A Class-Feature-Centroid Classifier for Text Categorization

Hu Guan
Computer Science Dept.
Shanghai Jiao Tong University
800 Dongchuan Road
Shanghai 200240, China
guanhu@sjtu.edu.cn

Jingyu Zhou
Computer Science Dept.
Shanghai Jiao Tong University
800 Dongchuan Road
Shanghai 200240, China
zhou-jy@cs.sjtu.edu.cn

Minyi Guo
Computer Science Dept.
Shanghai Jiao Tong University
800 Dongchuan Road
Shanghai 200240, China
guo-my@cs.sjtu.edu.cn

ABSTRACT

Automated text categorization is an important technique for many web applications, such as document indexing, document filtering, and cataloging web resources. Many different approaches have been proposed for the automated text categorization problem. Among them, centroid-based approaches have the advantages of short training time and testing time due to its computational efficiency. As a result, centroid-based classifiers have been widely used in many web applications. However, the accuracy of centroid-based classifiers is inferior to SVM, mainly because centroids found during construction are far from perfect locations.

We design a fast Class-Feature-Centroid (CFC) classifier for multi-class, single-label text categorization. In CFC, a centroid is built from two important class distributions: *inter-class term index* and *inner-class term index*. CFC proposes a novel combination of these indices and employs a denormalized cosine measure to calculate the similarity score between a text vector and a centroid. Experiments on the Reuters-21578 corpus and 20-newsgroup email collection show that CFC consistently outperforms the state-of-the-art SVM classifiers on both micro-F1 and macro-F1 scores. Particularly, CFC is more effective and robust than SVM when data is sparse.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation; Feature evaluation and selection*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms

Algorithms, Experimentation, Performance

Keywords

centroid, text classification, inter-class, inner-class, denormalized cosine measure

1. INTRODUCTION

Automatic text categorization (TC) is a process of assigning some class labels to text documents. In recent years, TC has been widely used in many web applications, for instance, query classification in search engines [2], deep classification of web documents [5, 38], and Blog classification [22]. For these web applications, it is often required that TC can be performed with a short training time and testing time, sometimes with incrementally updated data.

A number of TC techniques have been explored in the literature, for instance, Naive Bayes [6, 10], kNN [8], Neural Network [4], centroid-based approaches [2, 3, 16, 34], Decision Tree (DT) [17, 35, 7], Rocchio [26], and SVM [12, 5]. Among them, the centroid-based classifier is one of the most popular supervised approaches, due to the computational efficiency. However, previous work [31] has found that the performance of centroid-based classifiers is significantly lower than other approaches (e.g., SVM).

One of the reasons for the inferior performance of centroid-based classifiers is that centroids do not have good initial values. To solve this problem, many methods have been using feedback-loops to iteratively adjust prototype vectors, such as Dragpushing method [34], Hypothesis Margin method [34], and Weight Adjustment method [29]. These improved classifiers perform competitively compared to SVM classifiers.

Motivated by previous weight-adjustment efforts for centroid-based classifiers, we design a Class-Feature-Centroid (CFC) classifier, which strives to construct centroids with better initial values than traditional centroids. In our CFC classifier, we first extract inter-class and inner-class term indices from the corpus. Then both indices are carefully combined together to produce prototype vectors. Our experimental results on the skewed Reuters-21578 corpus and the balanced 20-newsgroup corpus demonstrate that our CFC classifier has a consistently better performance than SVM classifiers. In particular, CFC is more effective and robust than SVM when data is sparse. In summary, this paper has made following contributions:

- We propose a novel weight representation for centroid-based classifiers, which incorporates both inter-class term distribution and inner-class term distribution to determine term weights in prototype vectors.
- In the testing phase, we adopt a denormalized cosine measure for similarity calculation. Our experiments demonstrate that this approach is more effective for CFC than normalizing prototype vectors, because the discriminative capability of features is preserved.

The remainder of the paper is organized as follows. Section 2 reviews centroid-based text categorization methods. Section 3 elaborates the design of CFC. We evaluated the performance of our CFC in Section 4. Section 5 summarizes

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

related work. Finally, Section 6 concludes our work with future research directions.

2. CENTROID-BASED TC

In centroid-based TC, a text in a corpus is represented with a Vector Space Model (VSM), where each text is considered as a vector in term space [1]. A prototype vector (i.e., a centroid) is constructed for each category as a delegate vector for all documents belonging to that class. When classifying an unlabeled document, the vector representing the document is compared with all prototype vectors, and the document is assigned to the class whose prototype vector is most similar [15].

2.1 Prototype Representation

In the vector space model, all documents in a class form a lexicon set $\mathcal{F} = \{t_1, t_2, ..., t_{|\mathcal{F}|}\}$, and a document is represented as a vector. Normalization by document length is often performed when calculating a term's weight and scaling the vector to have L2-norm equal one is the most commonly used method. A traditional prototype vector is a delegate vector for each category, where a feature's weight should be a form of weight combination of all documents in the category.

To avoid over-fitting and high-computational complexity, many dimension reduction methods have been proposed for the term vector, such as stop words, stemming [37], word clustering [19], and document frequency [39].

2.2 Centroid Construction

Given a class C_j of a corpus, there are two classical methods to create C_j's prototype vector:

(1) Arithmetical Average Centroid (AAC):

$$\overrightarrow{Centroid_j} = \frac{1}{|C_j|} \sum_{\vec{d} \in C_j} \vec{d}, \quad (1)$$

where the centroid is the arithmetical average of all document vectors of class C_j. This is the most commonly used initialization method for centroid-based classifiers.

(2) Cumuli Geometric Centroid (CGC):

$$\overrightarrow{Centroid_j} = \sum_{\vec{d} \in C_j} \vec{d}, \quad (2)$$

where each term will be given a summation weight [15].

Compared to other TC methods, centroid-based approaches are more serious with the problem of inductive bias or model misfit [18, 36] — classifiers are tuned to the contingent characteristics of the training data rather than the constitutive characteristics of the categories. Centroid-based approaches are more susceptible to model misfit because of its assumption that a document should be assigned to a particular class when the similarity of this document and the class is the largest. In practice, this assumption often doesn't hold (i.e., model misfit).

