An Effective Method To Improve kNN Text Classifier*

Xiulan Hao[1], Xiaopeng Tao[1], Chenghong Zhang[2] and Yunfa Hu[1]
[1] Department of Computing and Information Technology, Fudan University
[2] School of Management, Fudan University
No. 220, Handan Road, Shanghai, China, 200433
hxl2221_cn@126.com,{xptao,chzhang,yfhu}@fudan.edu.cn

Abstract

Many of standard classification algorithms usually assume that the training examples are evenly distributed among different classes. However, unbalanced data sets often appear in many applications. As a simple, effective categorization method, kNN is widely used, but it suffers from biased data sets, too. In developing the Prototype of Internet Information Security for Shanghai Council of Information and Security, we detect that when training data set is biased, almost all test documents of some rare categories are classified into common ones. To alleviate such a misfortune, we propose a novel concept, critical point (CP), and adapt traditional kNN by integrating $CP's$ approximate value, LB or UB, training number with decision rules. Exhaustive experiments illustrate that the adapted kNN achieves significant classification performance improvement on biased corpora.

1 Introduction

With the rapid growth of Internet, more and more online documents become available. Text categorization is one of the key techniques aiming at organizing and processing this huge collection. In the past, many machine-learning techniques were applied to text categorization, including the Rocchio approach, decision trees, the naive Bayes method, neural networks, k-nearest neighbors (kNN), support vector machines, boosting [5, 6].

As a simple and efficient approach to text categorization, kNN is widely used and obtains a better result [1, 9, 11]. The idea behind the kNN algorithm is quite straightforward. To classify a new document, the system finds the k-nearest neighbors among the training documents and uses the categories of the k-nearest neighbors to label the new document. Its performance depends greatly on two factors, i.e., a suitable similarity function and an appropriate value for parameter k.

Biased or skewed distribution of data set is one of challenges in text categorization [6]. It often leads to lower performance [3, 4, 7, 10]. The main strategies to deal with this problem include feature optimizations [2], modification of traditional ones [3], and re-sampling [4], etc.

However, re-sampling usually removes training documents in larger categories, thus may lose some important information and always sacrifices the classification performance in some cases [7]. $WAKNN$ proposed by [2], in essence, is a method of feature optimization, and it has better classification results than many other classifiers, but it has a high computational cost.

We attempt to define decision functions according to the number of training samples and solve the problem of large classes overwhelming small classes. Experiments indicate, when training samples keep unchanged, Macro-F1 and Micro-Recall of categorization rise dramatically.

The remainder of the paper is organized as follows: Section 2 reviews traditional kNN categorization and its defects. Section 3 explains CP-based kNN categorization. Section 4 reports the test results using this method. Section 5 concludes the findings.

2 Traditional kNN Categorization

Assume that n-dimensional vector $X = (x_1, x_2, ..., x_n)$ represents a document, $C_i = (X_1^i, X_2^i, ..., X_q^i)$ represents a category (also known as class) containing q documents. Given a training set D consisting of m categories, $C_1, C_2, ..., C_m$, and a new arriving document X, kNN classifier will compute the similarity of each document in document set D to X and search the k neighbors nearest to X based on similarity. If there are k_i documents belonging to category C_i, define 2 decision functions as follows:

*This work was supported by the Natural Science Foundation of China (NSFC) under grant No. 70471011 and No. 60473070.

0-7695-2909-7/07 $25.00 © 2007 IEEE
DOI 10.1109/SNPD.2007.296

Function 1. Assume $f_i(X) = k_i, i = 1, ..., m$

$$f(X) = arg\max_j(f_j(X)), j = 1, ..., m \quad (1)$$

then X is classified into C_j, i.e., $X \in C_j$. We label the algorithm formulated by Function (1) as Trad1.

