Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 793490, 24 pages
doi:10.1155/2012/793490

Research Article

Multiclass Boosting with Adaptive Group-Based *k*NN and Its Application in Text Categorization

Lei La, Qiao Guo, Dequan Yang, and Qimin Cao

School of Automation, Beijing Institute of Technology, Beijing 100081, China

Correspondence should be addressed to Lei La, lalei1984@yahoo.com.cn

Received 31 December 2011; Revised 30 March 2012; Accepted 26 April 2012

Academic Editor: Serge Prudhomme

AdaBoost is an excellent committee-based tool for classification. However, its effectiveness and efficiency in multiclass categorization face the challenges from methods based on support vector machine (SVM), neural networks (NN), naïve Bayes, and *k*-nearest neighbor (*k*NN). This paper uses a novel multi-class AdaBoost algorithm to avoid reducing the multi-class classification problem to multiple two-class classification problems. This novel method is more effective. In addition, it keeps the accuracy advantage of existing AdaBoost. An adaptive group-based *k*NN method is proposed in this paper to build more accurate weak classifiers and in this way control the number of basis classifiers in an acceptable range. To further enhance the performance, weak classifiers are combined into a strong classifier through a double iterative weighted way and construct an adaptive group-based *k*NN boosting algorithm (AG*k*NN-AdaBoost). We implement AG*k*NN-AdaBoost in a Chinese text categorization system. Experimental results showed that the classification algorithm proposed in this paper has better performance both in precision and recall than many other text categorization methods including traditional AdaBoost. In addition, the processing speed is significantly enhanced than original AdaBoost and many other classic categorization algorithms.

1. Introduction

Machine learning- (ML-) based text categorization (TC) can be defined similar with other data classification tasks as the problem of approximating an unknown category assignment function $F : D \times C \rightarrow \{0,1\}$, where D is the set of all possible documents and C is the set of predefined categories [1]:

$$F(d,c) = \begin{cases} 1, & d \in D \text{ and } d \text{ belong to the class } c \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

Figure 1: Flow chart of text categorization.

The approximating function $M : D \times C \rightarrow \{0, 1\}$ is called a classifier, and the task is to build a classifier that produces results as "close" as possible to the true category assignment function F [2], for instance, whether an article belongs to fiction, whether a short message belongs to advertisement, or whether the author of a script is Shakespeare and so forth.

In text categorization projects, documents usually need be preprocessed to select suitable features. Then the document will be represented by their features. After the above steps, classifier will determine the category of the document. The flow chart of a TC task is shown in Figure 1.

In different tasks preprocessing contains some or all of the following aspects: transform unstructured document into structured or semistructured format, word segmentation, and text feature selection. Feature selection is the most important part of preprocessing [3]. Features can be characters, words, phrases, concepts, and so forth [4]. Document representation is the process of using features with different weights to show texts. Classifier's kernel is machine learning algorithm. It uses the document representation as its input and then outputs the categorization results.

Imagine an international IT corporation which is interested in job seekers' Java programming experience and English ability. The resume screening program of this company is actually a TC system. It can assist the managers choose appropriate employees. The system is shown in Figure 2.

Researchers made considerable achievements in the design of categorization algorithms because classifier is the key point in TC systems [5]. Several of the most important methods include naïve Bayes, support vector machine (SVM), k-nearest neighbor (kNN), decision tree (DT), neural networks (NN), and voting-based algorithms such as AdaBoost. Some comparative experiments revealed that SVM, kNN, and AdaBoost have the best precision, Naïve Bayes has the worst performance but very useful as baseline classifiers because of its ease of use. Performance of DT and NN are worse than the top 3 methods but the computational complexity is also lower [6–8].

In a word, the purpose of classifier design and research in TC is to improve the performance and maintain the balance between performance and cost.

The rest of this paper is organized as follows. Section 2 reviews related work and analyzes the goal of this paper. Section 3 improves classic kNN to build weak classifiers based on it. In Section 4, a double iterative weighted cascading algorithm is proposed to construct a strong classifier. Section 5 then modified the AdaBoost based on Sections 3 and 4 to solve multiclass problems. The application of the novel classification algorithm is presented and analyzed in Section 6. Finally, Section 7 summarizes the paper.

2. Related Work and Motivation

Voting-based categorization algorithms also known as classifier committees can adjust the number and professional level of "experts" in the committees to find a balance between performance and time-computational consumption. These algorithms give up the effort to build single powerful classifier but try to integrate views of many weak classifiers. The

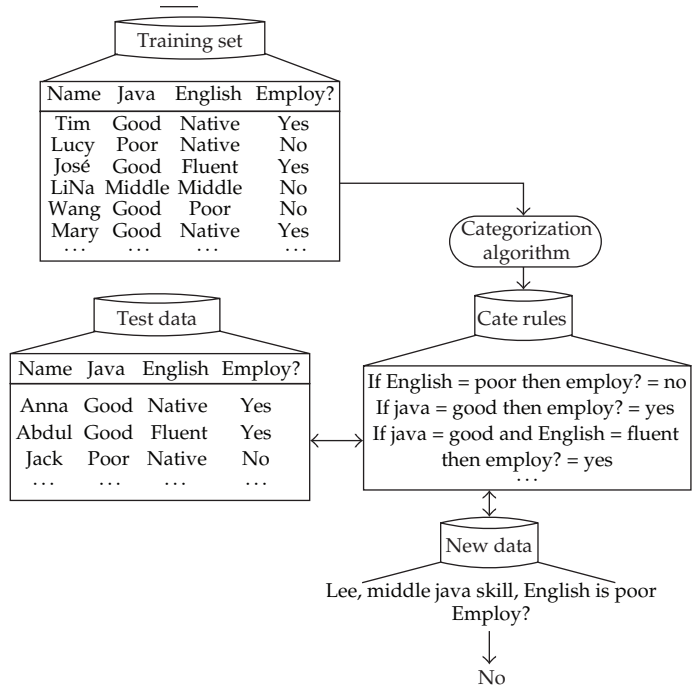


Figure 2: An instance of TC system.

philosophical principle of this methodology is *the truth always held in majority*. Bagging and boosting are the two kinds of most popular voting-based methods.

2.1. Boosting Algorithm

Unlike bagging method which trains the classifiers in parallel, in boosting the classifiers are trained sequentially. Before training the next classifier, the training set is reweighed for allocating greater weight to the documents that were misclassified by the previous classifiers [9]. Therefore, the system can pay serious attention on controversial texts and enhance the precision.

The original boosting algorithm uses three weak classifiers (c_1, c_2, c_3) to form a committee. It divides a large training set into three parts (X_1, X_2, X_3) randomly and use X_1 to train c_1 firstly. Then it uses the subset of X_1 which is misclassified by c_1 and the subset which is categorized rightly by c_1 together as the training set of c_2. The rest can be done in the same manner.

Scholars are committed to enhance the performance and reduce the overhead so a lot of improved boosting algorithm such as BrownBoost, LPBoost, LogitBoost, and AdaBoost were proposed. Majority comparative literatures proved that AdaBoost has the best performance among them [10].

2.2. Detail of AdaBoost

Boosting and its relative algorithms get big success in several practices such as image processing, audio classification, and optical characters recognition (OCR). At the same time,

boosting needs huge training sets, and thus sometimes the runtime consumption become unacceptable. Moreover, the weak classifiers' lower limit of accuracy needs to be predicted.

To control the computational cost in a reasonable range, Shapire and Singer [11] proposed AdaBoost. It uses a dual-weighted process to choose training sets and classifiers. The detailed steps of AdaBoost are as follows:

(1) Given training set $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where x_i is the training sample and $y_i \in \{1, -1\}$ denotes x_i's category label $(1 \le i \le n)$.

(2) Let $f_j(x_i)$ denote ith feature of jth document.

(3) Define the initial distribution of documents in the training set $D_I(i) = 1/N$.

(4) Searching weak classifier c_t $(t = 1, 2, \ldots, T)$: for jth feature of every sample, a weak classifier c_j can be obtained and thus get the threshold θ_j and orientation P_j to minimum the error ε_j as follows:

$$\varepsilon_j = \sum_{j=1}^{n} D_i(x_i) \left| c_j(x_i) \neq y_i \right|. \tag{2.1}$$

Therefore, the weak classifier c_j is

$$c_j(x) = \begin{cases} 1 & P_j f_j(x) < P_j \theta_j \\ -1 & \text{otherwise.} \end{cases} \tag{2.2}$$

(5) Choose c_j from the whole feature space which has the minimal error ε_j as the weak classifier.

