to the bootstrapping process is a large amount of unlabeled documents and a small amount of seed information to tell the learner about the specific task. Here, we consider a title word associated with a category as seed information. To automatically build up a text classifier with unlabeled documents, we must solve two problems; how we can automatically generate labeled training documents (*machine-labeled data*) from only a title word, and how we can handle incorrectly labeled documents in the machine-labeled data. This paper provides the solutions of both the problems. For the former, we employ the bootstrapping technique and, for the latter, we use the TCFP (*Text Categorization using Feature Projections*) classifier with robustness from noisy data (Ko and Seo, 2002).

1.1 How can an Automatic Text Classifier be Built from Unlabeled Documents?

Do you think that it is possible to build a text classifier with only unlabeled documents? Maybe we cannot gain any information from unlabeled documents for building a text classifier because the unlabeled documents do not contain the most important information, *their category*. In general, the existing supervised learning algorithms cannot construct any decision rules without the labeled data. Thus labeled training data must be obtained in order to use the existing supervised learning algorithms. Here, we explain how labeled data can be generated from the unlabeled data for text classification. Since text classification is a task based on the *pre-defined* categories, developers can at least know the categories for classifying documents. Knowing the categories means that they can at least choose a title word of each category. This is the starting point of the proposed method. As developers carry out a bootstrapping task from the title word, they can finally get labeled training data.

Suppose that we are going to classify documents into an ‘Autos’ category. First, the title word of this category is selected ‘automobile,’ and then the related keywords (e.g. ‘car’, ‘gear’, ‘transmission’, ‘sedan’) of ‘Auto’ are extracted by using co-occurrence information between the title word *(‘automobile’)* and the other words. In the proposed method, *context* is defined as a unit of meaning for the bootstrapping process from the title word; it has a middle size of sentences and documents (a sequence of 60 words in a document). Then the

bootstrapping process first extracts the most informative contexts for the category which include at least one among the title word and the keywords. The extracted contexts are called by *centroid-contexts* because they are regarded as contexts with the core meaning of each category. We can obtain many words directly co-occurred with the title word and the keywords from the centroid-contexts (e.g. ‘driver’, ‘clutch’, ‘trunk’, and so on); these words are in the first-order co-occurrence with the title word and the keywords. Since only the words in the first-order co-occurrence cannot sufficiently describe the meaning of the category, we must collect more contexts by measuring similarities between centroid-contexts and remaining contexts; the remaining contexts do not have any title word and any keywords. The collected contexts contain the words in the second-order co-occurrence with the title word and the keywords. As a result, the *context-cluster* of the category is constructed as the combination of the centroid-contexts and the contexts collected by the similarity method. A Naive Bayes classifier can learn from the created context-cluster. Since the Naive Bayes classifier can assign each unlabeled document its label, the labeled training documents are obtained automatically; it is called by *machine-labeled data*.

When the machine-labeled data is used to build up supervised mannered text classifiers, there is an additional problem in that the data has more incorrectly labeled documents than manually labeled data does. Thus we develop and employ the TCFP classifier with robustness from noisy data for learning from the machine-labeled data.

The rest of this paper is organized as follows. Section 2 presents previous related work. In section 3, we explain the bootstrapping technique to create machine-labeled data. Section 4 describes the TCFP classifier to learn from the machine-labeled data. Section 5 is devoted to the analysis of empirical results. In section 6, we discuss the proposed method and results. Finally, we describe conclusions and future work.

2. RELATED WORK

In this literature, there are various studies that aim to reduce efforts for labeling tasks. Some studies are based on models that learn from labeled and unlabeled documents (Nigam, 2001; Ghani, 2002; Lanquillon, 2000), models that perform a partially supervised classification (Jeon and Landgrebe, 1999; Liu et al., 2002), or active learning (Roy and McCallum, 2001; Tong and Koller, 2001). An alternative strategy is to employ unsupervised clustering for text classification (Adami et al., 2003; Slonim et al., 2002).

The studies to support the manual labeling of documents focus on the labeling task of a restricted set of documents. Then they fulfill to a requirement of a minimum amount of labeled data for each category using unlabeled data. Nigam (1998) studied an Expected Maximization (EM) technique for combining labeled and unlabeled data for text classification in his dissertation. Ghani (2002) developed a framework to incorporate unlabeled data in the Error-Correcting Output Coding (ECOC) setup by first decomposing multiclass problems into multiple binary problems and using Co-Training to learn the individual binary classification problems. Lanquillon (2000) presented another approach for learning from labeled and unlabeled data. Since the most straightforward way to make use of unlabeled data is through unsupervised learning, he exploited partitional clustering methods. Jeon and Landgrebe (1999) proposed a new partially supervised classification method using unsupervised clustering, and Liu et al. (2002) studied the problem of classification with only partial information, one class of labeled (positive) documents, and a set of mixed documents. Roy and McCallum (2001) presented an active learning method that directly optimizes expected future errors, and Tong and Koller (2001) introduced a new algorithm for performing active learning with SVM. These previous studies always require a first sample set of labeled data while the proposed method uses only unlabeled data.