Function 2. Assume

$$g_i(X) = \sum_{l=1}^{k_i} sim(X, X_l^i), i = 1, ..., m$$

$$g(X) = arg\max_j(g_j(X)), j = 1, ..., m \quad (2)$$

then X is classified into C_j, i.e., $X \in C_j$, where $sim(X, X_l^i)$ is similarity between X and training sample X_l^i. The algorithm described by Function (2) is labeled as Trad2.

However these two functions neglect distribution of training samples. The distribution of documents among different classes in a training set are usually not even, so the classifier may be biased towards larger classes (also called common classes). For example, when using the algorithm indicated by Function (2), many tiny similarity values will accumulate to a relatively larger one, which may, improperly, make a final decision favoring a larger class. The result is that a number of test samples in smaller classes (also known as rare classes) are classified into larger ones mistakenly, thus make system performance deteriorated.

3 CP-based kNN Categorization

In our project concerning Internet documents classification, we find almost all documents in smaller categories are classified into larger ones by traditional decision functions. To overcome this defect, we redefine decision functions by integrating the number of training documents in each category into them. Suppose that the smallest class has $minTrainNum$ samples, the largest one has $maxTrainNum$ samples.We shall use notations below:

SF	Shrink factor
CP	Critical point
LB	Lower approximation of CP
UB	Upper approximation of CP
N_j	Size of Category j in the Training Set
N_{ji}	$N_{ji} = (N_j)^{1/sf_i}, j = 1, \cdots, m$
x_i	$x_i = (MinTrainNum)^{1/sf_i}$
y_i	$y_i = (MaxTrainNum)^{1/sf_i}$
μ_i	Mean of N_{ji} , $j = 1, \cdots, m$
σ_i	Standard deviation of N_{ji} , $j = 1, \cdots, m$
λ_i	$\lambda_i = \frac{y_i}{x_i}, i \neq 0$

Function 3.

$$f'(X) = arg\max_j[\frac{k_j}{(N_j)^{1/sf}} \times \frac{(minTrainNum)^{1/sf}}{k}] \quad (3)$$

where $j = 1, ..., m$, then X is classified into C_j, $X \in C_j$.

Function 4. Assume

$$g_i(X) = \sum_{l=1}^{k_i} sim(X, X_l^i), i = 1, ..., m$$

$$g'(X) = arg\max_j[\frac{g_j(X)}{(N_j)^{1/sf}} \times \frac{(minTrainNum)^{1/sf}}{k}] \quad (4)$$

where $j = 1, ..., m$, then X is classified into C_j, $X \in C_j$.

Function 3, 4 are modifications of Function 1, 2 respectively. For convenience, we label the algorithm represented by Function (4) as $SF = sf_i$, where sf_i is the value of sf. For example, if sf takes 2.0, we denote it by $SF = 2.0$. The cosine value of two vectors is used to measure the similarity between the two documents, although other similarity measures are possible.

Perhaps due to relatively simple of exponential functions, our decision functions happen to have the similar components as [7]. But we focus on to find an optimal exponent according to the distribution of training data set. To distinct from [7], we shall call an exponent as a **shrink factor** below. From Function 3 or Function 4, we can infer,

- when sf is smaller, decision functions tend to favor smaller classes;
- when sf is larger, decision functions tend to prefer larger classes;
- when $sf \to \infty$, decision functions degenerate to traditional ones.

We conjecture there must be a definite value of sf that balance larger classes and smaller classes and have the optimal discriminative power. We call this value of sf as **critical point**. Because standard deviation of a data set reflects its distribution, we shall use it to find CP. For an unbalanced data set, when $N_{ji} = N_j$, x_i is smaller than σ_i. But when exponent operation is exerted on each N_j, that is, $N_{ji} = (N_j)^{1/sf_i}, sf_i > 1$, at a definite sf_i, x_i will equal to σ_i. We can empirically prove this sf_i is the CP to be found. Though there must be such a sf_i satisfied $x_i = \sigma_i$, it is hard to calculate it precisely. We can use a secondary optimal value to replace it. Let's see what happens when x_i=σ_i. Two special cases:
(1)Except y_i, all other N_{ji} have the same values as x_i, i.e., there are $m - 1$ $x_i's$ and one y_i. In this case,

$$\mu_i = \frac{(m-1)\times x_i + y_i}{m} = \frac{(m-1+\lambda_i)x_i}{m} \quad (5)$$

$$\sigma_i = \sqrt{\frac{(m-1)\times(x_i-\mu_i)^2 + (y_i-\mu_i)^2}{m-1}} = x_i \quad (6)$$

