Expert Systems with Applications 39 (2012) 1503–1509

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

An improved *K*-nearest-neighbor algorithm for text categorization

Shengyi Jiang *, Guansong Pang, Meiling Wu, Limin Kuang

School of Informatics, Guangdong University of Foreign Studies, 510420 Guangzhou, China

ARTICLE INFO

Keywords:
Text categorization
KNN text categorization
One-pass clustering
Spam filtering

ABSTRACT

Text categorization is a significant tool to manage and organize the surging text data. Many text categorization algorithms have been explored in previous literatures, such as KNN, Naïve Bayes and Support Vector Machine. KNN text categorization is an effective but less efficient classification method. In this paper, we propose an improved KNN algorithm for text categorization, which builds the classification model by combining constrained one pass clustering algorithm and KNN text categorization. Empirical results on three benchmark corpora show that our algorithm can reduce the text similarity computation substantially and outperform the-state-of-the-art KNN, Naïve Bayes and Support Vector Machine classifiers. In addition, the classification model constructed by the proposed algorithm can be updated incrementally, and it has great scalability in many real-word applications.

1. Introduction

Text categorization is the process of assigning predefined categories to text documents. Text categorization task is to approximate the unknown target function: $D \times C \rightarrow \{T, F\}$ (that describes how documents ought to be classified), where $D = \{d_1, d_2, d_3, \ldots, d_{|D|}\}$ is a domain of documents and $C = \{c_1, c_2, c_3, \ldots, c_{|C|}\}$ is a set of predefined categories. A value of T assigned to $< d_j, c_i >$ indicates a decision to document d_j under c_i, while a value of F indicates a decision not to document d_j under c_i (Sebastiani, 2002). With the exponential growth of online information, how to organize text data efficiently and effectively has become an important problem. Text categorization is a significant tool to solve this problem.

In recent years, text categorization techniques have drawn great attention and been widely used in many applications. A number of surveys had comprehensive introductions to text categorization (Sebastiani, 2002; Su, Zhang, & Xu, 2006) and numerous text categorization techniques have been proposed in the literature, for instance, decision tree (Apte, Damerau, Sholom, & Weiss, 1994), KNN (Guo, Wang, & Bell, 2004), Naïve Bayes (NB) (Frank & Bouckaert, 2006), neural network (NNet) (Ruiz & Srinivasan, 2002), (Rocchio, 1971), Support Vector Machine (SVM) (Chen & Hsieh, 2006), centroid-based approaches (Tan, 2008). Meanwhile, text categorization has been widely used in email spam filtering (Blanzieri & Bryl, 2008), opinion mining and sentiment analysis (Ye, Zhang, & Law, 2009).

Nowadays the major challenges of text categorization in solving real-world problems are hierarchical classification, imbalance corpus classification, classifying massive text data efficiently, etc (Su et al., 2006). There is a complicated and interconnected relation between categories in multi-categories classification issue. To further understand this relation, multiple hierarchical text classifications have been explored (Esuli, Fagni, & Sebastiani, 2008; Hao, Chiang, & Tu, 2007). When traditional text categorization algorithm encounters an imbalance document corpus, its performance decreases gravely. In dealing with skewed category distribution, the robustness of KNN and SVM is better than NB, Rocchio and NNet (Yang & Liu, 1999). In order to enhance the robustness of KNN, KNN with adaptive weight adjusted strategies based on the number and distribution of training text samples are proposed. It improves the KNN algorithm performance when dealing with imbalance corpus (Hao, Tao, & Xu, 2009; Tan, 2005).

KNN is a simple but effective method for text categorization, but it has three fatal defects: first, the complexity of its sample similarity computing is huge; second, its performance is easily affected by single training sample, such as noisy sample; third, KNN does not build the classification model since it is a lazy learning method. As a result, it is not suited in many applications well, where data is updated dynamically and required rigorously in real-time performance, such as email spam filtering. Many researchers sought ways to reduce the complexity of KNN, which can be divided into three methods generally: reducing the dimensions of vector text (Vries, Mamoulis, & Nes, 2002); reducing the amount of training samples (Li, & Hu, 2004); expediting the process of finding *K* nearest neighbors (Aghbari, 2005; Wang & Wang, 2007).

In this paper, we propose an improved KNN algorithm for text categorization (INNTC). We obtain classification model by

* Corresponding author. Tel.: +86 20 39328623; fax: + 86 20 39328032.
E-mail address: jiangshengyi@163.com (S. Jiang).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.040

employing constrained one pass clustering algorithm (Jiang ShengYi, 2006), which uses the least distance principle to constrainedly divide training text samples into hyper spheres with almost the same radius (i.e., threshold r). Then, we employ KNN approach to classify the test text collections based on the obtained model. INNTC builds the classification model, which reduces the test similarity computing complexity substantially and whittles the impact of its performance affected by single training sample. Meanwhile, INNTC is an incrementally modeling algorithm, which can update its classification model dynamically. Our empirical results on the Reuters-21578, Fudan Univ. text categorization corpus and the email spam filtering benchmark corpus demonstrate that INNTC significantly outperforms KNN and more effective and robust than NB and SVM.

