中文文件自動化分類方法之比較
Comparing Classifiers for Automatic Chinese Text Categorization

蔡志忠
Jyh-Jong Tsay
Dept. of Comp. Sci.
Natl. Chung Cheng Univ.
tsay@cs.ccu.edu.tw

王經篤
Jing-Doo Wang
Dept. of Comp. Sci.
Natl. Chung Cheng Univ.
jdwang@cs.ccu.edu.tw

Abstract

In this paper, we make an extensive comparison of three classifiers, naive Bayes(NB) probabilistic classifier, Rocchio linear classifier and k-Nearest Neighbor(kNN) classifier for Chinese text classification. Our goal is to compare their performance when they are integrated with term selection, term clustering and instance selection methods. Our experiment use one year CNA news articles to extract meaningful terms, one month news articles as training data, and 3-day news articles as testing data. When the dimension of term space is high, about $90,000$, the Rocchio linear classifier achieves the best average accuracy, 79.35%. The observation is different from previous research that Rocchio have relatively poor performance. When the dimension is reduced to $3,600$ by a combination of term selection and term clustering, kNN achieves the best average accuracy, 80.24%. We further use Generalized Instance Set(GIS) algorithm[13] to reduce the size of training data and hence speed up on-line classification of kNN. Experiment shows that the application of GIS can reduce the number of training data from $6,254$ to $1,195$, while improving the accuracy of kNN from 80.24% to 81.12%. The last accuracy achieved by previous related research is about 78%.

Keywords: Text Categorization, Term Selection, Term Clustering, naive Bayes, Rocchio, k-Nearest Neighbor.

1 Introduction

Text classification is the problem of automatically assigning predefined categories to free text documents, and is gaining more and more importance as the amount of text data available on World Wide Web grows dramatically. There are many studies [5, 22, 14, 15, 27, 8, 23, 24, 2, 13] of text classification for English texts but there are only a few studies for Chinese text classification until recently[3, 26, 9, 20, 19].

In this paper, we make an extensive comparison of three classifiers, naive Bayes(NB) probabilistic classifier, Rocchio linear classifier and k-Nearest Neighbor(kNN) classifier for Chinese text classification. Our goal is to compare their performance when they are integrated with term selection, term clustering and instance selection methods. Our experiment uses one year CNA news which consists of $73,420$ articles to extract meaningful terms, one month news which consists of $6,254$ articles as training data, and 3-day news which consists of 339 articles as testing data. Our experiment first build a SB-tree to extract significant terms, then use χ^2 statistic method to select a set of most representative terms, and finally use distributional clustering to cluster the selected terms into groups. Note that term selection as well as term clustering has been proposed to reduce the huge number of possible n-grams in Chinese to an practical level so that automatic classification is computationally practical. The combination of χ^2 statistic method and distributional clustering has been shown to work well in previous research[20, 19].

Let MicroAccuracy be the total average of classification accuracy and MacroAccuracy be the average classification accuracy of classes. When the dimension of term space is high, about $90,000$, the Rocchio linear classifier achieves the best MicroAccuracy, 79.35% as well as MacroAccuracy, 79.04%. The observation is different from previous research[23] that Rocchio have relatively poor performance. When the dimension of term space is less than or equal to $60,000$, kNN achieves the best MicroAccuracy, but the corresponding MacroAccuracy is relatively low. It implies that kNN prefers the large classes than the small ones. When the dimension is reduced from $90,000$ to $3,600$ by a combination of term selection and term clustering, kNN achieves the best MicroAccuracy, 80.24%, and the MacroAccuracy are also improved from 68.94% to 73.73%. We further use Generalized Instance Set(GIS) algorithm[13] to reduce the size of training data and hence speed up on-line classification of kNN. Experiment shows that the application of GIS can reduce the number of training data from 6,254 to 1,195, while improving the MicroAccuracy of kNN from 80.24% to 81.12%. Overall naive Bayes has the poorest performance.

The remainder of this paper is organized as follows. Section 2 describes the process to extract significant patterns. Section 3 reviews term selection method, χ^2 statistic. Section 4 reviews term clustering algorithm, distributional clustering. Section 5 introduces three classifiers, Rocchio algorithm, naive Bayes(NB) classifier, k-Nearest Neighbor(kNN) classifier. Section 6 reviews the Generalized Instance Set(GIS) algorithm. Section 7 gives our experimental results. Section 8 gives conclusion.