Hence,

$$\lambda_i = 1 + m/\sqrt{2} \doteq 1 + 0.707m \quad (7)$$

(2)Except x_i, all other N_{ji} have the same values as y_i, i.e., there are $m-1$ $y_i's$ and one x_i.

$$\mu_i = \frac{(m-1)\times y_i + x_i}{m} = \frac{[\lambda_i(m-1)+1]x_i}{m} \quad (8)$$

$$\sigma_i = \sqrt{\frac{(m-1)\times(y_i-\mu_i)^2 + (x_i-\mu_i)^2}{m-1}} = x_i \quad (9)$$

Hence,

$$\lambda_i = 1 + \sqrt{\frac{m^2}{2(m-1)}} \doteq 1 + \frac{0.707m}{\sqrt{m-1}} \quad (10)$$

In practice, rarely happen these 2 situations. But we can estimate λ_i by Function 7 and Function 10 when x_i=σ_i. That is,

$$1 + \frac{0.707m}{\sqrt{m-1}} \le \lambda_i \le 1 + 0.707m \quad (11)$$

Starting from $(\frac{y_0}{x_0})^{1/sf_0} = \lambda_0$, where $\lambda_0 = 1 + \frac{0.707m}{\sqrt{m-1}}$, $y_0 = MaxTrainNum$, $x_0 = MinTrainNum$, we can compute the approximate value of CP from sf_0 more quickly. Because λ_0 is the lower boundary of λ_i, value of sf_0 may larger than CP and we begin by decreasing sf_0. The evaluation process is shown in Algorithm 1. When classification, we can choose one of approximate values of CP as shrink factor, i.e. , UB or LB.

Algorithm 1: Calculate CP

INPUT $m, N_j(j = 1, \cdots, m)$
OUTPUT CP, LB, UB

$\lambda_0 = 1 + \frac{0.707m}{\sqrt{m-1}}$
$y_0 = \arg\max_{j=1,\cdots,m}(N_j), x_0 = \arg\min_{j=1,\cdots,m}(N_j)$
if $\lambda_0 > y_0/x_0$ **then**
  $LB = UB = CP = 1$ {Data is even}
  return
end if
$sf_0 = log_{\lambda_0}(y_0/x_0)$, $sf_i = floor(sf_0 * 10)/10.0$
repeat
  $x_i = (x_0)^{1/sf_i}$
  for $j = 1$ to m **do**
    $N_{ji} = (N_j)^{1/sf_i}$
  end for
  /*Calculating standard deviation of $(N_{j1}, \ldots, N_{jm})$ */
  $\sigma_i = stdev(N_{j1}, \ldots, N_{jm})$
  $sf_i = sf_i - 0.5$
until $x_i < \sigma_i$
$sf_i = sf_i + 0.5$
while $x_i < \sigma_i$ **do**
  $sf_i = sf_i + 0.1$, $x_i = (x_0)^{1/sf_i}$
  for $j = 1$ to m **do**
    $N_{ji} = (N_j)^{1/sf_i}$
  end for
  $\sigma_i = stdev(N_{j1}, \ldots, N_{jm})$
end while
$CP = sf_i$, $LB = floor(CP + 0.5)$
if $floor(CP) \neq LB$ **then**
  $LB = LB - 0.5$
end if
$UB = LB + 0.5$
return

