JCSE Journal of Computing Science and Engineering

Regular Paper

*Journal of Computing Science and Engineering,*
*Vol. 5, No. 2, June 2011, pp. 150-160*

# Issues and Empirical Results for Improving Text Classification

**Youngjoong Ko**
Deptartment of Computer Engineering, Dong-A University, Busan, Korea **yjko@dau.ac.kr**

**Jungyun Seo***
Deptartment of Computer Engineering, Sogang University, Seoul, Korea **seojy@sogang.ac.kr**

## Abstract

Automatic text classification has a long history and many studies have been conducted in this field. In particular, many machine learning algorithms and information retrieval techniques have been applied to text classification tasks. Even though much technical progress has been made in text classification, there is still room for improvement in text classification. In this paper, we will discuss remaining issues in improving text classification. In this paper, three improvement issues are presented including automatic training data generation, noisy data treatment and term weighting and indexing, and four actual studies and their empirical results for those issues are introduced. First, the semi-supervised learning technique is applied to text classification to efficiently create training data. For effective noisy data treatment, a noisy data reduction method and a robust text classifier from noisy data are developed as a solution. Finally, the term weighting and indexing technique is revised by reflecting the importance of sentences into term weight calculation using summarization techniques.

**Category:** Smart and intelligent computing

**Keywords:** Text classification; Semi-supervised learning; Noisy data reduction; Importance of sentence; Term weighting and indexing

## I. INTRODUCTION

In recent years, automatic content-based document management tasks have gained a prominent status in the field of information systems, due to the widespread and continuously increasing availability of documents in digital format. In particular, 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 according to the rapid growth of the World Wide Web [1]. This task is commonly referred to as text classification. Since there has been a recent explosion of electronic texts from not only the World Wide Web but also various online sources (electronic mail, corporate databases, chat rooms, digital libraries, etc.), 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. Within the machine learning paradigm, a general inductive process automatically builds an automatic text classifier by "learning" from a set of previously classified documents. The advantages of this approach are its accuracy comparable to human performance and considerable savings in terms of manpower. In addition, text classification is strongly related to information retrieval, as the foundation of an automated content-based document management. Thus, text classification may be seen as the meeting point of machine learning and information retrieval. Information gain and $\chi^2$ statistics, etc. have been used for feature selection [2], Naive Bayes (NB) [3, 4], Rocchio [5], nearest neighbor (*k*-NN) [6], support vector machine (SVM) [7], etc. have been broadly employed as text classifiers. After SVM was applied to text classification, it has dominated other text classifiers in terms of performance. However, text classification still has many points to be improved such as automatic

Open Access 10.5626/JCSE.2011.5.2.150 http://jcse.kiise.org



Received 28 February 2011, Accepted 28 March 2011
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

training data generation, noisy data reduction, a classifier with robustness from noisy data, term weighting and indexing, etc., especially according to application areas.

In this paper, we enumerate improvement issues for text classification and introduce improvement attempts and their empirical results to be actually applied in classification tasks. Among many improvement issues in text classification, we focus on three important improvement issues on automatic training data generation, noisy data treatment (noisy data reduction and a robust classifier from noisy data), and term weighting and indexing. In supervised-learning based text classification, obtaining good quality training data is very important, whereas labeling tasks for training data must be a painfully time consuming process. To reduce that kind of burden for labeling tasks, semi-supervised or unsupervised learning techniques have been applied to text classification [8, 9]. Even thought those labeling tasks are performed by experts and consume long time, almost all training data can include some noisy data. Removing noisy data is another key improvement issue for text classification. For this, we introduce two different solutions: noisy data reduction in training settings for positive and negative examples [10], and the development of a robust classifier from noisy data [11]. Finally, we pay attention to the term weighting and indexing phases of text classification. Most techniques related to the term weighting and the indexing phases are originated from the Information Retrieval literature. There exists a problem that these techniques are applied to text classification in the same methodology as information retrieval even though text classification and information retrieval have different properties and application environments. The importance measure of sentences in a document by using some text summarization techniques is used for a new term weighting scheme [12].

The rest of this paper is devoted to enumerate the issues of improving text classification and their empirical results step by step as follows. Section II presents the first improvement issue of labeling task and its one solution related to semi-supervised learning usage. In section III, we explain the necessity of noisy data treatment and its two clues to solution: the noisy date reduction method applied to binary text classification and the TCFP classifier with robustness to noisy data. Section IV describes the necessity of improved term weighting for text classification and introduces a new term weighting scheme. Finally, we present the conclusions.

## II. UTILIZATION OF UNLABELED DATA TO REDUCE THE PAINFUL AMOUNT OF MANUAL LABELING TASKS FOR OBTAINING TRAINING DATA

Generally, the supervised-learning based text classification requires a large, often prohibitive, number of labeled training data to achieve accurate learning. Since labeling tasks must be done by hand and the application area of text classification has been diversified, ranging from articles and web pages to emails and blogs, the labeling tasks for each application area are becoming harder and harder. Thus, most users of a practical system must obviously prefer algorithms that can achieve high accuracy but do not require a painful amount of manual labeling task. To satisfy the latter users, unsupervised learning [9] as well as semi-supervised learning [8, 13] and active learning [14] have been applied to text classification; they all attempt to utilize unlabeled data instead of labeled data. While labeled data is hardly obtained, unlabeled data is readily available and plentiful. Therefore, those learning methods are very useful to utilize unlabeled data for text classification. We proposed a new automatic machine-labeling method using *a bootstrapping technique*. The proposed method used only unlabeled documents and the title word of each category is used as initial data for learning text classification. Input to the bootstrapping process is a large number of unlabeled documents and a small amount of seed information to tell the learner about the specific task. Here, a title word associated with a category is considered as seed information.