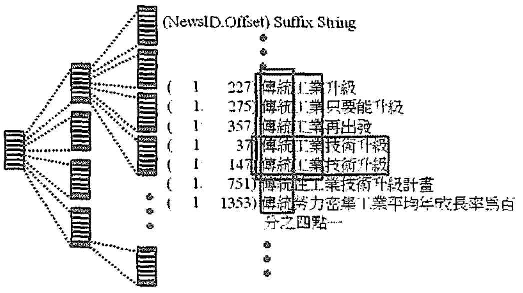
(NewsID.Offset) Suffix String
(1 227) 傳統工業升級
(1. 275) 傳統工業只要能升級
(1 357) 傳統工業再出發
(1 37) 傳統工業技術升級
(1 147) 傳統工業技術升級
(1. 751) 傳統性工業技術升級計畫
(1 1353) 傳統勞力密集工業平均年成長率為百分之四點一

Figure 1: SB-tree

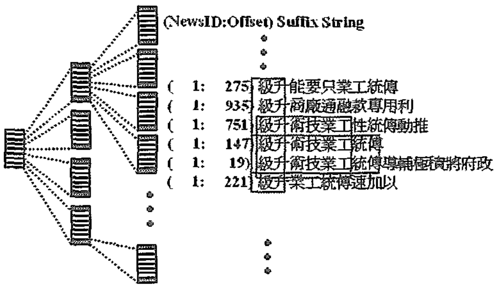
(NewsID:Offset) Suffix String
(1: 275) 級升能要只業工統傳
(1: 935) 級升商廠通融款專用利
(1: 751) 級升術技業工性統傳動推
(1: 147) 級升術技業工統傳
(1: 19) 級升術技業工統傳導輔極積將府政
(1: 221) 級升業工統傳速加以

Figure 2: Reverse SB-tree

2 Term Extraction

There are several research[21, 4, 16] on the extraction of meaningful terms from Chinese texts. In [21] Tseng proposed *a multi-linear term-phrasing* technique in which adjacent character sequences are merged pairwisely to form longer character sequences if they satisfy the criteria of the merging rules. This approach is simple but can not run incrementally when new news are added. In [4] Chien proposed *PAT-tree* method to extract keyword. PAT-tree is an incremental method but does not handle the I/O problem when the amount of memory is not large enough to store the whole tree. In this paper, we propose an approach based on SB-trees [6] which use B^+tree to store all the suffix strings[7] of the training documents. Note that SB-tree can grow incrementally, is I/O efficient and is scalable to store large amount of data.

We construct two SB-trees to locate the left and right boundary of terms respectively, and compute the statistics information of extracted term by scanning the leaves of SB-tree. We use *SB-trees* [6, 20] to store all suffix strings [7] of every sentences in the training corpus, and then search for all the repeated strings which appear more than once. To eliminate redundant strings, we gather only the repeated patterns that have, at least, two different kinds of successor Chinese characters. For example, in Figure 1, there are partial sorted suffix strings listed in the SB-tree. The " 傳統", " 傳統工業" and " 傳統工業技術升級"are considered as candidate patterns. Notice that the " 傳統工業技", " 傳統工業技術" and " 傳統工業技術升" are not considered as candidate patterns because they have only one successor Chinese character " 術"," 升" and " 級" respectively. This process determines the right boundary of terms.

To determine the left boundary of terms, we construct another SB-tree, called *Reverse-SB-tree*, with all suffix strings that come from each reversed sentences in the training corpus. For example, in Figure 2, there are candidate repeated patterns " 級升", " 級升術技業工" and " 級升術技業工統傳". Similarly, the " 級升術", " 級升術技" and " 級升術技業" are not considered as candidate patterns because they have only one successor Chinese character " 技"," 業" and " 工" respectively. This process determines the left boundary of terms. Terms identified in above process form an initial set of terms which are used for term selection.

3 Term Selection

After extracting terms from the training corpus as described in section 2, we apply term selection algorithms to select the most representative terms for each class. All terms are given scores by the term selection method, and are chosen according to the scores. In this paper, we use χ^2 statistic[25, 20, 19] as term selection method. The following reviews χ^2 statistic method.

For a term t and a class c, let A denote the number of times t and c co-occur, B is the number of times t occurs without c, C is the number of times c occurs without t, and N is the total number of documents. The χ^2 statistic measures the lack of independence between t and c, and can be compared to the χ^2 distribution with one degree of freedom to judge extremeness. The χ^2 statistic measure is defined in [12] as follows.

