ELSEVIER

Information Processing and Management 40 (2004) 65–79

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Improving text categorization using the importance of sentences

Youngjoong Ko *, Jinwoo Park, Jungyun Seo

Department of Computer Science, NLP lab., Sogang University, Sinsu-dong 1, Mapo-gu, Seoul 121-742, South Korea

Received 18 July 2002; accepted 18 July 2002

Abstract

Automatic text categorization is a problem of assigning text documents to pre-defined categories. In order to classify text documents, we must extract useful features. In previous researches, a text document is commonly represented by the term frequency and the inverted document frequency of each feature. Since there is a difference between important sentences and unimportant sentences in a document, the features from more important sentences should be considered more than other features. In this paper, we measure the importance of sentences using text summarization techniques. Then we represent a document as a vector of features with different weights according to the importance of each sentence. To verify our new method, we conduct experiments using two language newsgroup data sets: one written by English and the other written by Korean. Four kinds of classifiers are used in our experiments: Naive Bayes, Rocchio, *k*-NN, and SVM. We observe that our new method makes a significant improvement in all these classifiers and both data sets.

Keywords: Text categorization; Importance of sentence; Text summarization techniques; Indexing technique; Text classifier

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 categorized documents. For examples, there are Naive Bayes (McCallum &

* Corresponding author. Tel.: +82-2-706-8954; fax: +82-2-704-8273.

E-mail addresses: kyj@nlpzodiac.sogang.ac.kr (Y. Ko), jwpark@diquest.com (J. Park), seojy@ccs.sogang.ac.kr (J. Seo).

0306-4573/$ - see front matter
doi: 10.1016/S0306-4573(02)00056-0

Nigam, 1998; Ko & Seo, 2000), Rocchio (Lewis, Schapire, Callan, & Papka, 1996), Nearest Neighbor (Yang, Slattery, & Ghani, 2002), and Support Vector Machines (Joachims, 1998).

A text categorization task consists of the training phase and the text classification phase. The former includes the feature extraction process and the indexing process. The vector space model has been used as a conventional method for text representation (Salton, Fox, & Wu, 1983). This model represents a document as a vector of features using term frequency (TF) and inverted document frequency (IDF). This model simply counts TF without considering where the term occurs. But each sentence in a document has different importance for identifying the content of the document. Thus, by assigning a different weight according to the importance of the sentence to each term, we can achieve better results. For this problem, several techniques have been studied. In the first technique, term weights are differently weighted by the location of a term, so that the structural information of a document is applied to term weights (Murata et al., 2000). But this method supposes that only several sentences, which are located at the front or the rear of a document, have the important meaning. Hence it can be applied to only documents with a fixed form such as articles. The next technique uses the title of a document in order to choose the important terms (Mock, 1996). The terms in the title are handled importantly. But a drawback of this method is that some titles, which do not properly contain the meaning of the document, can rather increase the ambiguity of the meaning. This case often comes out in documents with an informal style such as *Newsgroup* and *Email*.

To overcome these problems, we have studied text summarization techniques. Among text summarization techniques, there are statistical methods and linguistic methods (Goldstein, Kantrowitz, Mittal, & Carbonell, 1999; Marcu, 1999; Radev, Jing, & Stys-Budzikowska, 2000). Since the former methods are simpler and faster than the latter methods, we choose the former methods to be applied to text categorization. Therefore, we employ two kinds of text summarization techniques; one measures the importance of sentences by the similarity between the title and each sentence in a document, and the other by the importance of terms in each sentence.

In this paper, we use the summarization techniques for classifying important sentences and unimportant sentences. The importance of each sentence is measured by them. Then the term weights in each sentence are modified in proportion to the calculated sentence importance. To test our proposed method, we used two different newsgroup data sets; one is a well known data set, the Newsgroup data set by Ken Lang, and the other was gathered from Korean UseNet discussion group. As a result, our proposed method showed a better performance than the basis system in both data sets.

The rest of this paper is organized as follows. Section 2 explains the proposed text categorization system in detail. In Section 3, we simply describe other classifiers used in our experiments. Section 4 presents the discussion of empirical results in our experiments and the analysis of our indexing method. The final section presents conclusions and future works.

2. The proposed text categorization system

The proposed system consists of two modules as shown in Fig. 1: one module for training phase and the other module for text classification phase. The each process of Fig. 1 is explained in the following sections in detail.

