# Guiding Practical Text Classification Framework to Optimal State in Multiple Domains

**Sung Pil Choi[1], Sung-Hyon Myaeng[2] and Hyun-Yang Cho[3]**
[1] Department of Information Technology Research, KISTI
335 Gwahangno, Yuseong-gu, Daejeon, 305-806, South Korea
[e-mail: spchoi@kisti.re.kr]
[2] School of Engineering, Information and Communications University,
119, Munjiro, Yuseong-gu, Daejeon, 305-732, South Korea
[e-mail: myaeng@icu.ac.kr]
[3] Library & Information Science Department, Kyonggi University,
94-6, Yiui-dong, Yeongtong-gu, Suwon, Kyonggi-do, 443-760, South Korea
[e-mail: hycho@kyonggi.ac.kr]
*Corresponding author : Sung Pil Choi

*Received April 17, 2009; revised June 5, 2009; accepted June 8, 2009;
published June 22, 2009*

## Abstract

This paper introduces *DICE*, a Domain-Independent text Classification Engine. *DICE* is robust, efficient, and domain-independent in terms of software and architecture. Each module of the system is clearly modularized and encapsulated for extensibility. The clear modular architecture allows for simple and continuous verification and facilitates changes in multiple cycles, even after its major development period is complete. Those who want to make use of *DICE* can easily implement their ideas on this test bed and optimize it for a particular domain by simply adjusting the configuration file. Unlike other publically available tool kits or development environments targeted at general purpose classification models, *DICE* specializes in text classification with a number of useful functions specific to it. This paper focuses on the ways to locate the optimal states of a practical text classification framework by using various adaptation methods provided by the system such as feature selection, lemmatization, and classification models.

**Keywords:** Document classification, test bed, machine learning, text categorization, feature selection, text mining, lemmatization

The research was supported by Korea Institute of Science and Technology Information (KISTI) under Institutional Grant and partially supported by Korea Research Council of Fundamental Science & Technology (KRCF) Grant, the Korean government. We would like to thank the anonymous reviewers for their critical and insightful comments.

**DOI: 10.3837/tiis.2009.03.005**

## 1. Introduction

Text classification can be characterized as a process of automatically assigning one or more of the pre-defined categories to a natural language document by inspecting and analyzing their inner contents, such as their terms and structural information. Sebastiani [1] describes the task as the activity of labeling natural language texts with thematic categories from a user-defined set. Much effort regarding text classification has been made in both academic and industrial circles due to its potential impact on other closely related areas, such as named entity recognition, document filtering, and word sense disambiguation. As such, many experimental text classification systems have been developed to show their relative advantages. However, few commercial or experimental classification systems are available for research, as in the case of information retrieval [1][2][3]. This situation makes it essential to develop a practical and efficient text classification system test bed where various extensions of existing approaches can be implemented and evaluated for their efficacy with minimal knowledge and experience in programming.

This paper introduces *DICE*, a Domain-Independent text Classification Engine. *DICE* is robust, efficient, and domain-independent in terms of software and architecture. Each module of the system is clearly modularized and encapsulated for extensibility. The clear modular architecture allows for simple and continuous verification and facilitates changes in multiple cycles, even after its major development period is complete.

The *DICE* can make system development processes clearer and more explicit. While parameter tunings are critical for achieving a high level of performance, especially with inductive learning modules for which parameters settings are of paramount importance (as in [1][4][5][6][7]), they are rarely mentioned explicitly in technical papers. In other words, it is neither clear nor explicit in past research how the systems were adapted and optimized for specific target domains. As a result, it is difficult to duplicate the same level of performance that may be necessary for comparisons in later research. This situation may indicate that their adaptation and optimization processes were not systematically done but were instead performed in an ad hoc manner.

Another situation the proposed test bed attempts to address is related to the fact that experimental text classification systems are not full-fledged, as mentioned in several examples in the literature [1][2][3]. For example, a typical approach to build an experimental system is to separate the entire system into several modules including a feature generation module (responsible for tokenization, lemmatization, and feature importance estimations, for instance) and a classification module (for training and classifying). At first glance, this approach appears to be reasonable with respect to the modularity and encapsulation concepts in software engineering. As the generation steps for feature sets, training results and classification results are not pipelined naturally in the system; however, manual interventions are required. This problem can be circumvented with the *DICE* environment, in which a full-fledged system is readily available and can be extended for extensions.

In order to ensure that the newly proposed system environment properly addresses the problems associated with past research prototypes in text classification, it is very important to provide a reasonable performance level of the system in terms of its overall effectiveness and efficiency. For maximum generality, such a system should not be biased toward or against a particular domain or task. To this end, an extensive series of experiments were conducted for four different domains.

This paper focuses on the ways to locate the optimal states of a practical text classification framework by using various adaptation methods provided by our system such as feature selection, lemmatization, classification models and so forth. By inspecting the experimental results of multiple datasets, we show that optimal configurations vary according to domains, which slightly differs from the previous results that concluded particular settings provided best performance without examining intensively in various test collections.

## 2. Related Work

### 2.1 Feature Selection

It is clear that a large number of previous studies on feature selection for text categorization exist, and the case can be made that the present study is similar to those. Careful and objective inspection, however, can reveal the difference between the present approach and existing studies. For example, the pioneering work in this field by Yang and Pedersen provides the evaluation result of comparing five different feature selection methods based on LLSF and kNN [6]. The present work can be distinguished from it in several ways. The first is that *DICE* provides all-in-one capabilities related to text categorization tasks; that is, it is no more than a commercialized system. Moreover, it is based on both naïve Bayesian and kNN models, which are considered the most promising models in terms of both efficiency and effectiveness. More importantly, their work appears to have focused on the decrease in the number of features (terms) due to the application of feature selection methods that maintain a similar level of effectiveness.

Forman G. [4] parallels the present work in many respects regarding the evaluation and inspection of feature selection mechanisms. In this model, the author showed the valuable results of conducting an empirical study of twelve feature selection methods via experiments on a large-scale, artificially generated test collection, which he termed a ”benchmark,” originating from the four well-known datasets. In spite of the far-reaching execution of his work, it is less likely that the results can be well interpreted and understood by those who want them to be used readily in further research. There may be two reasons for that. The first is that the target classification model of Forman G. [4] is a binary class decision model, whereas all conventional test collections assume a multi-class model. Another reason, which may be related to the first, is associated with the fact that he artificially created a new benchmark collection based on four conventional test collections (TREC, OHSUMED, Reuters, and WebACE). This experimental approach is differs greatly compared to general approaches in which the performance is computed independently by each dataset of the existing test collection. In contrast, the present study focuses on providing the explicit performance of each environmental setting of *DICE* one dataset at a time for a better understanding of the results. More importantly, the proposed architecture, which is capable of supporting multiple classification models, enables us to both compare the performance readily and accordingly conduct model tuning in a certain domain.

This experimental approach can be differentiated from Forman G. [4] in that the aim is to analyze the validity and universality of *DICE* through the concentrated behavioral inspections in certain domains, whereas Forman G. [4] gives general information only about the effectiveness of SVM without fully providing the inner settings of the classifier using datasets with greater variation.

In addition, a considerable number of studies have been performed by many researchers who sought to reveal the link between text classification models and feature selection methods [4][6][7][8][9][10][11][12][13][14][15][16]. Recently, many studies have been published related to the invention of new feature selection methods based on existing feature evaluation models which seeks to enhance the performance of text classification [20][21][21][22][23][24][25][26][27][28][29]. In particular, nearly all of these aforementioned approaches in are dependent upon the statistical characteristics of each target domain. Strictly speaking, they usually utilize the frequency information of the constituent terms of the entire dataset. The Term Information Storage Structure (TISS) of *DICE* is highly flexible; it is expressive enough to accommodate all the necessary information for estimating those feature selection models, which implies that new feature selection functions can be easily appended as the occasion demands. Further research by the authors, therefore, will include the embodiment of these various feature selection methods and the invention of novel methods of feature selection.

### 2.2 Implementation

The work of Lewis et al. [3] can be considered as an example of an early article on the subj ect of implementing practical text classification systems. In this paper, the authors introduced ATTICS, an extensible text classification system implemented in C++. However, experiment al results are not included. Moreover, they did not present the detailed architecture of ATTICS, which would be a key issue in their paper.

More recently, Aris et al. [30] showed that text classification tasks can be achieved by only conventional search engine functionals. One of the pivotal issues of their approach was to minimize the number of query terms coming from an input document, as the category assignment process should be realized by existing information retrieval operations. In the experimental results, however, a lack of clarity is apparent with respect to describing the actual performance values associated with this approach. First, they mainly used the "area under the curve (AUC) score" and the "cost estimation ratio" based on a ROC graph, which are not frequently used in current research. For instance, one of the figures in this paper that compares evaluation results based on both the AUC and *F*-measure shows that there is a striking contrast between the two measures (e.g., the values of the F-measure are very low compared to those of the AUC). Furthermore, as in a related study [3], the authors did not introduce the inner details of the data structures and the overall architecture of their system, which was also important given that they asserted the benefits of the search engine structure for classifying texts.

