# Pruning Training Corpus to Speedup Text Classification[1]

Jihong Guan[1] and Shuigeng Zhou[2]

[1] School of Computer Science, Wuhan University, Wuhan, 430079, China
jhguan@wtusm.edu.cn

[2] State Key Lab of Software Engineering, Wuhan University, Wuhan, 430072, China
Zhousg@whu.edu.cn

**Abstract**: With the rapid growth of online text information, efficient text classification has become one of the key techniques for organizing and processing text repositories. In this paper, an efficient text classification approach was proposed based on pruning training-corpus. By using the proposed approach, noisy and superfluous documents in training corpuses can be cut off drastically, which leads to substantial classification efficiency improvement. Effective algorithm for training corpus pruning is proposed. Experiments over the commonly used Reuters benchmark are carried out, which validates the effectiveness and efficiency of the proposed approach.

**Keywords**: text classification; fast classification; *k*-nearest neighbor (*k*NN); training-corpus pruning.

## 1 Introduction

As the amount of on-line textual information increases by leaps and bounds, effective retrieval is difficult without support of appropriate indexing and summarization of text content. Text classification is one solution to this problem. By placing documents into different classes according to their respective contents, retrieval can be done by first locating a specific class of documents relevant to the query and then searching the targeted documents within the selected class, which is significantly more efficient and reliable than searching in the whole documents repository.

Text classification has been a hot research topic in machine learning and information retrieval areas, and a number of methods for text classification were proposed [1, 2]. Among the existing methods, *k*NN is the simplest strategy that searches the *k*-nearest training documents to the test document and use the classes assigned to those training documents to decide the class of the test document [3, 4, 5, 6]. *k*NN classification method is easy to implement for it does not require the phase of classifier training that other classification methods must have. Furthermore, experimental researches show that *k*NN method offers promising performance in text

[1] This work was supported by the Natural Science Foundation of China (NSFC) (No. 60173027) and the Provincial Natural Science Foundation of Hubei of China (No. 2001ABB050).

classification [2, 6]. However, the *k*NN method is of low efficiency because it requires a large amount of computational power for evaluating a measure of the similarity between a test document and every training document and for sorting the similarities. Such a drawback makes it unsuitable for some applications where classification efficiency is pressing. For example, on-line text classification where the classifier has to respond to a lot of documents arriving simultaneously in stream format.

Some researchers in IR have addressed the problem of using representative training documents for text classification to improve classification efficiency. In [7] we proposed an algorithm for selecting representative boundary documents to replace the entire training sets so that classification efficiency can be improved. However, [7] didn't provide any criterion on how many boundary documents should be selected and it couldn't guarantee the classification performance. Linear classifiers [8] represent a category with a generalized vector to summarize all training documents in that category; the decision of the assignment of the category can be viewed as considering the similarity between the test document and the generalized vector. Analogously, [9] utilizes the centriod of each class as the only representative of the entire class. A test document is assigned to the class whose centroid is the nearest one to that test document. However, these approaches could not do well in the case that the sizes of different classes are quite different and distribution of training documents in each class is not regular in document space. Combining traditional *k*NN and linear classification methods, [5] uses a set of generalized instances to replace the entire training corpus, classification is based on the set of generalized instances. Experiments show this approach outperforms both traditional *k*NN and linear classification methods.

In this paper, our focus is also on the efficiency of *k*NN based text classification. We provide a robust and controlled way to prune noisy and superfluous documents so that the training corpuses can be significantly condensed while their classification competence is maintained as much as possibly, which leads to greatly improved classification efficiency. We design effective algorithm for text corpus pruning, and carry out experiments over the commonly used Reuters benchmark to validate the effectiveness and efficiency of our proposed approach. Our approach is especially suitable for on-line classification applications.

The rest of this paper is organized as follows. Section 2 introduces the vector space model (VSM), a clustering-based feature selection method and the *k*NN classification method. Section 3 first presents the concepts and algorithm for noisy and superfluous training documents pruning, and then gives a fast *k*NN classification approach based on the proposed training-corpus pruning algorithm. Section 4 describes some experiments for evaluating the proposed approach. Section 5 concludes the paper.