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

4 Distributional Clustering

One of the practical problems[2, 20, 19] in term selection is that a document may contain very few or even non of the selected terms(n-grams) if only a small number of significant terms are selected. However, a large number of selected terms will make automatic classification computationally impractical. To overcome the problems, we combine term(feature) selection with term clustering. In this paper we use distributional clustering [2] to cluster correlated terms. We modify the selection of initial seeds to have equal number of selected terms from each class to avoid a biased estimate of term probability distribution. The following reviews the distributional clustering used in this paper.

Consider, for example, the random variable over classes, C, and its distribution given a particular term, t_i. Let this distribution be $P(C|t_i)$. When term t_i and t_j are clustered together, the new distribution is the weighted average of the individual distributions described as following.

$$\begin{aligned} P(C|t_i \vee t_j) &= \frac{P(t_i)}{P(t_i)+P(t_j)} P(C|t_i) \\ &+ \frac{P(t_j)}{P(t_i)+P(t_j)} P(C|t_j) \end{aligned}$$

The measure of the difference between two probability distributions adapted by [2] is Kullback-Leibler divergence, which is an information-theoretic measure. The KL divergence between the class distributions induced by t_i and t_j is written $D(P(C|t_i)||P(C|t_j))$, and is defined as follows.

$$-\sum_{k=1}^{|C|} P(C_k|t_i) \log \frac{P(C_k|t_i)}{P(C_k|t_j)}$$

To avoid the odd properties of KL divergence, such as it is not symmetric, and it is infinite when an event with non-zero probability in the first distribution has zero probability in the second distribution, they modify the above formula as average KL divergence.

$$\begin{aligned} &\frac{P(t_i)}{P(t_i \vee t_j)} \cdot D(P(C|t_i)||P(C|t_i \vee t_j)) \\ + &\frac{P(t_j)}{P(t_i \vee t_j)} \cdot D(P(C|t_j)||P(C|t_i \vee t_j)) \end{aligned}$$

Instead of comparing the similarity of all possible pairs terms ($O(n^2)$ operation), Baker create clusters using a simple, greedy agglomerative approach that consider all pairs of a much smaller subset, of size c, where c is the final number of clusters desired. The clusters are initialized with c terms that have highest score, using information gain(IG) in [2]. The most similar two clusters are joined, the next term is added as a singleton cluster to bring the total number of clusters back up to c. Notice that the c terms as initial cluster may prefer some classes such that result in a biased estimate of term probability distribution to begin with. To avoid a biased estimate of term probability distribution to begin with, we have equal number of selected terms from each class as initial seeds of clusters. In this paper, the value of c tried for distributional clustering are 120, 240, 360, 600, 1200, 2400, 3600, 4800.

5 Classifiers

There are several studies[3, 26, 9, 20, 19] about Chinese text classification. In [3, 26, 9], they use the weight matrix approach to classify news in vector space model, but their keywords extraction method is not scalable. In [20, 19], they use a scalable approach, SB-tree [6], to extract keywords and classify news using naive Bayes(NB) classifer in probabilistic model. However, in [24], they claims that the performance of the NB is worse than the other classifiers, such as k-Nearest Neighbor(kNN). In this paper, we make an extensive comparison of three classifiers, the NB, the Rocchio [15] and the kNN. Our goal is to compare their performance when they are integrated with term selection and term clustering. To speed up on-line classification of the kNN, we use Generalized Instance Set(GIS) [13] algorithm to reduce the size of training data and to refine the original training data. The following reviews the classifiers evaluated in this paper.

5.1 naive Bayes(NB) Classifier

The naive Bayes classifier[26, 25, 20, 19] is one highly practical learning method and is based on the simplifying assumption that the probabilities of terms occurrences are conditionally independent of each other given the class value [17], though this is often not the case. The naive Bayes approach classifys a request document X to the most probable class, C_{NB} defined below.

$$C_{NB} = argmax_{C_k \in C} P(C_k|X)$$

By Bayes' theorem [10], the $P(C_k|X)$ can be represented as

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{\sum_{C_i \in C} P(X|C_i)P(C_i)}$$

Where $P(C_k) = |C_k|/\sum_{C_i \in C} |C_i|$ is the probability of the class C_k, and $|C_k|$ is the number of training documents in class C_k.