### *A. Bootstrapping Technique to Generate Machine-labeled Data*

The bootstrapping process consists of three modules: a module to preprocess unlabeled documents, a module to construct context-clusters for training, and a module to build up the NB classifier using context-clusters.

*1) Preprocessing:*

First of all, the Brill part of speech (POS) tagger is used to extract content words as words with noun or verb POS tags [15]. Since machine-labeled data has to be created from only a title word, *context* is defined as a new unit of meaning, and it is used as a new meaning unit to bootstrap the meaning of each category, i.e., a middle sized processing unit between a word and a document. A sequence of 60 content words within a document is regarded as a window size for one context. To extract contexts from a document, we use a sliding window technique [16]. The window slides from the first content word to the last content word of a document by the size of the window (60 words) and at the interval of each window (30 words).

*2) Constructing a Context-Cluster as the Training Data of Each Category:*

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

Contexts with a keyword or a title word of any category are 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. We next gather the second-order co-occurrence information by assigning the remaining contexts to the context-cluster of each category. For the assigning criterion, we calculate the similarities between the remaining contexts and centroid-contexts of each category. Thus, we propose that the similarity measure algorithm by Karov and Edelman [17] is reformed and applied to our context-cluster generation algorithm; the remaining contexts are assigned to each context-cluster by this algorithm.

In our similarity measure algorithm, words and contexts play complementary roles. Contexts are similar to the extent in that they contain similar words, and words are similar to the extent in 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 *CSM* correspond to the centroid-contexts and the columns to the remaining contexts. Each category has one *WSM* and one *CSM*. 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. The number of input contexts of row and column in *CSM* is limited to 200, considering execution time and memory allocation.

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: Update $CSM_n$ using $WSM_n$, and update $WSM_n$ using $CSM_n$. 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 $aff_n(X,W)$ by formulae (1) and (2) [17].

$$aff_n(W,X) = \max_{W_i \in X} sim_n(W,W_i) \tag{1}$$

$$aff_n(X,W) = \max_{W \in X_j} sim_n(X,X_j) \tag{2}$$

In the above formulae, $n$ denotes the iteration number, $W \in X$ means that a word $W$ belongs to a context $X$, 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.

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

$$sim_{n+1}(X_1,X_2) = \sum_{W \in X_1} weight(W,X_1) \cdot aff_n(W,X_2) \tag{3}$$

$$\begin{aligned} &if \quad W_1 = W_2 \\ &\quad sim_{n+1}(W_1,W_2) = 1 \\ &else \\ &\quad sim_{n+1}(W_1,W_2) = \sum_{W_1 \in X} weight(X,W_1) \cdot aff_n(X,W_2) \end{aligned} \tag{4}$$

The weights in formula (3) are calculated by a product of three factors: global frequency, log-likelihood factor, and part of speech. Since each weight in formula (4) 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 $CSM_n$.

The similarity between each remaining context and the centroid-contexts of a category is first estimated, and the similarity value is then averaged. Finally, each remaining context is assigned to the context-cluster of any category, when the category has a maximum similarity.

*3) Learning a NB Classifier using Context-Clusters:*

In the above section, we obtained labeled contexts training data: *context-clusters*. Since the training documents are labeled as a context unit, a NB classifier is selected to learn from *context-clusters* because it can be built by only estimating words probabilities in each category. Therefore, the NB classifier is constructed by estimating words distribution in the context-cluster of each category, and it finally classifies unlabeled documents into each category.

The NB classifier is built with minor modifications based on Kullback-Leibler Divergence [18]. This method exactly makes the same classifications as NB, but produces classification scores that are less extreme. Thus, the latter scores better reflect uncertainty than those produced by NB. A document $d_i$ is classified by the following formula:

$$\begin{aligned} P(c_j \mid d_i;\hat{\theta}) &= \frac{P(c_j \mid \hat{\theta})P(d_i \mid c_j;\hat{\theta})}{P(d_i \mid \hat{\theta})} \approx p(c_j \mid \hat{\theta}) \prod_{t=1}^{|V|} P(w_t \mid c_j;\hat{\theta})^{N(w_t,d_i)} \\ &\propto \frac{\log P(c_j;\hat{\theta})}{n} + \sum_{t=1}^{|V|} P(w_t \mid d_i;\hat{\theta}) \log\left(\frac{P(w_t \mid c_j;\hat{\theta})}{P(w_t \mid d_i;\hat{\theta})}\right) \end{aligned} \tag{5}$$

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 $d_i$.