In fact, many researchers have found that a centroid-based classifier should give more weight to term distributions among the corpus, i.e., inter-class, inner-class and in-collection distributions. Centroids considering characteristics of term distribution have shown improved results [14].

2.3 Classification

After centroids of different categories are determined, an unlabeled document is classified by finding the closest centroid to the document vector. The category of this centroid is then assigned to the test document. When the distance of two vectors is measured as the their dot product, the testing process is to calculate

$$C' = \arg\max_j (\vec{d} \bullet \overrightarrow{Centroid_j}).$$

That is, the test document d will be labeled as class C'. Besides the above method, other distances, such as Pearson Correlation Coefficients [24], Euclidean-based similarity computed by an exponential transformation, and Euclidean-based similarity computed by division [3], have also been employed in the literature.

2.4 Motivation of Our Work

The performance of centroid-based classifiers depends strongly on the quality of prototype vectors. To improve performance, many studies have attempted using feedbacks to adjust term weight in prototype vectors, such as Dragpushing [32], Hypothesis Margin [34], and CentroidW [29]. Combining Homogeneous [14] combined several centroid-based classifiers with different term distribution to improve the accuracy of classification. The performance of these adaptive methods is generally better than the traditional centroid-based methods. In particular, some of them can be comparable to SVM classifiers on micro-F1 and macro-F1 evaluations [32, 34].

The motivation of this work is from the observation that all of the above adaptive methods start with the same initial term weights obtained at the centroid construction phase. During the training phase, term weights are adjusted to get better prototype vectors. Different from these previous approaches, we try to obtain good centroids during the construction phase such that their classification capability is still competitive compared to those derived from adaptive methods. The following sections discuss details of our centroid-based classifier.

3. DESIGN

Different from previous approaches for constructing centroids, CFC puts more emphasis on term distributions in order to improve the quality of centroid. Specifically, CFC first extracts inter-class and inner-class term distributions from the corpus, and then uses these distributions to derive the centroid. In the rest of this section, we first give an overall description of CFC design, and then discuss our feature extraction method, followed by a discussion of our classification method.

3.1 Overview

Similar to other centroid-based approaches, CFC adopts the vector space model: all documents from a corpus form a lexicon set $\mathcal{F} = \{t_1, t_2, ..., t_{|\mathcal{F}|}\}$; and the centroid for class C_j is represented by a term vector $Centroid_j = (w_{1j}, w_{2j}, ..., w_{|\mathcal{F}|j})$, where w_{kj} $(1 \le k \le |\mathcal{F}|)$ represents the weight for term t_k.

The main difference between CFC and other approach is how term weight is derived. In CFC, weight for term t_k of

class j is calculated as:

$$w_{ij} = b^{\frac{DF_{t_i}^j}{|C_j|}} \times \log(\frac{|C|}{CF_{t_i}}), \quad (3)$$

where $DF_{t_i}^j$ is term t_i's document frequency in class C_j, $|C_j|$ is the number of documents in class C_j, $|C|$ is the total number of document classes, CF_{t_i} is the number of classes containing term t_i, and b is a constant larger than one.

In the above formula, the first component $b^{\frac{DF_{t_i}^j}{|C_j|}}$ is the inner-class term index, and the second component $\log(\frac{|C|}{CF_{t_i}})$ represents the inter-class term index. Both indices are discussed below.

3.2 Inner-Class Term Index

The inner-class distribution of a term can help classification. For instance, if a term appears many times in documents of category $\mathcal{C}$, and then a test document containing the term is more likely to be of category $\mathcal{C}$.

After comparing different forms for inner-class distribution, we finally choose $b^{\frac{DF_{t_i}^j}{|C_j|}}, (b > 1)$ to be CFC's the inner-class term index. The advantage of this form is to limit the inner-class feature weight within range $(1, b]$. The denominator $|C_j|$ in the formula smoothes the difference of document frequencies across categories.

The proposed inner-class term index can be easily computed by counting the occurrence of terms while traversing the corpus, which only incurs linear-time cost.

3.3 Inter-Class Term Index

Previous work [21] has studied many mathematical forms for scoring a term's distribution information. Intuitively, a good inter-class feature or term should distribute rather differently among classes. In other words, if a term only appears in a few categories, and then the term is a discriminative feature, thus a good feature for classification. Conversely, if a term appears in every category, and then the term is not a good inter-class feature.

Our method for extracting inter-class term index, i.e., $\log(\frac{|C|}{CF_{t_i}})$, produces more discriminative features. Note that such a form is similar to IDF, but the difference is that CFC is counting the number of categories containing the term. When a term occurs in every category, the value becomes 0 (because $|C| = CF_{t_i}$). When the term only occurs in one category, the value becomes $\log(|C|)$. In other cases, the value falls between them. We can observe that such a method favors rare terms and bias against popular terms.

The inter-class term index can be efficiently computed in linear time. By counting lexicons while traversing the corpus once, all inter-class term indices can be easily computed.

3.4 Testing Phase

After the prototype vector of each class is obtained, CFC classifies a document with a denormalized cosine measure, i.e.,

$$C' = \arg\max_j(\overrightarrow{d_i} \bullet \overrightarrow{Centroid_j}), \quad (4)$$

where $\overrightarrow{d_i}$ is the document vector for document i. The term weight for each feature in $\overrightarrow{d_i}$ uses the standard normalized TF-IDF score. As Figure 1 showing, $\cos\alpha$ is the standard similarity measure between two vectors, while we adopt $\cos\alpha \times ||\overrightarrow{Centroid_j}||_2$ as the similarity measure between a document vector and a prototype vector. Because the prototype vector is not normalized, this similarity is called denormalized cosine measure in this paper.

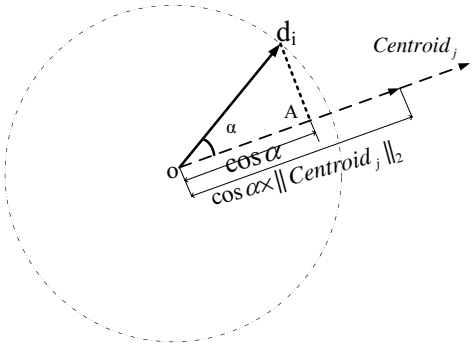

Figure 1: An illustration of denormalized cosine measure.