In the other hand, there are several studies which used the clustering algorithms to text classification for not doing any labeling tasks. Slonim et al. (2002) suggested using clustering techniques for unsupervised document classification. When a collection of unlabeled documents was given, he attempted to find clusters that are highly correlated with the true topics of documents by a new clustering method, the sequential Information Bottleneck (sIB) algorithm. Adami et al. (2003) proposed a semi-automatic process whose aim is

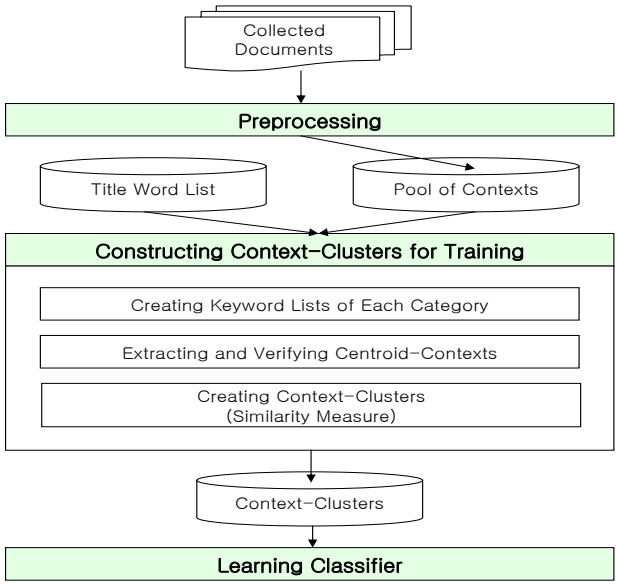
to minimize the work required to the administrators when creating, modifying, and maintaining taxonomy with labeled documents.

Basically, the problem we are attacking can be conceived as a bootstrapping technique using keywords for each category. Several bootstrapping techniques using keywords have studied in this literature (McCallum et al., 2000; Urena-lopez et al., 2001). McCallum et al. (2000) presented a bootstrapping method for construction of a domain-specific search engine. In the domain specific search engine, more specific information such as category hierarchy, keyword, 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. The bootstrapping iterations are EM steps that used unlabeled data and hierarchical shrinkage to estimate parameters of a Naive Bayes classifier. However, since they require specific preliminary information such as a category hierarchy and human-provided keywords to build a rule-list classifier, its application can be restricted. Urena-lopez et al. (2001) proposed an approach to text classification that is based on the integration of linguistic resources. They first presented the integration of the lexical database WordNet and the training collection for text classification, by means of the Vector Space Model and the Rocchio training algorithm. This integration is made with the help of word sense disambiguation (WSD). However, this method required the whole labeled training documents and did not exploit any unlabeled document.

3. THE BOOTSTRAPPING TECHNIQUE TO GENERATE MACHINE-LABELED DATA

The bootstrapping process consists of three modules as shown in Figure 1: a module to preprocess unlabeled documents, a module to construct context-clusters for training, and a module to build up the Naive Bayes classifier using context-clusters. Each module is described in the following sections in detail.

Figure 1. The overview of the bootstrapping process



3.1 Preprocessing

The preprocessing module has two main roles: extracting content words and reconstructing unlabeled documents into contexts. The Brill POS tagger is used to extract content words (Brill, 1995). 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. However, since machine-labeled data is created from only a title word, *context* is defined as a new unit of

meaning, and it is used as the meaning unit to bootstrap the meaning of each category; the linguistic definition of context is the part of a text that surrounds a particular word or a passage and determines its meaning (Manning and Schutze, 1999). Note that the final goal of the bootstrapping process is to build up labeled training documents from only title words automatically. Hence, the middle size processing unit, between word and document, is required. A sequence of 60 content words within a document is regarded as the window size for one context; we believe that words co-occurred with any keyword in a context have important meaning for each category. In order to choose this number of 60 words for the window size, we refer a study of bootstrapping technique applied to a WSD problem. Yarowsky recommended the optimal window size to be between 40 words and 100 words in his paper (Yarowsky, 1994). Thus we conducted simple experiments using various window sizes (40~100 words) and consequently chose 60 words for the window size. To extract the contexts from a document, we use a sliding window technique (Maarek et al., 1991). The window slides from the first content word to the last content word of the document in the size of the window (60 words) and with the interval of each window (30 words). That is, two successive contexts are overlapped in rear 30 words of the previous window and front 30 words of the next window. Therefore, the final output of preprocessing is a set of context vectors that are represented as content words of each context.

3.2 Constructing a Context-Cluster as the Training Data of Each Category

At first, keywords are automatically generated from a title word for each category using co-occurrence information. Then centroid-contexts are extracted by using the title word and keywords. Each centroid-context includes at least one of the title word and keywords. It is regarded as one of the most informative contexts for each category. Furthermore, more information of each category is obtained by assigning remaining contexts to each context-cluster by a similarity measure technique.

3.2.1 Creating Keyword Lists

A title word presents the main meaning of each category, but it could be insufficient in representing any category for text classification. Thus keywords, which are semantically related to the title word, are added for each category. The degree of semantic similarity is estimated for extracting keywords by using co-occurrence information between the title word and other words in the unlabeled documents.

The score of semantic similarity between a title word, *T*, and a word, *W*, is calculated by the cosine metric as follows:

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

where t_i and w_i represent the occurrence of words *T* and *W* in *i*-th document respectively; they are denoted by binary value, 0 or 1, and *n* is the total number of documents in the unlabeled documents. This formula calculates the similarity score between words based on the degree of their co-occurrence in the same document.