To estimate $P(X|C_k)$ is difficult since it is impossible to collect a sufficiently large number of training examples to estimate this probability without prior knowledge or further assumptions. However, the estimation become possible due to the assumption that a word's(term) occurrence is dependent on the class the document comes from, but that it occurs independently of the other words(terms) in the document. Therefore, the $P(X|C_k)$ can be written as follows [11]:

$$P(X|C_k) = \prod_{j=1}^{|X|} P(t_j|C_k)$$

where $|X|$ is the number of words (terms) in document X, and $P(t_j|C_k)$ is the conditional probability of t_j given Class C_k. Given the term $T = (t_1, t_2, \ldots, t_n)$ that describe the document X, the estimation of $P(X|C_k)$ is reduce to estimating each $P(t_j|C_k)$ independently. Notice above equation works well when every term appears in every document; otherwise, the product becomes 0 when some terms do not appear in that document. We use the following to approximate $P(t_j|C_k)$ to avoid the possibility that the product becomes 0, and still keeps the meaning of the equation.

$$P(t_j|C_k) = \frac{1 + TF(t_j, C_k)}{|T| + \sum_j^{|T|} TF(t_j, C_k)}$$

where $TF(t_j, C_k)$ is the frequency of term t_j in documents having class value k, $|T|$ is the number of all distinct terms used in the domain of document representation. The formula used to predict probability of class value C_k for a given document X is as the following :

$$P(C_k|X) = \frac{P(C_k)\prod_{t_j \in X} P(t_j|C_k)^{TF(t_j,X)}}{\sum_i P(C_i)\prod_{t_j \in X} P(t_j|C_i)^{TF(t_j,X)}}$$

5.2 Rocchio Linear Classifier

Given a class, the training document collection consists of positive and negative examples. Positive examples are those documents belonging to that class while the negative examples are those documents not belonging to that class. Each training example, represented by a vector, is regarded as an *instance*. Let D be an instance in the training collection and it is represented as $(d_1, \ldots, d_n)$ where n is the dimension of term space. Some common weighting schemes for determining the weights d_i include binary weights, term frequency(TF), and term frequency combined with inverse document frequency(TF-IDF)[18]. In this paper, we use TF-IDF weighting method as following.

$$d_i = \log_2(tf_i + 1) * \log_2(\frac{|D|}{df_i})$$

where $|D|$ is the total number of training documents; tf_i is the term frequency in the testing collection and df_i is the document frequency of d_i in training collection. With above definitions, we briefly describe the basic Rocchio[15] algorithm as following.

For every class(category) C_i, there is a feature weight vector

$$C_i = (c_{i1}, \ldots, c_{in})$$

where n is the dimension of term space and each element c_{ij} corresponds to the j-th feature. The elements in vector C_i are learned from all training examples including positive and negative instances. To determine whether or not a class is assigned to the request document X, it computes the inner product δ between the document vector X, represented as $(x_1, \ldots, x_n)$, and the feature vector C_i as follows:

$$\delta = \sum_{j=1}^{n} x_i \cdot c_{ij}$$

If the inner product is greater than a certain threshold value, the class(category) is assigned to X. Let P and N be the set of positive and negative instances respect to class C_i in the training corpus. The feature vector C_i is given as follows:

$$C_i = \frac{\sum_{D \in P} D}{|P|} - \eta \frac{\sum_{D \in N} D}{|N|}$$

In this paper, the value of η tried for Rocchio algorithm are 0.25, 0.5, 0.75, 1.0. We assume the X is assigned to only one class that has the largest value of δ.

5.3 k-Nearest Neighbor(kNN) Classifier

In a kNN algorithm, each training document D_i as well as the request document X are represented by vectors as $(d_{i1}, \ldots, d_{in})$ and $(x_1, \ldots, x_n)$ respectively. To conduct categorization, the similarity $\triangle(X, D_i)$ between each D_i and X is calculated. The training instances are sorted by the similarity metric in descending order. Then the k top-ranking instances are selected. The final score of the request document to each class(category) is calculated by considering the similarity metric of these k selected instances and their category association. The document is assigned to categories with the score greater than a certain threshold value. The cosine similarity often is used to measure similarity $\Delta(X, D_i)$ and is defined below :

$$\Delta(X, D_i) = \frac{\sum_{j=1}^{n} x_j \cdot d_{ij}}{\sqrt{\sum_{j=1}^{n} x_i^2}\sqrt{\sum_{j=1}^{n} d_{ij}^2}}$$

In this paper, the value of k tried for the kNN are 5, 10, 15, 20, 30, 50, 100, 150, 200, 300; we assign X to the class with largest Δ score.