This denormalized cosine measure preserves the discriminative capability of prototype vectors and improves the accuracy of classification (see Section 4.3.1).

3.4.1 A Classification Example

We use an example to illustrate how classification is performed in CFC and to demonstrate the difference between CFC and AAC. Assume a corpus contains four categories $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$. Three feature terms appear in 14 documents, and each document only contains one term. Table 1 gives the term distribution of this corpus.

Table 1: Term distribution for an example corpus.

Feature	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$
$term_1$	0	0	0	2
$term_2$	2	1	1	1
$term_3$	0	0	0	7

Then, Table 2 shows prototype vectors for each category. For CFC, b is set $e - 1$, and log uses function ln.

Table 2: Prototype vectors for the example corpus.

Classifier	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$
AAC	{0,1,0}	{0,1,0}	{0,1,0}	{0.2721,0.1361,0.9526}
CFC	{0,0,0}	{0,0,0}	{0,0,0}	{1.5445,0,2.0250}

Now, assume $\{0.6, 0.8, 0\}$ is the document vector for a document $\vec{d}$. We then calculate the similarity between $\vec{d}$ and all four categories and obtain $(0.8000, 0.8000, 0.8000, 0.2721)$ for AAC, and $(0, 0, 0, 0.9267)$ for CFC.

AAC would assign a class label $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ to $\vec{d}$, since the similarity for these three classes is 0.8000, the highest value. For CFC, label $\mathcal{C}_4$ is assigned to $\vec{d}$. For this pedagogic example, we can observe that:

- AAC should favor popular words in a corpus, which

is exemplified by the score of (0.8000, 0.8000, 0.8000, 0.2721) for $\vec{d}$.

- CFC is more favorable to rare terms and biases against popular terms. In the example, CFC gives $\vec{d}$ a score of $(0, 0, 0, 0.9267)$, because $term_1$ only appears in category $\mathcal{C}_4$ while $term_2$ occurs in all of the categories.

This distinct trait of CFC contributes to the quite different classification performance between AAC and CFC.

3.4.2 Discussion of Denormalized Measure

Enlarging $\cos\alpha$ by a factor of prototype vector's magnitude may seem to be unfair to some categories. A prototype vector with large norms could increase false positives, because documents from other categories may receive higher similarity scores. On the other hand, prototype vectors with small norms could have more false negatives. In fact, such a phenomenon can happen for AAC or CGC (see Section 4.3.1).

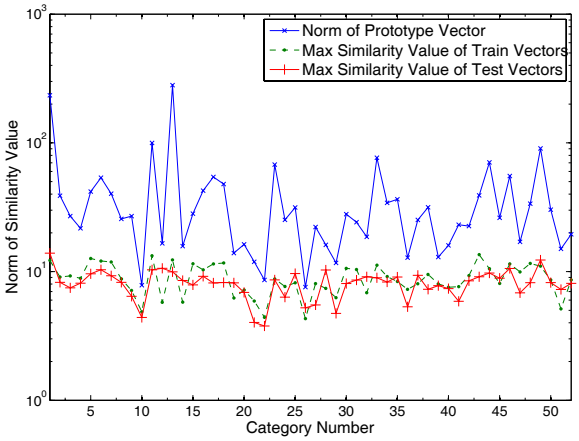

Figure 2: Norms of prototype vectors, similarity measure between documents (training and testing) and prototype vectors of CFC for Reuters-21578 corpus.

CFC, however, doesn't suffer from the above problem of incorrect classifications. We have found CFC's similarity measures do not change significantly across different categories. Figure 2 illustrates the comparison of the norms of prototype vectors, and similarity measure between documents (divided into training set and testing set) and prototype vectors for Reuters-21578 corpus. Even though the variations for the norms of prototype vectors can be quite significant (the average is 40.3 and standard deviation is 2363.0) the maximums of similarity measure for training documents and testing documents do not change significantly. The average and standard deviation are 8.96 and 5.69 for training set, and 8.17 and 3.88 for testing set. That is, similarity measures are roughly on the same scale for all categories.

4. EVALUATION

In this section, we evaluate our CFC classifier on the TC task by comparing CFC's performance with previous centroid-based approaches and the state-of-the-art SVM methods. Specifically, we compare the performance of four different approaches:

- AAC: centroid-based approach with arithmetical average centroids;
- CGC: centroid-based approach with cumuli geometric centroids;
- CFC: our class-feature centroids;
- SVM: SVM-based tools. We adopted SVMLight[1], SVMTorch[2], and LibSVM[3] in this paper.

We will use both skewed and balanced corpus for performance evaluation. For the skewed corpus experiments, we use the Reuters-21578 dataset. For balanced corpus, 20-newsgroup dataset is used. In this paper, we focus on classifying multi-class, single-label TC task and remove multi-label texts from the Reuters corpus.

The performance metrics used in the experiments are F1, micro-averaging F1, and macro-averaging F1. F1 is a combined form for precision (p) and recall (r), which is defined as

$$F1 = \frac{2rp}{r+p}.$$

We used F1 to evaluate the classification performance for individual category. The macro-averaging F1 (macro-F1) and micro-averaging F1 (micro-F1) were used to measure the average performance for the whole corpus. Macro-F1 gives the same weight to all categories, thus is mainly influenced by the performance of rare categories for the Reuters-21578 corpus, due to skewed category distribution. On the contrary, micro-F1 will be dominated by the performance of common categories for the Reuters-21578 corpus. Because the 20-newsgroup corpus is a much balanced corpus, its macro-F1 and micro-F1 are quite similar.

In the rest of this section, we use "μ-F1" and "M-F1" to represent micro-F1 and macro-F1, respectively.

4.1 Datasets and Experimental Settings

Reuters-21578. The Reuters-21578 dataset[4] are based on the Trinity College Dublin version. Trinity College Dublin changed the original SGML text documents into XML format text documents. After removing all unlabeled documents and documents with more than one class labels, we then retained only the categories that had at least one document in both the training and testing sets and got a collection of 52 categories.