The most important criterion for good keywords of a category is a similarity with the title word of each topic category, and it can be measured by formula 1. Then the ambiguity of words must be considered. That is, if any word has a high similarity with title words of two or more categories, the word must be excluded from keywords because it does not have the power to discriminate these categories. To apply the former criterion to the proposed method, each word is first assigned to the keyword candidate list of a category with the maximum similarity score. For latter criterion, the important score of each assigned word is recalculated by using the following formula:

$$Score\ (W, c_{\max}) = sim\ (T_{\max}, W) + (sim\ (T_{\max}, W) - sim\ (T_{second\ \max}, W)) \quad (2)$$

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

This formula means that a word in high ranking has a high similarity score with the title word ($sim(T_{max},W)$) and a high similarity score difference with other title words ($sim(T_{max},W)$- $sim(T_{secondmax},W)$). Words assigned to each category are sorted out according to the score calculated by formula 2 in a descending order. Then top *m* words are chosen as keywords in the category. Table 1 shows the list of keywords (top 5) for each category in the WebKB data set.

Table 1. The list of keywords 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

3.2.2 Extracting and Verifying Centroid-Contexts

A context with a keyword or a title word of any category is selected as a centroid-context. From the selected contexts, we can obtain a set of words in the first-order co-occurrence from centroid-contexts of each category. But, among the selected centroid-contexts, some contexts could not have effective features of a category even though they include a keyword or a title word of the category. Thus the importance score of each centroid-context is measured and it is ranked according to the calculated importance score. First of all, the weight of each word is calculated using Term Frequency (*TF)* within a category and Inverse Category Frequency (*ICF*) (Cho and Kim, 1997). Using word weights (TW_{ij}), the score of a centroid-context (S_k) in *j*-th category (c_j) is computed as follows:

$$Score\ (S_k, c_j) = \frac{TW_{1j} + TW_{2j} + \ldots + TW_{Nj}}{N} \quad (3)$$

where *N* is the number of content words in each centroid-context.

The centroid-contexts of each category are sorted in a descending order according to their calculated importance scores. This order of the centroid-contexts is used in the following section.

3.2.3 Creating the Context-Cluster of each category

We here gather the second-order co-occurrence information by assigning remaining contexts to the context-cluster of each category. For the assigning criterion, we calculate similarities between remaining contexts and the centroid-contexts of each category. Thus we employ the similarity measure algorithm by Karov and Edelman (1998). In the proposed method, a part of this algorithm is reformed for our purposes, and remaining contexts are assigned to each context-cluster by this algorithm.

1) Measurement of word and context similarities

As similar words tend to appear in similar contexts, the similarity is calculated by using contextual information. Words and contexts play complementary roles. 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, *Word Similarity Matrix (WSM)* and *Context Similarity Matrix* (*CSM)*; the rows and columns of *WSM* are labeled by all the content words encountered in the centroid-contexts of each category and input remaining contexts, and the rows of C*SM* correspond to the centroid-contexts and the columns to the remaining contexts. Each category has one *WSM* and one C*SM*. In each iteration *n*, WSM_n, whose cell (*i*,*j*) holds a value between 0 and 1, is updated, and the value of each cell indicates the extent to which the *i*-th word is contextually similar to the *j*-th word. Also, CSM_n, which holds similarities among contexts, is kept and updated. In this paper, the number of input contexts of row and column in *CSM* is limited to 200 as considering execution time and memory allocation, and the number of iterations is set as 3 as it is recommended by Karov and Edelman (1998).

To estimate the similarities, *WSM* is initialized 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 formula

To simplify the symmetric iterative treatment of similarities between words and contexts, an auxiliary relation between words and contexts is expressed as affinity and is represented by $aff_n(X,W)$. A word *W* is assumed to have a certain affinity to every context *X*, 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 (Karov and Edelman, 1998). In these formulae, $W \in X$ means that a word *W* belongs to a context *X*:

$$aff_n(W, X) = \max_{W_i \in X} sim_n(W, W_i) \quad (4)$$

$$aff_n(X, W) = \max_{W \in X_j} sim_n(X, X_j) \quad (5)$$

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 a 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:

$$sim_{n+1}(X_1, X_2) = \sum_{W \in X_1} weight(W, X_1) \cdot aff_n(W, X_2) \tag{6}$$

$$\begin{aligned} &\text{if} \quad W_1 = W_2 \\ &\quad sim_{n+1}(W_1, W_2) = 1 \\ &\text{else} \\ &\quad sim_{n+1}(W_1, W_2) = \sum_{W_1 \in X} weight(X, W_1) \cdot aff_n(X, W_2) \end{aligned} \tag{7}$$

The weights in formula 6 are calculated by a methodology described in Appendix. Since each weight in formula 7 is a reciprocal of the number of contexts that contain W_1, the sum of the weights is 1. These values are used to update the corresponding entries of WSM_n and CSM_n.

4) Assignment of remaining contexts to a category

The similarity value of each remaining context for each category is decided by using the following formula:

$$\underset{c_i \in C}{sim}(X, c_i) = aver\left\{ \underset{S_j \in CC_{c_i}}{sim}(X, S_j) \right\} \tag{8}$$

In formula 8, X is a remaining context, $C = \{c_1, c_2, ..., c_m\}$ is a category set, and $CC_{c_i} = \{S_1, ..., S_n\}$ is a controid-contexts set of category c_i.

Each remaining context is assigned to a category with the maximum similarity value. However, 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 (Ko and Seo, 2000):

$$\max\{ \underset{c_i \in C}{\text{sim}} (X, c_i) \} \geq \mu + \theta\sigma \quad (9)$$

where X is a remaining context, μ is an average of similarity values, $\underset{c_i \in C}{sim}(X, c_i)$, σ is a standard deviation of similarity values, and 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. In this paper, we empirically set a 15% threshold value from an experiment using a validation set.