## 3. Overall Architecture and Data Structure

**Fig. 1** shows the conceptual architecture of *DICE* as implemented in this paper. The system can contain manifold classification models, which are KNN and naïve Bayesian in this paper. In order to facilitate the preprocessing and training processes, document management module exists in the system. Also, classifier optimizer is used to adapt the system into a specific domain by means of its validation functions, configuration subsystem and other relevant utilities. Finally, DICE can also utilize domain specific language resources to maximize the performance in that domain. Instead of developing various different domain-dependent classifiers which might be restricted only to particular target domains, a general-purpose text classification framework was implemented so that it can be applied to any problem domain related to document classification. In this respect, the salient characteristics of the system are that it contains more than one document classification algorithm and that the feature generation module is separated from the algorithms. This approach can be justified by the fact that numerous document classification models share a similar feature structure and information. (e.g., a vector space model based on term frequencies, a conditional probability list of terms with respect to each class)

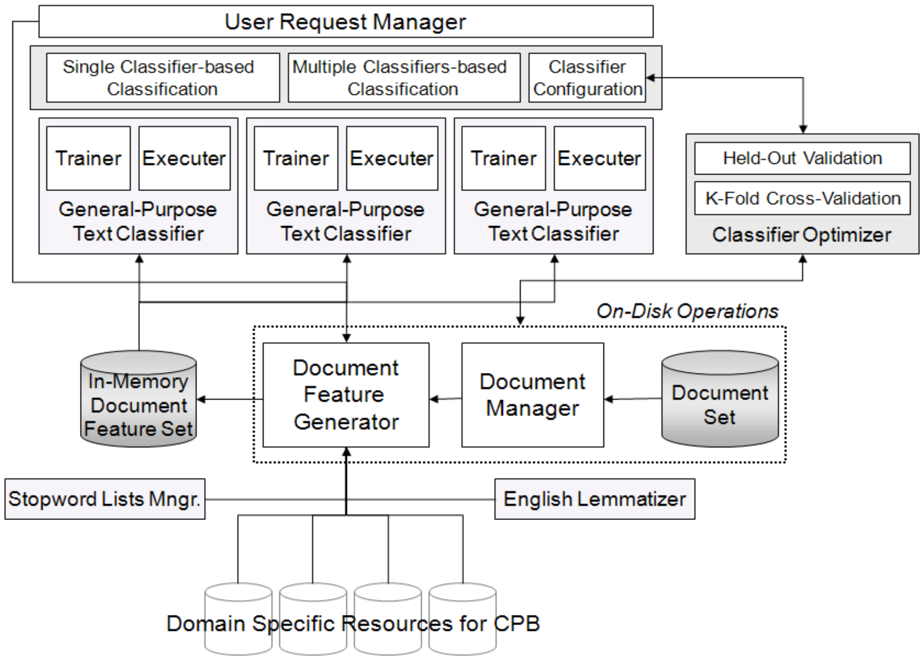

User Request Manager
Single Classifier-based Classification
Multiple Classifiers-based Classification
Classifier Configuration
Trainer
Executer
General-Purpose Text Classifier
Trainer
Executer
General-Purpose Text Classifier
Trainer
Executer
General-Purpose Text Classifier
Held-Out Validation
K-Fold Cross-Validation
Classifier Optimizer
On-Disk Operations
In-Memory Document Feature Set
Document Feature Generator
Document Manager
Document Set
Stopword Lists Mngr.
English Lemmatizer
Domain Specific Resources for CPB

**Fig. 1**. Overall Architecture of *DICE*

For a fast classification process, the inverted file structure as widely used in information retrieval systems is adopted here. Unlike conventional inverted file structures, however, this architecture is adequately optimized to text classification tasks; it includes a memory-based binary tree structure for the retrieval of massive terms and an efficient structure for saving and searching for class information.

### 3.1 Feature Storage Structures

Once each document has been analyzed (e.g., tokenization, lemmatization, and POS tagging), a stream of terms in each document is generated. **Fig. 2** shows the proposed storage structure for saving term information (*TISS*: Term Information Storage Structure). It exploits the general architecture and process of the inverted index structure used in IR. However, the concept is expanded so as to allow *TISS* to handle additional information related to classification tasks, including class identifiers, class frequencies, and class conditional probabilities, for which $P(w \mid c)$, where $w$ is a term and $c$ is a category.

As almost all of the classification tasks assume in-memory processing, implying that every analysis process is executed in main memory, a binary tree structure was adapted for saving and searching for each term in all target data rather than the B-tree or other locations. In the posting information part of each term, *TISS* maintains three types of information. These are the total term frequency; the general posting information, which is the mapping information between a term and document; and the conditional probability information in terms of each target class.

The total term frequency is the number of all occurrences of a term in the target collection. It will be used when calculating the probability of a term in both the learning process and the feature selection process. Posting information contains not only every document identifier in which a term appears but additional information such as the class identifier of each document, the term frequency and the term weight. This posting information is clearly related to similarity-based classification algorithms such as k-Nearest Neighbors and others. Finally, the conditional probability information includes exactly $C$ nodes if the number of target classes is $C$. Each node contains statistics concerning the class and term distribution, $P(w \mid c)$. Fig. 3 denotes the additional information that is necessary to estimate the probabilities of all classes. While the upper part of Fig. 3 relates to saving the number of documents belonging to each class, the lower part shows the storage structure for saving the total term frequency of each class.

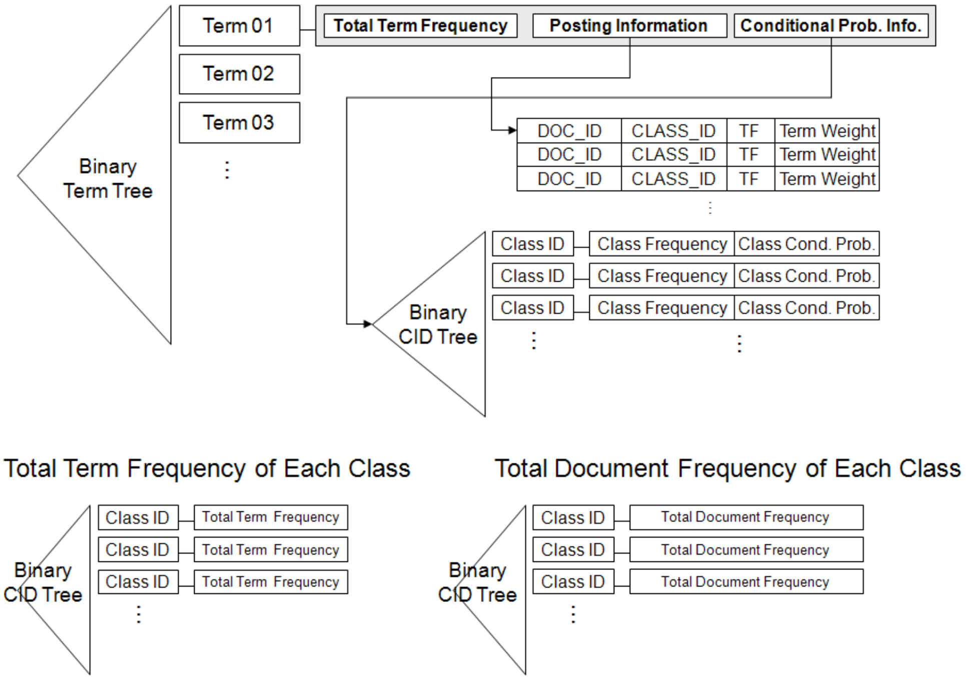


Fig. 2. Term Information Storage Structure

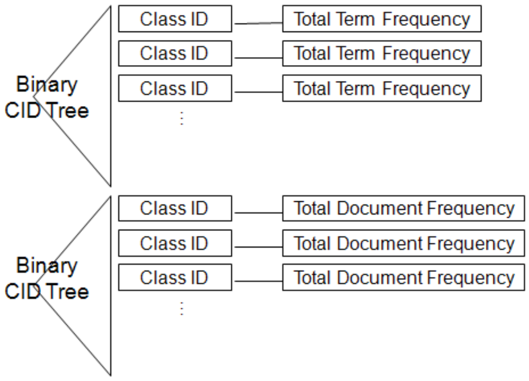


Fig. 3. Class Identifier vs. Total Term/Document Frequency Information Table Structure

Given the refined feature information after the learning process, the next step is to serialize or write the information into several files for later use. The feature information consists of three separate files, each of which plays an important role in the classification process. The first file is the set of the entire posting information of each term, as referred to earlier. The second file is a set of conditional probabilities of each term given the class, which can be used by a naïve Bayesian classifier. The last file contains the class probabilities. Users should specify the names of all three files in the configuration file.

On the other hand, in order to classify new documents based on the learned features, all feature information should be loaded into the main memory. Once the information is loaded, classification engines can use it repeatedly, which may be why this classification process should be configured as a server (daemon). As the loading process is quite time-consuming owing to the size of the feature information, this configuration will be beneficial when working with on-demand classification functionalities. While the term information storage structure in the learning process is quite complicated, in the classification (execution) process, the proposed system uses a very simplified structure to maintain the feature information. Fig. 4 shows the structure of the term information in the classification phase. All of the information is identical to that in *TISS*. Based on the structure, the classification engines can obtain all information regarding the classification of the target documents.

