11-2012

A generalized cluster centroid based classifier for text categorization

Guansong PANG
Singapore Management University, gspang@smu.edu.sg

Shengyi JIANG

Information Processing and Management 49 (2013) 576–586

Contents lists available at SciVerse ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

A generalized cluster centroid based classifier for text categorization

Guansong Pang [a,*], Shengyi Jiang [b]

[a] School of Management, Guangdong University of Foreign Studies, Guangzhou 510006, PR China
[b] School of Informatics, Guangdong University of Foreign Studies, Guangzhou 510420, PR China

ARTICLE INFO

Article history:
Received 16 August 2011
Received in revised form 21 October 2012
Accepted 22 October 2012
Available online 21 November 2012

Keywords:
Text categorization
KNN
Rocchio
Clustering
Generalized cluster centroid

ABSTRACT

In this paper, a Generalized Cluster Centroid based Classifier (*GCCC*) and its variants for text categorization are proposed by utilizing a clustering algorithm to integrate two well-known classifiers, i.e., the K-nearest-neighbor (*KNN*) classifier and the *Rocchio* classifier. *KNN*, a lazy learning method, suffers from inefficiency in online categorization while achieving remarkable effectiveness. *Rocchio*, which has efficient categorization performance, fails to obtain an expressive categorization model due to its inherent linear separability assumption. Our proposed method mainly focuses on two points: one point is that we use a clustering algorithm to strengthen the expressiveness of the *Rocchio* model; another one is that we employ the improved *Rocchio* model to speed up the categorization process of *KNN*. Extensive experiments conducted on both English and Chinese corpora show that *GCCC* and its variants have better categorization ability than some state-of-the-art classifiers, i.e., *Rocchio*, *KNN* and Support Vector Machine (*SVM*).

1. Introduction

With the exponential growth of online textual information, how to organize text data effectively and efficiently has become an important and demanding issue. Text categorization, a process of assigning predefined categories to test documents, is a significant tool for handling this issue. Many text categorization methods have been proposed in previous work (Baharudin, Lee, & Khan, 2010). Among these methods, *KNN* is shown to be a simple and effective method (Tan, 2006; Wang & Wang, 2007), but it brings with itself three fatal defects. First, the complexity of similarity computation is high. Second, its performance is easily biased by single training samples (e.g., noise samples). Third, *KNN* does not build a categorization model since it is a lazy learning method. As a result, *KNN* is not well suited in many applications where often call for strict requirements in real-time performance, such as email spam filtering. The top defect of *KNN* is high similarity computation, about which many algorithms have been proposed to reduce the complexity of *KNN*. These algorithms can be divided into three categories: reducing the dimensionality of document vectors (Vries, Mamoulis, & Nes, 2002), cutting down the amount of training samples (Li & Hu, 2004) and accelerating the process of finding *K* nearest neighbors (Liaw, Leou, & Wu, 2010; Wang & Wang, 2007).

In contrast, *Rocchio* is an efficient and easy-to-implement method for text categorization. It enjoys a good robustness since it summarizes original training samples into prototype vectors (i.e., generating one prototype vector for each category, named category-based prototype vectors) to classify test documents. However, *Rocchio* also has some limitations. For instance, it hypothesizes the data space is a set of linear separable hyperplane regions, but this hypothesis is often inconsistent with data distribution in many real-world applications where data is non-linearly distributed (Guo, Wang, Bell, Bi, & Greer, 2006; Lam & Han, 2003). Another defect is that one category-based prototype vector fails to fully represent a category within

* Corresponding author. Tel.: +86 13560089615.
E-mail address: pangguansong@163.com (G. Pang).

0306-4573/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ipm.2012.10.003

which there is a fine-grained and interconnected relationship, i.e., a category is a combination of several sub-categories. In this case, multi sub-category-based vectors for each category would have much better representation power.

We therefore propose a Generalized Cluster Centroid based Classifier (*GCCC*) to take full advantage of *KNN* and *Rocchio* via a clustering algorithm. We first employ a constrained single pass clustering algorithm (Jiang, 2006) and *Rocchio* to train a generalized cluster-based categorization model, and then use the *KNN* decision rule to classify test documents with the model. In particular, the constrained clustering algorithm and *Rocchio* are utilized to generate multi cluster-based vectors for each category. The clustering algorithm is expected to uncover the fine-grained relationship hidden within each category so that the constructed model can be more representative than the *Rocchio* model. Furthermore, *GCCC* could achieve an efficient online categorization when using the *KNN* decision rule to classify test documents with the model rather than all the original training samples. Extensive experiments are conducted on both English and Chinese corpora (i.e., Reuters-21578, Ling-Spam and Fudan Univ. text categorization corpora) so that we can compare the performance of the proposed method with other classifiers. The results show that our proposed method has better categorization ability than some state-of-the-art classifiers, i.e., *Rocchio*, *KNN* and *SVM*. In terms of online categorization efficiency, *GCCC* and its variants are more efficient than *KNN*, and have comparable time computation compared to *Rocchio*.

The rest of this paper is organized as follows: Section 2 describes the two well-known classifiers and introduces our proposed method. Section 3 presents the analysis of the empirical results. Related work is discussed in Section 4. The paper is concluded in Section 5.

2. Generalized cluster centroid based classifier for text categorization

2.1. KNN categorization

The process of the *KNN* classifier is as follows: given a test document x, find K nearest neighbors of x among training documents, and score category candidates for x based on the categories of K neighbors. Specifically, the similarity between x and each neighbor document is a score for the category of that neighbor document. If two or more neighbor documents belong to the same category, then the sum of the scores of that category is the total score of the category for x. Finally, system assigns the candidate category with the highest score to the test document x. The decision rule of *KNN* is as follows:

$$f(x) = \underset{j}{argmax}\ \text{Score}(x, C_j) = \sum_{d_i \in KNN^{doc}} sim(x, d_i) y(d_i, C_j)$$

where $f(x)$ is the label assigned to the test document x, Score (x, C_j) is the score of the candidate category C_j with respect to x, KNN^{doc} denotes a set of K nearest neighbor documents of x, $sim(x, d_i)$ is the similarity between x and the training document d_i, and $y(d_i, C_j) \in \{0, 1\}$ is a binary category value of the training document d_i with respect to C_j ($y = 1$ indicates document d_i belongs to category C_j; otherwise $y = 0$).