3.3 Learning a Naive Bayes Classifier Using Context-Clusters

In above section, we obtained labeled contexts training data: *context-clusters*. Since the training documents are labeled as the context unit, a Naive Bayes classifier is selected to learn from context-clusters because it can be built by only estimating words probabilities in each category. That is, the Naive Bayes classifier can learn not from word distribution within each document but from words distribution within each category. Therefore, the Naive Bayes classifier is constructed by estimating words distribution in the context-cluster of each category, and it finally classify unlabeled documents into each category.

The Naive Bayes classifier is built up with minor modifications based on Kullback-Leibler Divergence (Craven et al., 2000). This method makes exactly the same classifications as Naive Bayes, but produce classification scores that are less extreme. Thus better reflect uncertainty than those produced by Naive Bayes. A document d_i is classified by to the following formula:

$$P(c_j \mid d_i;\hat{\theta}) = \frac{P(c_j \mid \hat{\theta})P(d_i \mid c_j;\hat{\theta})}{P(d_i \mid \hat{\theta})} \approx P(c_j \mid \hat{\theta})\prod_{t=1}^{|V|} P(w_t \mid c_j;\hat{\theta})^{N(w,d_i)}$$
$$\propto \frac{\log P(c_j;\hat{\theta})}{n} + \sum_{t=1}^{|V|} P(w_t \mid d_i;\hat{\theta}) \log\left(\frac{P(w_t \mid c_j;\hat{\theta})}{P(w_t \mid d_i;\hat{\theta})}\right) \quad (10)$$

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

4. USING A FEATURE PROJECTION TECHNIQUE FOR HANDLING THE NOISY DATA OF THE MACHINE-LABELED DATA

The labeled data of a documents unit is finally obtained through the bootstrapping process, *machine-labeled data*. Now text classifiers can learn from the machine labeled data. But since the machine-labeled data is created by the proposed bootstrapping method, it generally includes more incorrectly labeled documents than the human-labeled data. In order to effectively handle them, a feature projection technique is applied to our text classifier (TCFP) (Ko and Seo, 2002). By the property of the feature projection technique, the TCFP classifier can have robustness from noisy data. In the experiment results, TCFP showed better performance than other conventional classifiers in using machine-labeled data.

4.1 The TCFP Classifier with Robustness from Noisy Data

We here explain our TCFP classifier using the feature projection technique. In this classifier, the classification knowledge is represented as a set of projections of training data on each feature dimension. The classification of a test document is based on the voting of each feature (word) of the test document. That is, the final prediction score is calculated by accumulating the voting scores of all features.

First of all, the voting ratio of each category must be calculated for all features. Since elements with a high TF-IDF value in projections of a feature must become more useful classification criteria for the feature, only

elements with TF-IDF values above the average TF-IDF value are used for voting. The selected elements participate in proportional voting with the same importance as the TF-IDF value of each element. Thus, the voting ratio of each category c_j in a feature f_m is calculated by the following formula:

$$r(c_j, f_m) = \sum_{f_{mi} \in V_m} w(f_m, \vec{d}_i) \cdot y(c_j, f_{mi}) \Big/ \sum_{f_{mi} \in V_m} w(f_m, \vec{d}_i) \tag{11}$$

In formula 11, f_{mi} denotes the projection element for a feature f_m in a document d_i, $w(f_m, \vec{d}_i)$ is the weight of a feature f_m in a document d_i, V_m denotes a set of elements selected for the voting of a feature f_m, and $y(c_j, f_{mi}) \in \{0,1\}$ is a function; if the category for an element f_{mi} is equal to c_j, the output value is 1. Otherwise, the output value is 0.

Next, since each feature separately votes on feature projections, contextual information is missing. Thus co-occurrence frequency is used to apply contextual information to the proposed classification algorithm. To calculate a co-occurrence frequency value between any two features f_i and f_j, the number of documents, which include both features, is counted. TF-IDF values of two features f_i and f_j in a test document are modified by reflecting the co-occurrence frequency of the two features. That is, terms with a high co-occurrence frequency value and a low category frequency value have higher term weights as the following formula:

$$fw(f_i, \vec{d}) = w(f_i, \vec{d}) \cdot \left(1 + \left(\frac{1}{\log(cf + 1)}\right) \cdot \left(\frac{\log(co(f_i, f_j) + 1)}{\log(maxco(f_k, f_l) + 1)}\right)\right) \tag{12}$$

where $fw(f_i,d)$ denotes a modified term weight assigned to term f_i, cf denotes the category frequency that is the number of categories in which f_i and f_j co-occur, $co(f_i, f_j)$ is a co-occurrence frequency value for f_i and f_j, and $maxco(f_k, f_l)$ is the maximum value among all co-occurrence frequency values. Note that the weight of feature f_j is also modified by the same formula using f_j instead of f_i and every $co(f_i, f_j)$ is calculated in the training phase.

Finally, the voting score of each category c_j in a feature f_m of a test document d is calculated by the following formula:

$$vs(c_j, f_m) = fw(f_m, \vec{d}) \cdot r(c_j, f_m) \cdot \log(1 + \chi^2_{\max}(f_m)) \quad (13)$$

where $fw(f_m,d)$ denotes a modified term weight by the co-occurrence frequency and $\chi^2_{\max}(f_m)$ denotes the maximum score of the calculated χ^2 statistics value of f_m in each category. These χ^2 statistics values in feature selection are calculated by using a two-way contingency table of a word f_m and a category c_i as follows:

$$\chi^2(f_m, c_i) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \quad (14)$$

where A is the number of times f_m and c_i co-occur, B is the number of times f_m occurs without c_i, C is the number of times c_i occurs without f_m, D is the number of times neither c_i nor f_m occurs, and N is the total number of documents

The outline of the TCFP classifier is as follows:

Input: test document: $\vec{d}$ =<f_1,f_2,…,f_n>

Main Process:

For each feature f_i

$fw(f_i,d)$ is calculated

For each feature f_i

For each category c_j

vote[c_j]=vote[c_j]+$vs(c_j,f_i)$ by Formula 13

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

5. EMPIRICAL EVALUATION

5.1 Data Sets and Experimental Settings

To test the proposed method, we used three different kinds of data sets: UseNet newsgroups (20 Newsgroups), web pages (WebKB), and newswire articles (Reuters 21578). For fair evaluation in Newsgroups and WebKB, we employed 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 the results of our experiments are averages of five runs.