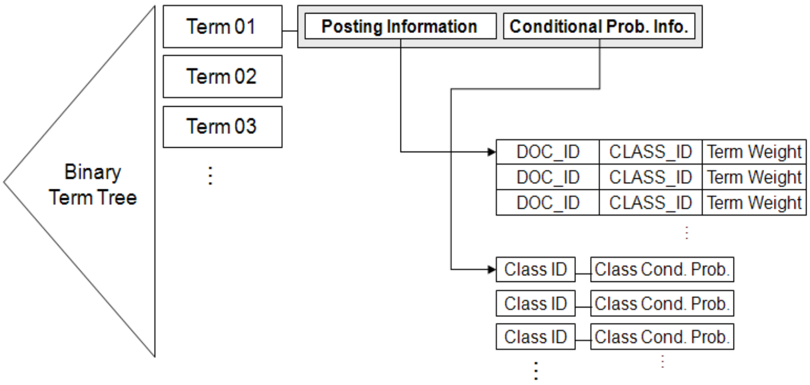


**Fig. 4**. Simplified Term Information Storage Structure for Classification (Execution)

### 3.2 Classification Models

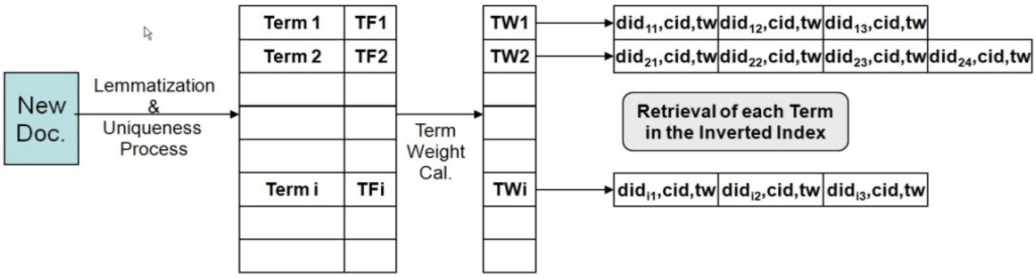
The proposed system now supports the two well-known classification settings that are highly efficient and promising: naïve Bayesian and *k*-nearest neighbor. If a function *f*(·) is defined as a text classifier which returns a relevant class given an input document, the naïve Bayesian classifier, which is the first algorithm, is

$$f(d) = \underset{c_i}{\arg\max}\, P(c_i)\prod_j P(x_j \mid c_i) = \underset{c_i}{\arg\max}\left(\log P(c_i) + \sum_j \log P(x_j \mid c_i)\right) \tag{1}$$

where a document $d$ is composed of and also can be represented by a set of terms $x_j$, with multiple target classes $c_i$. In order to estimate the conditional probability of a term given a class, $P(x_j \mid c_i)$, a *multinomial estimation method* is used:

$$P(x_j|c_i) = \frac{\sum_{d_k \in c_i} tf_{jk} + 1}{\sum_{w_t \in V}\sum_{d_k \in c_i} tf_{tk} + |V|} \tag{2}$$

In this case, $V$ is a set of all of the terms in the entire collection, and $tf_{ij}$ is the term frequency of the term $t_i$ in the document $d_j$.



$$\text{Distance}[\text{did}_{\text{ik}}] = \sqrt{\sum (\text{TW}_{\text{i}} - \text{tw}_{\text{ik}})^2}$$

$$\text{COS\_Measure}[\text{did}_{\text{ik}}] = \frac{\sum \text{TW}_{\text{i}} \times \text{tw}_{\text{ik}}}{|\text{new\_doc\_vec}| \times \text{doc\_vec\_len}[\text{did}_{\text{ik}}]}$$

**Fig. 5**. Document Vector Similarity Calculation based on a Retrieval Process

The second classification algorithm is the k-nearest neighbor algorithm, which is similar to the normal retrieval process based on the vector space model. A newly appearing document will be classified using the class information of the most similar documents as previously loaded in the document vector space. Based on the most similar k documents, the majority vote method, which favors the most frequent class in the nearest neighbors, was used.

$$f(d)=\underset{c_i}{\arg\max}\left[Freq\left(kNN(d),c_i\right)\right], NN(d,k)=\left\{d_i \in U \mid SimilarityRank\left(d_i,d\right) \le k\right\} \quad (3)$$

In this equation, $d$ is a newly appearing document, $c_i$ is a specific class and *SimilarityRank* ($d_i$, $d$) returns the ranking information based on the similarity between the two documents ($d_i$, $d$). The proposed system provides two similarity measurement approaches. With the assumption of the vector space model, a document $d$ can be expressed as the vector ($w_1$,…, $w_k$), where $w_i$ is the term weight value of the term $t_i$ in the document and $k$ is the number of terms in the entire collection.

**Fig. 5** depicts the simplified procedure for calculating the similarity between two documents using the retrieval process based on the vector space model, which is a very common and natural model in this domain. The focus here is not on the model itself but on the comprehensiveness and refinement of the framework. In general, it is not straightforward to develop a complete classification system that has multiple underpinning models. Moreover, literature regarding the implementation of classifiers based on retrieval models continues to be rare.

$$Sim_{\cos}\left(\vec{d}_1,\vec{d}_2\right) = \frac{\vec{d}_1 \cdot \vec{d}_2}{\left|\vec{d}_1\right| \times \left|\vec{d}_2\right|}$$
$$Sim_{dst}\left(\vec{d}_1,\vec{d}_2\right) = \sqrt{\sum_{i=1}^{k}\left(w_i^1 - w_i^2\right)^2} \quad (4)$$

*Sim*$_{cos}$ is the cosine value of the angle between $d_1$ and $d_2$, while *Sim*$_{dst}$ is the Euclidian distance between two document vectors. Users may select one of the two similarity methods according to their target domains.

## 4. User Configuration System

One of the strengths of *DICE* is its customization ability. Using the elaborate configuration function of *DICE*, a user can minutely control the entire system so as to ensure its highest level of performance in a certain domain. **Table 1** shows the details of each configuration item of the system. There is no graphical user interface, although such an interface will be implemented eventually, that enables us to specify these configuration items more conveniently. Among the elements, the pivotal elements are TCL_METHOD, where the preferred classification model is specified, and TCL_FS_METHOD, where the chosen feature selection method is written out of five methods.

**Table 1**. The Configuration Elements of *DICE*

| Configuration Items | Explanations |
|---|---|
| LEMMA_DIC_DIR | Directory of dictionaries that are required to lemmatize |
| LEMMA_AS_TERM | Whether a lemma is extracted as an index or not |
| STOP_WORDS | Whether stopwords will be used or not |
| STOP_WORD_FILE | File name containing the stopwords |
| TARGET_CLASSES | All of the categories in the current domain |
| TRAIN_DATA_DIR | Directory containing the training set |
| TEST_DATA_DIR | Directory containing the test set |
| CM_FILE_NAME | Confusion Matrix (Contingency Table) File Name |
| TCL_METHOD | Classification Model (kNN or naïve Bayesian) |
| TCL_CP_FILE | List of all Conditional Probabilities ($P(w \mid c)$) of terms |
| TCL_TW_FILE | List of all Posting Information (in IR) containing Term Weights |
| TCL_CI_FILE | List of all Classes and their Probabilities ($P(c)$) |
| TCL_KNN_K | The number of neighborhoods checked by kNN model |
| TCL_KNN_SIM | Similarity Measure (COS, DST) used in kNN model |
| TOTAL_TF_THRESH | Collection Frequency Threshold of terms |
| TCL_FS_METHOD | Feature Selection Method (NN, DF, IG, MI, CS) |
| TCL_FS_THRESH | Threshold for each Feature Selection Method |
| CPB | Whether Conditional Probability Boosting Method is used or not |

## 5. Experiments and Discussion

The experiment has two goals. First, it intends to demonstrate the location of the optimal settings of each collection using *DICE*'s flexible architecture; this can be achieved by the repetitive parameter alteration of the system. Secondly, though this intensive experiment, this paper presents the scalability and applicability of the system as it easily handles very large collections such as those at WebKB.

### 5.1 Datasets

In this paper, to evaluate the proposed *DICE* system, a series of intensive experiments were performed using the four distinct test collections consisting of Reuters-21578[1], 20 newsgroup[2],

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

four corpuses from four universities from the WebKB project[3] and blog postings with their emotional status (mood) attached[4]. By adapting various feature selection methods and altering the language processing approaches, the variations in the effectiveness and efficiency of *DICE* were investigated. **Table 2** gives a summary of the four target collections used in this paper.

**Table 2**. Experimental Datasets

| | Reuters-21578 | 20 Newsgroups | WebKB | Mood Corpus |
|---|---|---|---|---|
| **Subset** | ModApte | Total | Total | Total |
| **# classes** | 10 | 20 | 7 | 4 |
| **Total size** | 5.8 Mbytes | 35.7 Mbytes | 42.4 Mbytes | 6.6 Mbytes |
| **# documents** | 7,522 | 18,846 | 8,281 | 7,003 |
| **# training docs** | 6,022 | 15,077 | 6,624 | 5,604 |
| **# testing docs** | 1,502 | 3,769 | 1,657 | 1,399 |
| **# original features** | 5,859 (7,164) | 27,409 (32,533) | 20,391 (23,377) | 6,178 (7,549) |

**Table 3** shows the distributed (skewed) status of each document set with respect to the categories it contains. In particular, the variation in the number of documents of each class is very high in Reuter-21578 and WebKB, which might influence the final performance of the classification system.

**Table 3**. Document Distribution with respect to Classes

| Dataset | # of documents per class (sorted by number) |
|---|---|
| **Reuters-21578** | gold(99), coffee(114), sugar(135), ship(156), interest(211), money-fx(259), trade(333), crude(355), acq(2125), earn(3735) |
| **20 Newsgroups** | talk.religion.misc(628), talk.politics.misc(775), alt.atheism(799), talk.politics.guns(910), talk.politics.mideast(940), comp.sys.mac.hardware(963), comp.graphics(973), misc.forsale(975), comp.sys.ibm.pc.hardwarre(982), sci.electronics(984), comp.os.ms-windows,misc(985), sci.space(987), comp.windows.x(988), rec.autos(990), sci.med(990), sci.crypt(991), rec.sport.baseball(994), rec.motorcycles(996), soc.religion.christian(997), rec.sport.hockey(999) |
| **WebKB** | staff(137), department(181), project(504), course(930), faculty(1124), student(1641), other(3764) |
| **Mood Corpus** | fear(1011), sad(1031), angry(1136), happy(3825) |

## 5.2 Experimental Settings

**Table 4** describes the eight types of the experimental settings apart from the case in which the k nearest neighbor model is used. Therefore, a total of 32 experimental settings exist in this paper, although all are not considered here in the interest of brevity.