## 2 Preliminaries for *k*NN Based Text Classification

### 2.1 Documents Representation by Vector Space Model (VSM)

In *k*NN based text classification, the vector space model (VSM)[10] is used to represent documents. That is, a document corresponds to an *n*-dimensional document vector. Each dimension of the document vector corresponds to an important term appearing in the training corpus. These terms are also called document features. Given a document vector, its dimensional components indicate the corresponding terms' weights that are related to the importance of these terms in that document. Denote $D$ a training corpus, $V$ the set of document features, $V=\{t_1, t_2, \ldots, t_n\}$. For a document $d$ in $D$, it can be represented by VSM as follows.

$$\vec{d} = (w_1, w_2, \ldots, w_n). \tag{1}$$

Above, $\vec{d}$ indicates the vector of document $d$, $w_i$ ($i$=1~$n$) is the weight of term $t_i$. Usually, the weight is evaluated by *TFIDF* method. A commonly used formula is like this:

$$w_i = \frac{tf_i \times \log(N/n_i)}{\sqrt{\sum_{i=1}^{n}(tf_i)^2[\log(N/n_i)]^2}}. \tag{2}$$

Here, $N$ is the total number of documents in $D$, $tf_i$ is the occurrence frequency of $t_i$ in document $d$, and $n_i$ is the number of documents where $t_i$ appears. Obviously, document vectors calculated by (2) are unit vector. Given two documents $d_i$ and $d_j$, the similarity coefficient between them is measured by the inner product of their corresponding document vectors, *i.e.*,

$$Sim(d_i, d_j) = \vec{d}_i \bullet \vec{d}_j. \tag{3}$$

## 2.2 Clustering Based Feature Selection

To calculate document vectors for training documents, the first step is to select a set of proper document features. A number of statistic methods have been used for document features selection in the literature [11]. However, in this paper we use a new method, which is referred to as *clustering-based feature selection*.

From the point of geometry view, every document is a unit vector in document space ($n$-dimensional space). Basically, documents belong to the same class are closer to each other in document space than those that are not in the same class, that is, they have smaller distance (or larger similarity). Documents in the same class form a dense hyper-cone area in document space, and a training corpus corresponds to a cluster of hyper-cones each of which corresponds to a class. Certainly, different hyper-cones may overlay with each other. Intuitively, the goal of feature selection task here is to select a subset of documents features such that the overlaying among different training classes in the document space is as little as possible.

The basic idea of our clustering-based feature selection method is like this: treating each training class as a distinctive cluster, then using a genetic algorithm to select a subset of document features such that the difference among all clusters is maximized. We define the difference among all clusters as follows.

$$Diff = \frac{1}{m}\sum_{k=1}^{m}\Big(\frac{1}{|C_k|*(|C_k|-1)}\sum_{\substack{d_i, d_j \in C_k \\ i \neq j}} sim(d_{ik}, d_{jk})\Big) - \frac{1}{(\sum_{k=1}^{m} C_k)^2 - \sum_{k=1}^{m}|C_k|^2}\sum_{k=1}^{m}\sum_{\substack{d_i \in C_k \\ d_j \notin C_k}} sim(d_i, d_j). \tag{4}$$

Above, *m* is the number of clusters, the first part indicates the average intra-cluster similarity, and the second part means the average inter-cluster similarity. Due to space limitation, we omit the details of the clustering based feature selection algorithm.

### 2.3 *k*NN Based Text Classification

The *k*NN based text classification approach is quite simple [2]: given a test document, the system finds the *k* nearest neighbors among training documents in the training corpus, and uses the classes of the *k* nearest neighbors to weight class candidates. The similarity score of each nearest neighbor document to the test document is used as the weight of the classes of the neighbor document. If several of *k* nearest neighbors share a class, then the per-neighbor weights of that class are added together, and the resulting weighted sum is used as the likelihood score of that class with respect to the test document. By sorting the scores of candidate classes, a ranked list is obtained for the test document. By thresholding on these scores, binary class assignments are obtained. Formally, the decision rule in *k*NN classification can be written as:

$$score(d,c_i)=\sum_{d_j\in kNN(d)} Sim(\vec{d},\vec{d}_j)\delta(d_j,c_i)-b_i. \tag{5}$$

Above, *k*NN(*d*) indicates the set of *k* nearest neighbors of document *d*; $b_i$ is the class-specific threshold for the binary decisions, it can be automatically learned using cross validation; and $\delta(d_j,c_i)$ is the classification for document $d_j$ with respect to class $c_i$, that is,

$$\delta(d_j,c_i)=\begin{cases}1 & d_j\in c_i;\\ 0 & d_j\notin c_i.\end{cases}$$

Obviously, for a test document *d*, the similarity between *d* and each document in the training corpus must be evaluated before it can be classified. The time complexity of *k*NN classification is $O(n_t|D|\log(|D|))$ where $|D|$ and $n_t$ are the size of training corpus and the number of test documents. To improve classification efficiency, a possible way is to reduce $|D|$, which the goal of this paper.