The **Newsgroups** data set, collected by Ken Lang, contains about 20,000 articles evenly divided among 20 UseNet discussion groups (McCallum and Nigam, 1998; Nigam et al., 1998). Many of the categories fall into confusable clusters; for example, five of them are comp.* discussion groups, and three of them discuss about religion. In this paper, we used only 16 categories after removing 4 categories: three miscellaneous categories (talk.politics.misc, talk.religion.misc, and comp.os.ms-windows.misc) and one duplicate meaning category (comp.sys.ibm.pc.hardware)[1]. After removing words that occur only once and on a stop word list, the resulting average vocabulary from five training data has 43,249 words (no stemming).

The second data set comes from the **WebKB** project at CMU (Craven et al., 2000). 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 the data set was used in other studies (Joachims, 2001; McCallum and Nigam, 1998; Nigam, 2001; Lanquillon, 2000), we used the four most populous entity-representing categories: course, faculty, project, and student. The resulting data set consists of 4,198 pages. The resulting average vocabulary from five training data has 18,742 words.

[1] Since we used a category name as the title name of each category, we could not choose proper title words for these miscellaneous and duplicate categories. Thus these categories were removed for fair evaluation.

The **Reuters 21578** Distribution 1.0 data set consists of 12,902 articles and 90 topic categories from the Reuters newswire. Following other studies (Joachims, 2001; Nigam, 2001), the results of ten most populous categories were reported. To split train/test data, we followed a standard 'ModApte' split. We used all the words in the title and body, and we used 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 were selected for a validation set. After all parameter values of our experiments were set from the validation set, we evaluated the proposed method using these parameter values.

We applied a statistical feature selection method (χ^2 statistics) for each classifier at its preprocessing stage (Yang and Pedersen, 1997).

As performance measures, we followed the standard definition of recall, precision, and F_1 measures. For evaluation performance average across categories, we used the micro-averaging method (Yang, 1999). Results on Reuters are reported as precision-recall breakeven points, which is a standard information retrieval measure for binary classification (Joachims, 2001; Yang, 1999).

The title words in our experiment are selected from the category names of each data set for fair evaluation; because each category name of the Reuters data set has an abbreviated form, the title words are selected from the description of each category in the ReadMe file.

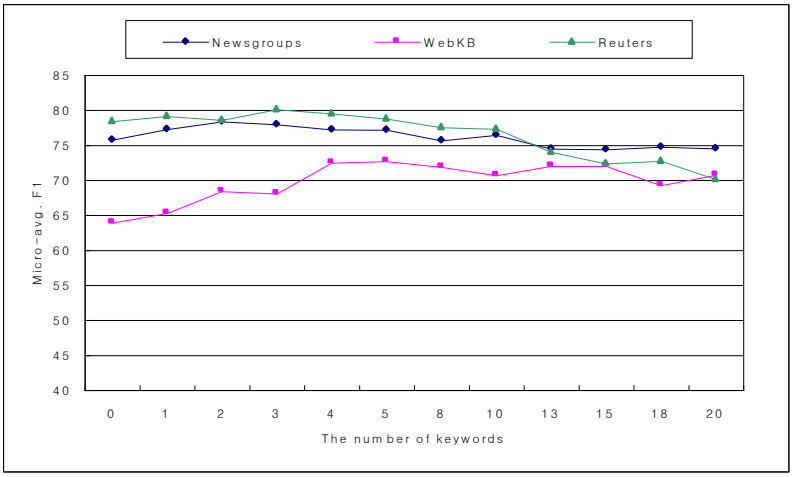
5.2 Experimental Results

We tested the proposed method the following steps. First, using the validation set of each data set, we observed the performance according to the number of keywords and verified our similarity measure technique for assignment of remaining contexts. Finally, the proposed method was compared with the ***sIB*** clustering algorithm (Slonim et al., 2002), a semi-supervised learning method, and a supervised learning method.

5.2.1 Observing the Performance According to the Number of Keywords

First of all, the number of keywords is determined to be used in the proposed method. The number of keywords is limited by the top *n*-th words from the ordered keyword list of each category. Figure 2 displays the performance at the different number of keywords (from 0 to 20) in each data set. If we use zero keywords, it means that only the title word is used.

Figure 2. The comparison of performance according to the number of keywords



As shown in Figure 2, 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 use the number of keywords with the best performance in each data set. Generally, we recommend the number of keywords to be from 2 to 5.