## 5.3 Effectiveness

*DICE* support the three evaluation methods of the micro-averaged F1 measure [4][32] the macro-averaged $F_1$ measure [32] and the traditional precision/recall method. During the optimization and adjustment of certain domains, any evaluation method can be used and, more importantly, compared any other at any time. This paper focuses mainly on the effectiveness in

[2] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

[3] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

[4] This dataset was created by *IR.NLP Lab. ICU*, and is not yet published.

terms of the micro-averaged $F_1$ measure. All of the experimental results include both the lemmatization effect (the usage of *DELEMMA*) and the influence of the feature selection methods (*DF*, *IG*, *CS*, and *MI*). Particularly, the number of features was not fixed in advance using the feature selection methods as in most other studies. To observe the detailed variation of the performance, the threshold values were changed directly in this experiment. Hence, the resulting graphs appear to be somewhat complicated and uneven with respect to the changes in the number of features. Given these graphs, however, the behavior of the entire system can be inspected and analyzed more precisely.

**Table 4**. Experimental Settings (in only the naïve Bayesian mode)

| Settings | Model[5] | Feature Selection | Lemmatization[6] |
|---|---|---|---|
| DF.nolem.NB | naïve Bayesian | Document Frequency | No |
| IG.nolem.NB | naïve Bayesian | Information Gain | No |
| MI.nolem.NB | naïve Bayesian | Mutual Information | No |
| CS.nolem.NB | naïve Bayesian | $X^2$ Statistics | No |
| DF.lem.NB | naïve Bayesian | Document Frequency | Yes |
| IG.lem.NB | naïve Bayesian | Information Gain | Yes |
| MI.lem.NB | naïve Bayesian | Mutual Information | Yes |
| CS.lem.NB | naïve Bayesian | $X^2$ Statistics | Yes |

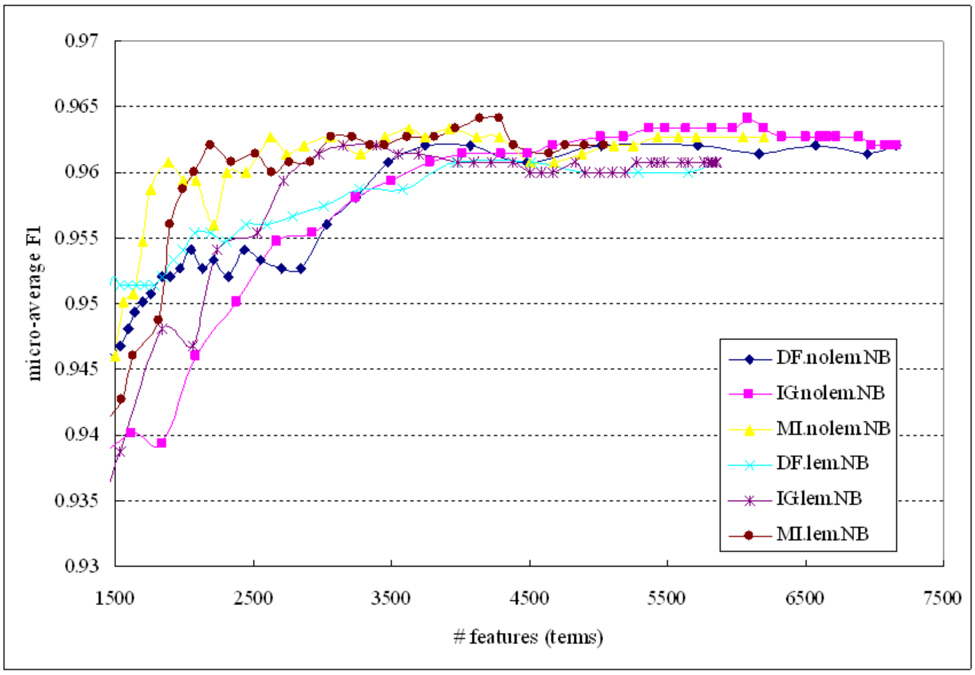


**Fig. 6**. Micro-average F1 values of the naïve Bayesian classifier (Reuter-21578)

**Fig. 6** shows the performance of the six settings of the naïve Bayesian classification of Reuter-21578. Each element in the legend in this figure indicates both the feature selection

[5] This indicates the classification model used in each setting. In the interest of brevity, we have omitted the settings related to the k nearest neighbor model.

[6] Whether lemmatization is applied or not

method that was used and whether or not it involved lemmatization. Given this figure, *MI.lem.NB* showed slightly better performance. Moreover, regardless of whether or not lemmatization occurred, it is clear that with fewer features, applying mutual information as the feature selection criteria generally works well, whereas the information gain shows better performance with larger features. In the range of the feature numbers between 2,000 and 4,000, *DELEMMA* had a beneficial effect on the performance. This can be explained by the fact that the lemmatization process helps to select the favorable features by adjusting the four feature selection methods.

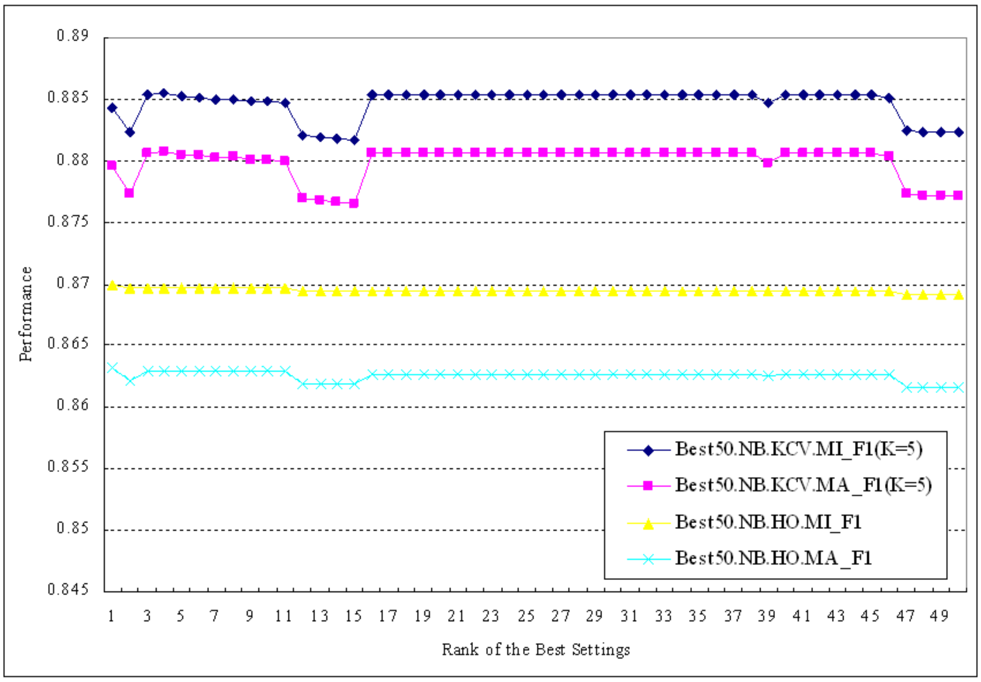


**Fig. 7**. Comparison in μF1 and mF1 between five-fold cross validation ("KCV") and held-out evaluation ("HO") using Reuter-21578

**Table 5** denotes the 10 settings that showing the best performance in terms of micro-averaged F1 values. The values in the parentheses indicate the thresholds that were applied in each feature selection method. Here, *MI.lem.NB* shows the best performance while using fewer features compared to other settings in the table. Moreover, in general, regardless of whether lemmatization applied, mutual information (*MI*) appears to be the best feature selection method in the Reuter-21578 collection in terms of both the number of features and its average performance.

In order to confirm the reliability of the performance of the best settings presented in **Table 5**, another experiment was run using five-fold cross validation over the top ranked 50 settings. The final performance was computed by averaging the micro-averaged $F1$ values of all five trials. **Fig. 7** summarizes the results of the experiments. It is clear from the figure that the rank given in **Table 5** should be altered when the cross validation method is used. Moreover, although this requires further investigation, the number of features each setting generates and uses is likely to have an impact on the degree of the difference in performance between the two evaluation methods. For example, the numbers of features for the 1*st*, 2*nd*, 4*th*, 5*th* settings are relatively small. In these settings, the micro-averaged $F1$ values are also smaller than in the other settings, which cannot be sufficiently explained by the fact that a higher number of features leads to better performance. Further research will include theoretical and empirical approaches to these phenomena.