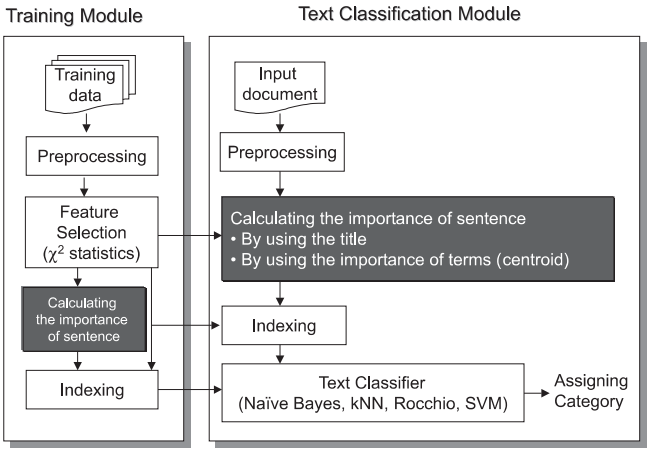

Fig. 1. Overview of the proposed system.

2.1. Pre-processing

A document from newsgroup data is composed of author, date, group, server, message ID, title, and body. In our system, we use only the contents of title and body.

The contents of documents are segmented into sentences. Then we extract content words from each sentence and represent each sentence as a vector of content words. To extract the content words, we use two kinds of POS taggers: Brill tagger for English and Sogang POS tagger for Korean. We consider words annotated with noun or verb tag as content words.

2.2. Feature selection

The size of vocabulary in our experiment is selected by ranking words according to their χ^2 statistics values with respect to the category (Yang & Pedersen, 1997). Using the two-way contingency table of a word t and a category c, the word-goodness measure is defined as follows:

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)} \tag{1}$$

where A is the number of times t and c co-occur, B is the number of times t occurs without c, C is the number of times c occurs without t, D is the number of times neither c nor t occurs, and 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:

$$\chi^2(t) = \max_{i=1}^{m} \{\chi^2(t,c_i)\} \tag{2}$$

where m denotes the number of categories.

The above χ^2 statistics values are calculated from training data.

2.3. Measuring the importance of sentences

The importance of each sentence is measured by two methods. First, the sentences, which are more similar to the title, have higher weights. In the next method, we first measure the importance of terms by TF, IDF, and χ^2 statistics values. Then we assign the higher importance to the sentence with more important terms. Finally, the importance of a sentence is calculated by combination of two methods.

2.3.1. The importance of sentences by the title

Generally, we believe that a title summarizes the important content of a document (Endres-Niggemeyer et al., 1998). By Mock (1996), terms that occur in the title have higher weights. But the effectiveness of this method depends on the quality of the title. In many cases, the titles of documents from Newsgroup or Email do not properly represent the contents of these documents. Hence we use the similarity between each sentence and the title instead of directly using terms in the title. The similar sentences to the title contain the important terms generally. For example, let us consider the following title; "I have a question." This title does not contain any meaning about the contents of a document. Nevertheless, sentences with the term, 'question', should be handled importantly because they can have key terms about the question.

We measure the similarity between the title and each sentence, and then we assign the higher importance to the sentences with the higher similarity. The title and each sentence of a document are represented as the vectors of content words. The similarity value of them is calculated by the inner product and the calculated values are normalized into values between 0 and 1 by a maximum value. The similarity value between the title T and the sentence S_i in a document d is calculated by the following formula:

$$\mathrm{Sim}(S_i, T) = \frac{\vec{S}_i \cdot \vec{T}}{\max_{S_j \in d}(\vec{S}_j \cdot \vec{T})} \tag{3}$$

where $\vec{T}$ denotes a vector of the title, and $\vec{S}_i$ denotes a vector of sentence.

2.3.2. The importance of sentences by the importance of terms

Since the method by the title depends on the quality of the title, it can be useless in the document with a meaningless title or no title. Besides, the sentences, which do not contain important terms, need not be handled importantly although they are similar to the title. On the contrary, sentences with important terms must be handled importantly although they are dissimilar to the title.

Considering these points, we first measure the importance values of terms by TF, IDF, and χ^2 statistics values, and then the sum of the importance values of terms in each sentence is assigned to the importance value of the sentence. In this method, the importance value of a sentence S_i in a document d is calculated as follows:

$$\mathrm{Cen}(S_i) = \frac{\sum_{t \in S_i} \mathrm{tf}(t) \times \mathrm{idf}(t) \times \chi^2(t)}{\max_{S_j \in d} \left\{ \sum_{t \in S_j} \mathrm{tf}(t) \times \mathrm{idf}(t) \times \chi^2(t) \right\}} \tag{4}$$

where $\mathrm{tf}(t)$ denotes the term frequency of term t, $\mathrm{idf}(t)$ denotes the inverted document frequency, and $\chi^2(t)$ denotes the χ^2 statistics values.