## B. Empirical Evaluation

*1) Data Sets and Experimental Settings:*

To test the proposed method, we used three different kinds of data sets: UseNet newsgroups (Newsgroups), web pages (WebKB), and newswire articles (Reuters 21578). For a fair evaluation of 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. Splitting data sets 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. About 25% of documents from the training data of each data set were selected for a validation set. After all parameter values of our experiments were set by the validation set, we evaluated the proposed method using these parameter values. We applied a statistical feature selection method ($\chi^2$ statistics) for each classifier at its preprocessing stage [2]. As performance measures, we followed the standard definition of recall *(r)*, precision *(p)*, and $F_1$ measure *(2rp/(r+p))*. For evaluating the average of performance across categories, we used the micro-averaging method that counts the decisions for all the categories in a joint pool and computes the global recall, precision, and $F_1$ values for that global pool [19]. Results on Reuters are reported as a precision-recall breakeven point, which is a standard information retrieval measure for bi-

nary classification [7, 19].

*2) Experimental Results:*

Here, we employ a supervised NB classifier to compare our method with the supervised method; the supervised NB classifier learns from human-labeled documents. Fig. 1 and Table 1 report the results of three data sets and compare the performance differences between the proposed method and supervised method.

As shown in Table 1, we obtained a 79.36% micro-average $F_1$ score in the Newsgroups data set, 73.63% micro-average $F_1$ score in the WebKB data set, and 88.62% micro-average precision-recall breakeven point in the Reuters data set. The differences between our method and the supervised NB classifier for each data set are 12.36% in the Newsgroups data set, 11.66% in the WebKB data set, and 3.02% in the Reuters data set. Since we use only unlabeled data and title words, the performance of our method is much more significant.

Particularly, the proposed method in the Reuters data set almost achieved comparable performance to that of the supervised method. As previously noted in [20], categories like wheat and corn are known for strong correspondence between a small set of words (like our title words and keywords) and the categories, while categories like acq are known for more complex characteristics. Since the categories with narrow definitions attain best classification with small vocabularies, we can achieve good performance in the Reuters data set with our method that depends on title words. In the Newsgroups and WebKB data sets, we could not attain comparable performance to that of the supervised method. In fact, the categories of these data sets are somewhat confusable. In the Newsgroups data set, many of the categories fall into confusable clusters; for example, five of them are comp.* discussion groups, and three of them discuss religion. In the WebKB data set, the meaningful words of each category also have high frequency in other categories. Worst of all, even title words (course, professor, faculty, and project) have confusing usage. We think these factors contributed to a comparatively poor performance of our method.

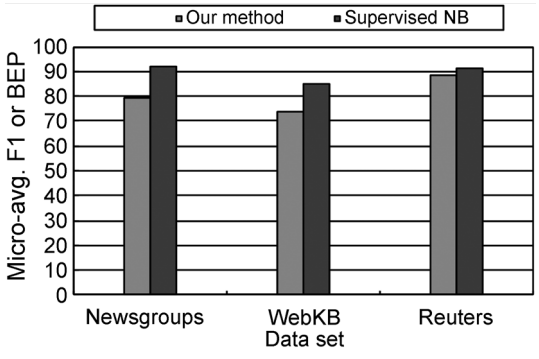


**Fig. 1.** Performance differences of the best micro-avg. $F_1$ scores or precision-recall break-even points (BEP) in three data sets: our method vs. supervised Naive Bayes (NB).

**Table 1.** Performance differences of the best micro-avg. $F_1$ scores or precision-recall break- even points in three data sets: our method vs. supervised Naive Bayes (NB)

| Data sets | Our method | Supervised NB |
|---|---|---|
| Newsgroups | 79.36 | 91.72 |
| WebKB | 73.63 | 85.29 |
| Reuters | 88.62 | 91.64 |

## III. NECESSITY OF NOISE REDUCTION AND ITS TWO CLUES: NOISY DATA REDUCTION AND THE TCFP CLASSIFIER WITH ROBUSTNESS FROM NOISY DATA

Effectively Dealing with noisy data is another key improvement issue of text classification. There are two different solutions: noisy data removal in binary training settings by the one-and-the-rest method [10], and the TCFP classifier with robustness from noisy data [11].

### *A. Improving the One-against-the-rest Method for Removing Noisy Data in Binary Text Classification*

In text classification, the binary setting or multi-class setting has been used to organize training examples for learning tasks. As the binary setting consists of only two classes, it is the simplest, yet most important formulation of the learning problem. Those two classes are composed of "relevant (positive)" and "non-relevant (negative)" for information retrieval applications [21]. Generally, some classification tasks involve more than two classes. When we apply the binary setting to the multi-class setting with more than two classes, there is a problem that the multi-class setting consists of only positive examples of each category; each category does not have negative examples. In order to solve this problem, the one-against-the-rest method has been used in many cases [22-24]; it can reduce a multi-class problem into many binary tasks. That is, while all the documents of a category are generated as positive examples by hand, documents that do not belong to the category are indirectly regarded as negative examples. This labeling task concentrates on only selecting positive examples for each category, and it does not label the negative examples that directly have the opposite meaning of counterpart positive category. Thus, the negative data set in the one-against-the-rest method probably includes noisy examples. In addition, because the negative data set consists of the different distributions of positive examples from various categories, it is hard to be considered as the exact negative example of each category. Those noisy documents can be one of the major causes of decreasing the performance for binary text classification. Therefore, classifiers need to efficiently handle the noisy training documents to achieve high performance.

*1) Detecting and Removing Noisy Data from the One-against-the-rest Method:*

In the one-against-the-rest method, the documents of one category are regarded as positive examples and the documents of the other categories as negative examples.