**Table 5**. Performance of Top 10 Settings on Reuter-21578 (naïve Bayesian)

| R | Settings | #feat. | µF1[a] | *m*F1[b] |
|---|---|---|---|---|
| 1 | MI.lem.NB(0.22) | 4,143 | 0.9640 | 0.8978 |
| 1 | MI.lem.NB(0.20) | 4,282 | 0.9640 | 0.8950 |
| 1 | IG.nolem.NB(0.52) | 6,081 | 0.9640 | 0.8944 |
| 4 | MI.nolem.NB(0.38) | 3,630 | 0.9633 | 0.8943 |
| 4 | MI.nolem.NB(0.34) | 3,923 | 0.9633 | 0.8942 |
| 4 | CS.nolem.NB(0.50) | 6,900 | 0.9633 | 0.8936 |
| 4 | CS.nolem.NB(0.40) | 7,028 | 0.9633 | 0.8928 |
| 4 | IG.nolem.NB(0.54) | 5,975 | 0.9633 | 0.8928 |
| 4 | IG.nolem.NB(0.56) | 5,832 | 0.9633 | 0.8928 |
| 4 | IG.nolem.NB(0.58) | 5,646 | 0.9633 | 0.8928 |

**Table 6**. Performance of Top 10 Settings on Reuter-21578 (kNN)

| R | Settings | #feat. | µF1 | *m*F1 |
|---|---|---|---|---|
| 1 | DF.nolem.KNN(28) | 1,534 | 0.9427 | 0.8476 |
| 2 | DF.nolem.KNN(29) | 1,477 | 0.9414 | 0.8466 |
| 2 | DF.nolem.KNN(27) | 1,589 | 0.9407 | 0.8462 |
| 4 | DF.nolem.KNN(25) | 1,693 | 0.9400 | 0.8480 |
| 5 | DF.nolem.KNN(34) | 1,298 | 0.9400 | 0.8515 |
| 5 | DF.nolem.KNN(32) | 1,369 | 0.9400 | 0.8478 |
| 5 | DF.nolem.KNN(31) | 1,395 | 0.9400 | 0.8466 |
| 5 | DF.nolem.KNN(30) | 1,435 | 0.9400 | 0.8462 |
| 5 | DF.nolem.KNN(26) | 1,635 | 0.9400 | 0.8414 |
| 10 | DF.lem.KNN(74) | 614 | 0.9394 | 0.8457 |

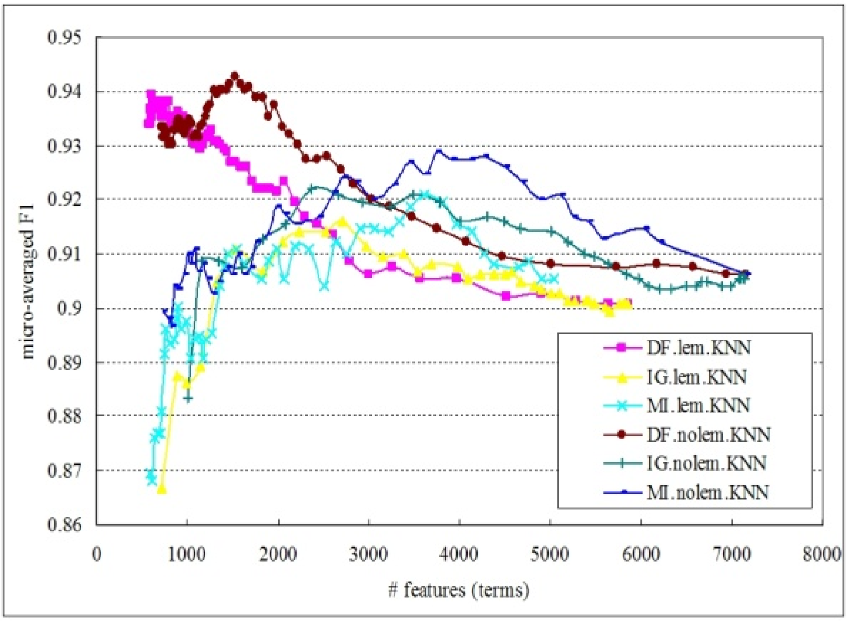


**Fig. 8**. Micro-averaged F1 values of the kNN classifier (k=9) (Reuter-21578)

**Fig. 8** depicts the results of *DICE*'s kNN classification on Reuter-21578. Over the past decade, a considerable number of studies have been conduced regarding the impacts of feature selection methods on the kNN classifiers, as described in the beginning of this paper. This may explain why the minor details for the moment are relatively less important. However, it is important to emphasize in this figure that *DF* is the best feature selection method in the kNN classification of *DICE*, which is a completely different result from the naïve Bayesian result. This result is somewhat surprising, as many studies have reported that *IG* or *MI* is more effective than others in the case of the kNN classification.

As in the naïve Bayesian classification, **Table 6** lists the top 10 settings with respect to the micro-averaged *F*1 among all settings of the kNN classification of *DICE*. As shown in **Fig. 8**, if lemmatization is used, the number of features is greatly reduced while a competitive performance level if retained. For example, the 10$^{th}$ setting, *DF.lem.KNN (74)*, has only 614 features out of a total of 5,859, whereas its micro-averaged and macro-averaged $F1$ values are nearly identical to the top-ranked setting. Due to space limitations, the full version of this ranked result is omitted. In this investigation, however, out of the best 30 settings, 16 settings use lemmatization, at an average of 693 features each.

**Fig. 9** shows the performance of the naïve Bayesian classification of *DICE* on the 20-newsgroup collection. This figure shows two important points: (1) The *DF.lem.NB* setting gives the most effective results, especially with a smaller number of features; (2) both *CS.lem.NB* and *CS.nolem.NB* show the best performance with a relatively large number of features. In particular, the benefits of lemmatization are demonstrated in this domain. As shown in the previous experiment, naïve Bayesian on Reuter-21578, lemmatization was unable to improve upon the performance level. In the 20-newsgroup domain, however, it helped raise the overall performance regardless of the feature selection method used. Although this cannot be proved systematically and theoretically, it is likely that if a domain has a sufficiently large number of features, lemmatization can improve the feature selection methods as they evaluate and choose the beneficial features from the entire set.

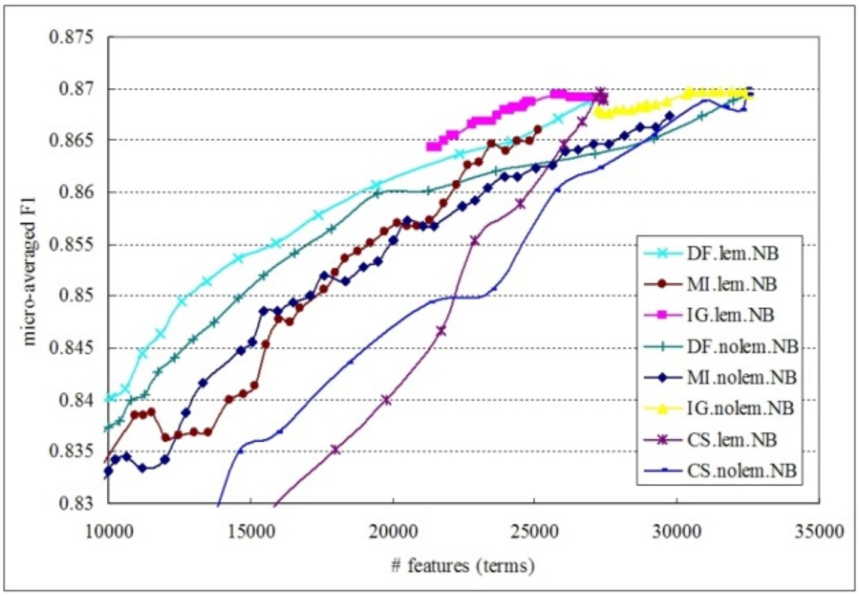


**Fig. 9.** Micro-averaged F1 values of the naïve Bayesian classifier (20 newsgroups)

More importantly, there are only very small performance gains using feature selection methods in this domain. As shown in **Table 7**, almost all of the best settings without lemmatization generate and utilize sets of features that are nearly equal in numbers. When lemmatization is applied, specifically the *CS.lem.NB* and *IG.lem.NB* settings, the features for these settings are reduced in number by nearly 5,000 on average. In contrast, the performance remains nearly identical to when the settings do not involve lemmatization. It is important to note that in the bottom line of **Table 7**, a setting without any feature selection and lemmatization is given which shows nearly the same micro-averaged $F1$ value as the others.

**Fig. 10** shows somewhat unexpected results when executing the five-fold cross validation on the best fifty settings which have the largest value given of $\mu F1$ based on the held-out evaluation (80% for training, 20% for testing). Far from being similar to the previous case

(Reuters-21578), what the figure indicates is in stark contrast to conventional knowledge regarding k-fold cross validation. Each $\mu F1$ ($mF1$) value of the five-fold cross validation is larger than its corresponding value in the held-out evaluation. In order to reveal the cause of this result, the resulting data including all the subsets' performance values generated during the five-fold cross validation were closely examined. It was found that *DICE*, taken altogether, gives poor performance values, particularly with the subset chosen as the test set in the held-out evaluation. Except for this subset, all of the other values were better classified by *DICE*, showing nearly equivalent or larger $\mu F1$ values compared to the averaged five-fold cross validation results. As a result, the task of choosing one subset was not as reliable in the held-out evaluation. In any event, **Fig. 10** shows the implications of confident evaluations of text classification systems.

**Table 7**. Performance of Top 10 Settings on 20 newsgroup (naïve Bayesian)

| Rank | Settings | # feat. | µF1 | *m*F1 |
|---|---|---|---|---|
| 1 | CS.nolem.NB(0.80) | 32,446 | 0.8699 | 0.8631 |
| 2 | CS.lem.NB(0.80) | 27,336 | 0.8697 | 0.8621 |
| 2 | IG.nolem.NB(0.52) | 32,516 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.54) | 32,432 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.56) | 32,246 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.58) | 31,938 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.60) | 31,552 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.62) | 31,216 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.64) | 30,889 | 0.8697 | 0.8629 |
| 2 | IG.nolem.NB(0.66) | 30,525 | 0.8697 | 0.8629 |

**Table 8**. Performance of Top 10 Settings on WebKB (naïve Bayesian)

| Rank | Settings | # feat. | µF1 |
|---|---|---|---|
| 1 | IG.nolem.NB(0.74) | 9,357 | 0.656401 |
| 2 | DF.nolem.NB(9) | 10,642 | 0.653382 |
| 2 | IG.nolem.NB(0.58) | 15,401 | 0.653382 |
| 4 | DF.nolem.NB(11) | 9,198 | 0.652778 |
| 4 | DF.nolem.NB(12) | 8,637 | 0.652778 |
| 4 | DF.nolem.NB(13) | 8,138 | 0.652778 |
| 4 | MI.nolem.NB(0.18) | 15,512 | 0.652778 |
| 4 | MI.nolem.NB(0.2) | 14,552 | 0.652778 |
| 9 | DF.nolem.NB(8) | 11,584 | 0.652174 |
| 9 | DF.nolem.NB(10) | 9,826 | 0.652174 |