2.3.3. The combination of two sentence importance values

Two kinds of sentence importance are simply combined by the following formula:

$$\mathrm{Score}(S_i) = 1.0 + k_1 \times \mathrm{Sim}(S_i, T) + k_2 \times \mathrm{Cen}(S_i) \tag{5}$$

In formula (5), the k_1 and k_2 are constant weights, which control the rates of reflecting two importance values. The 1.0 constant value is added to a calculated sentence importance value in order to prevent the modified TF value having lower value than original TF value by formula (6):

2.3.4. The indexing process

The importance value of a sentence by formula (5) is used for modifying the TF value of a term. That is, since a TF value of a term in a document is calculated by the sum of the TF values of terms in each sentence, the modified TF value ($\mathrm{WTF}(d,t)$) of the term t in the document d is calculated by formula (6):

$$\mathrm{WTF}(d,t) = \sum_{S_i \in d} \mathrm{tf}(S_i, t) \times \mathrm{Score}(S_i) \tag{6}$$

where $\mathrm{tf}(S_i, t)$ denotes TF of the term t in sentence S_i.

By formula (6), the terms, which occur in a sentence with the higher importance value, have higher weights than the original TF value. In our proposed method, we compute the weight vectors for each document using the WTF and the conventional TF–IDF scheme (Salton & Buckley, 1988). The weight of a term t in a document d is calculated as follows:

$$w(d,t) = \frac{\mathrm{WTF}(d,t) \times \log\left(\frac{N}{n_t}\right)}{\sqrt{\sum_{i=1}^{T}\left[\mathrm{WTF}(d,t_i) \times \log\left(\frac{N}{n_{t_i}}\right)\right]^2}} \tag{7}$$

where N is the number of documents in the training set, T is the number of features limited by feature selection, and n_t is the number of training documents in which t occurs.

The weight by formula (7) is used in k-NN, Rocchio, and SVM. But Naive Bayes classifier uses only WTF value.

3. Other classifiers used in our experiments

To verify our proposed system, we built Naive Bayes, Rocchio, k-NN, and SVM classifier. In this section, these classifiers are briefly described.

3.1. Naive Bayes

Naive Bayes probabilistic classifiers are also commonly used in text categorization (McCallum & Nigam, 1998; Ko & Seo, 2000; Yang et al., 2002). The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document. The naive part of such a model is the assumption of word independence. We use the Naive Bayes

classifier with minor modifications based on Kullback–Leibler divergence (Ko & Seo, 2000; Craven et al., 2000). Given a document d for classification, we calculate the probabilities of each category c_j as follows:

$$Pr(c_j|d) = \frac{Pr(c_j)Pr(d|c_j)}{Pr(d)} = \Pr(c_j)\prod_{i=1}^{T}\Pr(t_i|c_j)^{N(t_i|d)}$$
$$\propto \frac{\log Pr(c_j)}{n} + \sum_{i=1}^{T} Pr(t_i|d)\log\left(\frac{Pr(t_i|c_j)}{Pr(t_i|d)}\right) \quad (8)$$

where n is the number of words in document d, t_i is the ith word in the vocabulary, T is the size of the vocabulary, $N(t_i|d)$ is the frequency of word t_i in document d.

The category predicted by this classifier for a given document is simply a category with the highest score. Here, the expected likelihood estimator (Li & Yamanish, 1997) is used for the estimates of the probability of word t_i in category c and that of word t_i in document d as follows:

$$Pr(t_i|c) = \frac{N(t_i|c) + 0.5}{\sum_{j=1}^{T_c} N(t_j|c) + 0.5 \times T_c} \quad (9)$$

$$Pr(t_i|d) = \frac{N(t_i|d) + 0.5}{\sum_{j=1}^{T_d} N(t_j|d) + 0.5 \times T_d} \quad (10)$$

where $N(t_i|c)$ is the frequency of word t_i in category c. In the proposed method, $N(t_i|c)$ and $N(t_i|d)$ in formulae (9) and (10) are calculated by WTF instead of TF.

3.2. Rocchio

Rocchio is an effective method using relevance judgments for query expansion in information retrieval and filtering (Lewis et al., 1996; Salton & Buckley, 1990). 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 prototype or centroid, is constructed by combining document vectors into a prototype vector $\vec{c}_i$ for each category c_i. 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:

$$\vec{c}_i = \alpha\frac{1}{|c_i|}\sum_{\vec{d}\in c_i}\vec{d} - \beta\frac{1}{|N - c_i|}\sum_{\vec{d}\in N - c_i}\vec{d} \quad (11)$$

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

3.3. k-Nearest Neighbor (k-NN) algorithm

As an instance-based classification method, k-NN has been an effective approach to a broad range of pattern recognition and text classification problems (Duda, Hart, & Stork, 2001; Yang

et al., 2002). In k-NN algorithm, a new input instance should belong to the same category as their 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 the k-nearest neighbors among all stored training instances.

We use the conventional vector space model in text categorization, which represents each document as a vector of term weights and the similarity between two documents is measured by using the cosine value of the angle between the corresponding vectors (Yang et al., 2002).