2.2. Rocchio categorization

The basic idea of applying *Rocchio* to text categorization is to build a prototype vector for the documents of each category in the training text collection. The prototype vectors are computed with the following formula:

$$C_j = \alpha \frac{1}{|D_j|} \sum_{d_m \in D_j} d_m - \beta \frac{1}{|D - D_j|} \sum_{d_n \in D - D_j} d_n$$

where C_j is the category-based prototype vector for category C_j, and D is the whole set of documents in the training text collection, D_j and $|D_j|$ denote the set of documents in C_j and its size respectively. α and β are parameters that adjust the relative importance of positive and negative document samples. The step of producing prototype vectors can be regarded as a learning process. The *Rocchio* categorization model is composed of these prototype vectors. Given a test document x, it calculates the similarities between x and each prototype vector, and x is assigned to the category with the highest similarity.

2.3. Our proposed method

In this study, Vector Space Model (*VSM*) is used to represent documents. In *VSM*, each document is considered to be a vector in the term-space. Term weights in documents are computed via *TFIDF* (Zhang, Yoshida, & Tang, 2010).

2.3.1. Intuition of our proposed method

KNN is a sample-based learning method, which uses all the training documents to predict category labels of test documents. Because of high text similarity computation, *KNN* suffers from inefficiency in online categorization, which greatly impedes its applications. One effective way to improve its poor efficiency is to build a generalized categorization model. The model is then employed to replace the training samples to classify test documents via the *KNN* decision rule. The online categorization efficiency will be substantially enhanced when using the model to classify test documents instead of the

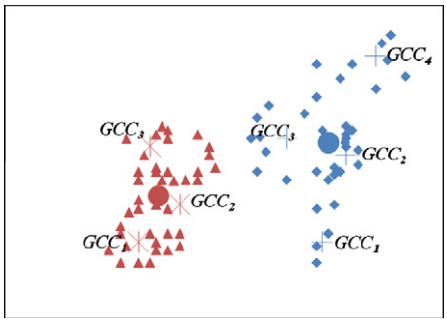

Fig. 1. An example of comparison of category-based prototype vectors and *GCCs*.

large-scale text collection. Meanwhile, the categorization ability will become much better since the model is the summarization of training documents.

Rocchio, on the other hand, is an efficient linear classifier, which builds its efficient categorization model by generalizing training samples into category-based prototype vectors. Thus, *Rocchio* is one of the best methods to build the categorization model for the improvement of *KNN* categorization efficiency. Nevertheless, *Rocchio* has two obvious drawbacks. One is that it hypothesizes the data space of samples can be linearly divided into different hyperplane regions. This hypothesis has less representative power than that of *KNN*. The other is that it only generalizes a prototype vector for each category. However, categories vary significantly in terms of scope in many contexts. In other words, some categories are essentially a combination of small categories, which are the so-called sub-categories or subtopics. If we could uncover this fine-grained relationship of the training samples and use different prototype vectors to represent different sub-categories of each category, the representation power of the *Rocchio* model would be strengthened. Clustering is a great tool to uncover this underlying relationship of training text documents since it is an unsupervised data learning process, aiming at partitioning data into clusters of similar samples. Therefore, we use a constrained clustering algorithm to group training samples into different clusters so that each category is denoted by pure clusters (i.e., documents in the same cluster have the same category). The clusters are regarded as sub-categories, which are used by *Rocchio* to generalize cluster-based prototype vectors, called Generalized Cluster Centroids (*GCCs*), and then the *GCCs* replace training samples to form a *GCCs*-based categorization model. As shown in Fig. 1, we assume that the *Triangle* category consists of three sub-categories and the *Square* category is composed of four sub-categories. In *Rocchio*, these two categories are denoted by their category-based prototype vectors which are denoted by two circles. In *GCCC*, ideally, the constrained clustering algorithm generate three and four pure clusters for the *Triangle* category and the *Square* category respectively, and then the *Rocchio* formula is used to generalize corresponding *GCCs* (i.e., cluster-based prototype vectors) for these two categories, which are denoted by the three star and four cross signs respectively. Finally, instead of using all the training samples, we make use of the *KNN* decision rule to classify test documents with the *GCCs* in the model.

Different clusters have very different term distributions so that the *GCCs* are non-linearly distributed at different positions in the data space. As a result, *GCCs* can better complement the defects of *Rocchio* and strengthen the expressiveness of the constructed model. Since the *GCCs* are derived from clusters and the clusters are a summarization of all the training samples, the model is insensitive to single training samples. Further, the *KNN* decision process is greatly accelerated since the number of *GCCs* in the model is a large reduction of training text documents. In this way, our proposed method can obtain impressive performance in terms of both efficiency and effectiveness.

2.3.2. Modeling and categorization

We combine a clustering algorithm with *Rocchio* to construct a *GCCs*-based model (i.e., an improved *Rocchio* model) for the latter *KNN* categorization. To ensure the scalability and applicability of the proposed method, we need to specify an effective and scalable clustering algorithm to perform clustering on large-scale text corpus. Clustering large-scale text corpus is a type of large and high-dimensional data clustering problems, which render most conventional clustering algorithms less effective. Varieties of clustering algorithms have been proposed in previous literature to handle large-scale and high-dimensional data, e.g., subspace clustering, correlation clustering (Kriegel, Kröger, & Zimek, 2009). Incremental clustering algorithms with less time-consumption can also deal with this problem since they are non-iterative and scan corpus in a single pass. The single pass clustering algorithm, a type of incremental clustering algorithms with approximately linear time complexity, is therefore adopted in this study (the whole clustering process is described in steps (1)–(12) of Fig. 2). To obtain pure clusters, we set a clustering constraint that the document is merged into the cluster only if the document has the same category as its nearest cluster, corresponding to steps (7)–(8) of Fig. 2. After the constrained clustering, we get a set of clusters $m_c = \left\{C_1^0, C_2^0, C_3^0, \ldots, C_i^0\right\}$ to produce the *GCCs*.