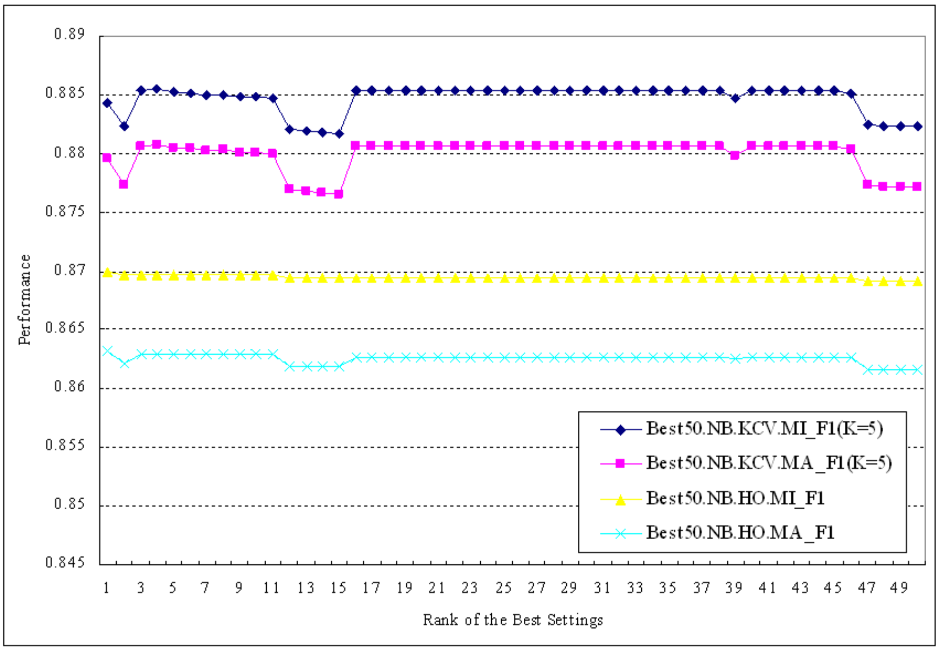


**Fig. 10**. Comparison in µF1 and mF1 between the six-fold cross validation ("KCV") and the held-out evaluation ("HO") (20 newsgroup)

The third test collection was *WebKB*, which is composed of a large number of HTML documents. This dataset was selected to investigate the filtering ability of *DICE*. In contrast to the other datasets used in this paper, it is clear that each document in this collection includes many meaningless tokens, specifically HTML tags. The entire dataset was used deliberately; by leaving in these HTML tags and unimportant tokens, the potential performance of *DICE* can be determined. To investigate the impact of its capabilities on unclean or messy datasets fully, an experiment should be performed using a clarified version of *WebKB* with a comparison between the two drawn afterward. This will be carried out in a future study.

Initially, it was expected that the feature selection methods provided by *DICE* would have a great impact on the overall performance in the domain of *WebKB*. However, **Fig. 11** indicates that there was no significant difference whether or not the feature selection methods were applied. Nevertheless, one important aspect of this finding is that *DICE* showed that it could reduce the number of features to approximately 10% of the entire feature set, specifically to approximately 23,000 using the two settings (*DF.lem.NB* and *DF.nolem.NB*) without a great hit on its performance. Moreover, although *IG.nolem.NB* shows the highest $\mu F1$ value while using nearly 10,000 features in this domain, its performance decreases abruptly when the number of features falls below 10,000. Therefore, it follows that *DF* is likely to be the most reliable and competitive feature selection method in the *WebKB* domain when the naïve Bayesian model is used as the classification model.

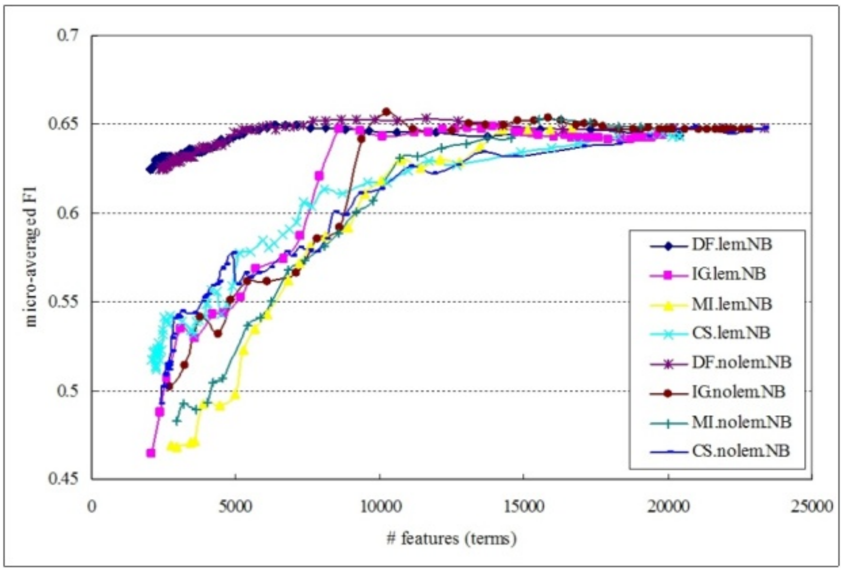


**Fig. 11**. Micro-averaged F1 values of the naïve Bayesian classifier on (*WebKB*)

As in the previous two experiments, a set of the best settings is also given in the *WebKB* domain. **Table 8** shows the 10 settings which had the highest micro-averaged $F1$ values. As the table indicates, except for the last setting, every setting did not involve lemmatization. Another distinguishing trait of the ranked list is that it includes all three feature selection methods (*DF*, *IG*, *MI*), which implies there is no prominent method for selecting useful features in this domain. In addition, the last setting of the list shows that lemmatization can be helpful in reducing the number of features while maintaining good performance.

The last dataset used was *Mood Corpus* which is composed of nearly 7,000 Web blog postings. Each posting (document) includes one of a total of four classes expressing the mood or emotional status of the current content; specifically these are angry, sad, happy, and fear. **Fig. 12** shows the global performance of *DICE* in this domain in terms of both the various numbers of features and feature selection methods. The cases in which lemmatization was applied are also included in the figure. This figure shows clearly that in this domain, *MI* (mutual information) is the best feature selection method. In addition, *DF* is also attractive, showing competitive performance. Related to this figure, **Table 9** contains the settings when using only the *MI* feature selection.

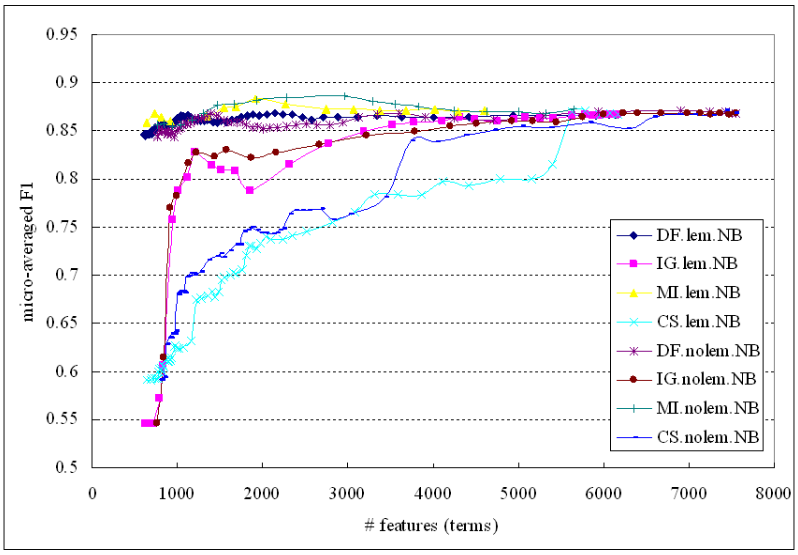


**Fig. 122**. Micro-averaged F1 values of the naïve Bayesian classifier (Mood Corpus)

**Table 9**. Performance of Top 15 Setting on *Mood Corpus* (Naïve Bayesian)

| Rank | Settings | # features | *F1 | *m*F1 |
|---|---|---|---|---|
| 1 | MI.nolem.NB (0.26) | 2,927 | 0.886347 | 0.840944 |
| 2 | MI.nolem.NB (0.28) | 2,288 | 0.884203 | 0.837813 |
| 3 | MI.lem.NB (0.26) | 1,917 | 0.883488 | 0.846795 |
| 4 | MI.nolem.NB (0.3) | 1,939 | 0.882059 | 0.838674 |
| 5 | MI.nolem.NB (0.24) | 3,298 | 0.879914 | 0.832808 |
| 6 | MI.lem.NB (0.24) | 2,278 | 0.87777 | 0.835196 |
| 6 | MI.nolem.NB (0.32) | 1,677 | 0.87777 | 0.83788 |
| 8 | MI.nolem.NB (0.22) | 3,565 | 0.877055 | 0.828827 |
| 9 | MI.nolem.NB (0.34) | 1,479 | 0.87634 | 0.839834 |
| 10 | MI.lem.NB (0.28) | 1,691 | 0.875625 | 0.837772 |

In this section, a series of very intensive analyses was discussed regarding the accuracy of *DICE* on four different datasets while varying the parameters that specify both the four feature selection methods and the two classification models. One of the contributions of this experiment related to the effectiveness of a text classification framework is that a new experimental procedure revealing the parametric details of the proposed system was invented and executed. Moreover, the optimal parameters for a particular domain were suggested. Through this approach, aspects related to the effectiveness of the current version of *DICE* were disclosed. Thus far, there is no known in-depth experimental study in the domain of text classification outside of the present work. Additionally, this paper does not simply aim to highlight the innate superiority of the proposed system; instead, it attempts to exhaustively reveal its current pros and cons.