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

$$s(c_i, \vec{d}) = \sum_{\vec{d}' \in R_k(\vec{d}) \cap D_i} \cos(\vec{d}', \vec{d}) \tag{12}$$

where $R_k(\vec{d})$ is the set of k-nearest neighbors of document d, and D_i is the set of training documents in category c_i.

By sorting the scores of all candidate categories, we obtain a ranked list of categories for each test document. Since documents of our data sets have only one category, we assign the only top ranking category to each document.

3.4. Support vector machines

Support vector machines (SVM) was proposed by Vapnic (1995) for solving two-class pattern recognition problems. 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:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{13}$$

$\vec{x}$ 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 (Vapnic, 1995). 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 multi-class classifier. There are several methods of multi-class classifier for SVM (Weston & Watkins, 1999). We employ the *One-Against-the Rest* method among them. This method requires k binary classifiers $f_{c_i}(x)$ $(1 \leqslant c_i \leqslant k)$ to be constructed for all categories. In training for category c_i, all data of category c_i 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:

$$c = \arg\max_{c_i} f_{c_i}(x) \tag{14}$$

Joachims (1998) implemented an efficient SVM toolkit, SVM$^{\text{light}}$. This toolkit was used in our experiments.

4. Empirical evaluation

4.1. Data sets and experimental settings

To test our proposed system, we used two newsgroup data sets written by two different languages: English and Korean. Each document in both data sets has only one category.

The *Newsgroups* data set, collected by Ken Lang, contains about 20,000 documents evenly divided among 20 UseNet discussion groups (McCallum & Nigam, 1998). 4000 documents (20%) were used for test data and the remaining 16,000 documents (80%) for training data. Then 4000 documents from training data were selected for a validation set. After all parameter values of our experiments were set from the validation set, we evaluated our method by using the fivefold cross-validation method. That is, each data set was split into five subsets, and each subset was used once as test data in a particular run while the remaining subsets were used as training data for that run. The split into training and test sets for each run was 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 vocabulary from training data has 51,018 words (with no stemming).

The second data set was gathered from the Korean UseNet group. This data set contains a total of 10,331 documents and consists of 15 categories. 3107 documents (30%) are used for test data and the remaining 7224 documents (70%) for training data. The resulting vocabulary from training data has 69,793 words. This data set is an uneven data set as shown in Table 1.

We used χ^2 statistics for the statistical feature selection (Yang & Pedersen, 1997). To evaluate our method, we implemented Naive Bayes, k-NN, Rocchio, and SVM classifier. The k in k-NN was set to 30, and $\alpha = 16$ and $\beta = 4$ were used in the Rocchio classifier. This choice was based on

Table 1
The constitution of Korean newsgroup data set

Category	Training data	Test data	Total
han.arts.music	315	136	451
han.comp.database	198	86	284
han.comp.devtools	404	174	578
han.comp.lang	1387	595	1982
han.comp.os.linux	1175	504	1679
han.comp.os.window	517	222	739
han.comp.sys	304	131	435
han.politics	1469	630	2099
han.rec.cars	291	126	417
han.rec.games	261	112	373
han.rec.movie	202	88	290
han.rec.sports	130	56	186
han.rec.travel	102	45	147
han.sci	333	143	476
han.soc.religion	136	59	195
Total	7224	3107	10,331

previous parameter optimization learned by the validation set. For SVM, we used the linear model offered by SVM^{light}.

As performance measures, we followed the standard definition of recall (r), precision (p), and F_1 measure ($2rp/(r+p)$). For evaluating an average performance across categories, we used the *micro-averaging method* and *macro-averaging method* (Yang, 1999). The recall, precision and F_1 measure can first be 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 compute the global recall, precision and F_1 values for that global pool. Since a document in our 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.

4.2. Experimental results

We tested our system through the following steps. First, using the validation set of *Newsgroup* data set, we set the number of features and the constant weights (k_1 and k_2) in the combination of two importance values in the Section 2.3.3. Then, using the resulting values, we conducted experiments and compared a basis system with our proposed system; the former system used the conventional TF for Naive Bayes and conventional TF–IDF for the other classifiers, and the latter system used WTF by formula (6) for Naive Bayes and the weights by formula (7) for the other classifiers.

4.2.1. Setting the number of features

First of all, we set the number of features in each classifier using the validation set of training data. The number of features in this experiment was limited by feature selection. Fig. 2 displays the performance curves for the proposed system and the basis system using SVM. We simply set both constant weights (k_1 and k_2) to 1.0 in this experiment.