Lastly, as the generalized objects can improve the robustness of a categorization model and distill certain relevant features to some extent (Lam & Han, 2003), we employ the *Rocchio* formula to generalize the clusters into *GCCs*, as in step (15) of Fig. 2. In this step, each cluster is generalized into a *GCC* via the following formula:

Input: the training set T; the category label of samples L_{sample}; clustering threshold r

Output: the *GCCs* set m_{gcc} (each *GCC* contains a category label $L_{C_i^0}$)

Procedure GCC (T, C)

(1) Let m_c be the set of clusters, and m_{gcc} be the *GCCs* set
(2) m_c and m_{gcc} are initialized as empty sets.
(3) Repeat
(4) Input a new document p.
(5) Compute the similarities between p and all the clusters $\overline{C}$ in m_c.
(6) Find the cluster C_i^0 in $\overline{C}$ that is closest to the document p.
(7) If $sim(p, C_i^0) \geq r$ and $L_p = L_{C_i^0}$
(8) Merge p into C_i^0 and update the weights of terms in cluster C_i^0.
(9) Else
(10) Create a new cluster C_i^0 with p and $L_{C_i^0} = L_p$
(11) Add cluster C_i^0 to m_c
(12) Until no text samples left in T
(13) If Boolean of integrating or filtering is true
(14) Integrate or filter out small clusters (where $|C_i^0|=1$) in m_c.
(15) Use the *Rocchio* formula to generalized cluster set m_c and get m_{gcc}.
(16) Return m_{gcc}.

Fig. 2. Pseudo-code of the proposed method.

$$gcc_i = \alpha \frac{1}{|C_i^0|} \sum_{d_m \in C_i^0} d_m - \beta \frac{1}{|D - C_i^0|} \sum_{d_n \in D - C_i^0} d_n$$

where gcc_i is the *GCC* for the ith cluster C_i^0, C_i^0 and $|C_i^0|$ denote the set of documents in C_i^0 and its size respectively. The intuition behind this formula is straightforward, where the documents of C_i^0 are regarded as positive samples and the rest as negative, then the documents are summed up and normalized, and finally a subtraction between positive and negative vectors is performed to distill distinguished features for gcc_i. α and β are parameters that adjust the relative importance of positive and negative text documents with respect to clusters ($\alpha = 2\beta$ is used in our study).

During clustering, some small clusters are produced, which only contains one document. These documents are likely to be outlier samples, which may contain important information for classification or may be just noisy samples. Thus, to strengthen the categorization model, we provide options to set a threshold to integrate or filter out these small clusters, as in steps (13) and (14) of Fig. 2. The methods that combine the modeling method with integrating or filtering strategies are called the variants of *GCCC*. Further details are described in Fig. 2.

The clustering threshold r in step (7) may influence the quality and time-efficiency of the clustering algorithm, e.g., as r increases, both the number of clusters and time-consumption will increase. In order to gain a stable threshold r, a sampling technique is employed to determine the threshold (Jiang, Song, & Wang, 2006). Further details are described as follows:

Step 1: Randomly choose N_0 pairs of documents in the corpus.
Step 2: Compute the similarities between each pair of documents.
Step 3: Compute the average similarity *ex* of the similarities derived from *Step 2*.
Step 4: Select r as $\varepsilon \times ex$, where $\varepsilon \geqslant 1$.

where N_0 denotes the number of pairs of selected documents, *ex* is the average similarity of the similarities of N_0 pairs of documents, and ε is a parameter that adjusts r values regarding different scenarios since *ex* varies from different text categorization tasks. When N_0 reaches a higher value, *ex* remains stable. In our experiments, we choose $N_0 = 8000$. The value of r is closely related to the corpus's size, and the empirical results show that high quality clustering results and satisfactory performance levels are achieved when ε varies between 5 and 13. Further details regarding ε values are given in Section 3.3.

In the categorization stage, we use the *KNN* decision rule to classify test documents with the *GCCs*-based model. Further details of the categorization are described as follows: given a test document x, score each *GCC* in m_{gcc} with respect to x using the following formula, and assign the category label of the *GCC* with the highest score to the test document x.

$$f(x) = \underset{j}{argmax}\ \text{ClusterScore}(x, C_j) = \sum_{gcc_i \in KNN^{doc}} sim(x, gcc_i) y(gcc_i, C_j)$$

where $\text{ClusterScore}(x, C_j)$ is the score of the candidate category C_j with respect to x, and $sim(x, gcc_i)$ is the similarity between x and gcc_i. $y(gcc_i, C_j)$ is an indicator function, $y = 1$ indicates gcc_i is part of category C_j; otherwise $y = 0$.

3. Experiments

3.1. Datasets

Reuters-21578[1] is a standard benchmark corpus for text categorization. The Reuters-21578 collection, which appeared on the Reuters newswire in 1987, contains 21578 English documents and 135 categories. According to Guo et al. (2006), we use the "ModApte" split version of Reuters-21578 and select the seven most frequent Reuters categories as our evaluation corpus. Stop words were removed using a stop word list.[2] Details of the chosen subset are described in Table 3.

Fudan Univ. text categorization corpus[3] is built by the Chinese natural language processing group in the Department of Computer Information and Technology of Fudan University. This corpus contains 20 categories with 9804 Chinese training documents and 9833 Chinese test documents, and the ratio of training documents and test documents is approximately 1:1. We choose 12 categories as our experimental corpus. During preprocessing, the *ICTCLAS* Chinese word segmentation is employed to filter prepositions, conjunctions, pronouns, interjections, auxiliary particles, and particles of speech stop words. Details of selected subset are described in Table 4.