Altogether, there are 6,495 training texts and 2,557 testing texts left in this 52-category corpus. The distribution of documents over the 52 categories is highly unbalanced. After removing 338 stop words (also provided by Trinity College dataset), and unigram terms that occur less than three times, we get 11,430 unique unigram terms. Words in titles are given ten times weight comparing to those in abstracts.

[1]http://svmlight.joachims.org
[2]http://www.idiap.ch/~bengio/projects/SVMTorch.html
[3]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools
[4]http://ronaldo.cs.tcd.ie/esslli07/sw/step01.tgz

20-newsgroup. This dataset[5] consists of 19,997 texts (about one thousand text documents per category), and approximately 4% of the articles are cross-posted. The stop words list [20] has 823 words, and we kept words that occurred at least once and texts that had at least one term.

Altogether, there are 19,899 texts (13,272 training and 6,627 testing) left in the corpus. When parsing documents, we only keep "Subject", "Keywords", and "Content". Other information, such as "Path", "From", "Message-ID", "Sender", "Organization", "References", "Date", "Lines", and email addresses, are filtered out. The total number of unigram terms is 29,557 unigrams. Words in "Subject" and "Keywords" are given ten times weight comparing to those in "Contents".

For both corpora, we used the tokenizer tool provided in the Trinity College sample. IDF scores for TF-IDF are extracted from the whole corpus. Stemming and word clustering were not applied.

Parameter Settings. For all SVM tools (SVMLight, SVMTorch, and LibSVM), the linear kernel and the default settings were used. All of SVM-based classifiers can cope with a sparse multi-class TC task directly with one-*vs*-others decomposition and default parameter values. For the Reuters corpus, χ^2-test method [39] was adopted to perform feature selection (top 9,000) for SVMLight (this setting yields the best result in our experiments), while SVMTorch classifier used all 11,430 features. For 20-newsgroup, we tuned parameter b for SVMLight for better performance.

In the experiments, the parameter b of CFC is set to $e - 1.7$, unless specified otherwise.

4.2 Overall Performance

We first compare the overall performance of different approaches. The results for both the Reuters and the 20-newsgroup are shown in Table 3.

Table 3 shows that CFC performs the best among all classifiers for the Reuters Corpus. Both micro-F1 and macro-F1 values are above 0.99, which is significantly better than SVM-based classifiers. Two classical centroid-based classifiers, AAC and CGC, perform the worst using normalized cosines and happen to have the same results.

For the 20-newsgroup corpus, Table 3 shows that CFC performs the best among all classifiers. CFC's micro-F1 and macro-F1 are 0.9272 and 0.9275, respectively, which are significantly better than all SVM classifiers. Again, centroid-based AAC and CGC perform the worst.

Table 3: Overall performance comparison of different classifiers.

Classifier	Reuters		20-newsgroup	
	μ-F1	M-F1	μ-F1	M-F1
CFC	**0.9941**	**0.9960**	**0.9272**	**0.9275**
SVMLight	0.9210	0.8335	0.8304	0.8297
SVMTorch*	0.9163	0.7875	0.8482	0.8479
LibSVM	0.8999	0.7190	0.8416	0.8425
AAC	0.8647	0.7344	0.8307	0.8292
CGC	0.8647	0.7344	0.8307	0.8292

*Previous work [33] has reported micro-F1 and macro-F1 for 20-newsgroup corpus could reach 0.8891 and 0.8876, respectively.

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

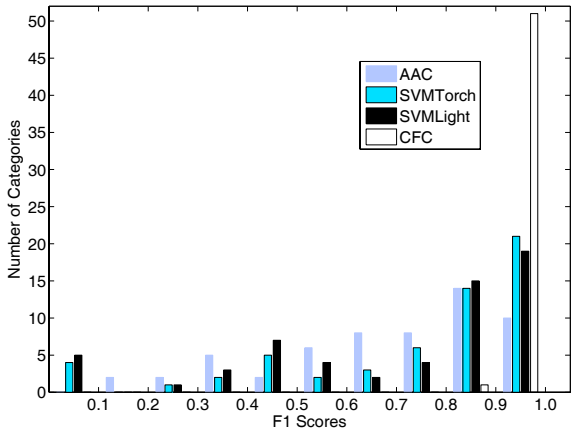

Figure 3: F1 comparison of AAC, SVMTorch, SVMLight, and CFC for the Reuters corpus.

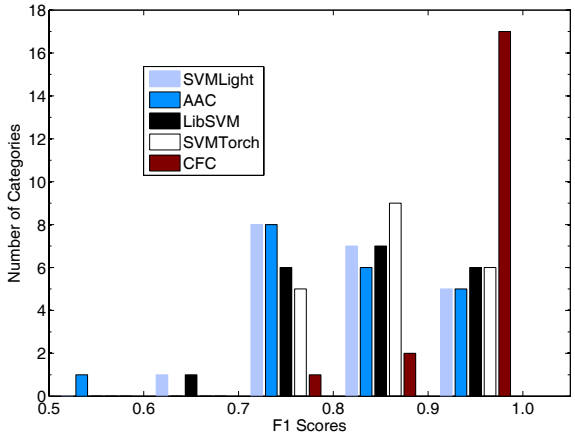

Figure 4: F1 comparison of AAC, SVMTorch, SVMLight, LibSVM, and CFC for the 20-newsgroup corpus.

To give a more visualized comparison, Figure 3 and Figure 4 give the distributions of F1 scores for the Reuters corpus and the 20-newsgroup corpus, respectively. From both figures, we can observe that CFC has much more categories within the range of $[0.9, 1.0]$. In fact, CFC consistently performs better than other approaches for most categories. In Figure 3, both SVMTorch and SVMLight have a few categories with zero F1 score. This is mainly for categories with sparse data, which is discussed below.

4.2.1 *Study on Sparse Data*

In this study, we selected the bottom 10 classes from Reuters corpus, where the number of training documents varies from one to five. Previous work [7] has found that most of conventional learning methods, such as SVM and kNN, have poor performance for sparse training cases, resulting in low macro-F1 scores, because macro-F1 treats every category equally.