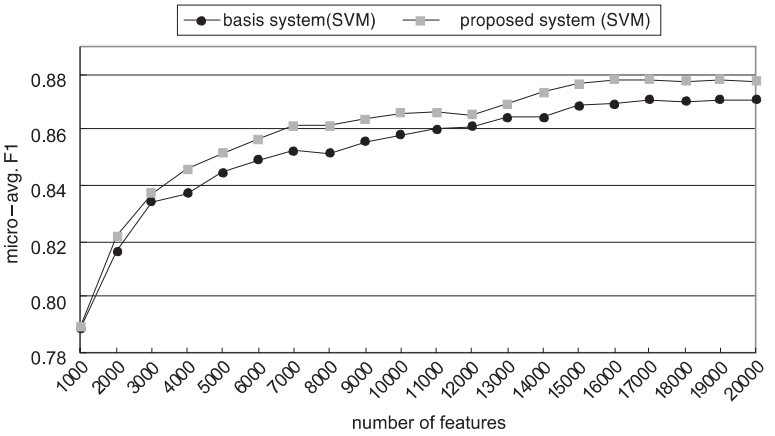

Fig. 2. Comparison of proposed system and basis system using SVM.

As shown in Fig. 2, the proposed system achieved a better performance than the basis system over all intervals. We set the number of features for SVM to 7000 with regard to the convergence of the performance curve and running time. By the similar method, the number of features in other classifiers was set: 7000 for Naive Bayes, 10,000 for Rocchio, and 9000 for k-NN. Note that, over all intervals and all classifiers, the performance of the proposed system was better than that of the basis system.

4.2.2. Setting the constant weights k_1 and k_2

Prior to the experiment for setting the constant weights, we evaluated two importance measure methods and their combination method individually; we used simply the same value for k_1 and k_2 ($k_1 = k_2$) in the combination method. We observed the results in each interval when constant weights were changed from 0.0 to 3.0. In Fig. 3, $\mathrm{Sim}(S,T)$ denotes the method by the title, $\mathrm{Cen}(S)$ denotes the method by the importance of terms, and $\mathrm{Sim}(S,T) + \mathrm{Cen}(S)$ denotes the method by the combination method.

In this experiment, we used SVM as a classifier and set feature number to 7000. We mostly obtained a best performance in the combination method.

In order to set the constant weights k_1 and k_2 in each classifier, we carried out a total of 900 trials on the validation set because each method had 30 intervals from 0.0 to 3.0 (interval size: 0.1). As a result, we obtained the best performance at 1.5 (k_1) and 0.4 (k_2) for SVM: 1.9 and 3.0 for Naive Bayes, 2.0 and 0.0 for Rocchio, and 0.8 and 2.8 for k-NN. These constant weights of each classifier were used in the following experiments.

4.2.3. Results in two newsgroup data sets

In this section, we report the results in two newsgroup data sets (see Tables 2 and 3).

In both data sets, the proposed system produced a better performance in all four classifiers. As a result, our proposed method can improve the classification performance with all four classifiers in both English and Korean.

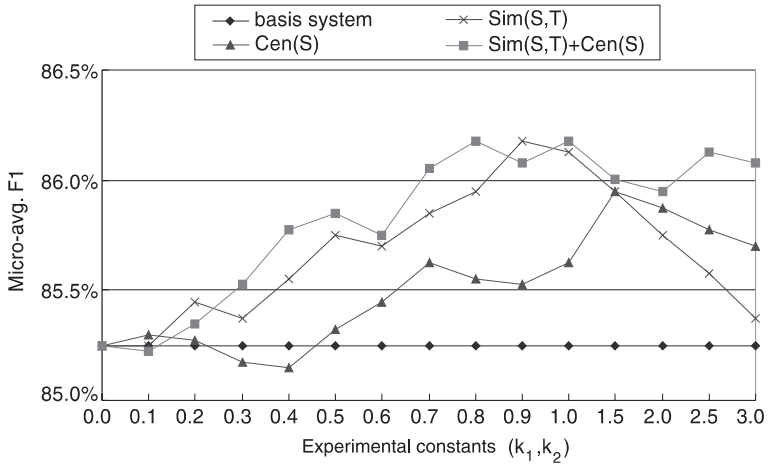

Fig. 3. Comparison of importance measure methods in different constant weights (k_1 and k_2).