Ling-Spam[4] contains 2893 messages collected from a moderated mailing list on profession and science of linguistics: 2412 legitimate messages and 481 spam messages (16.6%). Four versions of this corpus are given depending on whether stemming and stop word list were used. Each of these versions is partitioned into 10 stratified folds to facilitate evaluation using 10-fold cross validation. The *lemm* version (only stemming) is used in this experiment.

3.2. Performance metrics

Many evaluation metrics in text categorization are discussed in (Sokolova & Lapalme, 2009). We use F_1, *micro averaging* F_1 (*micro*-F_1) and *macro averaging* F_1 (*macro*-F_1) as evaluation metrics on Reuters and Fudan Univ. corpus. Details of these three metrics are given below:

$$F_1 = \frac{2 \times r \times p}{r + p}$$

where F_1 combines recall (r) and precision (p) into a single measure; F_1, *micro*-F_1 and *macro*-F_1 are to evaluate the classifier performance for individual categories and the whole corpus respectively.

In email spam filtering, weighted accuracy (*WAcc*), total cost ratio (*TCR*), spam recall (*SR*), spam precision (*SP*) and SF_1 (the combination of *SR* and *SP*) are widely used for evaluating spam filtering performance (Androutsopoulos, Koutsias, & Chandrinos, 2000; Androutsopoulos, Paliouras, & Karkaletsis, 2000; Koprinska, Poon, & Clark, 2007; Sakkis, Androutsopoulos, & Paliouras, 2001). These metrics are utilized to measure spam filtering performance on Ling-Spam. Details of the metrics are described as follows:

$$WAcc = \frac{\lambda N_{S\to S} + N_{L\to L}}{\lambda N_S + N_L} \quad WErr^b = \frac{N_S}{\lambda N_S + N_L} \quad WErr = \frac{\lambda N_{L\to S} + N_{S\to L}}{\lambda N_S + N_L} \quad WAcc^b = \frac{\lambda N_L}{\lambda N_S + N_L}$$

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda N_{L\to S} + N_{S\to L}} \quad SR = \frac{N_{S\to S}}{N_{S\to S} + N_{S\to L}} \quad SP = \frac{N_{S\to S}}{N_{S\to S} + N_{L\to S}} \quad SF_1 = \frac{2 \times SR \times SP}{SR + SP}$$

where N_S and N_L are the number of spam messages and legitimate messages, and $N_{Y\to Z}$ is the number of messages in category Y that the filter classified as Z; *WErr*, $WAcc^b$ and $WErr^b$ are the weighted error rate, baseline accuracy and baseline error rate respectively. *TCR* is the proportion of baseline error rate and weighted error rate. Greater *TCR* indicates better performance. λ indicates the cost sensitive ratio. Since our method is not a cost-sensitive classifier, λ is set to 1 in our experiments.

3.3. Results

The clustering threshold r and the K value of *KNN* are the two most influential parameters in our proposed method. Different clustering thresholds and K values are chosen to evaluate the sensitivity of our method. We also conduct a comparative study of the performance of our method and other classifiers, i.e., *KNN*, *Rocchio*, *SVM*. *KNN* and *Rocchio* are used as baseline classifiers in this study, and *SVM* is chosen as a comparative classifier since it is reported as one of the best methods in text categorization (Lee & Kageur, 2007). We use *LIBSVM*[5] and set $C = 5$ (the best performance is achieved when we set $C = 5$). The one-vs-one method is applied to produce multi-class labels for *SVM*, and the linear kernel is chosen since it outperforms non-linear kernels in text categorization (Zhang, Yoshida, & Tang, 2008).

[1] Reuters-21578 is available at http://archive.ics.uci.edu/ml/databases/reuters21578/
[2] Stop word list is available at http://download.csdn.net/source/1568518
[3] Fudan Univ. text categorization corpus is available at http://www.nlp.org.cn/docs/download.php?doc_id=294
[4] Ling-Spam is available at http://nlp.cs.aueb.gr/software_and_datasets/lingspam_public.tar.gz
[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

3.3.1. Results with different clustering thresholds

Figs. 3 and 4 show the *micro-F_1* and *macro-F_1* values of *GCCC* with different clustering thresholds on Reuters and Fudan Univ. corpora. It can be seen from the figures that *GCCC* gains stable *micro-F_1* and *macro-F_1* values with clustering thresholds ranging from $8.0 \times ex$ to $13.0 \times ex$. We use $r = 11.0 \times ex$ and $r = 12.0 \times ex$ in latter experiments on Reuters and Fudan corpora respectively.

Table 1 shows how the accuracy and *TCR* of *GCCC* vary on Ling-Spam with respect to different clustering thresholds. As we can see, *GCCC* obtains stable performance when r is selected in the interval $[2.0 \times ex, 6.0 \times ex]$. We use $r = 5.0 \times ex$ in the latter experiments.

Overall, the results conducted on these three corpora indicate that *GCCC* enjoys stable performance when it chooses the clustering threshold in a certain scope.

3.3.2. Results with different K values

The traditional *KNN* classifier is sensitive to the value of *K*. We conduct experiments to evaluate the sensitivity of the *K* value to *GCCC* with *KNN* as baseline.

Fig. 5 demonstrates the *micro-F_1* and *macro-F_1* values of *GCCC* and *KNN* with different *K* values on Reuters. The results show that *GCCC* consistently outperforms *KNN* in the performance of *macro-F_1* values, and achieves better performance than *KNN* in most *micro-F_1* values. It should be noted that *macro-F_1* is a more important metric than *micro-F_1* in measuring the performance on an imbalance corpus because *macro-F_1* evaluates the performance in terms of the average of all categories F_1 rather than the accuracy of classifying test documents as a whole. It is also clear that *GCCC* obtains more stable performance compared with the fluctuated performance of *KNN*.