Table 4 illustrates F1 values for different classifiers. Our

CFC performs the best with value one for all classes. I.e., all test documents are correctly classified. This is mainly because CFC can effectively exploit the discriminative capability of prototype vectors. This experiment can help to explain CFC's excellent macro-F1 values in the overall performance comparison.

For SVMLight or SVMTorch, more than half of classes have F1 values less than one, and both obtained zero for category "jet" and "dlr". The result could be caused by the lack of training data.

For AAC, all F1 values are well below one because AAC's prototype vectors don't have the same discriminative capability as CFC.

Table 4: Comparison of F1 metric for bottom 10 classes of Reuters corpus.

Class	Train	Test	AAC	CFC	Light	Torch
platinum	1	2	0.5714	**1**	0	**1**
jet	2	1	0.1538	**1**	0	0
potato	2	3	0.7500	**1**	0.8000	0.8000
tea	2	3	0.3636	**1**	0.5000	0.8000
cpu	3	1	0.3333	**1**	**1**	**1**
dlr	3	3	0.3529	**1**	0	0
nickel	3	1	0.3333	**1**	**1**	**1**
fuel	4	7	0.6364	**1**	0.4444	0.4444
lead	4	4	0.8000	**1**	0.9000	**1**
instal-debt	5	1	0.1667	**1**	**1**	**1**

4.2.2 Study on Sufficient Data

Reuters. To give an elaborated exhibition for categories with sufficient training data, F1 values of the top ten categories from the Reuters corpus were listed in Table 5. For all these ten categories, we observe that CFC consistently has the best F1 scores. In many categories, CFC is significantly better than SVM classifiers.

Table 5: Comparison of F1 metric for top ten classes of the Reuters corpus.

Class	Train	Test	AAC	CFC	Light	Torch
earn	2824	1076	0.9273	**0.9958**	0.9728	0.9647
acq	1594	695	0.9035	**0.9943**	0.9353	0.9266
crude	251	119	0.8860	**1**	0.8871	0.8525
trade	249	75	0.8951	**0.9796**	0.8589	0.8947
mny-fx	204	87	0.7886	**0.9884**	0.7561	0.7758
interest	189	81	0.8025	**0.9877**	0.8662	0.8701
mny-sp	115	28	0.6377	**0.9655**	0.8214	0.8571
ship	107	36	0.8537	**1**	0.6229	0.6349
sugar	97	25	0.8800	**1**	0.9796	0.9796
coffee	90	22	0.9565	**0.9778**	**0.9778**	**0.9778**

20-newsgroup. Table 6 gives a closer look at CFC's performance in a balanced and sufficient dataset — 20-newsgroup corpus. Similar to results from the Reuters corpus, we observe that CFC also consistently outperforms SVMLight, SVMTorch, and LibSVM classifiers. For most categories, the difference is significant.

In summary, when the training data is sufficient, our CFC classifier can consistently perform better than SVM classifiers for both corpus tested.

4.2.3 Discussion

We have attempted to improve SVMLight's performance for unbalanced training set by tuning parameter b, and such attempts can slightly improve F1 scores. For 20-newsgroup, when we tune parameter b (default value is 1.0), micro-F1 increases from 0.8124 to 0.8304, and macro-F1 rises from 0.8121 to 0.8297. However, tuning parameter b does not work well for the Reuters corpus.

SVM-based classifiers are proved powerful when they are working in a suitable environment. In our experiments, there are several factors that may contribute to the inferior performance of SVM. First, the training samples for small categories are seriously skewed when one-*vs*-others policy is adopted. Second, no feature extraction methods, such as latent semantic indexing (LSI) [11, 7] or linear discriminant analysis (LDA) [11], are performed. Third, we use a quite simple tokenizer with no stemming or word clustering. Features are kept if they appear more times than a threshold, i.e., based on term frequency. All of above factors could limit SVM to find the perfect hyperplane, thus lowering classification performance.

On the other hand, the better performance of CFC could hint that CFC has a discriminative ability on those "raw" terms (without feature extraction and selection) for text classification. Previous work [25] has found that document frequency is a powerful discriminative tool. In CFC, such discriminative capability is preserved.

Table 6: Comparison of F1 for all classes of 20-newsgroup corpus.

#	SVMLight	AAC	LibSVM	SVMTorch	CFC
1	0.7446	0.7616	0.7847	0.7814	**0.8359**
2	0.7829	0.7752	0.8055	0.8018	**0.9383**
3	0.7773	0.7890	0.7724	0.8049	**0.9293**
4	0.7100	0.7519	0.7489	0.7575	**0.9192**
5	0.7838	0.8050	0.8438	0.8397	**0.9469**
6	0.8000	0.8037	0.7297	0.8036	**0.9393**
7	0.7557	0.7907	0.7412	0.7585	**0.9257**
8	0.8871	0.8354	0.8839	0.8771	**0.9684**
9	0.8910	0.8750	0.8986	0.9164	**0.9572**
10	0.9302	0.9263	0.9325	0.9323	**0.9802**
11	0.9381	0.9170	0.9457	0.9399	**0.9894**
12	0.9385	0.8693	0.9475	0.9520	**0.9587**
13	0.8142	0.7829	0.8318	0.8158	**0.9450**
14	0.8806	0.9008	0.9198	0.8791	**0.9866**
15	0.9216	0.9132	0.9354	0.9453	**0.9790**
16	0.8854	0.7972	0.8886	0.8905	**0.9371**
17	0.8615	0.8038	0.8719	0.8694	**0.9102**
18	0.9096	0.9012	0.9251	0.9244	**0.9452**
19	0.7471	0.7370	0.7732	0.7687	**0.8168**
20	0.6111	0.5759	0.6696	0.7003	**0.7299**

Categories are: 1:alt.atheism, 2:comp.graphics, 3:comp.os.ms-windows.misc, 4:comp.sys.ibm.pc.hardware, 5:comp.sys.mac.hardware, 6:comp.windows.x, 7:misc.forsale, 8:rec.autos, 9:rec.motorcycles, 10:rec.sport.baseball, 11:rec.sport.hockey, 12:sci.crypt, 13:sci.electronics, 14:sci.med, 15:sci.space, 16:soc.religion.christian, 17:talk.politics.guns, 18:talk.politics.mideast, 19:talk.politics.misc, and 20:talk.religion.misc.