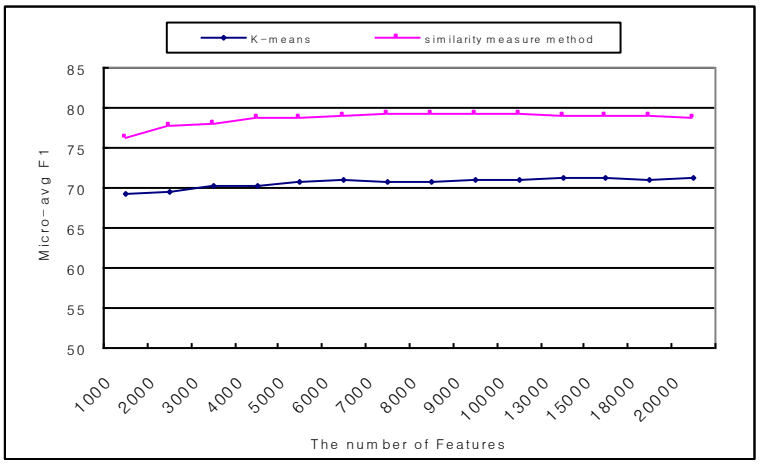
5.2.2 Verifying our Similarity Measure Algorithm for Assignment of Remaining Contexts through Comparing with the K-means Algorithm

We here verify our similarity measure algorithm for assignment of remaining contexts as mentioned in section 3.2.3. To verify the efficiency of our similarity measure algorithm, we exploit the standard *K*-means algorithm that uses the cosine metric as a similarity measure under a vector space model. But the *K*-means algorithm in this

paper has a different point from the standard *K*-means algorithm. In the beginning, each initial cluster center is set as the mean of centroid-contexts.

Figure 3 shows the performance curve of each algorithm. As shown in Figure 3, our similarity measure algorithm shows better performance over all intervals. Note that mutual information was used for feature selection to set the number of features in Figure 3.

Figure 3. The comparison of performance for context assignment algorithms



5.2.3 Comparing the Proposed Method Using TCFP to those Using other Classifiers

In this section, we prove the superiority of TCFP over the other classifiers (SVM, *k*-NN, Naive Bayes, Rocchio) in training data with much noisy data such as the *machine-labeled* data. As shown in Table 2, the best performance was obtained in using TCFP at all three data sets. For definition of notations in this section, *OurMethod(TCFP)* denotes the TCFP classifier using the machine-labeled data as training data. The same manner is applied for the other classifiers.

Table 2. The best micro-average F_1 scores for Newsgroup and WebKB and precision-recall breakeven points for Reuters of each classifier

Data Set	*OurMethod (NB)*	*OurMethod (Rocchio)*	*OurMethod (k-NN)*	*OurMethod (SVM)*	*OurMethod (TCFP)*
Newsgroups	83.46	83	79.95	82.49	***86.19***
WebKB	73.22	75.28	68.04	73.74	***75.47***
Reuters	88.23	86.26	85.65	87.41	***89.09***

5.2.4 Comparing Our Method with a Clustering Technique

In related work, there have been two approaches using unlabeled data in text classification; one approach combines unlabeled data and labeled data, and the other approach uses the clustering technique for text classification. Since the proposed method does not use any labeled data, it cannot be fairly compared with the former approaches. Therefore, the proposed method is compared to a clustering technique applied to text classification. Slonim et al. (2002) proposed a new clustering technique for unsupervised document classification and verified the superiority of his algorithm. They called his clustering technique *the sequential Information Bottleneck **(sIB)** algorithm.* In their experiments, the ***sIB*** algorithm was superior to other clustering algorithms. Moreover, its results were comparable to those by a supervised Naive Bayes Classifier. After we set the same experimental settings as those in Slonim's experiments and conduct experiments, we verify that the proposed method outperforms the ***sIB*** algorithm. In these experiments, the micro-averaging precision is used as performance measure and two revised data sets are used as the test data set: *revised_NG, revised_Reuters*. These data sets were revised in the same way according to Slonim's paper as follows:

In revised_NG, the categories of the Newsgroups data set were united with respect to 10 meta-categories: five comp categories, three politics categories, two sports categories, three religions categories, and two transportation categories into five big meta-categories.

The revised_Reuters used the 10 most frequent categories in the Reuters 21578 corpus under the ModApte split.

These experiments were conducted as a close test. That is, all the documents were used as test data as well as training data. The experimental results are shown in Table 3 in detail. As shown in Table 3, the experimental results of the proposed method show relative improvement as high as 8.36% in *revised NG* and 3.73% in *revised Reuters* over ***sIB***.

Table 3. The comparison of the proposed method and *sIB*

	sIB	***OurMethod (TCFP)***	***Improvement***
revised_NG	79.5	86.15	***+8.36***
revised_Reuters	85.8	89.0	***+3.73***

5.2.5 Comparing the proposed method to the Semi-Supervised Naive Bayes Classifier and the Supervised Naive Bayes Classifier

Like the experimental settings of other previous studies for classification using unlabeled data (Slonim et al, 2002; McCallum et al., 2000; Lanquillon, 2000), the Naive Bayes (NB) classifier is chosen as semi-supervised and supervised learning methods. For semi-supervised learning, Lanquillon (2000)'s experimental results are compared to our method because his experimental settings of the WebKB data set are nearly same as ours. He proposed the semi-supervised method based on partitional clustering for learning from labeled and unlabeled data in text classification; his training data set is composed of an unlabeled set of 2,500 pages and a non-overlapping labeled set. According to his experimental results with 20 labeled documents, the classification accuracy of the supervised NB classifier was 50% and that of the semi-supervised NB classifier was 61%. Finally, with 400 labeled training documents, the semi-supervised NB classifier achieved 76%. This classification accuracy is almost similar to our method's performance (75.47%) even though he used 400 labeled training documents. Moreover, he reported that the performance gain achieved by the semi-supervised learners decreases as the number of labeled training documents increases, because more accurate classifiers can be learned from the labeled data alone. Therefore, developers barely benefit from incorporating unlabeled documents through semi-supervised learning if they use more than a certain amount of labeled documents.