4 Experiments and Evaluations

4.1 Datasets

To verify the validity of our decision functions, we test on 2 Chinese DataSets. Both are compiled by our research group and are collections of news and papers. News is mainly downloaded from **http://www.sina.com.cn** and **http://www.chinadaily.com.cn**, and papers mainly come from **http://www.edu.cnki.net**. DataSet1 is composed of 13796 documents. As can be seen in Table 1, C34-Economy takes 18.1% of training set while C29-Transport, only takes 0.8%. The ratio of maximum samples and minimum samples is 21.8, thus distribution of DataSet1 is very skewed. DataSet2 consists of 2420 documents; the ratio of maximum samples and minimum samples is 10.375.

Besides, Reuter and TDT2 employed by [7] can be used directly to prove our methods.

4.2 Evaluation Measures

The category assignments of a binary classifier can be evaluated using a contingency table (Table 2) for each category.

Table 2. A contingency table

	Yes is correct	No is correct
Assigned Yes	a	b
Assigned No	c	d

Conventional performance measures are defined and computed from these contingency tables. These measures are recall (r), precision (p), fallout (f), accuracy (Acc), error (Err) and F1:

- $r = a/(a+c)$, if $a + c > 0$, otherwise undefined;
- $p = a/(a+b)$, if $a + b > 0$, otherwise undefined;
- $f = b/(b+d)$, if $b + d > 0$, otherwise undefined;

Table 1. Statistical Information of Training samples in DataSet1 and DataSet2

DataSet1					
Name of Class	No. of samples	Percentage	Name of Class	No. of samples	Percentage
C3-Art	530	5.8	C5-Education	669	7.3
C7-History	622	6.8	C11-Space	502	5.5
C19-Computer	1070	11.6	C29-Transport	76	0.8
C31-Environment	784	8.5	C32-Agriculture	988	10.7
C34-Economy	1663	18.1	C37-Military	99	1.1
C38-Politics	1329	14.4	C39-Sports	866	9.4
DataSet2					
ART	103	6.381	SPACE	80	4.956
C15-ENERGY	41	2.54	COMPUTER	40	2.478
MINE	40	2.478	TRANSPORT	77	4.77
ENVIRONMENT	69	4.275	AGRICULTURE	58	3.593
MEDICAL	69	4.275	ECONOMY	151	9.355
EDUCATION	72	4.46	MILITARY	104	6.443
POLITICS	415	25.712	SPORTS	234	14.498
HISTORY	61	3.779			

- $Acc = (a+d)/n$, where $n = a+b+c+d > 0$;
- $Err = (b+c)/n$, where $n = a+b+c+d > 0$;
- $F1 = 2rp/(r+p)$;

Macro-averaging and micro-averaging are two methods for evaluating performance average across categories. Micro-average is considered a per-document average while macro-average is a per-category average [9]. Assume there are C categories, then

- $Macro - F1 = (\sum_{c \in C} F1_c)/|C|$
- $Macro - Recall = (\sum_{c \in C} r_c)/|C|$
- $Macro - Precision = (\sum_{c \in C} p_c)/|C|$
- $Micro - Recall = (\sum_{c \in C} a)/(\sum_{c \in C} a + \sum_{c \in C} c)$

4.3 Experimental Results

We only perform experiments on DataSet1 and DataSet2. As to Reuter and TDT2, we cite results of [7] directly. To compare with [7], we choose $10,000$ features and use Information Gain as feature selection. We split each dataset into three approximately equal parts, then use two parts for training and the remaining third for test. We conduct the training-test procedure three times and use the average of the three performances as final result. This is so-called three-fold cross validation. We test at discrete points $k = 5, \ldots, 100$ respectively. Note that we use the mean of 20 test points as comparison values. Because Function 3 vs. Function 1 has the similar conclusions as Function 4 vs. Function 2, we only provide the experimental results of the latter. Using Algorithm 1, we can obtain the CP, LB, UB of the 4 DataSets respectively, as listed in Table 3.