6 Generalized Instance Set(GIS) Algorithm

The running time of the kNN is dependent on the size of training data. To speed up the on-line classification, there is an *generalized instance set (GIS)* algorithm[13], which unify the strengths of kNN algorithm and linear classifiers. The main idea is to construct a set of generalized instances(GI) to replace the original training examples. To perform categorization of a request document, kNN classifiers need to compute $|D|$ similarity scores where $|D|$ is the total number of training examples. However, the GIS algorithm only requires computation of $|G|$ similarity scores where $|G|$ is the total number of the learned generalized instances. Typically, $|G|$ is much less than $|D|$. Therefore, the GIS algorithm can be faster than the kNN algorithm. Notice that the Generalized Instance Set(GIS) algorithm not only could reduce the number of instances to save the running time but also make kNN more tolerable to noisy instance. We reviews the GIS algorithm[13] as follows.

For each class, the GIS algorithm automatically selects a representative positive instance and performs generalization process using g nearest neighbors which may include positive and negative instances. A generalized instance G is formed after the generation process. This G will be evaluated by the function *Rep* denoting the representative power. If the representative power of G is better than the old one(i.e. G'), than use G as a new point and repeat the search and generalization task again. The algorithm will continue to search for the best local generalized instance. If there is no further improvement, the last generalized instance G' is added to the generalized instance set GS and the corresponding g nearest neighbors are removed from the training collection. This process is repeated until no positive instance remained in the training collection. As the learning progresses, the GIS algorithm constructs a number of generalized instances and stores them in GS. The representative power function $Rep(G)$ for a generalized instance G is defined as follows:

$$Rep(G) = \sum_{I^+ \in K} (g - rank(I^+))$$

where K is the set of g nearest neighbor of G, I^+ is a positive instance in K and rank(I^+) denotes the ranking of instance I^+ in the set K according to the similarity. Large value for $Rep(G)$ implies that more positive instances are found in the set of g nearest neighbors of G. In this paper, we have basic Rocchio algorithm with $\eta = 0.25$ for the generalization process in the GIS algorithm,; the value of g tried for the GIS algorithm are 5, 10, 15, 20, 25, 30.

7 Experimental Results

Let n be the number of terms selected by term selection method, c be the number of term clusters, g be the g nearest neighbors in generalization process of the GIS algorithm. To determine the proper numbers of n, c and g, we first determine a proper number n by term selection, χ^2 statistic, and then use distributional clustering to determine a proper number c of term clusters. Finally, we experiment with the selected n terms, which are clustered into c term clusters, to determine the proper g for the GIS. In this paper, we use three measures, *MicroAccuracy*, *MacroAccuracy* and *AccuracyVariance*, to study the performance of each classifier. Notice that MacroAccuracy and AccuracyVariance are used to inspect the variation of accuracy between each class. The less value of AccuracyVariance is, the less difference of classification accuracy between each class is.

Let $|C|$ be the number of classes and C_i be the set of testing news that are belong to the i_th class. Let H_{ij} be the number of news in C_i that are classified to the j_th class. Let the accuracy of i_th class be $Acc(i) = \frac{H_{ii}}{|C_i|}$, where $|C_i|$ is the number of news in C_i. The MicroAccuracy is defined to be $\frac{\sum_{i=1}^{i=|C|} H_{ii}}{\sum_{i=1}^{i=|C|} |C_i|}$ that represents the total average of classification accuracy. The MacroAccuracy is defined to be $\frac{\sum_{i=1}^{i=|C|} Acc(i)}{|C|}$ that represents the average accuracy of classes. The Accuracy-Variance is defined to be $\frac{\sum_{i=1}^{i=|C|}(Acc(i)-MacroAccuracy)^2}{|C|}$ that represents the variance of the accuracy.