(6) Recalculate the feature of samples:

$$D_{t+1}(i) = \frac{D_t(i) e^{(-\alpha_i y_i c_i(x_i))}}{Z_t}, \tag{2.3}$$

where Z_t is a normalization factor which makes $\sum_{i=1}^{n} D_{t+1}(i) = 1$ and α_i is the weight.

(7) Repeat the steps above T times and get T optimal weak classifiers with different weights.

(8) Combine weak classifiers according to their weight to construct a strong classifier:

$$C_{\text{strong}}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t c_t(x) \right). \tag{2.4}$$

Training set utilization can be enhanced using the algorithm above through adjusting the weights of misclassified texts [12]. In addition, the performance of strong classifier is improved because it is constructed in a weighted way [13]. In a word, AdaBoost has lower training consumption and higher accuracy than original boosting algorithms.

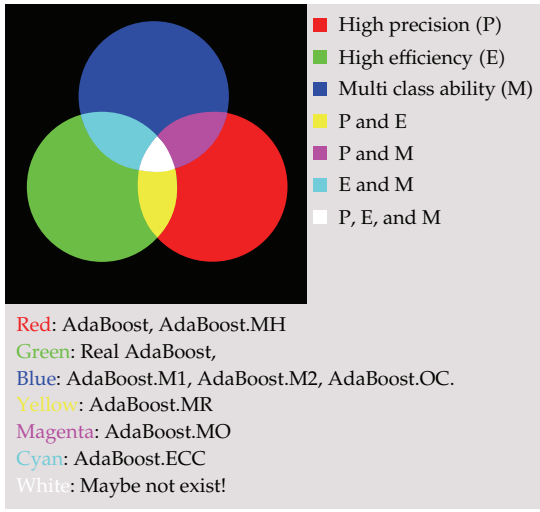


Figure 3: Performances of AdaBoost family members.

Researchers proposed some variants of AdaBoost focusing on different aspects such as precision, recall, robustness, computational overhead and multiclass categorization [14]. We called these algorithms AdaBoost family. The three most important indicators are precision, efficiency, and the ability of multiclass categorization. Performances of AdaBoost family members are shown in Figure 3 [15].

2.3. Problems of AdaBoost Family and Motivation of This Article

Figure 3 reveals that few algorithms in AdaBoost family can achieve high precision and high efficiency at the same time specifically in multiclass-oriented categorization problems. Unfortunately, multiclass is the main problem in classification tasks. Traditional methods which translate the multiclass problem into multiple two-class problems will reduce accuracy and increase complexity of the system [16].

To solve problems above, we design weak classifiers with high accuracy and low complexity to limit the number of experts and thus keep the precision while reduce the consumption. More professional expert should play a more important rule, and misclassified documents should attract greater attention to further improve system's performance. Therefore, more reasonable rules should be made to combine weak classifiers into strong classifier efficiently. In addition, this strong classifier should be used in multiclass classification tasks directly. Above is the motivation and purpose of this paper.

3. Weak Classifiers with AG*k*NN

Theoretically once weak classifiers are more accurate than guess randomly (1/2 in two-class tasks or $1/n$ in multiclass tasks), AdaBoost can integrate them into a strong classifier whose precision can infinitely be close to the true category distribution [17]. However, when the precision of weak classifiers are lower, more weak classifiers are needed to construct a strong classifier. Too many weak classifiers in the system sometimes increase its complexity and

computational consumption to intolerable level. Categorization systems use naïve Bayes or C4.5 as their weak classifiers may face this problem.

Some researchers tried to design weak classifiers based on more powerful algorithms such as neural networks [18] and support vector machine [19]. These algorithms can certainly achieve higher accuracy but lead to some new problems because they are over complex and thus contrary to the ideology of boosting.

3.1. k-Nearest Neighbor

Example-based classification algorithms keep a balance between performance and cost [20]. k-nearest neighbor is the most popular example-based algorithm as it has higher precision and lower complexity.

To make the classification, kNN transforms the target documents into representational feature vectors which have same formation with training samples. Then it calculates distance between the target document and the selected k neighbors [21]. Finally the category of target document is determined according to their neighbors' class. The schematic of two-class kNN is shown in Figure 4.

The distance between two documents is calculated by a distance function:

$$\mathrm{Sin}(d_i, d_j) \frac{\sum_{k=1}^{M} W_{ik} \times W_{jk}}{\sqrt{\left(\sum_{k=1}^{M} W_{ik}^2\right)\left(\sum_{k=1}^{M} W_{jk}^2\right)}}. \tag{3.1}$$

As shown, the above function calculates the Euclidean distance between two documents in a linear space. Choose nearest k neighbors as the reference documents, then the category C_j which includes most neighbors can be found as

$$p(\overline{x}, C_j) = \sum_{\overline{d_i} \in k\mathrm{NN}} \mathrm{Sim}\left(\overline{x}, \overline{d_i}\right) y\left(\overline{d_i}, C_j\right), \tag{3.2}$$

where $\overline{d_i}$ is the ith training document, $\mathrm{Sim}(a, b)$ is the similarity of document a, and document b, $y(\alpha, \beta)$ represent the probability of document α belong to category β.

3.2. Adaptive Group-Based kNN Categorization

Two main problems in traditional kNN are experience dependent and sample category balance. Experience dependent means k is an empirical value that need be preset [22]. Sample category balance notes that when the numbers of samples belonging to different categories have large gap, the classification results tend to be inaccurate. In other words, the system expects the category distribution of the samples as even as possible.

An adaptive group-based kNN (AGkNN) algorithm is proposed in this paper to solve problems above. The basic idea of AGkNN is to divide the training set into multiple groups. Use kNN to categorizing target documents parallel in each subset with a random initial value of k, and then compare classifying results. If the results of different groups are broadly consistent with each other, keep the group size and k's value. When the results are highly similar to each other, emerge groups and reduce the value of k. If the results are totally

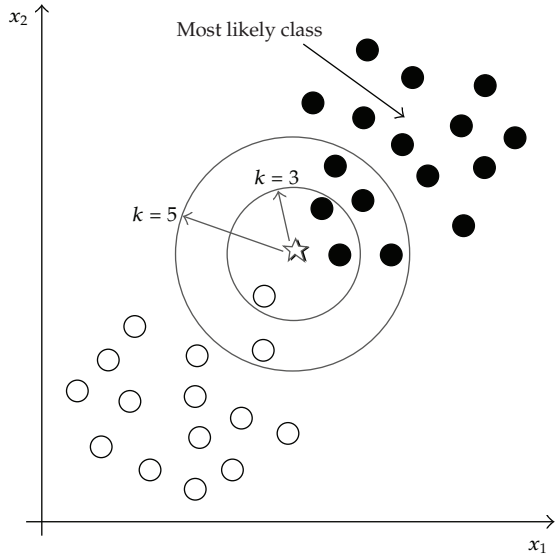


Figure 4: Schematic of two-class kNN.

different from each other, increase the value of k. Especially when two opposing views are counterbalanced, the system should increase the numbers of groups to make final decision.

Define $N_g(i)$ as the number of training groups when processing ith document. Give every category a number as its class name. For example, define a news reports set {financial news, sports news, entertainment news, political news, weather report} as $\{1,2,3,4,5\}$, $C_j(i)$ as the categorization result of ith document determined by jth group—for example $C_4(5)$ means in the forth group's opinion, the document which be classified is a weather report—and $\overline{C}$ as the average value of different categories calculated by feature distance in different groups. For instance, a document is classified as "sports news, sports news, entertainment news, entertainment news and sports news" in 5 training groups; the average value of categories $\overline{C}$ can be calculated as $\overline{C} = (2 + 2 + 3 + 3 + 2)/5 = 2.4$. The number of groups can be adaptively determined as

$$N_g(i+1) = \begin{cases} N_g + \operatorname{int}\left(\dfrac{1}{N_g}\displaystyle\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2\right), & \displaystyle\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2 > \overline{C} \\ N_g, & \dfrac{1}{\overline{\overline{C}}} < \displaystyle\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2 \le \overline{C} \\ N_g - \operatorname{int}\left(\dfrac{1}{\overline{\overline{C}}}\right), & \displaystyle\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2 \le \dfrac{1}{\overline{\overline{C}}}. \end{cases} \tag{3.3}$$

According to (3.3), the system can determine whether and how to adjust the grouping situation of samples by making reference to the variance of classification results given by different groups. When the variance of result given by each group is higher than the threshold, it means the categorization is not accurate enough because argument exists and more groups are needed to make a final decision. On the other hand, when the variance is

very low, it means there are almost no disputes in classification and the sample groups can be merged to saving time consumption. In this paper, we use $1/\overline{C}$ and $\overline{C}$ as the lower bound and higher bound because the average value of categories is empirically suitable and convenient to be used as threshold.