For supervised learning, the training data consists of 500 different documents randomly chosen from the appropriate categories; this means that the labeling task conducted for less time relative to that for all the training data. For this experiment, we consider not only the labeling task using a part of unlabeled documents but also the labeling task using the whole of them. As a result, the following table reports performances from two kinds of NB classifiers which are learned from 500 training documents and the whole training documents respectively.

Table 4. The comparison of the proposed method and the supervised NB classifier

Data Set	***OurMethod (TCFP)***	***NB (500)***	***NB (ALL)***
Newsgroups	86.19	72.68	91.72
WebKB	75.47	74.1	85.29
Reuters	89.09	82.1	91.64

In Table 4, the results of the proposed method are shown to be higher than those of *NB(500)* and they are comparable to those of *NB(All)* in all data sets. Especially, the result of the proposed method in the Reuters data set reached 2.55% closer to that of *NB(All)*. Note that NB(All) learned from the whole labeled training data.

6. DISCUSSION

We here discuss the weakness of the proposed method and propose the hybrid keywords extraction method to overcome this weakness. Then we observe how many human-labeled documents are required to obtain the performance of the proposed method in each data set.

6.1 Enhancing the Proposed Method from Choosing Keywords by Human Developers

The main problem of the proposed method is that its performance depends on the quality of the keywords and title words. As shown in Table 2, we obtained the worst performance in the WebKB data set. In fact, title words and keywords of each category in the WebKB data set also have high frequency in other categories. We think

these factors contribute to a comparatively poor performance of the proposed method. If keywords as well as title words are supplied by humans, the proposed method may be able to achieve better performance. However, choosing the proper keywords for each category is a much difficult task. Moreover, keywords from developers, who have insufficient knowledge about an application domain, do not guarantee a high degree of performance. In order to overcome this problem, we propose a hybrid method for choosing keywords. That is, a developer obtains 10 candidate keywords from the keyword extraction method and then he/she can choose proper keywords from them. Table 5 shows the results from the hybrid method in three data sets.

Table 5. The comparison of the original proposed method and the hybrid method

Data Set	***OurMethod (TCFP)***	***Hybrid (TCFP)***	***Improvement***
Newsgroups	86.19	86.23	***+0.046***
WebKB	75.47	77.59	***+2.81***
Reuters	89.09	89.52	***+0.48***

Especially, we could achieve significant improvement in the WebKb data set. Thus we find that the hybrid method for choosing keywords is more useful in a domain with confused keywords between categories such as the WebKB data set.

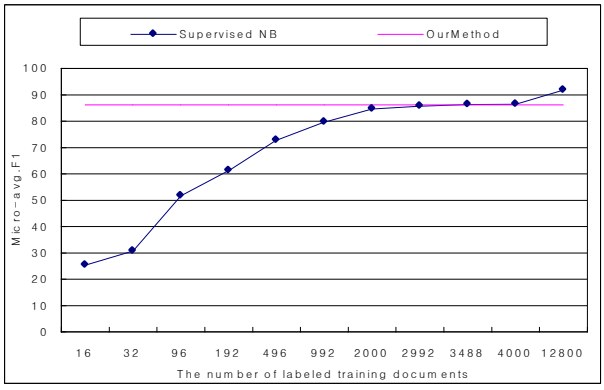
6.2 The Effect of Learning from Small Labeled Training Data to Whole Labeled Training Data

We here observe how many human-labeled documents are required to obtain the performance of the proposed method in each data set. Figure 4 shows the classification performance of the supervised Naive Bayes classifier on the three data sets selected when the number of human-labeled training documents is varied. The horizontal axes indicate the number of human-labeled training documents. Note that, for example, a total of 16 training documents for the Newsgroups data set correspond to one document per category and 12,800 training documents means that we made use of all labeled training documents. The vertical axes indicate the classification performance on the test sets.

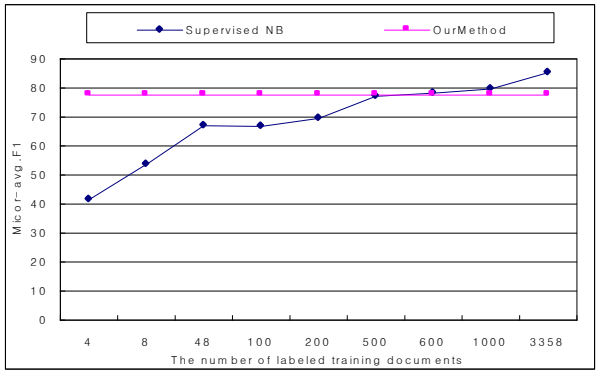
Notice that the achieved performances vary across the different data sets and different amounts of labeled data. A reason for this lies not only in the separateness of categories but also in the number of categories. Generally, the more labeled data there is, the better performance is. For each data 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 4, for the Newsgroups data set, the similar performance to that of the proposed method is achieved when 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. Moreover, the performances from the supervised classifiers with whole labeled training data do not show much difference in comparison with those of the proposed method.