In this paper, we assume that 1) the class space has flat structure and all classes are semantically disjointed; 2) each document in the training corpus belongs to only one class; 3) each test document can be classified into only one class. With these assumptions, for test document *d*, it should belong to the class that has the highest resulting weighted sum in (5). That is, $d\in c$ only if

$$score(d,c)=\max\{score(d,c_i)\mid i=1-n\}. \tag{6}$$

## 3 Training-Corpus Pruning for Fast Text Classification

Examining the process of *k*NN classification, we can see that outer documents or boundary documents (locating near the boundary) of each class (or document hyper-cone) play more decisively role in classification. On the contrary, inner documents or central documents (locating at the interior area) of each class (or document hyper-cone) are less important as far as *k*NN classification is concerned, because their contribution to classification decision can be obtained from the outer documents. In this sense, inner documents of each class can be seen as superfluous documents. Superfluous documents are just not tell us much about making classification decision, the job they do in informing classification decision can be done by other documents. Except for superfluous documents, there may be some noisy documents in training corpus, which are in-correctly labeled training documents. We seek to discard superfluous and noisy documents to reduce the size of training corpus so that classification efficiency can bee boosted. Meanwhile, we try to guarantee that the pruning of superfluous documents will not cause classification performance (*precision and recall*) degradation.

In the context of *k*NN text classification, for training document *d* in training corpus *D*, there are two sets of documents in *D* that are related to *d* in different way. Documents in one of the two sets are critical to the classification decision on *d* if *d* were a test document; for documents in the other set, *d* can contribute to the classification decisions on these documents if they were treated as test documents. Formal definitions for the two document-sets are as follows.

**Definition 1.** Given document *d* in training corpus *D*, the set of *k* nearest documents to *d* in *D* constitutes the *k-reachability* set of *d*, which is referred to as *k-reachability*(*d*). Formally, *k-reachability* (*d*)=$\{d_i \mid d_i \in D$ and $d_i \in k\mathrm{NN}(d)\}$.

**Definition 2.** Given document *d* in training corpus *D*, there is a set of documents in the same class that *d* belongs to, in which each document's *k-reachability* set contains *d*. We define this set of documents the *k-coverage* set of *d*, or simply *k-coverage* (*d*). Formally, *k-coverage* (*d*)=$\{d_i \mid d_i \in D$ and $d_i \in$ *class*(*d*) and $d \in$ *k-reachability* ($d_i$)$\}$. Here, *class*(*d*) indicates the class to which *d* belongs.

Note that in definition 2, *k-coverage* (*d*) contains only documents from the same class that *d* belongs to. The reason lies in the fact: our aim is to prune training-corpus while maintaining its classification competence. Obviously, pruning *d* may impact negatively the classification decisions on the documents in the same class that *d* belongs to; however, it can benefit the classification decisions on the documents in the other classes. Hence, we need take care only the documents in the same class that *d* belongs to and whose *k-reachability* sets contain *d*.

**Definition 3.** Given document *d* in training corpus *D*, if it could be correctly classified with *k-reachability*(*d*) based on the *k*NN method, in other words, *d* can be implied by *k-reachability*(*d*), then it is a *superfluous document* in *D*.

**Definition 4.** Given document *d* in training corpus *D*, it is a *critical document* if one of the following conditions is fulfilled:

a) at least one document $d_i$ in *k-coverage*(*d*) can not be implied by its *k-reachability*($d_i$);
b) after *d* is pruned from *D*, at least one document $d_i$ in *k-coverage*(*d*) cannot be implied by its *k-reachability*($d_i$).

**Definition 5.** Given document *d* in training corpus *D*, if it is not a superfluous document and its *k-coverage* (*d*) is empty, then it is a *noisy document* in *D*.