Table 2
Results in English newsgroup data set

	Naive Bayes, F_1		Rocchio, F_1		k-NN, F_1		SVM, F_1	
	Basis system	Proposed system	Basis system	Proposed system	Basis system	Proposed system	Basis system	Proposed system
Atheism	76.8	**77.4**	67.6	**68.8**	70.8	**71.6**	75.4	**76.8**
Graphics	75.2	**77.1**	70.1	**72.8**	**76.9**	76.5	79.9	**80.7**
windows.misc	65.8	**73.2**	73.3	**74.0**	74.9	**76.0**	81.8	**83.2**
pc.hardware	69.8	**72.0**	67.4	**68.4**	68.5	**72.7**	74.7	**75.5**
mac.hardware	81.6	**83.9**	80.0	**80.4**	77.3	**81.6**	86.6	**87.3**
windows.x	86.3	**87.3**	82.5	**82.7**	83.6	**83.7**	88.0	**88.3**
Forsale	79.7	**81.4**	77.9	77.9	69.1	**75.5**	**81.7**	81.1
Autos	89.7	**91.0**	85.6	**87.6**	86.2	**87.0**	90.9	**91.5**
Motorcycles	94.4	94.4	**91.2**	91.0	90.9	**91.4**	95.5	95.5
Baseball	96.5	**96.6**	92.7	**93.6**	**94.4**	94.0	95.3	**95.5**
Hockey	97.7	**97.9**	94.1	**94.4**	**94.6**	94.5	97.0	97.0
Crypt	93.9	**94.3**	88.8	**89.2**	92.3	**92.7**	94.6	**95.5**
Electronics	76.9	**78.1**	68.5	**71.3**	73.9	**74.6**	81.1	**81.3**
Med	**92.2**	91.8	87.8	**88.4**	89.0	**90.1**	92.0	92.0
Space	91.6	**92.2**	89.3	89.3	**91.3**	91.1	93.5	**93.8**
Christian	**91.9**	91.8	79.5	**80.3**	86.8	**90.4**	**97.1**	96.6
Guns	**83.5**	83.4	77.1	**79.2**	**83.6**	82.9	85.4	**86.2**
Midest	91.5	**92.6**	91.2	**91.8**	90.9	**92.3**	93.2	**93.5**
politics.misc	71.4	**71.9**	68.3	**69.3**	70.1	**72.2**	75.1	**75.6**
religion.misc	**51.6**	51.4	35.2	**41.0**	58.5	**62.0**	60.9	**64.6**
Macro-avg	83.0	**84.1**	78.6	**79.7**	81.2	**82.6**	86.0	**86.6**
Micro-avg	83.3	**84.4**	79.1	**80.0**	81.4	**82.7**	86.1	**86.6**

4.3. Verifying our indexing method

Salton argued that a collection of small tightly clustered documents with wide separation between individual clusters should produce the best performance (Salton, Yang, & Wang, 1975). In this light, we employed the method used by Salton et al. (1975) to verify our method. We then conducted experiments in English newsgroup data set (*Newsgroup* data set) and observed the resulting values.

We define the cohesion within a category and the cohesion between categories. The cohesion within a category is a measure for similarity values between documents in the same category. The cohesion between categories is a measure for similarities between categories. The former is calculated by formula (16), and the latter is calculated by formula (17):

$$\vec{C}_k = \frac{1}{|I_k|}\sum_{\vec{d}\in I_k}\vec{d}, \quad \vec{C}_{\text{glob}} = \frac{1}{|D|}\sum_{k=1}^{K}|I_k|\cdot\vec{C}_k \tag{15}$$

$$Co_{\text{within}} = \frac{1}{|D|}\sum_{k=1}^{K}\sum_{\vec{d}\in I_k}\vec{d}\cdot\vec{C}_k \tag{16}$$

Table 3
Results in Korean newsgroup data set

	Naive Bayes, F_1		Rocchio, F_1		k-NN, F_1		SVM, F_1	
	Basis system	Proposed system	Basis system	Proposed system	Basis system	Proposed system	Basis system	Proposed system
Music	68.8	**77.1**	61.0	**74.4**	72.3	**80.6**	86.5	**89.0**
Database	66.7	**71.7**	74.3	**75.0**	71.3	**76.6**	81.6	**82.7**
Devtools	**69.0**	68.0	**64.6**	63.2	57.3	**60.1**	72.6	**73.0**
Lang	80.6	**81.1**	79.2	**80.1**	79.8	**80.7**	85.1	**85.4**
Linux	73.9	**78.3**	79.8	**82.2**	78.0	**78.9**	86.9	86.9
Window	68.8	**71.0**	67.7	**69.7**	**70.6**	70.5	77.7	**78.7**
Sys	65.1	**68.8**	58.4	**62.8**	51.0	**62.3**	74.6	**75.2**
Politics	95.0	**95.2**	93.4	**94.7**	**94.0**	93.8	95.6	95.6
Cars	91.2	**92.6**	**88.0**	87.8	86.8	**87.8**	90.3	**90.7**
Games	65.0	**68.0**	68.2	**72.0**	70.2	**73.6**	76.4	**77.4**
Movie	79.8	**85.4**	84.0	**84.6**	88.4	**89.1**	89.0	**90.1**
Sports	82.9	**83.2**	**87.2**	84.7	**89.3**	88.5	91.9	91.9
Travel	81.6	**84.2**	**83.9**	77.2	76.7	**78.9**	88.4	**89.4**
Sci	84.9	**85.4**	74.7	**79.7**	**80.6**	80.0	79.7	**82.2**
Religion	86.4	**88.2**	82.4	**85.5**	**94.0**	93.2	91.8	**91.9**
Macro-avg	78.4	**80.8**	76.5	**78.2**	77.4	**79.6**	84.5	**85.3**
Micro-avg	79.1	**81.3**	78.7	**80.1**	79.9	**81.3**	86.0	**86.5**