4.3.1 Macro-Recall

Form Figure 1 , we can reach the same conclusion as [7] that the larger the shrink factor, the lower the Macro-Recall on DataSet2. Macro-Recall on DataSet1 has the similar conclusion, for page limitations, we omit it.

Table 3. Parameters of the 4 DataSets

Name of DataSet	CP	LB	UB
DataSet1	2.1	2.0	2.5
DataSet2	1.5	1.5	2.0
Reuter	3.7	3.5	4.0
TDT2	3.7	3.5	4.0

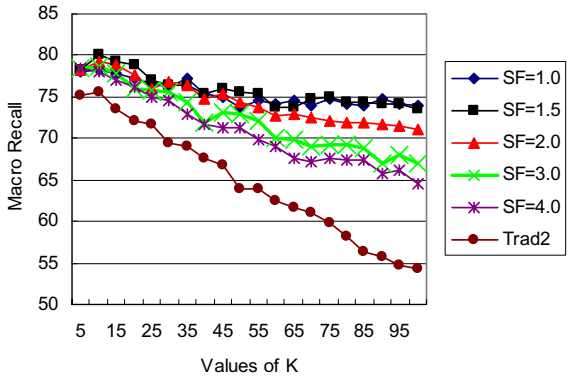

Figure 1. Macro-Recall on DataSet2

Performances of rare categories are accordance with Macro-Recall, we omit those figures. That recall of smaller categories achieve best results when $sf = 1.0$ justifies our suggestion that the smaller is sf, the more kNN classifier favors rare categories.

4.3.2 Precision

In most cases, precisions have the same conclusions as [7],that is, the larger the shrink factor, the higher the Macro-Precision, and Macro-Precision reaches the peak when kNN classifier works with traditional decision function. On DataSet2, Macro-Precision complies to this rule completely. But the rule doesn't fit DataSet1 well.