To effectively remove noisy data in the one-against-the-rest

method for a training setting, we have to find a boundary area that denotes a region including many noisy documents. First of all, using initial positive and negative data sets for each category from the one-against-the-rest method, we can learn a NB classifier and we can obtain a prediction score for each document via the following formula (6).

$$\text{Prediction\_Score}(c_i \mid d_j) = \frac{P(Positive \mid d_j)}{P(Positive \mid d_j) + P(Negative \mid d_j)} \tag{6}$$

where $c_i$ is a category and $d_j$ is a document of $c_i$. $P(Positive|d_j)$ is the probability of document $d_j$ to be positive in $c_i$, and $P(Negative|d_j)$ is the probability of document $d_j$ to be negative in $c_i$.

According to the calculated prediction scores, the entire documents of each category are sorted out in descending order. Probabilities, $P(Positive|d_j)$ and $P(Negative|d_j)$, of formula (6) is generally calculated by the NB formula as follows [12, 18, 25]:

$$\begin{aligned} P(Positive \mid d_j) &= \frac{P(Positive)P(d_j \mid Positive)}{P(d_j)} \\ &= P(Positive)\prod_{i=1}^{T} P(t_i \mid Positive)^{N(t_i|d_j)} \\ &\propto \frac{\log(P(Positive))}{n} + \sum_{i=1}^{|T|} P(t_i \mid d_j)\log\left(\frac{P(t_i \mid Positive)}{P(t_i \mid d_j)}\right) \end{aligned} \tag{7}$$

where $t_i$ is the $i$-th word in the vocabulary, $T$ is the size of the vocabulary, and $N(t_i|d_j)$ is the frequency of word $t_i$ in document $d_j$.

A boundary between positive and negative examples can be detected in a block with the most mixed degree of positive and negative documents. The sliding window technique is first used to detect the block [26]. In this technique, windows of a certain size are sliding from the top document to the last document in a list ordered by the prediction scores. An entropy value is calculated for estimating the mixed degree of each window as follows [27]:

$$Entropy(W) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{8}$$

where, given a window ($W$), $p_+$ is the proportion of positive documents in $W$ and $p_-$ is the proportion of negative documents in $W$.

Two windows with the highest entropy values are picked up; one window is first detected from the top and the other is first detected from the bottom. If there is no window or only one window with the highest entropy value, then windows with the next highest entropy values become targets of the selected windows. Then, the maximum (*max*) and minimum (*min*) threshold values can be searched from selected windows respectively. The *max* threshold value is found as the highest prediction score of a negative document in the former window, and the *min* threshold value is as the lowest prediction score of a positive document in the latter window. We regard the documents between *max* and *min* threshold values as unlabeled documents; these documents are considered as potentially noisy documents. Now, three classes for training documents of each category are constructed as definitely positive documents, unlabeled documents, and definitely negative documents. By applying the revised expectation-maximization (EM) algorithm to those three data sets, we can extract the actual noisy documents and remove them.

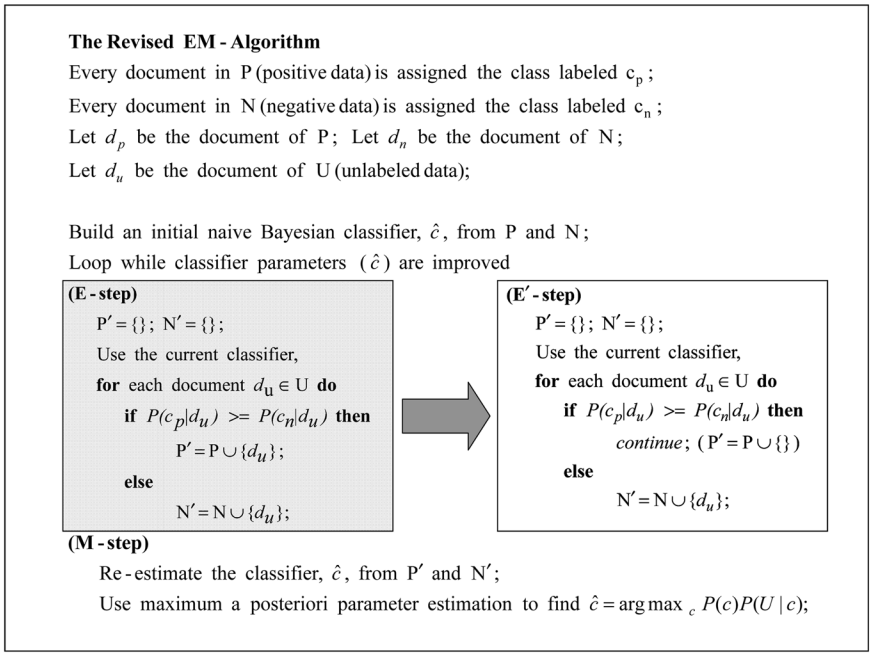


**Fig. 2.** The revised expectation-maximization (EM) agorithm.

The EM algorithm is used to pick out noisy documents from unlabeled data and to remove them. The general EM algorithm consists of two steps, the *Expectation* step and *Maximization* step [28]. This algorithm first trains a classifier using the available labeled documents and labels the unlabeled documents via hard classification (*Expectation* [E or E´] step). It then trains a new classifier using the labels of all the documents (*Maximization* [M] step), and iterates to convergence. The NB classifier is used in the two steps of the EM algorithm. Fig. 2 shows how the EM algorithm is revised in our method.