$$Co_{\text{between}} = \frac{1}{|D|} \sum_{k=1}^{K} |I_k| \cdot \left(\vec{C}_{\text{glob}} \cdot \vec{C}_k \right) \tag{17}$$

where D denotes the total training document set, I_k denotes training document set in kth category, $\vec{C}_k$ denotes a centroid vector of kth category, and $\vec{C}_{\text{glob}}$ denotes a centroid vector of the total training documents.

In formulae (15), $\vec{C}_k$ is described by the mean vector of each category, and $\vec{C}_{\text{glob}}$ is represented by the mean vector of all training vectors. The cohesion within a category in formula (16) is calculated by averaging cosine similarity values between $\vec{C}_k$ and each document of kth category, and the cohesion between categories in formula (17) is calculated by averaging cosine similarity values between $\vec{C}_{\text{glob}}$ and $\vec{C}_k$.

An indexing method with a high cohesion within a category and a low cohesion between categories should produce a better performance in text categorization. First, we measured the cohesion within a category in each indexing method: the basis method by the conventional TF–IDF, the method by the title ($\text{Sim}(S, T)$), the method by the importance of terms ($\text{Cen}(S)$), and the method by the combination method ($\text{Sim}(S, T) + \text{Cen}(S)$). Fig. 4 shows the resulting curves in each different constant weight; we simply used the same values for k_1 and k_2 in the combination method.

As shown in Fig. 4, $\text{Cen}(S)$ shows the highest cohesion value, but $\text{Sim}(S, T)$ does not have an effect on the cohesion values in comparison with the method by the conventional TF–IDF.

Fig. 5 displays the resulting curves of the cohesion between categories as the same manner in Fig. 4.

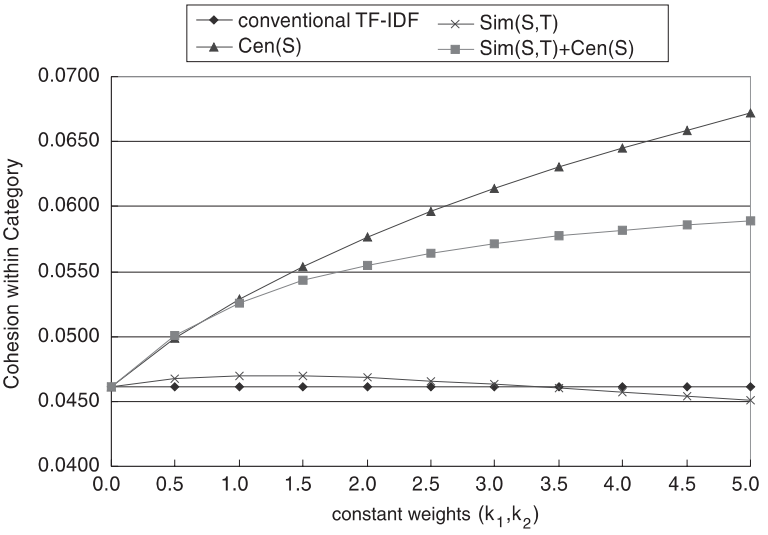

Fig. 4. The cohesion within a category.

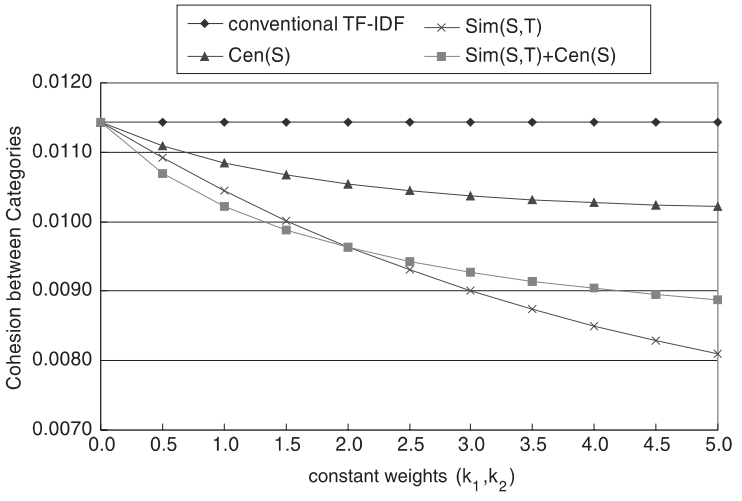

Fig. 5. The cohesion between categories.