4.3 Study on Denormalized Cosine Measure

4.3.1 Normalize vs. Denormalize

CFC.

Our CFC classifier adopts a denormalized prototype vector, i.e., $\cos\alpha \times ||\overrightarrow{Centroid_j}||_2$. To study the effects of this design, we compare the performance of denormalized and normalized prototype vectors for CFC and other centroid-based classifiers. The results are illustrated in Table 7.

For CFC, Table 7 shows that both micro-F1 and macro-F1 have significant changes when switching from a normalized form to the denormalized one. In fact, when prototype vectors are normalized, CFC's performance is even much lower than AAC and CGC. This is because normalization smoothes the prototype vectors, thus damages their discriminative capability of enlarging selected term features in a text vector.

Table 7: Comparison of normalized and denormalized prototype vectors for the Reuters corpus.

Classifier	μ-F1	M-F1
normalized CFC	0.6058	0.5769
CFC	0.9941	0.9960
AAC	0.8647	0.7344
denormalized AAC	0.5604	0.5454
CGC	0.8647	0.7344
denormalized CGC	0.8510	0.4408

AAC and CGC.

For AAC and CGC, Table 7 shows that denormalization can cause significant performance degradations. For AAC, micro-F1 drops from 0.8647 to 0.5604 and macro-F1 fall from 0.7344 to 0.5454. For CGC, macro-F1 drops from 0.7344 to 0.4408. Here we select the CGC classifier for an elaborated illustration.

Table 8: Comparison of denormalized CGC and CGC for the top five categories of the Reuters corpus.

Class	Train	Test	Pre	DPre	F1	DF1
earn	2824	1076	0.9989	0.8227	0.9273	0.8993
acq	1594	695	0.9750	0.8971	0.9035	0.8939
crude	251	119	0.9266	0.8110	0.8860	0.8374
trade	249	75	0.9412	0.8023	0.8951	0.8571
mny-fx	204	87	0.7841	0.8800	0.7886	0.8148

Note: "Pre" means precision, "DPre" means precision for denormalized CGC, and "DF1" means F1 evaluation for denormalized CGC.

Table 8 compares the original CGC approach and denormalized CGC for the top five categories. We can observe that after denormalization, the precision for top four categories significantly declines. The reason is that denormalization results in larger norms for prototype vectors, as top categories have more documents. When calculating similarity, a big norm increases false positives for those top categories, thus lower precision. The top five categories represent nearly 80% of the whole corpus, so a larger number of false positives in these categories also correspond to the performance deterioration of small categories.

On the other hand, Table 9 compares original CGC and CGC using denormalized prototype vectors for the bottom ten categories. The most prominent change after denormalization is that recall for many categories have dropped to zero. The reason is that the norms of these rare categories are small, causing lower values of similarity measure for documents within these categories and increasing the number of false negatives.

Though the F1 value of the top five categories are not significantly declining, those rare categories have significant drop on F1 evaluation, which results in significant degradation of macro-F1.

Table 9: Comparison of denormalized CGC and CGC for bottom ten categories of the Reuters corpus.

Class	Train	Test	Recall	DRecall	F1	DF1
platinum	1	2	1	0	0.5714	0
jet	2	1	1	0	0.1538	0
potato	2	3	1	0	0.7500	0
tea	2	3	1	0.6667	0.5455	0.8000
cpu	3	1	1	0	0.3333	0
dlr	3	3	1	0	0.3529	0
nickel	3	1	1	0	0.3333	0
fuel	4	7	1	0.1429	0.6364	0.2500
lead	4	4	1	0	0.8	0
instal-debt	5	1	1	0	0.1667	0

Note: "DRecall" means recall evaluation for denormalized CGC.

In summary, a denormalized prototype vector can help

in 20-newsgroup. For instance, the length of texts in class "alt.atheism" varies from 1 KB to 51 KB, and this variation of text length is quite common among all categories. As a result, the number of selected features in text vectors has much greater variations: 1 to 678 for 20-newsgroup and 1 to 44 for Reuters. Considering the unique terms (or features) in 20-newsgroup and Reuters are 29,577 and 11,430, respectively, texts in 20-newsgroup corpus may select up to 2.29% features, while texts in Reuters may only use 0.38% features. As a result, the similarity measure of 20-newsgroup is more varied.

4.4 Study on Feature Weight

We carried out a series of experiments to study the feature weight for CFC. Table 10 illustrates the results of some representative combinations of inner-class and inter-class indices using the Reuters corpus.

The first construction, $\ln(\frac{|C|}{CF_t})$, yields above 0.89 values for both micro-F1 and macro-F1 metrics, indicating that the inter-class term distribution works well for classification. The second and third formulas show that a simple combination of inner-class and inter-class distributions may actually hurt the performance.

The fourth formula only uses inner-class index and yields values above 0.99, indicating the inner-class index is highly discriminative in the classification. Combined with inter-class term index, the fifth formula shows slight improvement. The sixth formula shows that changing parameter b can further improve the results. We will discuss parameter b in the next subsection.

Table 10: Performance study on different forms of feature weight.

No.	Formula	μ-F1	M-F1
1	$\ln(\frac{\|C\|}{CF_t})$	0.8909	0.8982
2	$\frac{DF_t^j}{\|C_j\|)} \times \ln(\frac{\|C\|}{CF_t})$	0.6676	0.4996
3	$\ln(1+DF_t^j) \times \ln(\frac{\|C\|}{CF_t})$	0.9711	0.8520
4	$\exp(\frac{DF_t^j}{\|C_j\|})$	0.9930	0.9938
5	$\exp(\frac{DF_t^j}{\|C_j\|}) \times \ln(\frac{\|C\|}{CF_t})$	0.9941	0.9951
6	$(e-1)^{\frac{DF_t^j}{\|C_j\|}} \times \ln(\frac{\|C\|}{CF_t})$	0.9941	0.9960

Note: $|C|$ is the total number of categories in a corpus. $|C_j|$ is the number of documents within jth category. DF_t^j is term t's document frequency within category C_j. CF_t is number of categories containing term t.