The value of k can be calculated adaptively as:

$$k = \begin{cases} \gamma_i, & \frac{1}{N_g}\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2 < 1 \\ \gamma_i + \operatorname{int}\left(\frac{1}{n}\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2\right), & \frac{1}{N_g}\sum_{j=1}^{N_g}\left(C_j(i) - \overline{C}\right)^2 \geq 1, \end{cases} \tag{3.4}$$

where γ_i is the random initial value of k. The system can test if the random initial k is suitable. It can judge whether majority classifier reached agreement according to the variance to inferring if the categorization result is precise enough or not, moreover, to adjust the value of k adaptively to get a more accurate result.

The detail work steps are shown in Figure 5.

In this way, the algorithm can set value of k adaptively and take full use of training set because the training set is grouped automatically and the value of k is initialed randomly. Furthermore, the system can adjust the number of groups and reference neighbors adaptively by calculate and update variance of categorization results given by different groups in real time. There is no condition missing in the algorithm; in addition, the core of the algorithm is still kNN algorithm whose convergence had been proofed in [23], so the AGkNN converges. The runtime complexity of solving the variance of n elements is $3n - 1$, so the computational complexity T of one document classification in this algorithm is

$$T = N_g \cdot k \cdot (3N_g - 1), \tag{3.5}$$

Therefore the main problems which limit the effectiveness of original kNN for a long time are eliminated in AGkNN. Experience-dependent problem be solved means the algorithm can achieve higher efficiency and robustness. Overcoming the drawback of need category balance means the system can improve its accuracy. In summary, AGkNN has better performance and lower complexity than classic kNN. It is wonderful as the weak classifier in AdaBoost.

3.3. Generating Weak Classifiers

Weak classifier design is critical for differentiating positive samples and negative samples in training set. The precision of weak classifiers must be better than 50% to ensure the convergence of strong classifier. Therefore, the threshold θ needs to be set for weak classifiers to guarantee the system performance by combining them into a more powerful strong classifier [24].

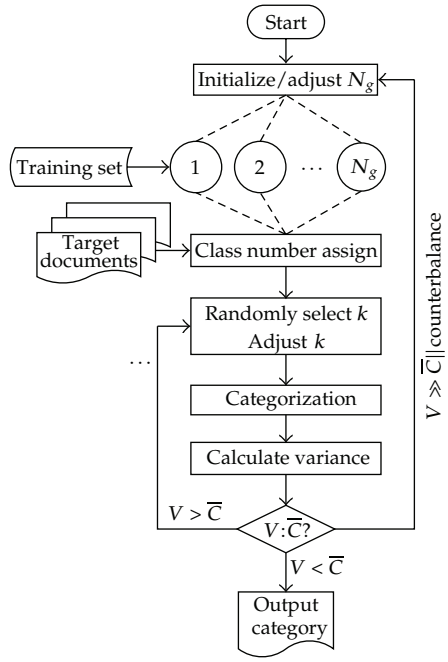


Figure 5: Detail work steps of AGkNN.

Define the weight of positive document x as w_p, the weight of negative document y as w_n, the positive threshold as θ_p, and the negative threshold as θ_n. The threshold can be calculated as

$$\begin{aligned} \theta_p &= \min\left(\max\left(w_p(x)\right) - w_p(x) + w_n(x)\right), \\ \theta_n &= \min\left(\max\left(w_n(x)\right) - w_n(x) + w_p(x)\right), \\ \theta &= \begin{cases} \theta_p, & \theta_p \le \theta_n \\ \theta_n, & \text{otherwise.} \end{cases} \end{aligned} \tag{3.6}$$

Accuracy of weak classifiers can be maintained above 0.5 by introducing and updating the threshold θ. Therefore, weak classifiers based on AGkNN can be generated by following the steps below.

(1) Calculate threshold θ.

(2) Call AGkNN for categorization.

(3) Calculate the classification error e.

(4) Randomly choose and compare e with θ, **if** $e > \theta$ **go to** step (2), **else**, continue.

(5) Update the threshold θ.

(6) **End**.

4. DIWC Algorithm: A Tool for Constructing Strong Classifier

Whether strong classifier has a good performance depends largely on how weak classifiers are combined. To build a powerful strong classifier, basis classifiers which have higher precision must take more responsibility in categorization process. Therefore categorization system should distinguish between the performances of weak classifiers and give them different weights according to their capabilities. Using these weights, boosting algorithms can integrate weak classifiers as the strong classifier in a more efficient way and achieve excellent performance [25].

4.1. Review Weighting Mechanism in Original AdaBoost

Original AdaBoost algorithm uses a linear weighting way to generate strong classifier. In AdaBoost, strong classifier is defined as:

$$\begin{aligned} f(x) &= \sum_{t=1}^{T} \alpha_t h_t(x), \\ H(x) &= \operatorname{sign}\big(f(x)\big), \end{aligned} \tag{4.1}$$

where $h_t(x)$ is a basis classifier, α_t is a coefficient, and $H(x)$ is the final strong classifier.

Given the training documents and category labels $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, $x_i \in X$, and $y_i = \pm 1$. The strong classifier can be constructed as [26]

Initialize weight $D_1(i) = 1/m$, for $t = 1, 2, \ldots, T$.

(1) Select a weak classifier with the smallest weighted error:

$$h_t = \arg\min_{h_j \in H} \varepsilon_j = \sum_{i=1}^{m} D_t(i)\big(y_i \neq h_j(x_i)\big), \tag{4.2}$$

where ε_j is the error rate.

(2) Prerequisite: $\varepsilon_t < 1/2$, otherwise stop.

(3) Upper bounded ε_t by $\varepsilon_t(H) \leq \prod_{t=1}^{T} Z_t$, where Z_t is a normalization factor.

(4) Select α_t to greedily minimize $Z_t(\alpha)$ in each step.

(5) Optimizing $\alpha_t = (1/2)\log((1 + r_t)/(1 - r_t))$, where $r_t = \sum_{i=1}^{m} D_t(i)h_t(x_i)y_i$ by using the constraint $Z_t = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \leq 1$.

(5) Reweighting as

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)\exp\big(-\alpha_t y_i h_t(x_i)\big)}{Z_t} = \frac{\exp\Big(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i)\Big)}{m\prod_{q=1}^{t} Z_q}, \\ \exp\big(-\alpha_t y_i h_t(x_i)\big) &\begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i). \end{cases} \end{aligned} \tag{4.3}$$

The above-mentioned steps demonstrated that AdaBoost gives classifiers which have better classification performance higher weights automatically, especially by step (5). In this

way, AdaBoost can be implemented simply. Its feature selection is on a large set of features. Furthermore, it has good generalization ability.

However, this weighting algorithm does not check the precision of former classifiers using the later training documents. In other words, the strong classifier generation is a single iterative process. Weak classifiers probably have different performances in different training samples. The weak classifiers which are considered should get higher weights by AdaBoost actually have better performance in former part of the training set. However, the basis classifiers may have good performance in later part but be ignored unreasonably. Therefore, credibility of weights is decreasing with the test sequence. This phenomenon can be called *weight bias*. Weight bias could lead to suboptimal solution problem and make the system oversensitive to noise. Accuracy is affected by the above problems and the robustness of system is decreased.

To overcome these drawbacks, boosting algorithm should use a double iterative process for allocating weights to basis classifiers more reasonable.

4.2. Double Iterative Weighted Cascading Algorithm

In AdaBoost, weak classifiers with higher weights certainly can correctly process the documents which were misclassified by lower weights classifiers. It is important but not enough to improve categorization performance—two crucial problems are ignored as follows.

(1) Could basis classifiers with higher weight classify samples which already are rightly categorized by classifiers with lower weight in a high accuracy?

(2) If weak classifiers with lower weights also do not have the power to process documents which are misclassified by high-weight classifiers?

Weights' credibility is reduced when the answers of these two problems are not absolutely positive. Therefore it is worth introducing the problems-aforementioned into weak classifiers weights allocation.

This paper proposed a double iterative weighted cascading (DIWC) algorithm to solve the two problems above and make the utilization of basis classifiers more efficient. The kernel ideal of DIWC is adding a weighting process by input training samples in reverse order. Comparing with original AdaBoost algorithm, we can call this process double iterative. Using the average weight of a basis classifier calculated in the two weighting process as the final weight. Introducing the average weight of two iterations to replace the weight using in traditional AdaBoost can avoid the weight bias problem because it takes the two problems above into account. It defines "powerful" for basis classifiers by using not only the former part but also the full training samples. The sketchy procedure chart of DIWC is shown in Figure 6.