We use the news from Central News Agency(CNA)[1] as experimental resource. First, we use one year news, 1991/1/1-1991/12/31, to extract *Chinese frequent strings*(CFS) [16] via scanning the leaves of SB-trees as described in Section 2, and then use χ^2 statistic method to select the most representative n terms to form the bases of term vector space. Each document D is represented as $(d_1, \ldots, d_n)$ where n is the dimension of term space and d_i is corresponding weight of the i_th term. we use the next month news, 1992/1/1-1992/1/31, as training data , and the next 3 days news, 1992/2/1-1992/2/3, as testing data. The news consists

	CNA News Group	1991/1/1-12/31	1992/1/1-31	1992/2/1-3
1.政治	cna.politics.*	23516	1744	90
2.經濟	cna.economics.*	10160	1047	40
3.交通	cna.transport.*	3423	336	34
4.文教	cna.edu.*	6064	517	26
5.體育	cna.l*	4929	387	25
6.社會	cna.judiciary.*	5679	612	38
7.股市	cna.stock.*	3313	277	7
8.軍事	cna.military.*	4646	319	15
9.農業	cna.argriculture.*	3217	279	22
10.宗教	cna.religion.*	1315	109	3
11.財政	cna.finance.*	3622	287	11
12.社福	cna.health-n-welfare.*	3536	340	28
	Total	73420	6254	339

Table 1: CNA News Statistics

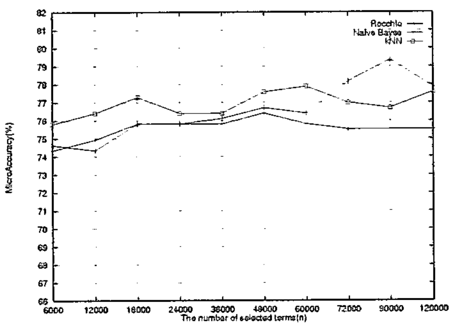

Figure 3: MicroAccuracy : integration with term selection

of 12 classes as listed in Table 1.

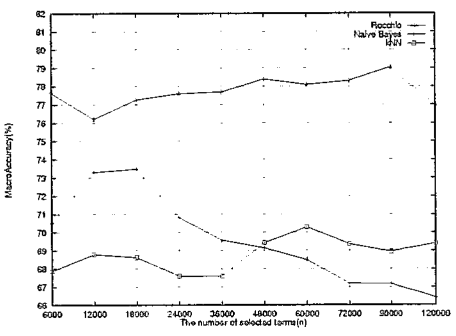

Figure 4: MacroAccuracy : integration with term selection

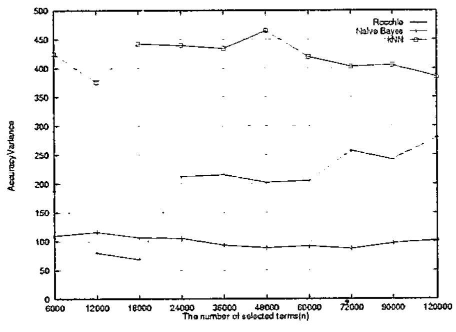

Figure 5: AccuracyVariance : integration with term selection

7.1 Integration with Term Selection

We first measure the performance of each classifier when it is integrated with χ^2 statistic term selection method. Let n be the number of terms selected. The best MicroAccuracy achieved by the NB, the kNN, the Rocchio are 76.40%, 77.88% and 79.35% when n is $48,000, 60,000$ and $90,000$ respectively as shown in Figure 3. kNN outperforms Rocchio when the dimension of term space $n \leq 60,000$, but Rocchio achieves the best MicroAccuracy, 79.35%, when n is $90,000$. The reason is that kNN is sensitive to the noisy examples and do not cope with irrelevant terms effectively[13]. On the other hand, Rocchio can deal with noise to some extend by summarizing the contribution of positive and negative training examples. Although kNN achieves the best MicroAccuracy when $n \leq 60,000$, its MacroAccuracy is relatively low. This implies that kNN prefers the large classes than the small ones. The Rocchio achieves the best MacroAccuracy for all n, and the best MicroAccuracy when n is $90,000$. It performs quite stable when integrated with term selection. This observation is different from previous research[23].

7.2 Integration with Term Clustering

We next measure the performance of each classifier when term clustering is integrated. We experiment for $n = 48,000$, $n = 60,000$ and $n = 90,000$ according to the best MicroAccuracy achieved by the NB, the kNN and the Rocchio respectively as shown in Figure 3. Let c be the number of term clusters. The range of c experimented is from 120 to 4,800. In this paper, we only show the experimental results for the case $n = 90,000$. The performance in the other cases are similar. As shown in Figure 6, the kNN achieves the best MicroAccuracy, 80.24%, when c is 3,600. Furthermore, both the MacroAccuracy(68.94% to 73.73%) and the AccuracyVariance (405.53 to 343.8) of the kNN are improved via term clustering as shown in Figure 7. Notice that the performance of the NB drops because clustering terms into one group might distort term distributions.