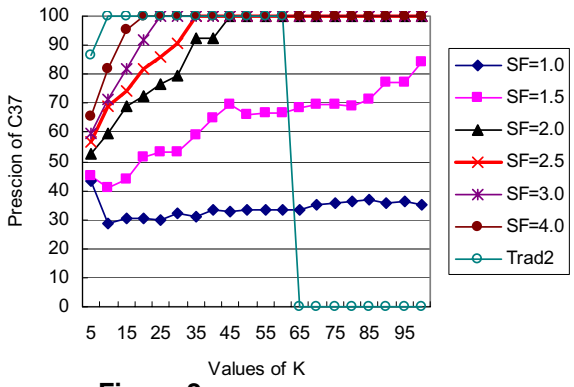

Figure 2. Precision of C37 in DataSet1

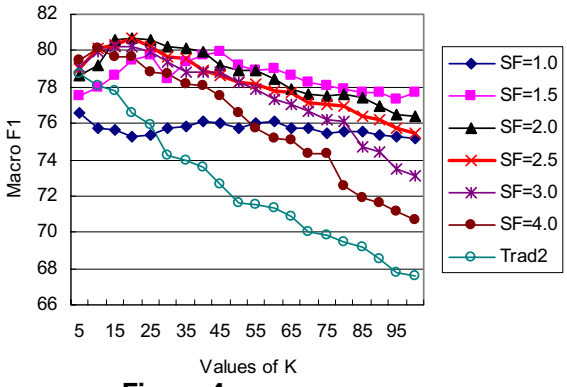

Figure 4. Macro F1 on DataSet1

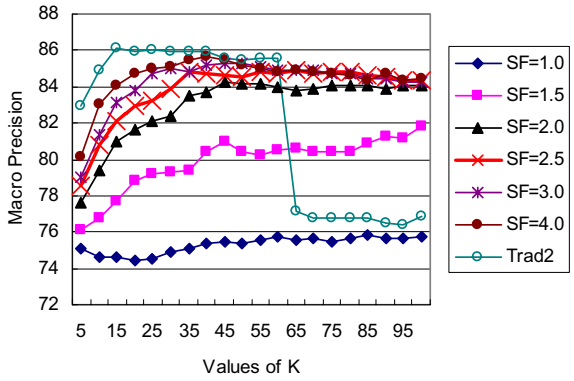

Figure 3. Macro-Precision on DataSet1

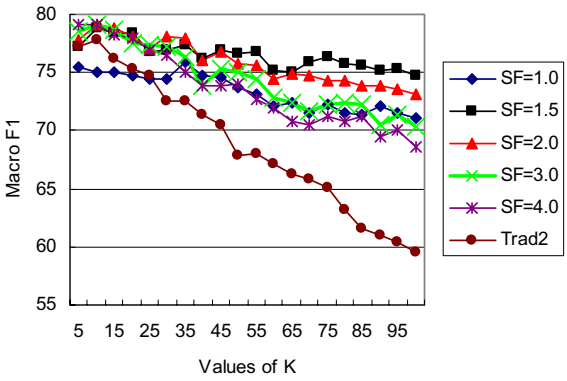

Figure 5. Macro F1 on DataSet2

As indicated in Figure 2, when kNN classifier runs with traditional decision function on DataSet1, its precision fluctuates fiercely: at some k's, it reaches the peak of 100%, and at some k's, it hits a low of 0. The fluctuation of precision of rare categories, in reverse, will affect the Macro-Precision, as shown in Figure 3. Therefore, when rare categories exists, precision will not be an appropriate measure.

4.3.3 F1 Measure

As can be seen from Figure 4, when $sf = 2.0$, F1 reaches a peak of 78.662%. At the same time, Trad2 hits a low of 72.463%. That is, $sf = 2.0$ beats Trad2 by 6.2% on DataSet1. Note that 2.0 is the value of LB of DataSet1. The performance remain stable when $15 \leq k \leq 30$.

Again, F1 of Trad2 declines to 68.716%, which is the worst score on DataSet2. When $sf = 1.5$, F1 obtains its best score on DataSet2, 76.476%, about 7.8% higher than Trad2. Note that 1.5 is the value of LB of DataSet2. When $k > 20$, F1 scores drop sharply.

According to [7], when exponent takes 4.0, their algorithm $NWKNN$ achieves the best results on Reuter and TDT2 and beats KNN by 10% on TDT2. Again 4.0 is just the value of UB on these two datasets.

As to rare categories, F1 measures are improved dramatically. The worst score of rare categories in DataSet2 is C37-Military, when $sf < UB$, all F1 scores climb to above 50%, which is an acceptable value[6]. See Figure 6, for more detail. Similar conclusion can be reached on DataSet1.

4.3.4 Micro-Recall

As indicated by 7, on DataSet2, except $sf = 1.0$, all scores under Function 4 are higher than Trad2. Therefore, our decision function is excellent by micro-average measure. Note Micro-Recall falls steeply when $k > 25$. There are the same findings on DataSet1.

4.3.5 Overall Evaluation

From experimental results, we can see that when $sf > LB$, all curves have the similar tendency to traditional ones and when $sf \leq LB$, the curves exhibit another trend. This fact justifies our assumption that when sf is smaller, decision functions tend to favor smaller classes; and when sf is larger, decision functions tend to prefer larger classes.