DIWC can achieve weighting procedure shown in Figure 6 by the following steps below.

(1) **Start**: initialize documents weights $w_d(i)$ and weak classifier weights $w_c(j)$.

(2) Training the first classifier c_1 with first sample documents subset s_1, mark the set of documents which is misclassified by c_1 in s_1 as e_1.

(3) **Loop**: training c_i with s_i and e_{i-1}

(4) Calculation: calculating weights of basis classifiers according to the first round of loops (trainings).

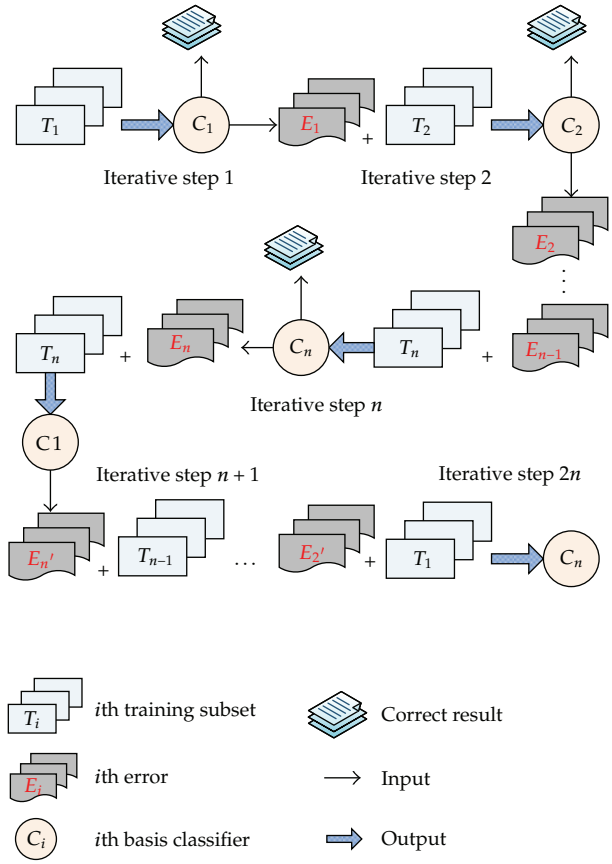


Figure 6: Procedure of DIWC algorithm.

(5) Reverse iterative: training c_1 with s_i and e_i.

(6) **Loop**: training c_i with s_i and e'_{i+1}.

(7) Calculation: calculating weights of basis classifiers according to the second round of loops (trainings).

(8) Calculate final weights of basis classifiers according to step (4) and step (7)

(9) Cascade: combine basis classifiers according to their final weights and construct strong classifier.

(10) **End**.

There are three methods that can be used, respectively, to calculate the final weights with different accuracy and complexity.

The first method is quite simple: calculate the arithmetic mean of weights in two iterative loops and use it as weak classifiers' final weights. This method has a very low computational cost. In this paper, it is called DIWC-1.

Note that some basis classifiers may have a very high weight both in the first and the second rounds of loops. It means these classifiers have global high categorization ability and should play a more important role in classification process instead of using the average weight simply. In this case, an upper bound value is set as the final weight of significantly

powerful classifiers. On the other hand, some classifiers may have a very low weight in both two iterative loops. The utility of these classifiers must be limited by using a lower bound value to enhance system's accuracy. This method spends more time on computing but has higher precision. It is called DIWC-2.

The third method concerns the situation that some weak classifiers may have a very high weight in one round of loops but a very low weight in another round of loops. One more iterative process is needed to determine the final weight. Especially, if the weights' variance of three rounds is significantly large, the system will consider the weak classifiers as noise oversensitive and deduce its weight. This method can achieve the best precision and robustness. However its training consumption is also highest. We call it DIWC-3 in this paper.

The computational complexity of DIWC-1, DIWC-2, and DIWC-3 can be calculated by referring (3.5). Set m as the number of documents would be classified. The runtime complexity T_1 of DIWC-1 is quite simple as

$$T_1 = 2m * N_g * k * (3N_g - 1). \tag{4.4}$$

In DIWC-2, weights of two iterative processes will be compared, an upper bound σ_h will be introduced when classifiers have a very high weight both in the first and the second rounds of loops, and a lower bound σ_l will be introduced when classifiers have a very low weight both in the first and the second rounds of loops. Because not every basis classifier needs an upper bound/lower bound and introduces bounds leading to extra computational consumption, so the runtime complexity T_2 ranges in

$$3m * N_g * k * (3N_g - 1) \leq T_2 \leq 4m * N_g * k * (3N_g - 1). \tag{4.5}$$

The DIWC-3 considers not only upper bound and lower bound but also the difference between weights in the two iterative loops. When the weights determined in the two loops have big difference, a third loop may be needed for final decision making. Similar to DIWC-2, the range of runtime complexity T_3 can be described as

$$3m * N_g * k * (3N_g - 1) \leq T_2 \leq 6m * N_g * k * (3N_g - 1). \tag{4.6}$$

As the analysis above, the computational complexity is proportional to k, m, and N_g^2; when the number of classification objects increases, the time consumption will increase linearly. Therefore the algorithms avoid index explosion problem and have an acceptable runtime complexity. In addition, the algorithms are converged because no condition is missing and the values of weights are infinity.

4.3. Using Training Sets More Efficiently

As per the review in Section 4.1, traditional AdaBoost gives documents which is misclassified by former weak classifiers with higher importance to improve the system's ability to categorize "difficult" documents. This ideal is helpful for making AdaBoost achieves better precision than former boosting algorithms. However, AdaBoost still leaves space for improving the efficiency of using training documents.

Actually, all the training documents which are categorized incorrectly should be gathered into an error set and use it to train every basis classifier. The accuracy will further

progressing by using training documents in this way. The implementation of this method is quite convenient. Integrating this method with DIWC-1, DIWC-2, and DIWC-3 constructs the complete double iterative weighted cascade algorithm. The pseudocode of DIWC is shown in Algorithm 1 where e_i is the error set of the ith basis classifier, ω_i^1 is the weight of the ith classifier in the first iterative loop, ω_i^2 is the weight of the ith classifier in the second iterative loop, ε is the lower threshold of the difference between ω_i^1 and ω_i^2, σ_h is the upper threshold of weight, W_{MAX} is the upper bound, σ_l is the lower threshold of weight, W_{MIN} is the lower bound, δ is the upper threshold of the difference between ω_i^1 and ω_i^2, and ω_i' is the final weight of the ith classifier.

5. Multiclass Classification

Majority members of AdaBoost family are oriented to two-class classification tasks. When solving multiclass problem, they often transform it into multiple two-class problems. These algorithms tend to have shortcomings in accuracy or efficiency and difficulty to achieve perfection when faced to multiclass categorization tasks. However, multiclass is a main problem in classification tasks. In many occasions simply using two-valued logic as yes or no can or cannot be hard to satisfy the requirements of categorization tasks. For instance, a news report may belong to politics, economics, sports, culture, new scientific discovery, or entertainment. In other words, processing multiclass classification tasks with higher performance should be the most important purpose of the boosting-based algorithm development.

5.1. kNN in Multiclass Classification

As per the kernel algorithm of weak classifiers, k-nearest neighbor has a nature advantage to solve multiclass problems. The mathematical expression of kNN is

$$p(d_i, C_j) = \sum_{d \in k\mathrm{NN}} \mathrm{Sim}(d_i, d_j) y(d_j, C_j). \tag{5.1}$$

The above function reveals that kNN algorithm can easily be used in multiclass classification problems, because unlike other categorization algorithms, kNN does not divide the problem into two subspaces or two subparts, but it records the class label C_j directly. Therefore, it need not to be premodified much to satisfy the multiclass categorization problem.