4.5 Study on Parameter b

In this section, we first arrange the inner-class term index solely in the prototype vectors to exhibit the effect of the inner-class term index for the Reuters corpus. Then, we study the effects of CFC's parameter b.

4.5.1 Sole Inner-Class Term Index

In this experiment, we only use inner-class index as feature weight (i.e., $b^{\frac{DF_t^j}{|C_j|}}$) in a prototype vector. Figure 6 shows that when b is between one and e, CFC's performance is generally good. When b is $e-1.8(\approx 0.918)$, the performance is significantly lower than $e-1.7$ (≈ 1.018).

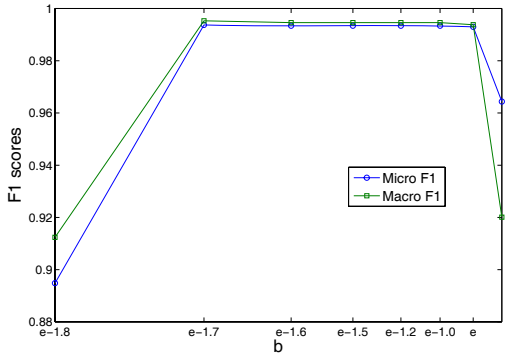

Figure 6: Sensitivity study parameter b of formula $b^{\frac{DF_t}{|C_j|}}$ for the Reuters corpus.

4.5.2 Study on CFC's Parameter b

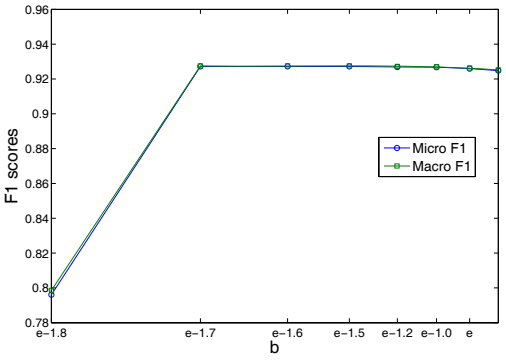

Figure 7: Sensitivity study of CFC's parameter b for the 20-newsgroup corpus.

Previous experiment hints that parameter b performs best when slightly larger than one. In this experiment, we study CFC's performance with varying values of b. Figure 7 and Figure 8 shows the results for the 20-newsgroup corpus and the Reuters corpus, respectively. We can observe values larger than one are significantly better than $e-1.8$. When larger than one, increasing b lowers performance. So, $e-1.7$ works well in practice and is used as the default value for parameter b in our experiments.

5. RELATED WORK

A number of automated TC approaches train their models from both positive and negative examples, such as Decision Tree (DT) [17, 35, 7], Rocchio [26], and SVM [12, 5]. DT employs a recursive growing and pruning branch method until every leaf node has only one class label. The Rocchio method builds a model that rewards a document for its closeness to positive examples and its distance from negative examples. SVM selects support vectors in both negative and positive examples to find a hyperplane that optimizes decision surface. Different from these approaches, CFC does

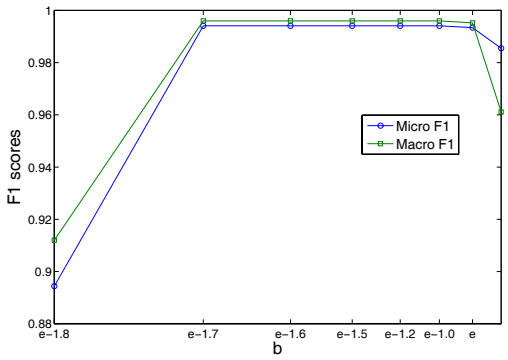

Figure 8: Sensitivity Study of CFC's parameter b for the Reuters corpus.

not use the negative and positive examples directly in its model construction. Instead, the distributions of terms are used to derive CFC's model, i.e., the centroid. As a result, constructing centroids in CFC only incurs linear-time cost.

Previous work extracts many different features from documents, such as TF, IDF, information gain [13], mutual information [27, 23], Chi-square [39, 28], and odds ratio [30, 28]. Usually, feature selection metrics are functions of four dependency tuples [30]:

1. (t, c_i): presence of term t and membership in category c_i;
2. $(t, \overline{c_i})$: presence of term t and non-membership in category c_i;
3. $(\bar{t}, c_i)$: absence of t and membership in category c_i;
4. $(\bar{t}, \overline{c_i})$: absence of t and non-membership in category c_i.

CFC counts the occurrence of a term in a category, i.e., only the first tuple, while some metrics (e.g., Chi-square and odds ratio) also use other tuples. CFC's inter-class term index is most similar to IDF [25, 9], but differ in the fact that CFC counts the number of categories containing specific terms. Unlike TF-IDF, CFC proposes a novel combination of inter-class and inner-class indices for computing feature weights.

Many web applications [2, 38, 22] have used TC for various purposes. Xue et al. [38] discussed the problem of classifying documents into a large-scale hierarchy, which first acquires a candidate category set for a given document, and then performs classification on a small subset of the original hierarchy. Search engines can use TC to improve their query classification results [2]. Our CFC classifier is fast on training and classification, and can be easily updated with incremental data. Thus, CFC could be used in these web applications, providing TC support.

6. CONCLUSIONS

We designed the Class-Feature-Centroid (CFC) classifier for text categorization and compared its performance with SVM and centroid-based approaches. Experimental results on Reuters-21578 corpus and 20-newsgroup email collection show that CFC consistently outperforms SVM and centroid-based approaches with both micro-F1 and macro-F1 evaluations on multi-class, single-label TC tasks. Additionally, when data is sparse, CFC has a much better performance and is more robust than SVM.

Our CFC classifier proposes a novel centroid incorporating both inter-class and inner-class term indices. Both can be efficiently computed from a corpus in linear time and can be incrementally updated.

In the testing phase, CFC adopts a denormalized cosine measure, instead of a normalized prototype vector. This is to preserve prototype vectors' discriminative ability and enlarging effect. Experimental results demonstrate that this denormalized approach is more effective for CFC.