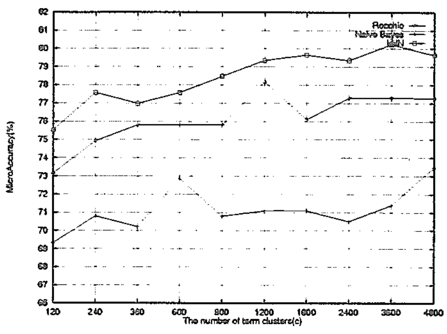

Figure 6: MicroAccuracy : integration with term clustering

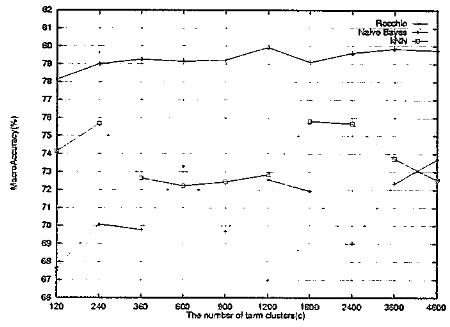

Figure 7: MacroAccuracy : integration with term clustering

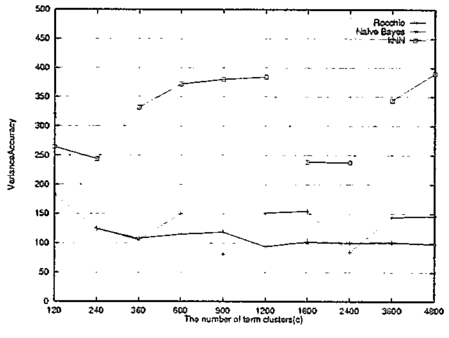

Figure 8: AccuracyVariance : integration with term clustering

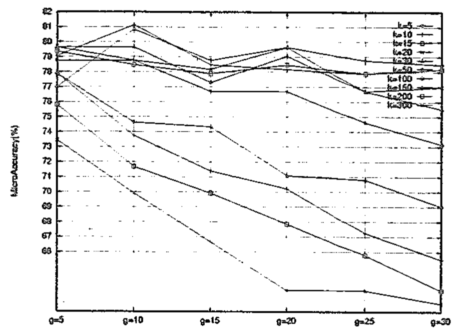

Figure 9: kNN with GIS : MicroAccuracy

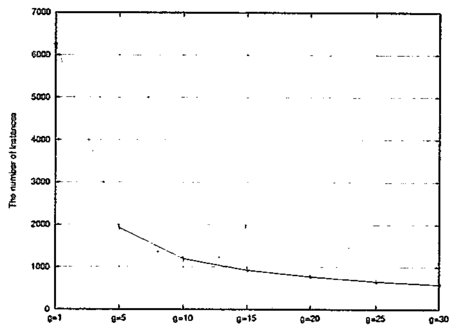

Figure 10: The reduced size of the instances set

7.3 kNN with GIS

We choose $n = 90,000$ and $c = 3,600$, and experiment for various g. As indicated in previous section, kNN achieves the best performance when term selection and term clustering are integrated. A major problem of kNN is that its classification time is high especially when the number of training data is large. To speed up kNN, we apply GIS algorithm to reduce the number of training instances.

As shown in Figure 9, the best MicroAccuracy achieved by the kNN is 81.12% when $g = 10$. Notice that the MicroAccuracy increases from 80.24% to 81.12% while the number of instances set is reduced from 6,254 to 1195 as shown in Figure 10. Note that $g = 1$ means the original training instances. To show the effect of speedup due to the GIS algorithm, we measure the running times on a PC with Celeron 333A CPU and 98MB RAM. The running times of Rocchio, NB, kNN(k=100) and kNN with GIS(k=100) are 21, 22, 104 and 60 seconds respectively.