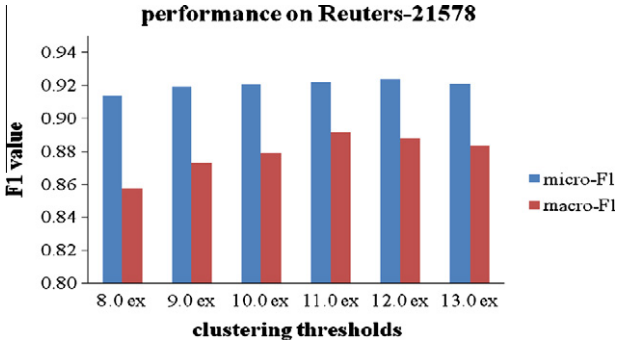

Fig. 3. *Micro-F_1* and *macro-F_1* values of *GCCC* with different thresholds on Reuters.

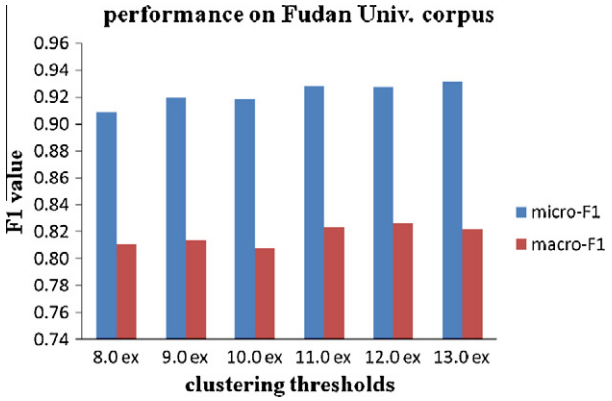

Fig. 4. *Micro-F_1* and *macro-F_1* values of *GCCC* with different thresholds on Fudan corpus.

Table 1
W_{Acc} and *TCR* of *GCCC* with different clustering thresholds on Ling-Spam.

Metric	2.0 *ex*	3.0 *ex*	4.0 *ex*	5.0 *ex*	6.0 *ex*
W_{Acc}	0.9841	0.9837	0.9851	0.9889	0.9900
TCR	10.45	10.23	11.19	15.03	16.60

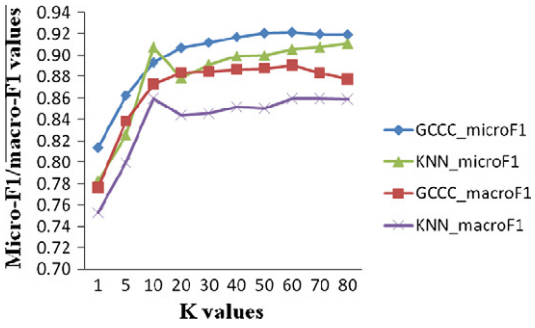

Fig. 5. *Micro-F_1* and *macro-F_1* of *GCCC* and *KNN* with different *K* values on Reuters.

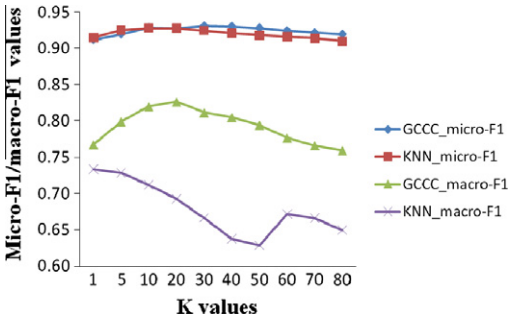

Fig. 6. *Micro-F_1* and *macro-F_1* value of *GCCC* and *KNN* with different *K* values on Fudan.

Table 2
TCRs of *GCCC* and *KNN* with different *K* values on Ling-Spam.

Algorithm	*K* = 1	*K* = 2	*K* = 10
GCCC	15.03	15.03	30.1
KNN	5.35	5.12	1.53

Fig. 6 shows the *micro-F_1* and *macro-F_1* values of *GCCC* and *KNN* with different *K* values on the Fudan corpus. The figure shows that *GCCC* outperforms *KNN* in most *micro-F_1* values, and all *macro-F_1* values of *GCCC* dramatically outperform *KNN*. These results are consistent with that reported on the Reuters corpus.

The effectiveness of *KNN* (*TiMBL*) classifier applied in spam filtering has been studied in many previous work. Androutsopoulos, Paliouras, et al. (2000) conducted a comprehensive study of *KNN* on the Ling-Spam corpus, and the results demonstrated that *KNN* achieved favorable performance in spam filtering. To better investigate the possible improvement of *GCCC* in spam filtering, experiments are conducted to evaluate its performance given by three different *K* values (with *K* = 1, *K* = 2, and *K* = 10 respectively in accordance to Androutsopoulos, Paliouras, et al. (2000)). Table 2 shows that *GCCC* greatly outperforms *KNN* in all *K* values.

3.3.3. Comparisons with other classifiers

Tables 3 and 4 show performance comparisons in *F_1*, *micro-F_1* and *macro-F_1* values of *Rocchio*, *KNN*, *SVM*, and *GCCC* on Reuters and Fudan corpora. Below we report the best results we obtain from this set of experiments. It can be observed from Tables 3 and 4 that *GCCC* has a competitive performance compared to other classifiers. They show that *GCCC* consistently outperforms other three classifiers in *macro-F_1* value on both corpora, though it has less effective performance in *micro-F_1*. In particular, *GCCC* has about 5–10% improvement over *KNN*, *Rocchio* and *SVM* classifiers in *macro-F_1* value. Recall that *macro-F_1* is a much more important metric than *micro-F_1* in imbalance corpus classification. *GCCC_F1* and *GCCC_I1* are variants of *GCCC*, and their performance analysis is given in Section 3.3.4.