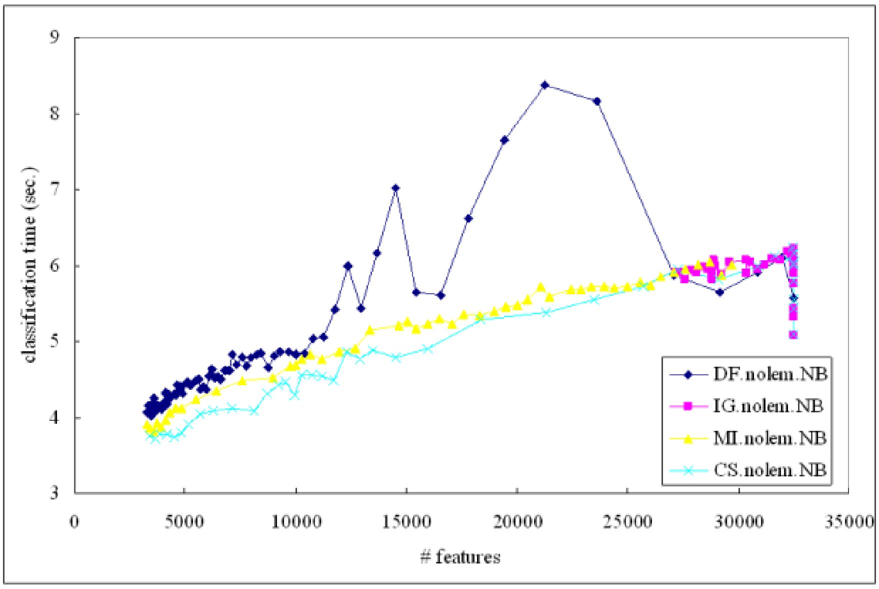
## 5.4 Efficiency

In order to be used as a practical application, it is critical for *DICE* to show competitive speed in both training and classification. In this subsection, complete and detailed results of experiments related to the training and classification time are given. In actuality, measuring and comparing the speed of some applications is not trivial because it is completely dependent on the specifications of the hardware and operating system with which these types of experiments are executed. In this paper, a Linux server equipped by 4 GB of memory and four 2.8 GHz Intel Zeon CPUs was used.

**Table 10**. Summary of Training and Classification Time

| Dataset | Tr. Size (Mbytes) | Training (sec.) | Cl. Size (Mbytes) | Testing (sec.) | T-Speed(1) (docs/sec.) | T-Speed(2) (Mbytes/sec.) |
|---|---|---|---|---|---|---|
| Reuter-21578 (NB) | 4.404 | 5.238 | 0.965 | 0.878 | 1710.706 | 1.099 |
| 20 Newsgroup | 26.704 | 39.073 | 6.481 | 5.149 | 731.987 | 1.292 |
| WebKB | 32.601 | 45.972 | 8.577 | 4.146 | 399.421 | 2.069 |
| Mood Corpus | 5.551 | 5.003 | 1.252 | 1.037 | 1349.083 | 1.207 |
| Reuter-21578 (KNN) | 4.404 | 4.378 | 0.965 | 24.007 | 62.565 | 0.040 |

**Table 10** shows summary of the efficiency test, which includes the five-time average of the training and classification time of *DICE*, the number of documents that were classified in one second, and other information. Due to the variations in the size of one document in each dataset, document sizes are given based on the classification time as well so as to evaluate the system accurately and objectively. In the *NB* classifiers, except for the *WebKB* dataset, *DICE* can classify close to 1.1 Mbytes of text per second. As expected, *WebKB* contains many special characters denoting HTML tags. Therefore, the resulting size is exceedingly large. On the other hand, in spite of the reputation of *KNN* in relation to its efficiency, its speed is very slow compared to *NB*. Further research will include algorithmic and implementation evaluations of the two classification models.



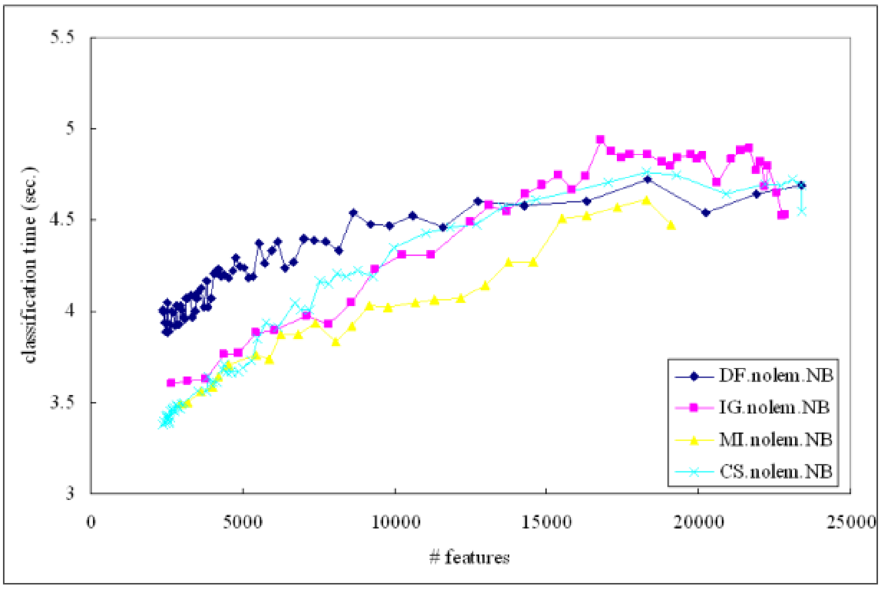


**Fig. 13**. Classification Time (Naïve Bayesian, Concurrent Environment), Top: 20 Newsgroup (3,769), Bottom: WebKB (1,657)

As shown in **Fig. 13**, the classification time is a somewhat susceptible to the size of the feature set,. Each caption in the sub-figure includes both the name of the target dataset and the number of the documents in it. Although some fluctuations and increments in the running time are shown in these graphs, as in the case of the training time, *DICE* can most likely be operated in the concurrent environment without any large penalty in speed.

Finally, a figure is presented which compares the training and classification times of both the *NB* and *KNN* models. This is shown in **Fig. 14**. This figure is the final figure regarding the comparative experiment of the efficiency of the two models. To demonstrate the reliability of this test, the two large datasets of *WebKB* and the 20-newsgroup set were combined to generate the largest dataset used in this experiment. This figure shows that the classification time of *KNN* monotonically increases, whereas the running time of *NB* is nearly constant. From a practical viewpoint, *NB* is superior to *KNN* with respect to both effectiveness (shown in the previous section) and efficiency. Moreover, with these objective experimental results, it is clear that *NB* would be a more attractive selection for the development of a real-time text classification system.

## 6. Conclusion

This paper presents a domain-independent text classification framework (*DICE*). Although there is no newly created approach in this system, an open-source-based framework system was developed which enables many researchers to invent creative approaches to text classification. To allow the developed system to be easily analyzed and used, the architecture of *DICE* was clarified explicitly in that many conventional structures related to the management and retrieval of information concerning text classification were adapted and modified. Although the source code of the modules involved is not presented, the extendable algorithmic formation of attaching additional feature selection methods can facilitate active research regarding the impact of various approaches to selecting useful and important terms from the training documents. Those who want to make use of the system without any modifications will be able to optimize it for a particular domain simply by adjusting the configuration file.

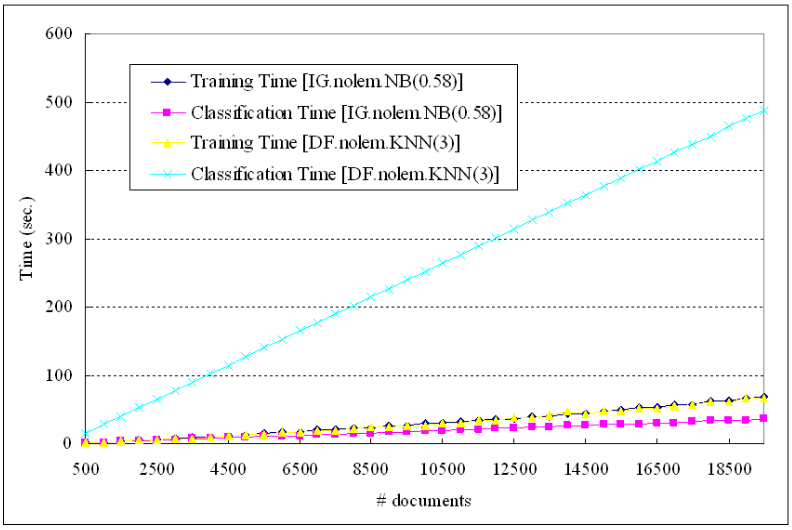


**Fig. 14**. Training and Testing Speed w.r.t. the increase in the number of documents (*IG.nolem.NB (0.58)*, *DF.nolem.KNN (3)*)

This experimental approach can be considered unique in that the optimal points (a set of parameter values) are suggested for each dataset (domain), while other papers and researchers have presented only the abstracted results which came from a hidden procedure of optimizing their systems or approaches. There has been no known attempt to reveal the inner part of the text classification systems directly. Moreover, an in-depth analysis is given of an efficiency test on *DICE* in which its speed is demonstrated.

There are a number of application frameworks or libraries of general purpose classification models open to the public. However, few of them focus on the text classification. They merely provide a classification function itself based on well-made feature spaces. *DICE* concentrates on text classification while providing many meaningful functions related to it. It is hoped that *DICE* will be released via the Internet so that many researchers can harness and improve it by adapting numerous creative theories and mechanisms.