Traditional text categorization research often use the Euclidean distance or the Manhattan distance to measure the similarity between samples. However, when faced to multiclass categorization problems, these distance definitions cannot distinguish the importance between weights effectively [27]. To solve this problem, the Mahalanobis distance is used in this paper:

$$D(X) = \sqrt{(X - \mu)^T S^{-1} (X - \mu)}. \tag{5.2}$$

And the distance between vector X_i and X_j is defined as

$$D(X_i, X_j) = \sqrt{(X_i - X_j)^T S^{-1} (X_i - X_j)}. \tag{5.3}$$

Input: training set $S = \{s_1, s_2, \ldots, s_m\}$ and weak classifier $C = \{c_1, c_2, \ldots, c_n\}$
OutPut: strong classifier H

```
1  begin
2      test c_1 with s_1
3      for (i = 2; i ≤ m; i++)
4          for (j = 2; j ≤ n; j++)
5              test c_j with s, and e_{j^2-1}
6      calculate W_j^1,
7      test c_1 with s_m and e_m
8      for (i = m − 1; i ≥ 1; i−−)
9          for (j = 2; j ≤ n; j++)
10             test c_j with s_{i−1} and e_i^2
11     calculate w_i^2
12     for (j = 1; j ≤ m; j++)
13         if |w_i^1 − w_i^2| ≤ ε&&w_i^1, w_i^2 ≥ σ_h
14             w_i' = w_MAX
15         else if |w_i^1 − w_i^2| ≤ ε&&w_i^1, w_i^2 ≤ σ_1
16             w_i' = w_MIN
17         else if ε ≤ |w_i^1 − w_i^2| ≤ δ
18             W_i' = (w_i^1 − w_i^2)/2
19         else
20             int t = 0; t++
21             while (t ≤ 1 )
22                 goto 2
23             |w_i^1 − w_i^2| ≤ δ? w_i' = (w_i^1 − w_i^2)/2 : w_i' = w_MIN
24 end
```

Algorithm 1: Pseudocode of DIWC Double iterative weighted cascading.

In this way, the importance between weights can be distinguished effectively [28]. Because kNN can be easily used in multiclass situation, we can construct strong classifier without big modification of the basis classifier itself.

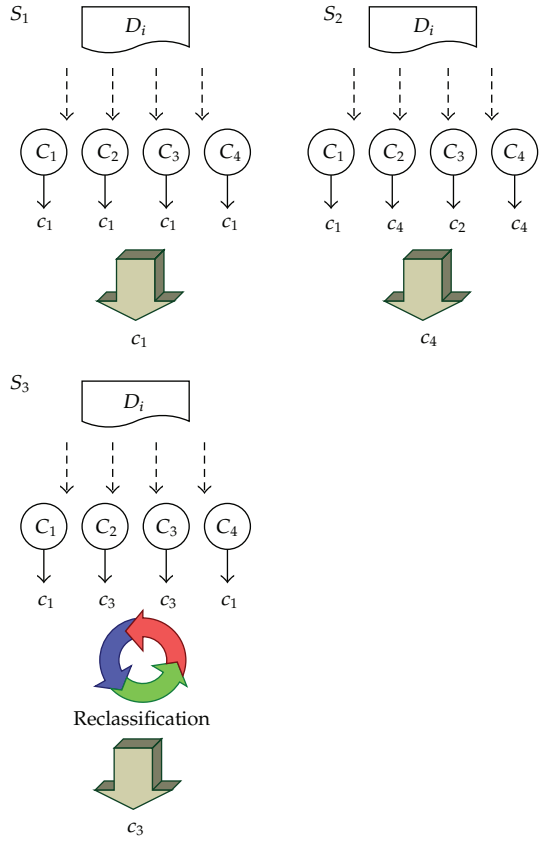
5.2. Integrating Strong Classifiers

According to the analysis of the former subsection, weak classifiers in this paper are easily used in multiclass classification problems. However the performance can be further improved by changing the way of using strong classifiers.

Strong classifier tends to be used directly to solve two-class problem or independently to divide multiclass problem into several two-class problems in the AdaBoost family. This is probably the simplest way but certainly not the best way because the accuracy of two-class categorization cannot be further enhanced in strong classifying step and the complexity of multiclass categorization problem cannot be constraint efficiently.

Several strong classifiers can work together to solve the problems above. In this paper, we proposed a strong classifiers' cascading method to further improve the precision and limit the consumption in multiclass classification tasks.

The method of integrating strong classifiers can be explained clearly by using examples. For instance, we can use four strong classifiers in series sequentially to determine which category a document belonged to. When they make the same judgment, use it as the



S_1: same result

S_2: the minority is subordinate to the majority

S_3: counterpart results and reclassification

Figure 7: Work logic of cascading strong classifiers.

final result. When they get different results, the principle of *the minority is subordinate to the majority* could be used. Especially, when two different determinations are counterparts, a reclassification process is needed to get the final result. The work logic of this method is shown in Figure 7.

Using the method of integrating strong classifiers in series can improve the classification accuracy because the Cramer-Rao bound is lower in this situation [29]. The derivation and definition of original Cramer-Rao bounds contain too many integral functions and thus very complex, so we use the modified Cramer-Rao bounds (MCRBs) in this paper as below [30]:

$$\mathrm{MCRBs}(\tau) = E_{X,\mu}\left\{\left[\frac{\partial}{\partial\tau}\ln p(X \mid \mu, \tau)\right]^2\right\}^{-1}, \tag{5.4}$$

where $p(X \mid \mu, \tau)$ is the conditional probability of X when given variables μ and τ. Reference [31] had proved that, in text categorization, the average-result of multiple classifiers has lower MCRBs than result by single classifier. Therefore, system's precision can improved by this method. However, input documents to strong classifiers in series will significantly extend the categorization time. To save process time, strong classifiers can be combined in parallel, but in this way, the computational consumption will be increased. To keep balance between time and computational consumption, when implement the strong classifiers integrating method in real systems, users should decide combine them in series or in parallel according the size of documents collection, mutual information (MI) of different categories, the hardware capability, and time consumption tolerance of different systems and different projects.

6. Application and Analysis

The novel text categorization tool in this paper—adaptive group k nearest neighbor-based AdaBoost (AGkNN-AdaBoost) algorithm—is fully proposed in the former sections. To evaluate its performance we tested its training time by Matlab with different submodules and parameters. We also measured the time consumption of other algorithms and made comparison to analyze whether and why AGkNN-AdaBoost is better than many other tools, furthermore, which parts make the contributions for its efficiency beyond some algorithms and what mechanisms make it spend more training time than other algorithms.

A text categorization system based on AGkNN-AdaBoost is implemented, and plenty of standard corpora texts are used to measure its precision, recall, F_1. and so forth with different submodule and different initial parameters. We compared AGkNN-AdaBoost's performance not only with the AdaBoost family algorithms but also with some other classic classification algorithms such as SVM, decision tree, neural networks, and Naïve Bayes. We analyzed all data carefully and took the time consumption into account to make our final conclusion about the novel tool's performance.

6.1. Experiment for Time Consumption Analysis

In training step, we use documents set in which each document is a bit more or less than 2 KB and selected from standard copora to test the time needed for modeling when using AGkNN-AdaBoost. The relationship between the number of documents and time consumption by using different parameters and random model is shown in Figure 8.

In Figure 8, we test the modeling time with training sets containing 10 documents, 50 documents, 200 documents, 1000 documents, 5000 documents, and 20000 documents. We select $k = 3$, $k = 4$, $k = 5$, and random k as the number of reference neighbors. In each situation, we set the number of group as 4, 5, and 6 to evaluate the novel tool's performance with different parameters. In this test step, the stochastic strategy is used for strong classifier generation. That means system would use DIWC-1, DIWC-2, or DIWC-3 randomly. For ease of view, the logarithms of the documents numbers are used to draw the curve.