The rest of this paper is organized as follows: in Section 2 we describe the traditional KNN algorithm and outline the algorithm we propose. Section 3 describes the empirical results. Section 4 discusses and concludes the paper.

2. Improved *K*-nearest-neighbor algorithm for text categorization

KNN and SVM have much better performance than other classifiers (Yang & Liu, 1999). However, KNN is a sample-based learning method, which uses all the training documents to predict labels of test document and has very huge text similarity computation. As a result, it cannot be widely used in real-world applications. To tackle this problem, we propose an improved KNN algorithm for text categorization based on one pass clustering algorithm and KNN algorithm.

2.1. Traditional K-nearest-neighbor algorithm for text categorization

The process of KNN algorithm is as follows: given a test document x, find the K nearest neighbors of x among all the training documents, and score the category candidates based the category of K neighbors. The similarity of x and each neighbor document is the score of the category of the neighbor document. If several of the K nearest neighbor documents belong to the same category, then the sum of the score of that category is the similarity score of the category in regard to the test document x. By sorting the scores of the candidate categories, system assigns the candidate category with the highest score to the test document x. The decision rule of KNN can be written as:

$$f(\mathrm{x}) = \arg\max_j \mathrm{Score}(x, C_j) = \sum_{d_i \in kNN} sim(x, d_i) y(d_i, C_j),$$

where $f(x)$ is the label assigned to the test document x; $\mathrm{Score}(x, C_j)$ is the score of the candidate category C_j with respect to x; $sim(x, d_i)$ is the similarity between x and the training document d_i; $y(d_i, C_j) \in \{0, 1\}$ is the binary category value of the training document d_i with respect to C_j ($y = 1$ indicates document d_i is part of category C_j, or $y = 0$).

This approach is effective, non parametric and easy to implement. However, its classification time is long and the accuracy is severely degraded by the presence of noisy training document.

2.2. Improved K-nearest-neighbor algorithm for text categorization based on clustering

We use VSM (Vector Space Model) (Salton, Wong, & Yang, 1975) to represent the documents. In this model, each document is considered to be a vector in the term-space. Per-word weight of the document is computed using TFIDF (Salton & Buckley, 1988). INNTC obtains classification model by using constrained one pass clustering algorithm. Then, we employ KNN approach to classify the test text collections. The details of our algorithm are described as follows:

2.2.1. Step 1: Build classification model based clustering

Clustering is a process of partitioning data into clusters of similar objects. It is an unsupervised learning process of hidden data. In text clustering, it assumes the similarity degree of the content of the documents in the same cluster is the most, while in different clusters to the least. Therefore, to preprocess the documents using clustering is useful for discovering the distribution and structure of corpus. The state-of-the-art clustering approaches were reported in the thorough survey (Jain, Murty, & Flynn, 1999). A majority of clustering algorithms proposed in previous literatures cannot handle large and high-dimensional data. However, incremental clustering algorithms with less time consuming can deal with it, since they are non-iterative and scan corpus in single pass. One pass clustering algorithm is a kind of incremental clustering algorithm with approximately linear time complexity. To build the classification model with the training text documents, we use one pass clustering algorithm to constrainedly cluster the text collections. The details about the clustering are described as follows:

(1) Initialize the set of clusters m_0, as the empty set, and read a new text p.
(2) Create a new cluster with the p; its label is regarded as the label of the new cluster.
(3) If no texts are left in the text collections, go to (6), otherwise read a new text p, compute the similarities between p and all the clusters $\overline{C}$ in m_0 using the cosine function, and find the cluster C_i^0 in m_0 that is closest to the text p. Namely, find a cluster C_i^0 in m_0, such that $sim(p, C_i^0) \geqslant sim(p, \overline{C})$ for all $\overline{C}$ in m_0.
(4) If $sim(P, C_i^0) < r$ or the label of text p difference from the label of the nearest cluster, go to (2).
(5) Merge text p into cluster (C_i^0) and update the weight of words of cluster C_i^0, go to (3).
(6) Stop clustering, get the clustering results $m_0 = \{C_1^0, C_2^0, C_3^0, \ldots, C_n^0\}$, each cluster in m_0 is consisted of weighted words and cluster label, and m_0 is the classification model.