E´-step is reformed to effectively remove the noise documents located in the boundary area. Unlike the original E-step, E´-step does not assign an unlabeled document, $d_u$, to the positive data set, $P$, because it regards $d_u$ as anther noisy document. Since the positive documents are labeled by hand and have enough information for a category, additional positive documents can decrease performance. Finally, we can learn the text classifiers with binary training data generated by the revised EM algorithm.

*2) Empirical Evaluation:*

The experimental results show that the proposed method achieved better performance than the original one-against-the-rest method in all the three training data sets and all four classifiers. To evaluate the effectiveness of the proposed method, we implemented four different text classifiers (*k*-NN, NB, Rocchio, and SVM). Further, performance of the original one-against-the-rest method is compared to that of the proposed method on three test data sets (Newsgroups, WebKB, and Reuters data sets). As performance measures, the standard definition of recall and precision is used, and the *micro-averaging method* and *macro-averaging method* are applied to evaluate the average of performance across categories; in the macro-averaging method, the recall, precision, and $F_1$ measures are first computed for individual categories and then averaged over categories as a global measure of the average performance over all categories [6]. The results are reported as the precision-recall breakeven points (BEP), which is a standard information retrieval measure for binary classification; given a ranking of documents, the precision-recall BEP is a value at which precision and recall are equal [6, 11, 29]. Tables 2-4 show the experimental results from each text classifier in Newsgroups, WebKB, and Reuters data sets respectively.

As shown in the Tables 2-4, SVM achieved less improvement than the other classifiers. This is caused by the fact that the performance of SVM using the original one-against-the-rest method is too high in all data sets. Note that it is more difficult to improve a classifier with higher performance. As a result, the proposed method achieved better performances than the original method over all classifiers and all data sets. This is an obvious proof that the proposed method is more effective than the original one-against-the-rest method.

## B. The TCFP Classifier with Robustness from Noisy Data Using the Feature Projection Technique

To effectively handle out noisy data, a new text classifier using a feature projection technique was developed and the

**Table 2.** Results of the Newsgroups data set

| | *k*-NN | | NB | | Rocchio | | SVMs | |
|---|---|---|---|---|---|---|---|---|
| | Original method | Proposed method | Original method | Proposed method | Original method | Proposed method | Original method | Proposed method |
| Micro-avg. BEP | 86.07 | **87.96** | 83.17 | **84.86** | 82.84 | **84.48** | 88.34 | **89.08** |
| Macro-avg. BEP | 84.58 | **87.03** | 82.87 | **84.55** | 81.5 | **83.57** | 87.73 | **89.08** |

NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine, BEP: break even points.

**Table 3.** Results of the WebKB data set

| | *k*-NN | | NB | | Rocchio | | SVMs | |
|---|---|---|---|---|---|---|---|---|
| | Original method | Proposed method | Original method | Proposed method | Original method | Proposed method | Original method | Proposed method |
| Micro-avg. BEP | 84.97 | **86.74** | 85.67 | **87.21** | 86.52 | **88.26** | 92.12 | **92.64** |
| Macro-avg. BEP | 82.13 | **85.55** | 83.58 | **86.53** | 83.71 | **87.03** | 91.52 | **92.17** |

NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine, BEP: break even points.

**Table 4.** Results of the Reuters data set

| | *k*-NN | | NB | | Rocchio | | SVMs | |
|---|---|---|---|---|---|---|---|---|
| | Original method | Proposed method | Original method | Proposed method | Original method | Proposed method | Original method | Proposed method |
| Micro-avg. BEP | 91.47 | **94.27** | 90.80 | **93.86** | 89.24 | **91.80** | 94.66 | **95.52** |
| Macro-avg. BEP | 82.66 | **85.432** | 81.26 | **86.38** | 77.56 | **83.55** | 89.86 | **90.72** |

NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine, BEP: break even points.

classifier was named TCFP [11]. By the property of the feature projection technique, the TCFP classifier can achieve robustness against noisy data. In the experiment results, TCFP showed better performance than other conventional classifiers in noisy data.

*1) The TCFP Classifier with Robustness from Noisy Data:*

In the TCFP classifier, 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 high term frequency-inverse document frequency (TF-IDF) values 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:

$$r(c_j, f_m) = \sum_{f_{mi} \in V_m} w(f_m, \vec{d}_i) \cdot y(c_j, f_{mi}) \Big/ \sum_{f_{mi} \in V_m} w(f_m, \vec{d}_i) \qquad (9)$$

In formula (9), $f_{mi}$ 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 $c_j$, then 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 the 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 high co-occurrence frequency values and low category frequency values have higher term weights as the following formula:

$$fw(f_i, \vec{d}) = w(f_i, \vec{d}) \cdot \left( 1 + \left( \frac{1}{\log(cf + 1)} \right) \cdot \left( \frac{\log(co(f_i, f_j) + 1)}{\log(maxco(f_k, f_i) + 1)} \right) \right) \qquad (10)$$

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 $f_i$.

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

$$vs(c_j, f_m) = fw(f_m, \vec{d}) \cdot r(c_j, f_m) \cdot \log(1 + \chi^2_{\max}(f_m)) \qquad (11)$$

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 $\chi^2$ statistics value of $f_m$ in each category.