As shown in the chart above, the time consumption increased when the number of neighbors or groups increased. Note that logarithmic coordinates are used in the figure, so time consumption increased significantly with the change of N_g and k. Therefore, adjust the number of neighbors and groups adaptively has great significance to improve system's efficiency in the conditions of guaranteeing the performance. In random k mode, system's training time is longer than $k = 3$ mode and shorter than $k = 5$ mode. Whether the efficiency

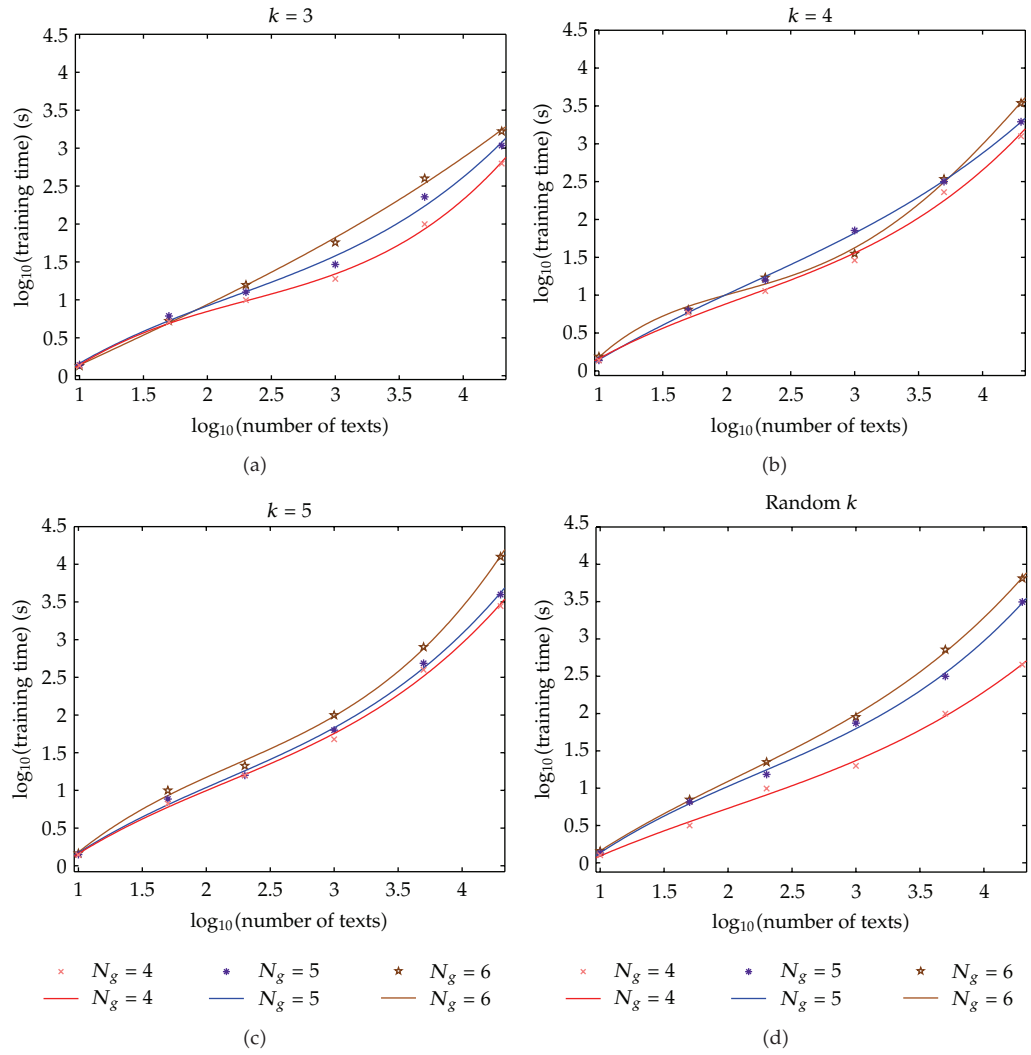


Figure 8: Time consumption of different size.

of the system is higher than $k = 4$ mode mostly depends on the number of groups, the number of documents, the size of each document, and document types.

To compare AGkNN-AdaBoost whose strong classifier is based on DIWC-1, DIWC-2, and DIWC-3 with other classic categorization algorithm, we designed experimental control groups including our novel algorithm, SVM, neural networks, and naïve Bayes. Similar to the former part, 5 thousands of documents (each text's size is about 2 KB) downloaded from standard copara are used for the comparison. The result is shown in Figure 9.

Time consumption will change with parameters and the way which combined strong classifiers. The training time of DIWC-1, DIWC-2, and DIWC-3 with different size of training set is tested as shown in Figure 9. The red dashed line represents DIWC-1, blue dash-dotted line represents DIWC-2, and brown dotted line represents DIWC-3.

Figure 9 reveals that time consumption increases fast when the training set becomes larger. In addition, more training time is needed when using more complex way to integrate

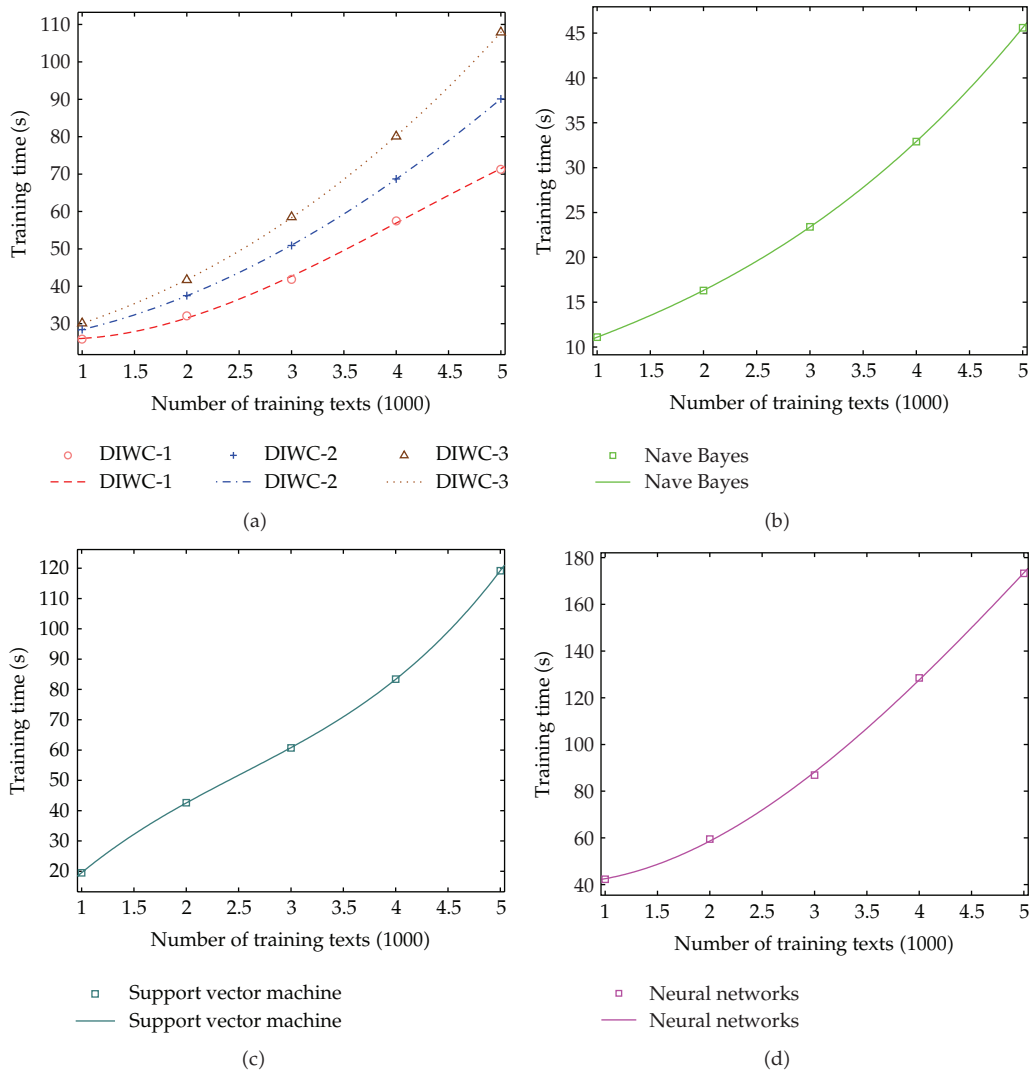


Figure 9: Training time of different algorithms.

strong classifiers. As shown above, the system even combines strong classifiers with DIWC-3 which need less training time than classification tools based on SVM and neural networks. However, three kinds of AG*k*NN-AdaBoost all need longer training time than naïve Bayes.

AdaBoost is a big algorithm family. We choose most classic and most efficient algorithms—original AdaBoost, AdaBoost.M1, AdaBoost.MR, and AdaBoost.ECC to evaluate the runtime complexity level of our novel algorithms proposed in this paper. We used the same training set as the former experiment, and the result is shown in Figure 10.