During the clustering processes, each cluster is represented as cluster vector in accordance with the centroid vector for each cluster. The strategy of updating the words weights of the clusters in step (5) is described as follows:

$$w_{c_i^0}^{j+1}(t) = \frac{w_{c_i^0}^{j}(t) \times |c_i^0| + w(t)_p}{|c_i^0| + 1},$$

where $w_{c_i^0}^{j+1}(t)$ indicates the new weight of word t in cluster C_i^0; $w_{c_i^0}^{j}(t)$ is the weight of word t in cluster C_i^0; $w(t)_p$ is the weight of word t in text p; $|c_i^0|$ is the number of texts contained in cluster C_i^0.

2.2.2. Step 2: Text categorization

Since KNN is an effective method, we integrate KNN method to classify the test documents by using the obtained classification model. The details about the categorization are described as follows: given a test document x, score each cluster in m_0 with respect to x using the following formula, and assign the label of the cluster with the highest score to the test document x.

$$f(x) = \operatorname*{argmax}_{\mathrm{j}} \ \mathrm{ClusterScore}(x, C_j) = \sum_{C_i^0 \in kNN} sim(x, C_i^0) y(C_i^0, C_j),$$

where $f(x)$is the label assigned to the test document x; $\mathrm{ClusterScore}(x, C_j)$ is the score of the candidate category C_j with

respect to x; $sim(x, C_i^0)$ is the similarity between x and the cluster C_i^0 in model m_0; $y(C_i^0, C_j) \in \{0, 1\}$ is the cluster C_i^0 with respect to C_j ($y = 1$ indicates cluster C_i^0 is part of category C_j, or $y = 0$).

2.2.3. Step 3: Updating the classification model

Incrementally update the model $m_0 = \{C_1^0, C_2^0, C_3^0, \ldots, C_n^0\}$ with the new training corpus by the method of setting up the model, and obtain the new classification model.

INNTC builds the classification model using constrained one pass clustering algorithm, it changes the learning way of KNN algorithm. The number of the clusters obtained by constrained clustering is much less than the number of training samples. As a result, when we use KNN method to classify the test documents, the text similarity computation is substantially reduced and the impact of its performance affected by single training sample is also whittled.

Many previous text categorization algorithms are hard to or cannot update their classification model dynamically, such as NB, so they need to rebuild the model if new training documents are given. However, the size of text data is huge and increases constantly in the real-world applications, and rebuilding model is very time-consuming. INNTC is an incrementally modeling algorithm, which can update its classification model quickly based on the new training samples. It is valuable in practical applications.

2.3. Selecting clustering threshold r

The threshold r may influence the quality of clustering and the time-efficiency of the algorithm. As r increases, both the number of produced clusters and time-costing will increase. In order to gain a reasonable and relative stable threshold r, we employ sampling techniques (Jiang, Song, & Hui, 2006) to determine the threshold. The details are described as follows:

Step 1: Choose randomly N_0 pairs of texts in the corpus.
Step 2: Compute the similarities between each pair of texts.
Step 3: Compute the average ex of similarities from Step 2.
Step 4: Selecting r as $\varepsilon \times ex$, where $\varepsilon \geqslant 1$.

When N_0 reaches a higher value, ex remains stable. In our experiments, we choose $N_0 = 8000$. The value of threshold r is closely related to the size of corpus and its application areas. The empirical results show that it gets a high quality clustering results and good classification performance when ε is selected in [3, 10], the detail about the value of ε is given in Chapter 3.1.

3. Evaluations

In this section, we use Reuters-21578, Fudan Univ. text categorization corpus and Ling-Spam email spam filtering corpus to evaluate our algorithm. The environment settings of our empirical computer are as follows: Pentium(R) D CPU 2.80 GHz, 2.79 GHz, 1.00 GB; Operation System is Microsoft Windows XP SP2.

Many evaluation metrics in text categorization are used (Sebastiani, 2002). We use F_1 and *Macro-*F_1 (Lewis, Schapire, & Callan, 1996; Tan, 2005) to measure our algorithm performance. The details about F_1 and *Macro-*F_1 are described as follows:

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

where F_1 combines recall (r) and precision (p) into a single measure, *Macro-*F_1 is the average of all individual category F_1 values. We use F_1 and *Macro-*F_1 to evaluate the classifier performance for individual category and the whole corpus respectively.

In the email spam filtering, weighted accuracy (WAcc), total cost ratio (TCR), spam recall (SR), spam precision (SP), SF_1 (the combination of SR and SP) are widely used to evaluate the spam filter performance (Androutsopoulos, Koutsias, & Chandrinos, 2000a; Androutsopoulos, Paliouras, & Karkaletsis, 2000b; Koprinska, Poon, & Clark, 2007; Sakkis, Androutsopoulos, & Paliouras, 2001). The details about the metrics are described as follows:

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

where N_S and N_L are corresponding to the number of spam messages and legitimate messages; $N_{Y\to Z}$ is the number of messages in category Y that the filter classified as Z; $WErr$, $WAcc^b$ and $WErr^b$ is 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 cost sensitive ratio. Since our algorithm is not a cost-sensitive classifier, in our experiment λ is assigned to 1.