The outline of the TCFP classifier is as follows:

**Input:** test document: $\vec{d} = \langle f_1, f_2, \ldots, f_n \rangle$
**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 11

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

Robustness to nosy data of the TCFP classifier is due to its voting mechanism. The voting mechanism of the TCFP classifier depends on separate voting in each feature, and it can reduce the negative effect of possible noisy data and irrelevant features in text classification. That is, when a document contains irrelevant features or an incorrect label, the document may be in the wrong location of vector space model. This document can negatively impact performance, especially for *k*-NN and SVM. On the other hand, an irrelevant feature in the TCFP classifier contributes to only voting of the feature. Moreover, the only feature elements with a TF-IDF weight over an average weight value can take part in our voting mechanism, and this process makes the TCFP classifier more effective in handling irrelevant features.

*2) Experimental Evaluation:*

We provide empirical proofs that TCFP is a useful text classifier and is robust from noisy data. In our experiments, we used

**Table 5.** Comparison of performance results for each classifier on each data set

| | TCFP | *k*-NN | NB | Rocchio | SVM |
|---|---|---|---|---|---|
| Newsgroups | ***85.52*** | 85.15 | 82.51 | 81.68 | 87.32 |
| WebKB | ***88.07*** | 84.83 | 85.22 | 85.98 | 91.75 |
| Reuters | ***90.01*** | 88.93 | 88.62 | 86.47 | 93.32 |

NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine.

**Table 6.** Comparison of performance results of each classifier on four noisy data sets

| Noisy degree (%) | TCFP | *k*-NN | NB | Rocchio | SVM |
|---|---|---|---|---|---|
| 10 | 85.13 | 84.78 | 82.42 | 81.56 | 86.26 |
| 20 | 84.64 | 84.04 | 81.87 | 81.17 | 84.49 |
| 30 | 84.5 | 83.9 | 81.09 | 81.26 | 81.73 |
| 40 | 83.3 | 82.39 | 79.83 | 80.9 | 76.02 |

NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine.

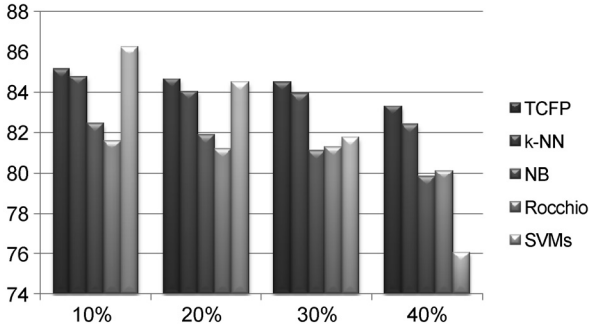


**Fig. 3.** Comparison of performance results of each classifier on four noisy data sets. NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine.

three test data sets (Newsgroups, WebKB, and Reuters data sets) and employed four other classifiers (k-NN, NB, Rocchio, and SVMs) or comparison with the TCFP classifier. As performance measures, we followed the standard definition of recall *(r)*, precision *(p)*, and $F_1$ measure *(2rp/(r+p))* for the Newsgroups and WebKB data sets and results on Reuters are reported as precision-recall breakeven points [19].

First, Table 5 reports the performance comparison between classifiers on the three data sets to verify the general usefulness of TCFP.

The results show that TCFP is superior to *k*-NN, NB, and Rocchio classifiers. However, TCFP achieved lower performance than SVMs, which has been reported as a classifier with the best performance in this paper. In order to verify the superiority of TCFP in terms of robustness to noisy data, we conducted experiments for evaluating the robustness of each classifier from noisy data. For this experiment, we generated four data sets by increasing the number of noisy documents from 10% to 40% using the **Newsgroups** data set: these noisy documents were randomly chosen from each category and randomly assigned into other categories. The results of each classifier on each noisy data set are shown in Fig. 3 and Table 6. These results are also obtained by a five-fold cross-validation method.

As shown in Fig. 3 and Table 6, TCFP showed the best performance beginning from 20% noisy data set, and the decreasing rate of performance of TCFP is less than that of *k*-NN and SVMs. Especially, we observed that the performance of SVMs degraded rapidly when the number of noisy documents increased.

As a result, the experimental results show that TCFP achieves good performance and a special characteristic with regards to robustness to noisy data.

## IV. IMPROVING TERM WEIGHT SCHEME FOR TEXT CLASSIFICATION

We here focus on the term weighting and indexing scheme of text classification as another improvement issue. The vector space model has been used as a conventional method for text representation [30]. This model commonly 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. However, each sentence in a document has different importance for identifying the content of the document. Thus, text classification can be improved by assigning a different weight according to the importance of the sentence into each term. For this, we apply text summarization techniques to classify important sentences and unimportant sentences in a document. The importance of each sentence is measured by those text summarization techniques, and 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, and the other was gathered from the Korean UseNet discussion group.