8 Conclusions

In this paper, we make an extensive comparison of three classifiers, naive Bayes(NB) probabilistic classifier, Rocchio linear classifier and k-Nearest Neighbor(kNN) classifier for Chinese text classification. Experimental result shows that the kNN achieves the highest MicroAccuracy, 77.88% when the dimension of term space, n, is less than or equal to $60,000$. This observation is consistent with previous study[24]. However, it is worthy to notice that the Rocchio algorithm achieves the highest MicroAccuracy, 79.35%, when $n \geq 72,000$. The reason is that kNN is sensitive to the noisy examples and do not cope with irrelevant terms effectively[13]. On the other hand, Rocchio can deal with noise to some extend by summarizing the contribution of positive and negative training examples. To reduce the high dimension of term space to an practical level while maintaining similar classification accuracy, we combine term selection, χ^2 statistic, and term clustering, distributional clustering. The combination improves the MicroAccuracy of the kNN from 77.88% to 80.24% while n is reduced from $90,000$ to $3,600$. Furthermore, we use GIS algorithm to speed up on-line classification of the kNN by reducing the number of training data. GIS also improves the MicroAccuracy of kNN from 80.24% to 81.12%.

References

[1] Central news agency. http://www.cna.com.tw/index.html.

[2] L.Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *SIGIR 98*, 1998.

[3] S. M. Chen. Automatic classification of economics news. Master's thesis, National Taiwan University, Taiwan, R.O.C., 1992.

[4] Lee-Feng Chien. Pat-tree-based keyword extraction for chinese information retrieval. In *SIGIR 97*, 1997.

[5] David D.Lewis and William A. Gale. A sequential algorithm for training text classifier. In *SIGIR 94*, 1994.

[6] Paolo Ferragina and Roberto Grossi. An experimental study of sb-trees. In *ACM-SIAM symposium on Discrete Algorithms*, 1996.

[7] William B. Frakes and Rick Kazman. *Information Retrieval Data Structures & Algorithm.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1992.

[8] Marko Grobelink and Dunja Mladenic. Feature selection for classification based on text hierarchy. In *Conference on Automated Learning and Discovery CONALD-98*, 1998.

[9] Sen-Yuan Huang, Yu-Ling Chou, and Ja-Chen Lin. Automatic classification for news written in chinese. *Computer Processing of Oriental Languages*, 12(2):143–159, 1998.

[10] M. James. *Classification Algorithms.* Wiley, 1985.

[11] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 1997.

[12] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data Analysis : An Introduction to Cluster Analysis.* John Wiley and Sons,Inc., New York, 1990.

[13] Wai Lam. Using a generalized instance set for automatic text categorization. In *SIGIR 98*, 1998.

[14] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *SIGIR 96*, 1996.

[15] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *SIGIR 96*, 1996.

[16] Yih-Jeng Lin, Ming-Shing Yu, Shyh-Yang Hwang, and Ming-Jer Wu. A way to extract unknown words without dictionary from chinese corpus and its applications. In *Research on Computational Linguistics Conference(ROCLING XI)*, 1998.

[17] Tom M. Mitchell. *Machine Learning.* The McGraw-Hill Companies, Inc, 1997.

[18] Amitabh Kumar Singhal. *Term Weighting Revisited.* PhD thesis, Cornell University, 1997.

[19] Jyh-Jong Tsay and Jing-Doo Wang. Term selection with distributional clustering for chinese text categorization using n-grams. In *ROCLING XII*, 1999.

[20] Jyh-Jong Tsay, Jing-Doo Wang, Chun-Fu Pai, and Ming-Kuen Tsay. Implementation and evaluation of scalable approaches for automatic chinese text categorization. In *The 13th Pacific Asia Conference on Language, Information and Computation*, 1999.

[21] Yuen-Hsien Tseng. Fast keyword extraction of chinese document in a web environment. In *Information Retrieval with Asian Languages(IRAL 1997)*, 1997.

[22] Yiming Yang. Effective and efficient learning from human decisions in text categorization and retrieval. In *SIGIR 94*, 1994.

[23] Yiming Yang. An evaluation of statistical approaches to text categorization. In *Journal of Information Retrieval*, 1999.

[24] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR 99*, 1999.

[25] Yiming Yang and Jan O. Pedersen. A comparative study on feature in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 1998.

[26] Yun-Yan Yang. A study of document autoclassification in mandarin chinese. In *Research on Computational Linguistics Conference(ROCLING VI)*, 1993.

[27] Bo-Hyun Yun, Min-Jeung Cho, and Hae-Chang Rim. Korean information retrieval model based on the principle of word formation. In *Information Retrieval with Asian Languages(IRAL 1997)*, 1997.