It is clearly shown in Figure 10 that AG*k*NN-AdaBoost has higher efficiency than original AdaBoost and AdaBoost.M1. Moreover, AG*k*NN-AdaBoost using DIWC-1, DIWC-2 and DIWC-3 as its strong classifier construction strategy makes them all spend training time equal to or less than AdaBoost.MR and AdaBoost.ECC—the leader of the efficiency in AdaBoost family [32]. That is because the adaptive grouping mechanism can better fit

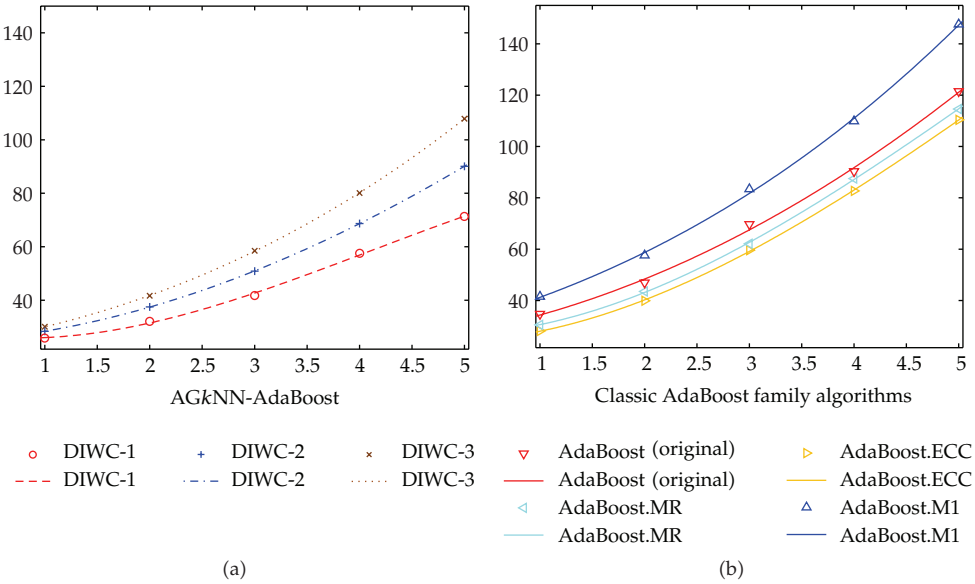


Figure 10: Time consumption of the AdaBoost family.

different characteristics of different training sets in grouping and k selection. Comparing adaptive value of k with experience-dependant k, the adaptive k has higher efficiency because it can upload its value in real time according to the variation of different situation such as document size, document type, and feature sparsity. In addition, the strategy which strong classifiers used—solving multiclass problem directly instead of transforming it into multiple two-class problems—further reduced the system complexity and time consumption.

It should be noted that the performance difference of efficiency between AGkNN-AdaBoost and AdaBoost.ECC is not obvious. The main reason is that the advanced reweight process of DIWC-3 spends a lot of time. However, AGkNN-AdaBoost still has advantages compared with AdaBoost.ECC, because AGkNN-AdaBoost with DIWC-1 and DIWC-2 has significantly lower runtime complexity. What's more, the precision of AGkNN-AdaBoost will be proved better than AdaBoost.ECC no matter DIWC-1, DIWC-2, or DIWC-3 is used.

6.2. Performance Comparison and Analysis

Experiment is made to evaluate performance of the system. Chinese news corpus support by Sogou Labs [33] is used as the training set and test set. kernel conditional random fields [34] (KCRFs) are used for preprocessing (word segmentation, feature extraction, and representation) the documents. The corpus can be divided into six categories—economics, politics, sports, weather report, entertainments, and culture. 20000 documents are randomly selected as the training samples and 10000 documents are randomly selected as the test texts in each category. Experimental results of system's precision, recall, and F_1-measure with comparative data [35–37] as shown in Tables 1, 2, 3, and 4.

As shown in the above-mentioned tables, AGkNN-AdaBoost has better performance than other text categorization algorithms. The performance of AGkNN-AdaBoost is far beyond naïve Bayes, neural networks, and decision tree. In addition, the novel classification tool has better performance than other AdaBoost family members. Spatially strong classifiers'

Table 1: Precision comparison.

Algorithms	Text type					
	Economics	Politics	Sports	Weather	Entertainment	Culture
AdaBoost	0.848	0.855	0.851	0.860	0.851	0.859
AdaBoost.M1	0.857	0.859	0.863	0.847	0.858	0.866
AdaBoost.MR	0.854	0.862	0.847	0.865	0.855	0.862
AdaBoost.ECC	0.848	0.854	0.841	0.843	0.840	0.856
Naïve Bayes	0.769	0.794	0.783	0.806	0.811	0.772
SVM	0.867	0.862	0.870	0.877	0.865	0.871
Neural network	0.832	0.807	0.819	0.824	0.828	0.803
Decision tree	0.809	0.792	0.786	0.831	0.799	0.812
AGkNN DIWC-1	0.887	0.894	0.882	0.911	0.877	0.893
AGkNN DIWC-2	0.899	0.906	0.903	0.921	0.898	0.902
AGkNN DIWC-3	0.918	0.905	0.917	0.924	0.903	0.907

Table 2: Recall comparison.

Algorithms	Text type					
	Economics	Politics	Sports	Weather	Entertainment	Culture
AdaBoost	0.852	0.857	0.849	0.863	0.859	0.861
AdaBoost.M1	0.852	0.864	0.863	0.877	0.862	0.865
AdaBoost.MR	0.858	0.863	0.849	0.866	0.851	0.867
AdaBoost.ECC	0.851	0.844	0.845	0.850	0.846	0.849
Naïve Bayes	0.761	0.798	0.782	0.804	0.817	0.805
SVM	0.868	0.865	0.874	0.874	0.867	0.876
Neural network	0.834	0.809	0.823	0.811	0.825	0.807
Decision tree	0.815	0.798	0.784	0.819	0.799	0.813
AGkNN DIWC-1	0.897	0.888	0.890	0.913	0.885	0.905
AGkNN DIWC-2	0.909	0.914	0.914	0.922	0.891	0.908
AGkNN DIWC-3	0.921	0.917	0.919	0.923	0.911	0.916

integrates according to DIWC-3 method have the best accuracy and recall. It takes more than six percentage increment to average F_1-measure in the six categories than AdaBoost family members. Although the support vector machine is an excellent classification tool, AGkNN-AdaBoost is even more accurate than it. Moreover take its runtime complexity into consideration, AGkNN-AdaBoost is much better than SVM.

Therefore, AGkNN-AdaBoost is an ideal tool for text categorization, it achieves really high accuracy while controls the runtime complexity in a very low degree. That is because the adaptive characteristics improve the performance, the double iterative mechanism enhances the efficiency, and the multiclass classification ability improves precision and reduces the time consumption at the same time.

It is interesting to note that classification in weather reports has the best precision and recall. It is probably because weather reports are quite simple and always contain similar features or key words such as *sunny*, *rainy*, *cloudy*, *temperature*, and *heavy fog warning*. AGkNN-AdaBoost does not perform as excellent in entertainment topic categorization as in other categories perhaps because documents belongs to this topic containing too many new words

Table 3: F_1-measure comparison.

Algorithms	Text type					
	Economics	Politics	Sports	Weather	Entertainment	Culture
AdaBoost	0.850	0.856	0.850	0.862	0.855	0.860
AdaBoost.M1	0.855	0.862	0.863	0.876	0.860	0.866
AdaBoost.MR	0.856	0.863	0.848	0.866	0.853	0.865
AdaBoost.ECC	0.849	0.849	0.848	0.847	0.843	0.847
Naïve Bayes	0.765	0.796	0.783	0.805	0.814	0.804
SVM	0.868	0.864	0.872	0.874	0.876	0.864
Neural network	0.833	0.808	0.821	0.808	0.827	0.805
Decision tree	0.812	0.795	0.785	0.825	0.799	0.813
AGkNN DIWC-1	0.896	0.888	0.896	0.912	0.881	0.899
AGkNN DIWC-2	0.895	0.910	0.909	0.922	0.906	0.903
AGkNN DIWC-3	0.907	0.911	0.918	0.924	0.920	0.912

Table 4: Average performance of algorithms.

Algorithms	Index		
	Precision	Recall	F_1-measure
AdaBoost	0.854	0.857	0.856
AdaBoost.M1	0.858	0.864	0.861
AdaBoost.MR	0.857	0.862	0.860
AdaBoost.ECC	0.847	0.857	0.852
Naïve Bayes	0.789	0.795	0.792
SVM	0.869	0.871	0.870
Neural network	0.819	0.818	0.819
Decision tree	0.805	0.805	0.705
AGkNN DIWC-1	0.899	0.896	0.898
AGkNN DIWC-2	0.905	0.910	0.908
AGkNN DIWC-3	0.916	0.918	0.917

and formal phrases which lead documents in this topic to have more complex features and the feature space of them possibly are extremely sparse.