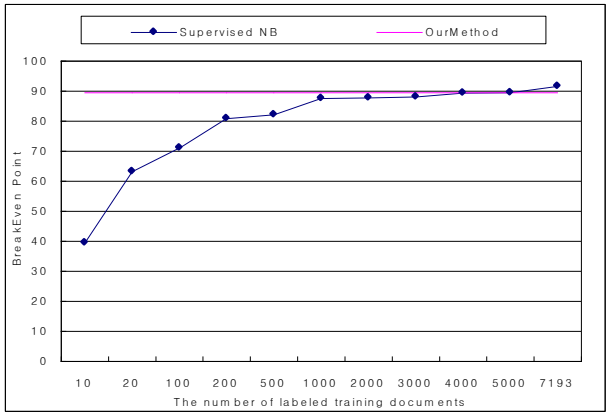
Figure 4. The effect of the training set size on the classification performance of a supervised Naive Bayes classifier in each data set and the comparison with the performance of the proposed method. The horizontal axes indicate the number of labeled training documents



(a) The Newsgroups data set



(b) The WebKB data set



(C) The Reuters data set

7. CONCLUSIONS AND FUTURE WORK

This paper has addressed a new unsupervised or semi-supervised text classification method. Though the proposed method uses only title words and unlabeled data, it shows reasonably comparable performance to the supervised Naive Bayes classifier. Moreover, it outperforms a clustering method, ***sIB***. Labeled data is expensive while unlabeled data is inexpensive and plentiful. Therefore, the proposed method is useful for low-cost text classification. Furthermore, if some text classification tasks require high accuracy, the proposed method can be used as an assistant tool for easily creating training data.

Since the proposed method depends on title words and the number of keywords, we need additional studies for the characteristics of candidate words for title words and the number of input keywords according to different kinds of data set.

Acknowledgement

This paper was supported by Dong-A University Research Fund in 2008.

REFERENCES

Adami, G., Avesani, P., and Sona, D. (2003). Bootstrapping for Hierarchical Document Classification. In *Proceedings of the International Conference on Information Knowledge Management*.

Brill, E. (1995). Transformation-Based Error-driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*, Vol. 21, No. 4.

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

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

Ghani, R., (2002). Combining Labeled and Unlabeled Data for MultiClass Text Categorization, In *Proceedings of International Conference on Machine Learning* (ICML 2002), pp. 8-12.

Jeon, B. and Landgrebe, D. (1999). Partially Supervised Classification Using Weighted Unsupervised Clustering. *IEEE Transaction On Geoscience and Remote Sensing*, Vol. 37, No. 2, pp.1073-1079.

Joachims, T. (2001). *Learning to Classify Text Using Support Vector Machines.* The dissertation for the degree of Doctor of Philosophy, Kluwer Academic Publishers.

Karov, Y. and Edelman, S. (1998). Similarity-based Word Sense Disambiguation. *Computational Linguistics*, Vol. 24, No. 1, pp. 41-60.

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

Ko, Y. and Seo, J. (2002). Text Categorization using Feature Projections. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'2002)*, pp. 467-473.

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

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

Liu, B., Lee, W., Yu, P., and Li, X. (2002). Partially Supervised Classification of Text Documents. In *Proceedings of 19th International Conference on Machine Learning*, pp. 387-394.

Maarek, Y., Berry, D., and Kaiser, G. (1991). An Information Retrieval Approach for Automatically Construction Software Libraries. *IEEE Transaction on Software Engineering*, Vol. 17, No. 8, pp. 800-813.

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

McCallum, A. and Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *AAAI '98 workshop on Learning for Text Categorization*, pp. 41-48.

McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (1999). A Machine Learning Approach to Building Domain-Specific Search Engines. In *Proceedings of The Sixteenth International Joint Conference on Artificial Intelligence* (IJCAI-99).

McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, Vol. 3, No. 2, pp. 127-163.

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

Nigam, K.P. (2001) *Using Unlabeled Data to Improve Text Classification*. The dissertation for the degree of Doctor of Philosophy.

Roy, N. and McCallum, A. (2001). Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Proceedings of 18th International Conference on Machine Learning*, pp.441-448.

Salton, G. and Buckley, C. (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, Vol. 24, pp.513-523.

Slonim, N., Friedman, N., and Tishby, N. (2002). Unsupervised Document Classification using Sequential Information Maximization. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 129-136.

Tong, S., and Koller, D. (2001). Support Vector Machine Active Learning with Application to Text Classification. *Journal of Machine Learning Research*, Vol. 2, pp. 45-66.

Urena-lopez, L.A., Buenaga, M., and Gomez, J.M. (2001). Integrating Linguistic Resources in TC through WSD. *Computers and the Humanities*, Vol. 35, pp. 215-230.

Yang, Y. and Pedersen, J.P. (1997). A Comparative Feature Selection in Statistical Learning of Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412-420.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, Vol. 1, No. 1/2, pp. 67-88.

Yang, Y., Slattery, S., and Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, Vol. 18, No. 2.

Yarowsky, D. (1994). Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, pp. 88-95.

Appendix

We here explain how to calculate the word weight of formula 6 in section 3.2.3. The weight of a word in formula 6 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.

Global frequency: Frequent words in total contexts are less informative of context similarity as follows:

$$1 - \frac{freq\ (w_i)}{\max\ freq} \tag{A.1}$$

where *maxfreq* is the value of the highest frequency in total contexts.

Log-likelihood factor: Generally, 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 as follows:

$$\log \left\{ \frac{\Pr(w_i \mid CC)}{\Pr(w_i)} + 1 \right\} + 1 \tag{A.2}$$

where $Pr(w_i)$ is estimated from the frequency of w_i in the total contexts, and $Pr(w_i|CC)$ from the frequency of w_i in centroid-contexts. 1 is assigned to the words which do not appear in centroid-contexts.

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

The weight of a word, $F(w_i, X)$, is the product of the above factors and each weight are normalized by the sum of weights of words in a context as follows:

$$weight \quad (w_i, X) = \frac{F(w_i, X)}{\sum_{w_j \in X} F(w_j, X)} \tag{A.3}$$