### *A. Measuring the Importance of Sentences*

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

*1) Importance of Sentences by the Title:*

Generally, we believe that a title summarizes the important content of a document [31]. Hence, we measure the similarity between the title and each sentence, and we then 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, and their similarity value is calculated by the inner product, while 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:

$$Sim(S_i, T) = \frac{\vec{S}_i \cdot \vec{T}}{\max_{S_j \in d}(\vec{S}_j \cdot \vec{T})} \quad (12)$$

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

*2) 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 a document with a meaningless title or no title. Besides, sentences with important terms must be also handled importantly although they are dissimilar to the title. Considering these points, we first measure the importance values of terms by TF, IDF, and $\chi^2$ statistic values, and the sum of the importance values of terms in each sentence is then 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:

$$Cen(S_i) = \frac{\sum_{t \in S_i} tf(t) \times idf(t) \times \chi^2(t)}{\max_{S_j \in d} \left\{ \sum_{t \in S_j} tf(t) \times idf(t) \times \chi^2(t) \right\}} \quad (13)$$

where *tf(t)* denotes the term frequency of term *t*, *idf(t)* denotes the inverted document frequency, and $\chi^2(t)$ denotes the $\chi^2$ statistic value.

*3) Combination of Two Sentence Importance Values:*

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

$$Score(S_i) = 1.0 + k_1 \times Sim(S_i, T) + k_2 \times Cen(S_i) \quad (14)$$

In formula (14), $k_1$ 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 form having a lower value than the original TF value by formula (15).

*4) Indexing Process:*

The importance value of a sentence by formula (14) is used to modify 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 (*WTF(d,t)*) of the term *t* in the document *d* is calculated by formula (15).

$$WTF(d,t) = \sum_{S_i \in d} tf(S_i, t) \times Score(S_i) \quad (15)$$

where $tf(S_i,t)$ denotes TF of the term *t* in sentence $S_i$.

The weight by formula (15) is used in *k*-NN, NB, Rocchio, and SVM.

## B. Empirical Evaluation

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 20 Newsgroups data set is the same one used in the previous sections. 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. 3,107 documents (30%) are used for test data and the remaining 7,224 documents (70%) for training data. The resulting vocabulary from training data has 69,793 words.

As performance measures, we followed the standard definition of recall *(r)*, precision *(p)*, and $F_1$ measure (*2rp/(r+p)*). For evaluating the average performance across categories, we used the *micro-averaging method* and *macro-averaging method* [19].

Tables 7 and 8 list a comparison of performances of each classifier using different indexing schemes on two newsgroup data sets. Here, the basis method used the conventional TF for NB and conventional TF-IDF for the other classifiers.

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

# V. CONCLUSIONS

This paper has been devoted to present three important points of improvement and their actual empirical results. They can be

**Table 7.** Results of the English Newsgroups data set

| | *k*-NN | | NB | | Rocchio | | SVMs | |
|---|---|---|---|---|---|---|---|---|
| | **Basis method** | **Proposed method** | **Basis method** | **Proposed method** | **Basis method** | **Proposed method** | **Basis method** | **Proposed method** |
| Micro-avg. $F_1$ | 81.2 | **82.6** | 83.0 | **84.1** | 78.6 | **79.7** | 86.0 | **86.6** |
| Macro-avg. $F_1$ | 81.4 | **82.7** | 83.3 | **84.4** | 79.1 | **80.0** | 86.1 | **86.6** |

NN: nearest neighbor, NB: Naive Bayes, SVM: support vector machine.

**Table 8.** Results of the Korean Newsgroups data set

| | *k*-NN | | NB | | Rocchio | | SVMs | |
|---|---|---|---|---|---|---|---|---|
| | **Basis method** | **Proposed method** | **Basis method** | **Proposed method** | **Basis method** | **Proposed method** | **Basis method** | **Proposed method** |
| Micro-avg. $F_1$ | 77.4 | **79.6** | 78.4 | **80.8** | 76.5 | **78.2** | 84.5 | **85.3** |
| Macro-avg. $F_1$ | 79.9 | **81.3** | 79.1 | **81.3** | 78.7 | **80.1** | 86.0 | **86.5** |

summarized as follows:

- Training Data Generation: semi-supervised learning or active learning techniques can be applied to text classification. They give us a way to utilize inexpensive and plentiful unlabeled data. In our experiment, we achieved comparable performance to the supervised method, even though we only used the title word of each category and unlabeled data.
- Noisy Data Reduction: effective noisy data reduction and robust classifier development from noisy data can resolve some noisy data problems. In our experiments, the proposed noisy data reduction method led to higher performances in all three test data sets and four different conventional classifiers and the TCFP classifier, which are developed as a robust classifier from noisy data, also led to good performance in the environment with much noisy data.
- Term Weighting and Indexing: the development of a new term weighting and indexing scheme is needed because that of text classification is different from that of information retrieval. Thus, we advocated a new term weighting and indexing method for text categorization using two kinds of text summarization techniques: one uses the title while the other uses the importance of terms. In our experiments, the proposed method achieved better performance than the basis system in all these classifiers and both languages, English and Korean.

## ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0003780).

## REFERENCES

1. F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, Mar. 2002.
2. Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," *Proceedings of the 14th International Conference on Machine Learning*, Nashville, TN, 1997, pp. 412-420.
3. Y. Ko and J. Seo, "Automatic text categorization by unsupervised learning," *Proceedings of the 18th Conference on Computational Linguistics*, Saarbrucken, Germany, 2000, pp. 453-459.
4. A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," *AAAI/ICML Workshop on Learning for Text Categorization*, Madison, WI, 1998, pp. 41-48.
5. D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka, "Training algorithms for linear text classifiers," *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, 1996, pp. 298-306.
6. Y. Yang, S. Slattery, and R. Ghani, "A study of approaches to hypertext categorization," *Journal of Intelligent Information Systems*, vol. 18, no. 2-3, pp. 219-241, Mar. 2002.
7. T. Joachims, "Learning to classify text using support vector machines," Ph.D. dissertation, University of Dortmund, Dormnund, 2001.
8. Y. Ko and J. Seo, "Text classification from unlabeled documents with bootstrapping and feature projection techniques," *Information Processing and Management*, vol. 45, no. 1, pp. 70-83, Jan. 2009.
9. N. Slonim, N. Friedman, and N. Tishby, "Unsupervised document classification using sequential information maximization," *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, 2002, pp. 129-136.
10. H. Han, Y. Ko, and J. Seo, "Using the revised EM algorithm to remove noisy data for improving the one-against-the-rest method in binary text classification," *Information Processing and Management*, vol. 43, no. 5, pp. 1281-1293, Sep. 2007.
11. Y. Ko and J. Seo, "Using the feature projection technique based on a normalized voting method for text classification," *Information Processing and Management*, vol. 40, no. 2, pp. 191-208, Mar. 2004.
12. Y. Ko, J. Park, and J. Seo, "Improving text categorization using the importance of sentences," *Information Processing and Management*, vol. 40, no. 1, pp. 65-79, Jan. 2004.
13. K. Nigam, A. McCallum, and T. Mitchell, "Semi-supervised text classification using EM," *Semi-Supervised Learning*, Cambridge, MA: MIT Press, pp. 33-56, 2006.
14. S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, pp. 45-66, Nov. 2001.
15. E. Brill, "Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging," *Computational Linguistics*, vol. 21, no. 4, pp. 543-565, Dec. 1995.
16. Y. S. Maarek, D. M. Berry, and G. E. Kaiser, "An information retrieval approach for automatically constructing software libraries," *IEEE Transactions on Software Engineering*, vol. 17, no. 8, pp. 800-813, Aug. 1991.
17. Y. Karov and S. Edelman, "Similarity-based word sense disambiguation," *Computational Linguistics*, vol. 24, no. 1, pp. 41-59, Mar. 1998.
18. M. Craven, D. Dipasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, "Learning to construct knowledge bases from the World Wide Web," *Artificial Intelligence*, vol. 118, no. 1-2, pp. 69-113, Apr. 2000.
19. Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1-2, pp. 69-90, 1999.
20. T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN, 1997, pp. 143-151.
21. T. Joachims, Learning to Classify Text Using Support Vector Machines, Boston: Kluwer Academic Publishers, 2002.
22. B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naïve Bayesian classifiers," *Proceedings of the Eighteenth International Conference on Machine Learning*, Williamstown, MA, 2001, pp. 609-616.
23. B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, AB, 2002, pp. 694-699.
24. C. W. Hsu and C. J. Lin, "A comparison of methods of multi-class support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415-425, Mar. 2002.

25. D. D. Lewis, "Naive (bayes) at forty: the independence assumption in information retrieval," *Proceedings of the 10th European Conference on Machine Learning*, Chemnitz, Germany, 1998, pp. 4-15.
26. C. H. Lee, C. R. Lin, and M. S. Chen, "Sliding-window filtering: an efficient algorithm for incremental mining," *Proceedings of the ACM CIKM: 10th International Conference on Information and Knowledge Management*, Atlanta, GA, 2001, pp. 263-270.
27. T. M. Mitchell, Machine Learning, New York: McGraw-Hill, 1997.
28. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via em algorithm," *Journal of the Royal Statistical Society Series B-Methodological*, vol. 39, no. 1, pp. 1-38, 1977.
29 T. Joachims, "Text categorization with support vector machines: learning with many relevant features," *Machine Learning: ECML-98. Lecture Notes in Computer Science vol. 1398*, Heidelberg: Springer Verlag, pp. 137-142, 2002.
30 G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513-523, 1988.
31 B. Endres-Niggemeyer, Summarizing Information, New York: Springer, pp. 307-338, 1998.

**Youngjoong Ko**

Youngjoong Ko received the B.S. degree in mathematics, the M.S. and Ph.D degrees in computer science from Sogang University in 1996, 2000, and 2003, respectively. From 1996 to 1997, he worked at the LG-EDS systems as a software engineer. Currently, he is an associate professor of computer engineering at Dong-A university, Busan, Korea. His research interests include natural language processing, dialogue interface, text mining, opinion mining, and information retrieval.

**Jungyun Seo**

Professor Jungyun Seo is a professor at the Department of Computer Science and Engineering, Sogang University, Seoul, Korea. He received B.S. degree in Mathematics from Sogang University in 1981, and M.S. and Ph.D. degree in Computer Science from the University of Texas at Austin in 1985 and 1990, respectively. He was an assistant professor at the department of Computer Science, KAIST, Daejeon, Korea. He started a company, named DiQuest Inc. where he developed intelligent Question/Answering and Information Retrieval (IR) solutions. He served in several professional societies, numerous conferences and several editorial boards; Currently, the president of Korea Cognitive Science Society, and a Vice President of Korean Institute of Information Scientists and Engineers (KIISE). He published more than 200 papers including about 110 international journal and conference papers in the area of Natural Language Processing, Artificial Intelligence and Database. His research interests include dialogue interface including multi-modal dialogues, statistical methods for NLP, Machine Translation and Information Retrieval.