To further investigate the effectiveness of *GCCC*, the comparison of *GCCC* and other state-of-the-art spam filters is conducted in Table 5. As can be seen from the table, *GCCC* with *K* = 10 consistently outperform other classifiers in *SF_1*, accuracy and *TCR*.

3.3.4. Results of the variants of GCCC

In this part, we focus on the performance of the variants of *GCCC*. The results are given in Tables 3–5, where *GCCC_F1* denotes the filtering strategy is chosen and *GCCC_I1* represents *GCCC* plus the integrating strategy. *GCCC_I1* not only

Table 3
Performance of different classifiers on Reuters-21578.

Category	Train	Test	*GCC* (*1*)	*GCCC*	*GCCC_F1*	*GCCC_I1*	*Rocchio*	*KNN*	*SVM*
Acq	1650	719	477	0.9241	0.9505	0.9419	0.9441	0.8998	0.9608
Corn	181	56	21	0.9346	0.9369	0.9298	0.9474	0.8870	0.8807
Crude	389	189	67	0.7859	0.8529	0.8256	0.7957	0.8235	0.7930
Earn	2877	1087	543	0.9571	0.9698	0.9698	0.9763	0.9500	0.9799
Interest	347	131	39	0.9306	0.9268	0.9524	0.8632	0.8945	0.9243
Ship	197	89	81	0.7465	0.6957	0.7222	0.7476	0.6503	0.6115
Trade	369	117	81	0.9106	0.9136	0.9295	0.9113	0.8943	0.8739
Micro-F_1				0.9217	0.9393	0.9389	0.9322	0.9112	0.9359
Macro-F_1				0.8910	0.8947	0.8967	0.8862	0.8597	0.8628

Table 4
Performance of different classifiers on Fudan.

Category	Train	Test	*GCC* (*1*)	*GCCC*	*GCCC_F1*	*GCCC_I1*	*Rocchio*	*KNN*	*SVM*
Space	640	642	124	0.8902	0.8906	0.9033	0.8891	0.8983	0.9470
Electronics	27	28	22	0.6667	0.6341	0.7347	0.4587	0.5500	0.5000
Communication	25	27	19	0.6818	0.5500	0.8302	0.7324	0.7273	0.6383
Computer	1357	1357	298	0.9446	0.9494	0.9550	0.9371	0.9489	0.9676
Transport	57	59	43	0.9204	0.8235	0.8679	0.8710	0.8155	0.7750
Environment	1217	1218	175	0.9647	0.9556	0.9641	0.9478	0.9616	0.9670
Law	51	52	34	0.7470	0.6742	0.7292	0.6202	0.5854	0.4923
Medical	51	53	38	0.7529	0.7229	0.7955	0.8793	0.6341	0.4918
Military	74	76	36	0.4490	0.6615	0.7328	0.4084	0.6050	0.6222
Politics	1024	1026	210	0.9398	0.9316	0.9412	0.8784	0.9291	0.9348
Sports	1253	1254	233	0.9525	0.9414	0.9437	0.8628	0.9436	0.9590
Education	59	61	55	0.5487	0.0317	0.2222	0.3393	0.1429	0.4578
Micro-F_1				0.9274	0.9240	0.9341	0.8676	0.9248	0.9339
Macro-F_1				0.8263	0.7601	0.8203	0.7764	0.7285	0.7294

Table 5
Performance of different classifiers on Ling-Spam.[a]

Algorithm	*SR*	*SP*	*SF_1*	Accuracy	*TCR*
GCCC (5.0EX_10NN)	0.9773	0.9900	0.9836	0.9945	30.10
GCCC_F1	0.9481	0.9852	0.9663	0.9889	15.04
GCCC_I1	0.9669	0.9733	0.9701	0.9900	16.60
Rocchio	0.9959	0.7827	0.8765	0.9433	2.93
TiMBL (1-NN)	0.8527	0.9592	0.9028	0.9689	5.35
TiMBL (2-NN)	0.8319	0.9710	0.8961	0.9675	5.12
TiMBL (10-NN)	0.3454	0.9964	0.5130	0.8908	1.52
NB	0.8235	0.9902	0.8992	0.9693	5.41
Stacking	0.9170	0.9650	0.9404	N/A	8.44
Outlook patterns	0.5301	0.8793	0.6614	0.9098	1.84
RF	0.9750	0.9830	0.9790	0.9931	N/A
DT	0.9520	0.9560	0.9540	0.9848	N/A
SVM	0.8190	0.9900	0.8960	0.9685	N/A
Baseline	0	∞	0	0.8337	1.00

[a] Outlook patterns, TiMBL, Stacking, RF, DT, NB and SVM spam filtering results are from Androutsopoulos, Koutsias, et al. (2000), Androutsopoulos, Paliouras, et al. (2000), Sakkis et al. (2001) and Koprinska et al. (2007).

outperforms *KNN*, *Rocchio* and *SVM* classifiers on all the three corpora, but also has a competitive performance compared to *GCCC* and *GCCC_F1*. Although *GCCC_F1* performs less effectively than *GCCC_I1*, it achieves comparable performance compared to *KNN*, *Rocchio* and *SVM*.

To investigate the reason for the diverse results of *GCCC_F1* and *GCCC_I1*, we carry out experiments on Reuters and Fudan corpora to get the number of *GCCs* which contains only one document, denoted as *GCC* (*1*) given in Tables 3 and 4. Intuitively, if *GCC* (*1*) are noisy text samples, the performance would be improved when filtering out these *GCCs*. However, the results show that the minority categories (i.e., small categories) are more likely to produce *GCC* (*1*) than the majority categories (i.e., large categories). It means that when the filtering strategy is employed, most of *GCCs* of minority categories are filtered out so that the minority categories lose substantial prior knowledge. As a result, the performance on minority categories degrade, and the overall performance is therefore dragged down. For example, *GCCC_F1* performs worse on Fudan than that

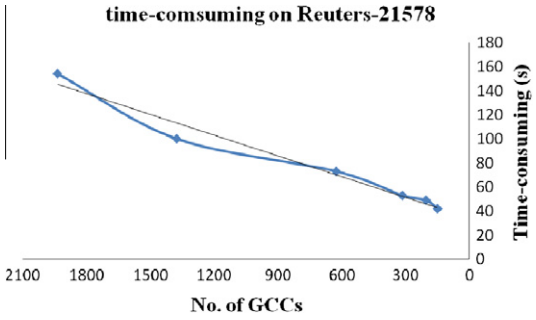

Fig. 7. Time–cost of *GCCC* on Reuters with respect to the number of *GCCs*.