In summary, a superfluous document is *superfluous* because its class assignment can be derived from other documents; a critical document is *critical* to other documents because it can contribute to making correct classification decisions about these documents; and a noisy documents is *noise* as far as classification is concerned because it is incorrectly labeled. In *k*NN classification, noise documents must be given up; superfluous documents can be discarded; however, critical documents must be kept in order to maintain training corpus' classification competence. Based on this consideration, we give a rule for training corpus pruning as follows.

**Rule 1.** The rule for training-document pruning.

For document *d* in training corpus *D*, it can be pruned from *D* if

1) it is a noisy document in *D*, or
2) it is a superfluous document, but not a critical document in *D*.

For the second case in Rule 1, the first constraint is the prerequisite for pruning a certain document from the training corpus, while the second constraint is put to guarantee that the pruning of a certain document will not cause degradation of classification competence of the training corpus.

While pruning superfluous documents, it is worthy of pointing out that the order of pruning is also critical because the pruning of one document may impact the decision on whether other documents can be pruned. Intuitively, inner documents of a class in the training corpus should be pruned before outer documents. This strategy can increase the chance of retaining outer documents as many as possible. Otherwise, if outer documents were pruned before inner documents, it would be possible to cause the Domino effect that a lot of documents are pruned from the training corpus, including outer documents, which would degrade greatly the classification competence of the training corpus. Therefore, some rule is necessary to control the order of documents pruning.

Generally speaking, inner documents of a certain class in the training corpus have some common features: 1) inner documents may have more documents of their own class around themselves than outer documents can have; 2) inner documents are closer to the center of their class than the outer documents are; 3) inner documents are further from the documents of other classes than the outer documents are. Based on these observations, we give a rule about superfluous document's pruning priority as follows. Here, we denote *H-k*NN(*d*) the number of documents in *k*NN(*d*) that belongs to the class of *d*; *similarity-c*(*d*) the similarity of document *d* to the center of its own class, and *similarity-ne*(*d*) the similarity of document *d* to the nearest document that does not belong to its own class.

**Rule 2.** The rule for setting priority of pruning superfluous documents.

Given two documents $d_i$, $d_j$ in a class of the training corpus, both $d_i$ and $d_j$ are superfluous documents that can be pruned according to Rule 1.

1) if $H\text{-}k\text{NN}(d_i) > H\text{-}k\text{NN}(d_j)$, then prune $d_i$ before $d_j$;
2) if $\textit{similarity-c}(d_i) > \textit{similarity-c}(d_j)$, then prune $d_i$ before $d_j$;
3) if $\textit{similarity-ne}(d_i) < \textit{similarity-ne}(d_j)$, then prune $d_i$ before $d_j$;

4) if they have similar *H-k*NN, *similarity-c* and *similarity-ne*, then any one can be pruned first;
5) the priority of using *H-k*NN, *similarity-c* and *similarity-ne*: *H-k*NN>*similarity-c*> *similarity-ne*.

Following is an algorithm for training corpus pruning. In algorithm 1, we assume that there is only one class in the training corpus. If there are multiple classes in the training corpus, just carrying out the pruning process in algorithm 1 over one class after another.

**Algorithm 1.** Pruning-training-corpus (*T*: training corpus, *P*: pruned corpus)

```
1)  $P=T$; $S=\Phi$;
2)  for each document $d$ in $T$
3)      compute $k$-reachability($d$); compute $k$-coverage($d$);
4)  for each noisy document $d$ in $T$
5)      $S=S \cup \{d\}$; $T=T-\{d\}$; $P=P-\{d\}$;
6)      for each document $d_i$ in $k$-coverage($d$)
7)        remove $d$ from $k$-reachability($d_i$) and update $k$-reachability($d_i$) in $T$;
8)      for each document $d_i$ in $k$-reachability($d$)
9)        remove $d$ from $k$-coverage ($d_i$);
10) for each document $d$ in $T$ but not in $S$
11)     if $d$ can be pruned and have the highest priority to be pruned, then
12)         $S=S \cup \{d\}$; $P=P-\{d\}$;
13)         for each document $d_i$ in k-coverage($d$)
14)             update k-reachability($d_i$) in $T$.
15) return $P$.
```

Based on the technique of training corpus pruning, a fast algorithm for *k*NN text classification is outlined as follows.