We obtained the lowest cohesion value in $\mathrm{Sim}(S,T)$. Using $\mathrm{Cen}(S)$, the resulting cohesion values are slightly lower than those of the method by the conventional TF–IDF. In both Figs. 4 and 5, the cohesion values of the combination method show the middle values between $\mathrm{Sim}(S,T)$ and $\mathrm{Cen}(S)$.

By the results in Figs. 4 and 5, we can observe that our proposed indexing method reforms the vector space for a better performance: the high cohesion within a category and the low cohesion between categories. Using the proposed indexing method, the document vectors in a category are located more closely and individual categories are separated more widely. These effects were also

Table 4
Top performance by two different methods

	k-NN		Rocchio	
	$\mathrm{Sim}(S,T)$	$\mathrm{Cen}(S)$	$\mathrm{Sim}(S,T)$	$\mathrm{Cen}(S)$
Macro-avg F_1	80.6	**82.4**	**79.7**	78.8

observed in our experiments. Since k-NN constructs only a local approximation to the target function that applies in the neighborhood of the new document, it has an advantage in a vector space with the high cohesion within a category. On the other hand, Rocchio has an advantage in a vector space with the low cohesion between categories because the prototype or centroid vector of each category in Rocchio is used as a criterion vector for classification. We achieved the similar results in our experiments. That is, k-NN produced a better performance by using $\mathrm{Cen}(S)$, and Rocchio produced a better performance by using $\mathrm{Sim}(S,T)$. Table 4 shows the summarized results in each individual method of k-NN and Rocchio.

5. Conclusions

In this paper, we have presented a new indexing method for text categorization using two kinds of text summarization techniques: one uses the title and the other uses the importance of terms. For our experiments, we used two different language newsgroup data sets and four kinds of classifiers. Our system achieved a better performance than the basis system did in all these classifiers and both two languages. Then we verified the effect of the proposed indexing method by measuring two kinds of cohesion. We confirmed that the proposed indexing method can reform the document vector space for a better performance in text categorization. As a future work, we need the additional research for applying more structural information of document to text categorization techniques.

Acknowledgement

This work is supported by Institute for Applied Science and Technology of Sogang University.

References

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (2000). Learning to construct knowledge bases from the world wide web. *Artificial Intelligence, 118*(1–2), 69–113.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). New York: John Wiley and Sons.

Endres-Niggemeyer, B., Haseloh, K., Müller, J., Peist, S., Sigel, I. S., Sigel, A., Wansorra, E., Wheeler, J., & Wollny, B. (1998). *Summarizing information* (pp. 307–338). Berlin: Springer-Verlag.

Goldstein, J., Kantrowitz, M., Mittal, V. O., & Carbonell, J. (1999). Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of the 23rd international ACM SIGIR conference on research and development in information retrieval (SIGIR'99)*.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In *European conference on machine learning (ECML)* (pp. 137–142).

Ko, Y., & Seo, J. (2000). Automatic text categorization by unsupervised learning. In *Proceedings of the 18th international conference on computational linguistics (COLING'2000)* (pp. 453–459).

Lewis, D. D., Schapire, R. E., Callan, J. P., & 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).

Li, H., & Yamanish, K. (1997). Document classification using a finite mixture model. In *Proceedings of the 35th annual meeting of the association for computational linguistics*, *ACL'97*.

Marcu, D. (1999). Discourse trees are good indicators of importance in text. In *Advances in automatic text summarization* (pp. 123–136). Cambridge, MA: MIT Press.

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In *AAAI'98 workshop on learning for text categorization* (pp. 41–48).

Mock, K. J. (1996). Hybrid hill-climbing and knowledge-based techniques for intelligent news filtering. In *Proceedings of the national conference on artificial intelligence (AAAI'96)*.

Murata, M., Ma, Q., Uchimoto, K., Ozaku, H., Isahara, H., & Utiyama, M. (2000). Information retrieval using location and category information. *Journal of the Association for Natural Language Processing, 7*(2).

Radev, D. R., Jing, H., & Stys-Budzikowska, M. (2000). Summarization of multiple documents: clustering, sentence extraction, and evaluation. In *Proceedings of ANLP-NAACL workshop on automatic summarization*.

Salton, G., Yang, C., & Wang, A. (1975). A vector space model for automatic indexing. *Communications of the ACM, 18*(11), 613–620.

Salton, G., Fox, E. A., & Wu, H. (1983). Extended Boolean information retrieval. *Communications of the ACM, 26*(12), 1022–1036.

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management, 24*, 513–523.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of American Society for Information Sciences, 41*, 288–297.

Vapnic, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer.

Weston, V., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the seventh European symposium on artificial neural networks (ESANN-99)*.

Yang, Y., & Pedersen, J. P. (1997). Feature selection in statistical learning of text categorization. In *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, 1*(1/2), 67–88.

Yang, Y., Slattery, S., & Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems, 18*(2).