Table 6
Effectiveness and efficiency of *Rocchio*, *KNN*, our method on Reuters.

Methods	No. of *GCCs* or instances in the model	Time-consumption(s)	*Micro-F_1*	*Macro-F_1*
Our method	1937	154	0.9217	0.8910
	635	111	0.9389	0.8967
	628	73	0.9393	0.8947
	317	53	0.9355	0.8819
Rocchio	7	37	0.9113	0.8862
KNN	6010	291	0.8943	0.8597

on Reuters due to the fact that the Fudan corpus is more skewed than Reuters. On the other hand, the integrating strategy summarizes all these scattered *GCCs* so that the categorization model has more prior knowledge to classify test documents. As a result, the integrating strategy improves the overall performance. Thus, it is necessary to consider the characteristic of corpus when deciding to choose *GCCC* or its variants for categorization tasks. *GCCC_I1* is more appropriate for imbalance text categorization tasks than *GCCC_F1*, and *GCCC* tends to obtain stable performance in different scenarios.

3.4. Time efficiency

Let n_{train} be the number of text samples in the training collection, and n_a be the average number of none-zero features per sample. In the processing of building model, single pass clustering and *Rocchio* computing are the critical parts. To simplify the analysis, the final number of clusters is assumed to be m. The clustering algorithm scans the training set in a single pass, and the number of clusters varies from 1 to m during clustering. In the worst case, the clustering has the time complexity $O(n_{\text{train}}\, n_a\, m)$. The time complexity of generalizing m cluster centroids take $O(m)$. n_a can be regarded as a constant with respect to a corpus. Thus, the time complexity of building model is $O(n_{train}\, m)$.

To classify a test document, the critical parts are scoring and ranking the *GCCs*. The time complexity of these parts takes $O(mlogm)$. However, the time complexity of *KNN* is $O(n_{train}\ log\ n_{train})$, where n_{train} is significantly larger than m. Therefore, *GCCC* requires much less similarity computation than *KNN* during online text categorization. Besides, the number of *GCCs* is larger than the number of category-based prototype vectors in *Rocchio*, but *GCCs* has less none-zero features than that of prototype vectors. Thus, *GCCC* can obtain comparable time computation compared to *Rocchio*.

Fig. 7 shows the classification time-consumption of *GCCC* on Reuters. It demonstrates that the online time-consumption of *GCCC* has near linear decrease with the reduction of the number of *GCCs*. Table 6 presents the changes in the effectiveness and efficiency of *GCCC* and its variants on Reuters with *KNN* and *Rocchio* as comparative classifiers. As we can see from Table 6 that the performance of our method is closely related to the number of *GCCs*. Larger number of *GCCs* results in more effective but less efficient performance of *GCCC* and vice versa. Similar results can be observed from Fudan and Ling-Spam. Thus, there has to be a trade-off between the effectiveness and efficiency in terms of different contexts.

4. Related work

There has been much work done on developing new methods for text categorization over the last decade, e.g., decision tree (Appavu & Rajaram, 2009), *KNN* (Wang & Wang, 2007), *Rocchio* (Guo et al., 2006), Naïve Bayes (*NB*) (Frank & Bouckaert, 2006), neural network (*NNet*) (Li & Park, 2009), Support Vector Machine (*SVM*) (Lee & Kageur, 2007), and centroid-based approaches(Tan, 2008). Meanwhile, the applications of text categorization algorithms have been widely spread in email spam filtering (Blanzieri & Bryl, 2008; Lai, 2007), opinion mining and sentiment analysis (He & Zhou, 2011).

As one of the most popular text categorization algorithms, *KNN* has drawn great attention. In previous work, most researchers focused on constructing a faster search tree to accelerate the finding of K nearest neighbors (Liaw et al.,

2010; Vries et al., 2002; Wang & Wang, 2007). It is hard to accurately compare the computation complexity of building search tree since they are influenced by various factors such as the required number of training samples, feature dimensions and corresponding thresholds. However, these methods often need to at least scan all training samples once to build the search tree, and then scan the search tree several times to prune. From this perspective, the efficiency of *Rocchio* and our method has slight advantage since they only call for scanning once. In terms of effectiveness, many studies on improving the *KNN* algorithm just keep the same effectiveness as *KNN* because they aim to find the exact K nearest neighbor as *KNN* (Liaw et al., 2010; Vries et al., 2002; Wang & Wang, 2007). As the proposed method can build a more representative model than *KNN* and *Rocchio*, we can achieve more effective performance.

Other similar work is Lam and Han (2003) and Guo et al. (2006), which exploited *KNN* and *Rocchio* to build a more effective categorization model. They attempted to build multi-representatives for each category by search global and local neighbors of each document, and then utilized *Rocchio* to generalize a *GIS*-based (Generalized Instance Set) model. A category label is assigned to documents when the similarity score of that category is the highest, or higher than a predefined threshold. To find the neighbors of each document for modeling, they need to search the training text collection iteratively. Compared with these two work, using the single pass clustering algorithm to assist modeling process has less time consumption since the clustering algorithm only needs to perform data scanning once. To classify documents, unlike Lam and Han (2003) and Guo et al. (2006), we use the *KNN* decision rule to determine the category labels of test documents since the *KNN* categorization method can achieve remarkable effectiveness. Therefore, our method can obtain competitive performance compared with these work. We compare our experimental results with that of the more recent work (i.e., Guo et al. (2006)). According to the limited similar experiments conducted on the English corpus (i.e., Reuters), our proposed method achieves more effective performance than the method presented in Guo et al. (2006).