- LB **or** UB**?**

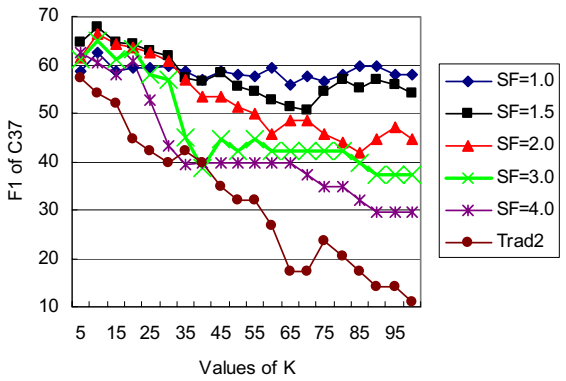

Figure 6. F1 of C37-Military in DataSet2

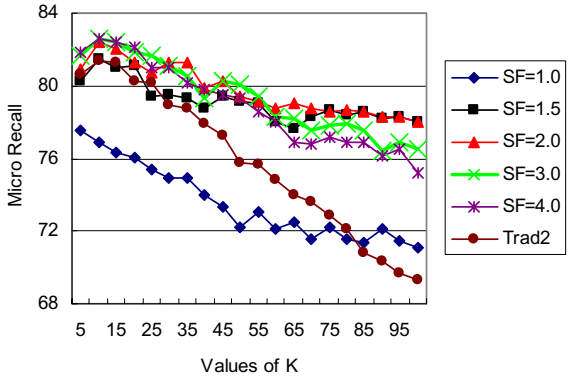

Figure 7. Micro Recalls on DataSet2

On all four corpus, F1 measure reaches its peak at $sf = LB$ or $sf = UB$. We recommend to choose LB for 2 reasons. Firstly, when $sf \leq LB$, all measures are less insensitive to k. Secondly, Macro-Recall and recall of rare categories tend to have better scores at LB than at UB.

- **Selection of k**

  Though [8] argued that the performance reached peak when $k = 30$ and when k was between 30 and 200, scores remained constant, our results do not conform to this claim. Due to different datasets and decision functions, our classifier remains stable when k varies between 15 and 25. When $k > 30$, all tend to decline. The fact verifies that many tiny similarity values will accumulate to a relatively large one, which may, improperly, make a final decision favoring a larger category. Notice when $k \leq 10$, Trad2 works well. This, too, explains that when k is bigger, more noise data will be added to decision functions.

5 Conclusion

When distribution of training samples is biased, our kNN classifier improves recall dramatically and alleviates the misfortune that almost all test documents in smaller classes are judged as some larger classes.

To our knowledge, [3] also combines training numbers with their decision functions. But when selecting top k for different categories, they introduced a parameter α which should be dealt with deliberately. Our decision function is more simple and decided by distribution of training documents completely.

Our main contributions are,

- Propose a novel concept, Critical Point (CP), and give an algorithm to evaluate the value of CP according to distribution of training samples;
- Integrate $CP's$ approximate value, LB or UB, and number of training samples with traditional decision rules;
- Verify its validity by exhaustive experiments on 4 corpus. Note that results on Reuter and TDT2 are quoted from [7].

References

[1] A. Cardoso-Cachopo and A. L. Oliveira. An empirical comparison of text categorization methods. In *Proceedings of the 10th International Symposium on String Processing and Information Retrieval*, pages 183–196, 2003.

[2] E.-H. S. Han, G. Karypis, and V. Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53–65, 2001.

[3] B. Li, Q. Lu, and S. Yu. An adaptive k-nearest neighbor text categorization strategy. In *ACM Transactions on Asian Language Information Processing*, volume 3, pages 408–421, December 2004.

[4] R. Li and Y. Hu. Noise reduction to text categorization based on density for $k$nn. In *the 2th International Conference on Machine Learning and Cybernetics*, Xi'an, China, Nov. 2003.

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

[6] J.-S. Su, B.-F. Zhang, and X. Xu. Advances in machine learning based text categorization. *Jounal of Software*, 17(9):1848–1859, Sep. 2006.

[7] S. Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667–671, 2005.

[8] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of 17th Ann Int ACM SIGIR Conference on Research and Development in Informat,on Retrieval (SIGIR '94)*, pages 13–22, 1994.

[9] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, (1):69–90, 1999.

[10] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer. Improving text categorization methods for event tracking. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval*, pages 65–72, 2000.

[11] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. of the 22nd ACM Intl Conf. on Research and Development in Information Retrieval (SIGIR-99)*, pages 42–49, 1999.