## References

[1] F. Sebastiani, “Machine learning in automated text categorization,” *ACM Computing Surveys*, Vol. 34, No. 1, pp. 1–47, 2002.

[2] J. Dörre, P. Gerstl, and R. Seiffert, “Text mining: finding nuggets in mountains of textual data,” in *Proceedings of KDD-99, 5th ACM International Conference on Knowledge Discovery and Data Mining*. San Diego, US: ACM Press, New York, USA, pp. 398–401.

[3] D. D. Lewis, D. L. Stern, and A. Singhal, “ATTICS: a software platform for on-line text classification,” in *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, M. A. Hearst, F. Gey, and R. Tong, Eds. Berkeley, US: ACM Press, New York, US, 1999, pp. 267–268.

[4] G. Forman, “An extensive empirical study of feature selection metrics for text classification,” *Journal of Machine Learning Research*, Vol. 3, pp. 1289–1305, March 2003.

[5] Moulinier, “Feature selection: a useful preprocessing step,” in *Proceedings of BCSIRSG-97, the 19th Annual Colloquium of the British Computer Society Information Retrieval Specialist Group*,

ser. Electronic Workshops in Computing, J. Furner and D. Harper, Eds. Aberdeen, UK: Springer Verlag, Heidelberg, DE, 1997.

[6] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of ICML-97, 14th International Conference on Machine Learning*, D. H. Fisher, Ed. Nashville, US: Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 412–420.

[7] Moschitti, "A study on optimal parameter tuning for rocchio text classifier," in *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, F. Sebastiani, Ed. Pisa, IT: Springer Verlag, pp. 420–435, 2003.

[8] H. Paijmans, "Text categorization as an information retrieval task," *The South African Computer Journal*, vol. 21, pp. 4–15, 1999.

[9] D. Mladenić, "Feature subset selection in text learning," in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, C. N´edellec and C. Rouveirol, Eds. Chemnitz, DE: Springer Verlag, Heidelberg, DE, 1998, pp. 95–100, published in the "Lecture Notes in Computer Science" Series Number 1398.

[10] M. Radovanović and M. Ivanović, "Interactions between document representation and feature selection in text categorization," in *Proceedings of DEXA-06, 17th International Conference on Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, vol. 4080. Krakow, Poland: Springer-Verlag, pp.489–498, 2006.

[11] M. Makrehchi and M. S. Kamel, "Text classification using small number of features," in *Proceedings of MLDM-05, 4th International Conference on Machine Learning and Data Mining in Pattern Recognition*, ser. Lecture Notes in Artificial Intelligence, Vol. 3587. Leipzig, Germany: Springer-Verlag, pp. 580–589, 2005.

[12] D. Mladenić and M. Grobelnik, "Feature selection on hierarchy of web documents," *Decision Support Systems*, Vol. 35, No. 1, pp. 45–87, 2003.

[13] Kolcz, V. Prabakarmurthi, and J. K. Kalita, "String match and text extraction: Summarization as feature selection for text categorization," in *Proceedings of CIKM-01, 10th ACM International Conference on Information and Knowledge Management*, H. Paques, L. Liu, and D. Grossman, Eds. Atlanta, US: ACM Press, New York, US, pp. 365–370, 2001.

[14] G. Forman, "A pitfall and solution in multi-class feature selection for text classification," in *Proceedings of ICML-04, 21st International Conference on Machine Learning*, C. E. Brodley, Ed. Banff, CA: Morgan Kaufmann Publishers, San Francisco, US, 2004.

[15] E. Gabrilovich and S. Markovitch, "Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5," in *Proceedings of ICML-04, 21st International Conference on Machine Learning*, C. E. Brodley, Ed. Banff, CA: Morgan Kaufmann Publishers, San Francisco, US, 2004.

[16] L. Galavotti, F. Sebastiani, and M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization," in *Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries*, J. L. Borbinha and T. Baker, Eds. Lisbon, PT: Springer Verlag, Heidelberg, DE, 2000, pp. 59–68, published in the "Lecture Notes in Computer Science" Series Number 1923.

[17] E. Montañés, I. Díaz, J. Ranilla, E. F. Combarro, and J. Fern´andez, "Scoring and selecting terms for text categorization," *IEEE Intelligent Systems*, Vol. 20, No. 3, pp. 40–47, 2005.

[18] H. Taira and M. Haruno, "Feature selection in svm text categorization," in *Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence*. Orlando, US: AAAI Press, Menlo Park, US, pp. 480–486, 1999.

[19] Y.-S. Lai and C.-H. Wu, "Meaningful term extraction and discriminative term selection in text categorization via unknown-word methodology," *ACM Transactions on Asian Language Information Processing*, Vol. 1, No. 1, pp. 34–64, 2002.

[20] Lee and G. G. Lee, "Information gain and divergence-based feature selection for machine learning-based text categorization," *Information Processing and Management*, Vol. 42, No. 1, pp. 155–165, 2006.

[21] P. Soucy and G. W. Mineau, "A simple feature selection method for text classification," in *Proceeding of IJCAI-01, 17th International Joint Conference on Artificial Intelligence*, B. Nebel,

Ed., Seattle, US, pp. 897–902, 2001.

[22] S. Cohen, E. Ruppin, and G. Dror, “Feature selection based on the shapely value,” in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Edinburgh, Scotand, August 2005, pp. 665–670.

[23] Kolcz and A. Chowdhury, “Avoidance of model re-induction in svmbased feature selection for text categorization,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, Hyderabad, India, pp. 889–894, 2007.

[24] J. Yan, N. Liu, B. Zhang, S. Yan, Z. Chen, Q. Cheng, W. Fan, and W.-Y. Ma, “OCFS: optimal orthogonal centroid feature selection for text categorization,” in *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, Salvador, Brazil, August 2005, pp. 122–129.

[25] Z. Zheng, X. Wu, and R. Srihari, “Feature selection for text categorization on imbalanced data,” *SIGKDD Explorations*, Vol. 6, No. 1, pp. 80–89, 2004.

[26] R. Basili, A. Moschitti, and M. T. Pazienza, “An hybrid approach to optimize feature selection process in text classification,” in *Proceedings of AI*IA-01, 7th Congress of the Italian Association for Artificial Intelligence*, F. Esposito, Ed. Bari, IT: Springer Verlag, Heidelberg, DE, 2001, pp. 320–325, published in the “Lecture Notes in Computer Science” Series Number 2175.

[27] W. Wibowo and H. E. Williams, “Simple and accurate feature selection for hierarchical categorisation,” in *Proceedings of the 2002 ACM Symposium on Document engineering*. McLean, US: ACM Press, New York, US, pp. 111–118, 2005.

[28] Dhillon, S. Mallela, and R. Kumar, “A divisive information-theoretic feature clustering algorithm for text classification,” *Journal of Machine Learning Research*, Vol. 3, pp. 1265–1287, March 2003.

[29] G. Wang and F. H. Lochovsky, “Feature selection with conditional mutual information maximin in text categorization,” in *Proceedings of CIKM-04, 13th ACM International Conference on Information and Knowledge Management*, D. A. Evans, L. Gravano, O. Herzog, C. Zhai, and M. Ronthaler, Eds. Washington, US: ACM Press, New York, US, pp. 342–349, 2004.

[30] Anagnostopoulos, A. Broder, and K. Punera, “Effective and efficient classification on a search-engine model,” in *CIKM*, 2006.

[31] (2001) The online plain text english dictionary. [Online]. Available: http://www.mso.anu.edu.au/ralph/OPTED/

[32] F. Sebastiani, “A tutorial on automated text categorisation,” in *Proceedings of ASAI-99, 1st Argentinean Symposium on Artificial Intelligence*, A. Amandi and R. Zunino, Eds., Buenos Aires, AR, 1999, pp. 7–35, an extended version appears as [1].

[33] (2008) Korean Indexing Engine. [Online]. Available:www.kristalinfo.com/K-Lab/idx.

[34] (2008) Korean Morphological Analyzer. [Online]. Available:www.kristalinfo.com/K-Lab/ma.

**Sung-Pil Choi** is a senior researcher at Korea Institute of Science and Technology Information (KISTI). And also he is a Ph.D. student in the School of Engineering at Information and Communications University (ICU), Korea. He received his MS from the Pusan National University (PNU). His current research interests include Text Mining, Machine Learning, Natural Language Processing, Information Retrieval and other related fields.

**Sung Hyon Myaeng** is currently a professor at Information and Communications University (ICU), Korea. Prior to this appointment, he was a faculty at Chungnam National University, Korea, and Syracuse University, USA, where he was granted tenure. He has served on program committees of many international conferences in the areas of information retrieval, natural language processing, and digital libraries, including his role as a chair for ACM SIGIR, 2002 and 2008, and for AIRS, 2004. He is on editorial boards of several international journals, including ACM Transactions on Asian Information Processing as an associate editor, Information Processing and Management, Journal of Natural Language Processing, and Journal of Computer Processing of Oriental Languages. Domestically, he was the chair of SIG-HLT (Human Language Technology), Korea Information Science Society, between 2005 and 2007. He has published numerous technical articles on conceptual graph-based IR, cross-language IR, automatic summarization, text categorization, topic detection and tracking, and distributed IR in the context of digital libraries.

**Hyun Yang Cho** is a professor of Department of Library & Information Science at Kyonggi University, Korea. Prior to this appointment, he was a senior researcher at Korea Institute of Science and Technology Information (KISTI). He had been served as aGeneral Secretary, an auditor, and board of trustee of Korea Society for Library and Information Science from 2005 to 2006. His primary research interests include evaluation of information retrieval systems and knowledge organization in general. His secondary research interests are focused on aspects of formal and informal scholarly communication such as bibliometric studies of scholarly literatures, social network analysis, and knowledge transfer in the area of science and technology.