**Algorithm 2.** Fast *k*NN classification based on training documents pruning (outline)

1) Selecting document feature with the proposed clustering based feature selection method; building training document vectors with the selected features;
2) Pruning the training corpus by using algorithm 1;
3) For each test document *d*, calculate its similarity to each training document in the pruned training corpus;
4) Sorting the computed similarities to get *k*NN(*d*);
5) Deciding *d*'s class based on formula (5) and (6).

## 4 Experimental Results

We evaluate the proposed approach by using the Reuters benchmark compiled by Apte *et al.* for their evaluation of the SWAP-1 by removing all of the unlabeled documents from the original Reuters corpus and restricting the categories to have a training-set frequency of at least two [12]. Usually this corpus is simply referred to as

Apte[1]. We do not use the Apte corpus directly, instead we first remove training and test documents that belong to two or more categories, and then select the top 10 categories to form our own compiled Apte corpus. Statistic information of the compiled Apte corpus is listed in Table 1.

**Table 1.** Our complied Apte corpus (*TC-Apte*)

| Category | Number of training docs | Number of test docs |
|---|---|---|
| Acq | 1597 | 750 |
| Coffee | 93 | 25 |
| Crude | 255 | 124 |
| Earn | 2840 | 1170 |
| Interest | 191 | 90 |
| money-fx | 215 | 100 |
| money-supply | 123 | 30 |
| Ship | 111 | 38 |
| Sugar | 97 | 32 |
| Trade | 251 | 88 |
| Total | 5773 | 2447 |

We implemented a prototype with VC++ 6.0 under Windows 2000. Experiments were carried out on a PC with P4 1.4GHz CPU and 256MHz memory. The goal of experiments is to evaluate the performance (effectiveness and efficiency) of our approach. For simplicity, we denote the complied Apte corpus *TC-Apte*. In experiments, *TC-Apte* is pruned first by using our pruning algorithm; the pruned result corpus is denoted as *TC-Apte-pruned*. Classifiers are trained with *TC-Apte* and its corresponding pruned corpus *TC-Apte-pruned*, the trained classifiers' performances are then measured and compared.

Three performance parameters are measured: *precision* (*p*), *recall* (*r*), and *classification speedup* (or simply *speedup*), in which *p* and *r* are used for effectiveness measurements, and *speedup* is used for efficiency improvement measurement of our approach. Here we use the *micro-averaging* method for evaluating performance average across multiple classes. In the context of this paper (*i.e.* each document, either for training or for test, belongs to only one category), *micro-average p* (or simply *micro-p*) and *micro-average r* (or simply *micro-r*) have similar values. In this paper, we use *micro-p*, which can be evaluated as follows:

$$micro-p = \frac{the\ \ number\ \ of\ \ \ correctly\ \ assigned\ \ test\ documents}{the\ \ number\ \ of\ \ \ test\ ocuments}.$$

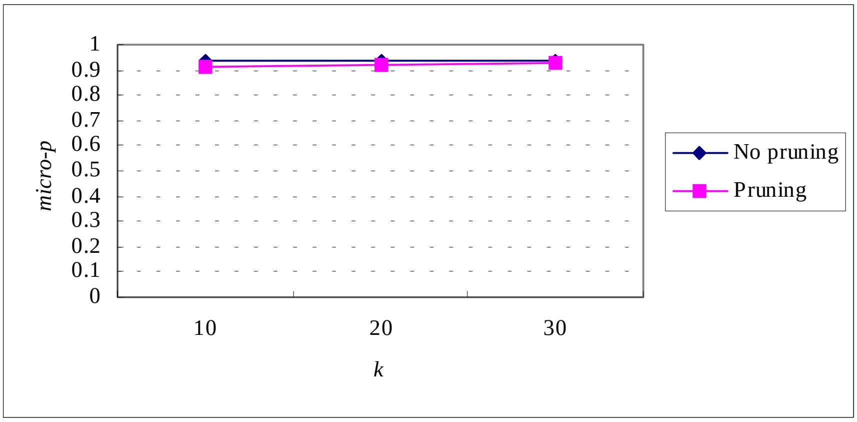
We define efficiency *speedup* in the following formula:

$$speedup = \frac{t_{TC-Apte}}{t_{TC-Apte-pruned}}.$$

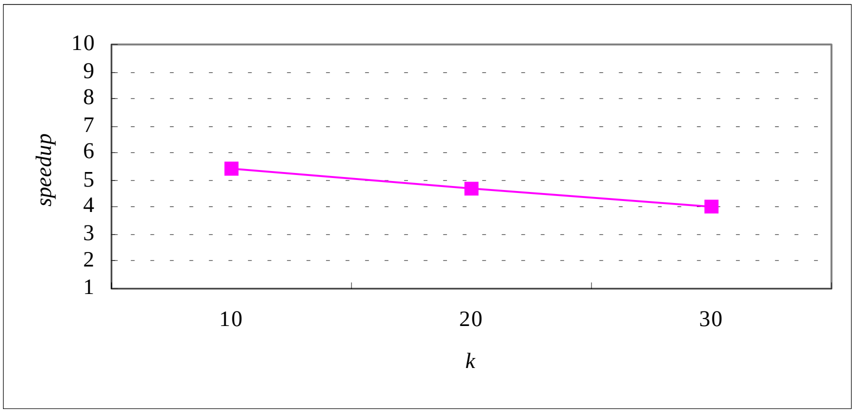
Above, $t_{\text{TC-Apte}}$ and $t_{\text{TC-Apte-pruned}}$ are the time cost for classifying a test document (or a set of test documents) based on *TC-Apte* and *TC-Apte-pruned* respectively.

[1] Apte corpus is available at: http://moscow.mt.cs.cmu.edu:8081/reuters_21450/apte

Due to space limitation, here we give only partial experimental results. Fig. 1 illustrates the results of the impact of *k* value on pruning effectiveness and classification performance over *TC-Apte*. From Fig.1, we can see that by using our corpus-pruning technology, classification efficiency can get improved at a factor of larger than 4, with less than 3% degradation of micro-averaging performance. Obviously, this result is acceptable.



(a) *k* value vs. *micro-p*



(b) *k* value vs. *speedup*

**Fig. 1.** Impact of *k* value on pruning effectiveness and classification performance

## 5 Conclusion

The rapid growth of text information available arises the requirement of efficient text classification method. Although *k*NN based text classification is a good method as far as performance is concerned, it is inefficient for it has to calculate the similarity of the test document to each training document in the training corpus. In this paper, we propose a training-corpus pruning based approach to speedup the *k*NN method. By using our approach, the size of training corpus can be reduced significantly while classification performance can be kept at a level close to that of without training documents pruning. Experimental results validate the efficiency and effective of the proposed approach.

## References

1. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1): 1-47, 2002
2. Y. Yang and X. Liu. A re-examination of text categorization. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), 1999.
3. B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory-based reasoning. Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92), 1992, 59-65.
4. Y. Yang. Expert network: effective and efficient learning from human decisions in text categorization and retrieval. Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94), 1994, 13-22.
5. W. Lam and C. Y. Ho. Using a generalized instance set for automatic text categorization. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98), 1998, 81-89.
6. S. Zhou, J. Guan. Chinese documents classification based on N-grams. A. Gelbukh (Ed.): Intelligent Text Processing and Computational Linguistics, LNCS, Vol. 2276, Spring-Verlag, 2002, 405-414.
7. S. Zhou. Key Techniques of Chinese Text Database. PhD thesis of Fudan University, China. 2000.
8. D. D. Lewis, R. E. Schapore, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96), 1996, 298-306.
9. E. H. Han and G. Karypis. Centroid-based document classification algorithm: analysis & experimental results. Technical Report TR-00-017, Dept. of CS, Uni. of Minnesota, Minneapolis, 2000. http://www.cs.umn.edu/~karypis
10. G. Salton, A. Wong, and C. S. Yang. A vector space model got automatic indexing. K. S. Jones and P. Willett (Eds.), Readings in Information Retrieval. Morgan Kaufmann, 1997. 273-280.
11. Yang, Y., Pedersen J.P. A Comparative Study on Feature Selection in Text Categorization Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997.
12. C. Apte, F. Damerau, and S. Weiss. Text mining with decision rules and decision trees. Proceedings of the Conference on Automated Learning and Discovery, Workshop 6: Learning from Text and the Web, 1998.