5. Conclusion

In this paper, we propose a generalized cluster centroid based classifier for text categorization by utilizing a constrained clustering algorithm to integrate *KNN* and *Rocchio*. More specifically, to strengthen the expressiveness of the *Rocchio* model, we combine the constrained clustering with *Rocchio* to build a generalized cluster-based categorization model. The model is then employed to classify test documents by the *KNN* decision rule. Experiments conducted on three corpora show that *GCCC* and its variants achieve impressive performance when the parameters r and K choose their values in the interval $[5.0 \times ex, 13 \times ex]$ and the interval $[5, 50]$ respectively. This indicates that our method can achieve relatively stable and favorable performance. In terms of efficiency, our method can obtain near linear time complexity in modeling; during online categorization, *GCCC* and its variants are of better efficiency than *KNN*, and they also have comparable time consumption compared to *Rocchio*. A limitation of *GCCC* is that its modeling stage is more time-consuming than *KNN* and *Rocchio*. One simple but effective solution is to execute the modeling stage off-line.

Further work will be conducted to investigate effective imbalance text categorization strategies based on the different sizes of *GCCs*, e.g., how to adjust the weights of *GCCs* in accordance to the size of the categories and their belonging *GCCs*.

Acknowledgements

We thank Dr. Warren Jin for his valuable and constructive suggestions on this research, as well as the anonymous reviewers for their insightful suggestions. We also thank Eric Maldonado, Prof. Aiping Mo, Prof. Xinying Qiu, Lisha Zhang and Jiajun Chen for their helpful comments on this manuscript's revision. This work was supported by the National Natural Science Foundation of China (No. 61070061, No.61202271), the Ministry of Education of China Project (No. 12YJAH103, No.11YJCZH086) and the Graduate Research Project of GDUFS (No.12GWCXXM-11).

References

Androutsopoulos, I., Koutsias, J., & Chandrinos, K. V. (2000a). An evaluation of Naive Bayesian anti-spam filtering. In *Proceedings of 11th European conference on machine learning* (pp. 9–17).

Androutsopoulos, I., Paliouras, G., & Karkaletsis, V. (2000b). Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In *Proceedings of the workshop on machine learning and textual information access* (pp. 1–13).

Appavu, S., & Rajaram, R. (2009). Knowledge-based system for text classification using ID6NB algorithm. *Knowledge-Based Systems, 22*(1), 1–7.

Baharudin, B., Lee, L. H., & Khan, K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology, 1*(1), 4–20.

Blanzieri, E., & Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review, 29*(1), 63–92.

Frank, E., & Bouckaert, R. (2006). Naive bayes for text classification with unbalanced classes. *Knowledge Discovery in Databases*, 503–510.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2006). Using KNN model for automatic text categorization. *Soft Computing-A Fusion of Foundations, Methodologies and Applications, 10*(5), 423–430.

He, Y., & Zhou, D. (2011). Self-training from labeled features for sentiment analysis. *Information Processing & Management, 47*(4), 606–616.

Jiang, S. Y. (2006). Efficient classification method for large dataset. In *Proceedings of the fifth international conference on machine learning and cybernetics* (pp. 1190–1194). Dalian.

Jiang, S. Y., Song, X. Y., & Wang, H. (2006). A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letters, 27*(5), 802–810.

Koprinska, I., Poon, J., & Clark, J. (2007). Learning to classify e-mail. *Information Sciences, 177*, 2167–2187.

Kriegel, H. P., Kröger, P., & Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD), 3*(1), 1–58.

Lai, C. C. (2007). An empirical study of three machine learning methods for spam filtering. *Knowledge-Based Systems, 20*(3), 249–254.

Lam, W., & Han, Y. (2003). Automatic textual document categorization based on generalized instance sets and a metamodel. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(5), 628–633.
Lee, K. S., & Kageur, K. A. (2007). Virtual relevant documents in text categorization with support vector machines. *Information Processing & Management, 43*(4), 902–913.
Li, R. L., & Hu, Y. F. (2004). A density-based method for reducing the amount of training data in KNN text classification. *Journal of Computer Research and Development, 41*(4), 539–545.
Li, C. H., & Park, S. C. (2009). Combination of modified BPNN algorithms and an efficient feature selection method for text categorization. *Information Processing & Management, 45*(3), 329–340.
Liaw, Y. C., Leou, M. L., & Wu, C. M. (2010). Fast exact k nearest neighbors search using an orthogonal search tree. *Pattern Recognition, 43*(6), 2351–2358.
Sakkis, G., Androutsopoulos, I., & Paliouras, G. (2001). Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of the 6th conference on empirical methods in natural language processing* (pp. 44–50).
Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437.
Tan, S. (2006). An effective refinement strategy for KNN text classifier. *Expert Systems with Applications, 30*(2), 290–298.
Tan, S. (2008). An improved centroid classifier for text categorization. *Expert Systems with Applications, 35*(1), 279–285.
Vries, A. d., Mamoulis, N., & Nes, N. (2002). Efficient KNN search on vertically decomposed data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (pp. 322–333).
Wang, Y., Wang, Z. O. (2007). A fast KNN algorithm for text categorization. In *Proceedings of the 6th international conference on machine learning and, cybernetics* (pp. 3436–3441).
Zhang, W., Yoshida, T., & Tang, X. (2008). Text classification based on multi-word with support vector machine. *Knowledge-Based Systems, 21*(8), 879–886.
Zhang, W., Yoshida, T., & Tang, X. (2010). A comparative study of TF* IDF, LSI and multi-words for text classification. *Expert Systems with Applications, 8*(38), 2758–2765.