We are applying the CFC approach to the multi-class, multi-label TC task, preliminary results have been promising. We will continue this effort in the future. Additionally, we plan to investigate the performance of CFC for large-scale web documents.

Source code for this work is available at `http://epcc.sjtu.edu.cn/~jzhou/research/cfc/`.

Acknowledgment

We would like to thank Tao Yang and the anonymous reviewers for their insightful comments on this paper. This work was supported in part by 863 Program of China (Grant No. 2006AA01Z172, 2006AA01Z199, and 2008AA01Z106), National Natural Science Foundation of China (Grant No. 60533040, 60811130528, and 60773089), and Shanghai Pujiang Program (Grant No. 07pj14049).

7. REFERENCES

[1] M. Benkhalifa, A. Mouradi, and H. Bouyakhf. Integrating external knowledge to supplement training data in semi-supervised learning for text categorization. *Information Retrieval*, 4(2):91–113, 2001.

[2] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 231–238, Amsterdam, The Netherlands, 2007.

[3] Z. Cataltepe and E. Aygun. An improvement of centroid-based classification algorithm for text classification. *IEEE 23rd International Conference on Data Engineering Workshop*, 1-2:952–956, 2007.

[4] R. N. Chau, C. S. Yeh, and K. A. Smith. A neural network model for hierarchical multilingual text categorization. *Advances in Neural Networks, LNCS*, 3497:238–245, 2005.

[5] S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proceedings of the 23rd Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 256–263, Athens, Greece, 2000.

[6] P. Frasconi, G. Soda, and A. Vullo. Text categorization for multi-page documents: a hybrid naive Bayes HMM approach. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 11–20. ACM Press New York, NY, USA, 2001.

[7] S. Gao, W. Wu, C. H. Lee, and T. S. Chua. A maximal figure-of-merit (MFoM)-learning approach to

robust classifier design for text categorization. *ACM Transactions on Information Systems*, 24(2):190–218, 2006.

[8] G. D. Guo, H. Wang, D. Bell, Y. X. Bi, and K. Greer. Using kNN model for automatic text categorization. *Soft Computing*, 10(5):423–430, 2006.

[9] B. How and K. Narayanan. An Empirical Study of Feature Selection for Text Categorization based on Term Weightage. *WI 2004*, pages 599–602, 2004.

[10] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes. Multinomial naive bayes for text categorization revisited. *AI 2004: Advances in Artificial Intelligence*, 3339:488–499, 2004.

[11] H. Kim, P. Howland, and H. Park. Dimension Reduction in Text Classification with Support Vector Machines. *Journal of Machine Learning Research*, 6(1):37–53, 2006.

[12] R. Klinkenberg and T. Joachims. Detecting Concept Drift with Support Vector Machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 487–494, 2000.

[13] C. K. Lee and G. G. Lee. Information gain and divergence-based feature selection for machine learning-based text categorization. *Information Processing and Management*, 42(1):155–165, 2006.

[14] V. Lertnattee and T. Theeramunkong. Combining homogeneous classifiers for centroid-based text classification. *ISCC 2002*, pages 1034–1039, 2002.

[15] V. Lertnattee and T. Theeramunkong. Effect of term distributions on centroid-based text categorization. *Information Sciences*, 158:89–115, 2004.

[16] V. Lertnattee and T. Theeramunkong. Class normalization in centroid-based text categorization. *Information Sciences*, 176(12):1712–1738, 2006.

[17] D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156, 1994.

[18] Y. Liu, Y. Yang, and J. Carbonell. Boosting to correct inductive bias in text classification. In *Proc. of the 11th International Conference on Information and Knowledge Management*, pages 348–355, McLean, VA, 2002.

[19] S. Martin, J. Liermann, and H. Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37, 1998.

[20] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[21] E. Montanes, I. Diaz, J. Ranilla, E. F. Combarro, and J. Fernandez. Scoring and selecting terms for text categorization. *IEEE Intelligent Systems*, 20(3):40–47, 2005.

[22] X. Ni, G. Xue, X. Ling, Y. Yu, and Q. Yang. Exploring in the weblog space by detecting informative and affective articles. In *WWW*, Banff, Canada, 2007.

[23] Z. L. Pei, X. H. Shi, M. Marchese, and Y. C. Liang. An enhanced text categorization method based on improved text frequency approach and mutual information algorithm. *Progress in Natural Science*, 17(12):1494–1500, 2007.

[24] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, NC, 1994.

[25] S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60:503–520, 2004.

[26] R. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, Australia, 1998.

[27] K. M. Schneider. Weighted average pointwise mutual information for feature selection in text categorization. *Knowledge Discovery in Databases: PKDD 2005*, 3721:252–263, 2005.

[28] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[29] S. Shankar and G. Karypis. *Weight Adjustment Schemes for a Centroid Based Classifier*. Army High Performance Computing Research Center, 2000.

[30] P. Soucy and G. W. Mineau. Feature selection strategies for text categorization. *Advances in Artificial Intelligence, Proceedings*, 2671:505–509, 2003.

[31] V. Tam, A. Santoso, and R. Setiono. A comparative study of centroid-based, neighborhood-based and statistical approaches for effective document categorization. *16th International Conference on Pattern Recogniton*, Iv:235–238, 2002.

[32] S. Tan. Large margin DragPushing strategy for centroid text categorization. *Expert Systems with Applications*, 33(1):215–220, 2007.

[33] S. Tan. Using hypothesis margin to boost centroid text classifier. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 398–403, Seoul, Korea, 2007.

[34] S. Tan. An improved centroid classifier for text categorization. *Expert Systems with Applications*, 35(1-2):279–285, 2008.

[35] S. Weiss, C. Apte, F. Damerau, D. Johnson, F. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, pages 63–69, 1999.

[36] H. Wu, T. Phang, B. Liu, and X. Li. A refinement approach to handling model misfit in text categorization. In *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 207–216, Alberta, Canada, 2002.

[37] J. Xu and W. Croft. Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems (TOIS)*, 16(1):61–81, 1998.

[38] G. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st Annual International ACM SIGIR Conference*, pages 627–634, Singapore, 2008.

[39] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. *Machine Learning*, pages 412–420, 1997.