7. Conclusion

An improved boosting algorithm based on k-nearest neighbors is proposed in this paper. It uses an adaptive group-based kNN categorization algorithm as basis classifiers and combines them in a double iterative weighed cascading method which contains three alternative modes. The strong classifiers are also modified for better satisfying multiclass classification tasks. The AGkNN-AdaBoost algorithm is implemented in a text categorization system, and several experiments are made. Experimental results shows that the algorithm proposed in this paper has higher precision, recall, and robustness than traditional AdaBoost. Furthermore, the time and computational consumption of AGkNN-AdaBoost are lower than many other categorization tools which are not limited to AdaBoost family

algorithms. Therefore the algorithm proposed in former sections is quite a useful tool in text categorization, including the Chinese TC problems.

However, support vector machine as one of the best classification algorithm and its usage as weak classifier combined by ideas which are similar with DIWC is a virgin land in text categorization. Moreover, there is space for improving the accuracy and efficiency of AGkNN-AdaBoost, and the performance of AGkNN-AdaBoost in other classification tasks such as image processing, speech categorization, and writer identification should be evaluate. These will be undertaken as future works on this topic.

Acknowledgments

The material presented in this paper is partly based upon work supported by the China Association for Science and Technology. Experimental data is offered by the Sogou Labs.

References

[1] H. Al-Mubaid and S. A. Umair, "A new text categorization technique using distributional clustering and learning logic," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 9, pp. 1156–1165, 2006.

[2] M. Lan, C. L. Tan, J. Su, and Y. Lu, "Supervised and traditional term weighting methods for automatic text categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 721–735, 2009.

[3] M. Lan, C. L. Tan, and H. B. Low, "Proposing a new term weighting scheme for text categorization," in *Proceedings of the 21st National Conference on Artificial Intelligence*, pp. 763–768, July 2006.

[4] L. Galavotti, F. Sebastiani, and M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization," in *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 59–68, 2000.

[5] P. Soucy and G. W. Mineau, "Beyond tfidf weighting for text categorization in the vector space model," in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI '05)*, pp. 1130–1135, August 2005.

[6] X. Quan, L. Wenyin, and B. Qiu, "Term weighting schemes for question categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 1009–1021, 2011.

[7] Y. Liu, J. Bian, and E. Agichtein, "Predicting information seeker satisfaction in community question answering," in *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR '08)*, pp. 483–490, July 2008.

[8] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*, pp. 412–420, 1997.

[9] Y. Chao, W. Yuanqing, L. Jiuxue, and Z. Zhaoyang, "Theory deduction of AdaBoost classification," *Journal of Southeast University*, vol. 41, no. 4, 2011.

[10] R. Qahwaji, M. Al-Omari, T. Colak, and S. Ipson, "Using the real, gentle and modest AdaBoost learning algorithms to investigate the computerised associations between coronal mass ejections and filaments," in *Proceedings of the International Conference on Communications, Computers and Applications*, pp. 37–42, Bradford, UK, August 2008.

[11] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[12] R. E. Schapire, M. Rochery, M. Rahim, and N. Gupta, "Boosting with prior knowledge for call classification," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 2, pp. 174–181, 2005.

[13] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security*, D. Barbara and S. Jajodia, Eds., Kluwer, Norwell, Mass, USA, 2002.

[14] W. Hu, W. Hu, and S. Maybank, "AdaBoost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics B.*, vol. 38, no. 2, pp. 577–583, 2008.

[15] K. M. Ting and Z. Zheng, "Boosting cost-sensitive trees," in *Proceedings of the 1st International Conference on Discovery Science*, pp. 244–255, Springer, December 1998.

[16] F. U. Zhong-Liang, "Cost-sensitive AdaBoost algorithm for multi-class classication problems," *Journal of Automation*, vol. 37, no. 8, pp. 973–983, 2011.

[17] E. Song, D. Huang, G. Ma, and C. C. Hung, "Semi-supervised multi-class Adaboost by exploiting unlabeled data," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6720–6726, 2011.

[18] I. Maglogiannis, H. Sarimveis, C. T. Kiranoudis, A. A. Chatziioannou, N. Oikonomou, and V. Aidinis, "Radial basis function neural networks classification for the recognition of idiopathic pulmonary fibrosis in microscopic images," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 1, pp. 42–54, 2008.

[19] C. T. Lin, C. M. Yeh, S. F. Liang, J. F. Chung, and N. Kumar, "Support-vector-based fuzzy neural network for pattern classification," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 1, pp. 31–41, 2006.

[20] J. M. Yang, P. T. Yu, and B. C. Kuo, "A nonparametric feature extraction and its application to nearest neighbor classification for hyperspectral image data," *IEEE Transactions on Geoscience and Remote Sensing*, no. 3, pp. 1279–1293, 2010.

[21] R. H. Yuhas, A. F. H. Goetz, and J. W. Boardman, "Discrimination among semi-arid landscape endmembers using spectral angle mapper (SAM) algorithm," in *Proceedings of the Summaries of the 4th Annual JPL Airborne Geoscience Workshop*, vol. 1, pp. 147–150, AVIRIS Workshop, R. Green, Ed., Pasadena, Calif, USA, October 1992.

[22] Z. Chun-hong and X. Wei, "The approach to text automatic classification technology of characteristic catabases on the SVM-KNN," *Information Science*, vol. 11, 2009.

[23] D. Coomans and D. L. Massart, "Alternative k-nearest neighbour rules in supervised pattern recognition. Part 2. Probabilistic classification on the basis of the kNN method modified for direct density estimation," *Analytica Chimica Acta*, vol. 138, pp. 153–165, 1982.

[24] N. Boonyanunta and P. Zeephongsekul, "Improving the predictive power of AdaBoost: a case study in classifying borrowers," in *Proceeding of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE '03)*, pp. 674–685, June 2003.

[25] H. J. Lin, Y. T. Kao, F. W. Yang, and P. S. P. Wang, "Content-based image retrieval trained by adaboost for mobile application," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 20, no. 4, pp. 525–541, 2006.

[26] J. Mitéran, J. Matas, E. Bourennane, M. Paindavoine, and J. Dubois, "Automatic hardware implementation tool for a discrete adaboost-based decision algorithm," *Eurasip Journal on Applied Signal Processing*, vol. 2005, no. 7, pp. 1035–1046, 2005.

[27] L. P. Dinu and A. Rusu, "Rank distance aggregation as a fixed classifier combining rule for text categorization," *Lecture Notes in Computer Science*, vol. 6008, pp. 638–647, 2010.

[28] S. M. Namburu, T. Haiying, L. Jianhui, and K. R. Pattipati, "Experiments on supervised learning algorithms for text categorization," in *Proceedings of the 2005 IEEE Aerospace Conference*, March 2005.

[29] D. Modgil and P. J. La Riviére, "Optimizing wavelength choice for quantitative optoacoustic imaging using the Cramer-Rao lower bound," *Physics in Medicine and Biology*, vol. 55, no. 23, pp. 7231–7251, 2010.

[30] A. N. D'Andrea, U. Mengali, and R. Reggiannini, "Modified Cramer-Rao bound and its application to synchronization problems," *IEEE Transactions on Communications*, vol. 42, no. 2, pp. 1391–1399, 1994.

[31] M. Sansone, R. Fusco, A. Petrillo, M. Petrillo, and M. Bracale, "An expectation-maximisation approach for simultaneous pixel classification and tracer kinetic modelling in dynamic contrast enhanced-magnetic resonance imaging," *Medical and Biological Engineering and Computing*, vol. 49, no. 4, pp. 485–495, 2011.

[32] G. Lie, G. Ping-Shu, Z. Ming-Heng, L. Lin-Hui, and Z. Yi-Bing, "Pedestrian detection for intelligent transportation systems combining AdaBoost algorithm and support vector machine," *Expert System with Applications*, vol. 39, no. 4, pp. 4274–4286, 2012.

[33] http://www.sogou.com/labs/resources.html .

[34] L. Fengcheng, H. Degen, and J. Peng, "Chinese word sense disambiguation with AdaBoost.MH algorithm," *Journal of Chinese Information Processing*, vol. 20, no. 3, pp. 6–13, 2005.

[35] H. Zhi-Kun, G. Wei-Hua, Y. Chun-Hua, D. Peng-cheng, and D. X. Steven, "Text classification method for inverter based on hybrid support vector machines and wavelet analysis," *International Journal of Control Automation and Systems*, vol. 9, no. 4, pp. 797–804, 2011.

[36] W. Wang and B. Yu, "Text categorization based on combination of modified back propagation neural network and latent semantic analysis," *Neural Computing and Applications*, vol. 18, no. 8, pp. 875–881, 2009.

[37] Z. Xuan and T. Da-Gang, "Study on text classification methods," in *Proceedings of the International Conference of China Communication*, pp. 123–125, October 2010.