3.1. Datasets and parameter settings

Reuters-21578[1] is a standard benchmark for text categorization. The Reuters-21578 collection contains 21578 documents and 135 categories appeared on the Reuters news wire in 1987. 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 stop word list.[2] In this corpus, ε is assigned to 8. The details about the subset we choose are described in Table 1.

The Fudan Univ. text categorization corpus[3] is from Chinese natural language processing group in Department of Computer Information and Technology in Fudan University. The collection contains 20 categories: 9804 training documents and 9833 test documents. The proportion of training documents and test documents is approximately 1:1, and the whole corpus is used in this paper. During preprocessing, we use the ICTCLAS2009 Chinese word segmentation to filter preposition, conjunction, pronoun, interjection, auxiliary particle, particles of speech stop words. In this experiment, ε is assigned to 9. The details about this corpus are described in Table 2.

Ling-Spam[4] corpus 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 in order to facilitate evaluation using 10-fold cross validation. We use the *lemm* version (only stemming) and $\varepsilon = 3$ in this experiments.

3.2. Performance analysis on two text categorization benchmarks

We use F_1 and *Macro-*F_1 to evaluate our algorithm performance on Reuters-21578 and Fudan Univ. text categorization corpus. We have a comparative study with the performance of INNTC and KNN with the different K values, and their best performances are selected to compare with NB and SVM. The classification results on Reuters-21578 and Fudan Univ. text categorization corpus with

[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.

Table 1
The details about the seven most frequent categories and its classification results.

Categoryname	Train	Test	INNTC ($K = 45$)			KNN ($K = 10$) $F1$ value	NB $F1$ value	SVM $F1$ value
			Cluster vectors	Compression ratio	$F1$ value			
ACQ	1650	719	686	58.42%	0.9360	0.8998	0.9674	0.9608
Corn	181	56	41	77.35%	0.9126	0.8870	0.9444	0.8807
Crude	389	189	119	69.41%	0.7821	0.8235	0.8429	0.7930
Earn	2877	1087	915	68.20%	0.9643	0.9500	0.9843	0.9799
Interest	347	131	70	79.83%	0.9143	0.8945	0.9231	0.9243
Ship	197	89	98	50.25%	0.7364	0.6503	0.6154	0.6115
Trade	369	117	120	67.48%	0.8872	0.8943	0.8561	0.8739
	6010	2388	2049	65.91%	0.8850	0.8597	0.8843	0.8628

Table 2
The details about the Fudan Univ. corpus and its classification results.

Category name	Train	Test	INNTC($k = 10$)			KNN ($k = 5$) $F1$ value	NB $F1$ value	SVM $F1$ value
			Cluster vectors	Compression ratio	$F1$ value			
C11-Space	640	642	199	68.91%	0.9038	0.8927	0.8967	0.9406
C15-Energy	32	33	18	43.75%	0.6250	0.6154	0.0000	0.5000
C16-Electronics	27	28	20	25.93%	0.4444	0.5641	0.0000	0.4500
C17-Communication	25	27	18	28.00%	0.7755	0.7556	0.0714	0.6250
C19-Computer	1357	1358	469	65.44%	0.9500	0.9443	0.9540	0.9626
C23-Mine	33	34	23	30.30%	0.7586	0.6349	0.0000	0.7317
C29-Transport	57	59	41	28.07%	0.8929	0.7429	0.0656	0.7250
C3-Art	740	742	191	74.19%	0.9237	0.9244	0.8683	0.9217
C31-Enviornment	1217	1218	388	68.12%	0.8863	0.8692	0.9166	0.9474
C32-Agriculture	1021	1022	196	80.80%	0.9337	0.9221	0.8689	0.9363
C34-Economy	1600	1601	302	81.13%	0.8836	0.8980	0.8757	0.9231
C35-Law	51	52	39	23.53%	0.8548	0.8880	0.0000	0.4516
C36-Medical	51	53	41	19.61%	0.6022	0.6190	0.0000	0.4746
C37-Military	74	76	45	39.19%	0.7500	0.6429	0.0260	0.6458
C38-Politics	1024	1026	273	73.34%	0.5263	0.5574	0.8674	0.8851
C39-Sports	1253	1254	339	72.94%	0.8585	0.8463	0.9002	0.9501
C4-Literature	33	34	30	9.09%	0.3256	0.2051	0.0000	0.1111
C5-Education	59	61	46	22.03%	0.4211	0.1408	0.0000	0.4286
C6-Philosophy	44	45	27	38.64%	0.5507	0.4928	0.0000	0.5205
C7-History	466	468	265	43.13%	0.6806	0.6555	0.6503	0.7626
	9804	9833	2970	69.71%	0.7512	0.7185	0.4590	0.7011

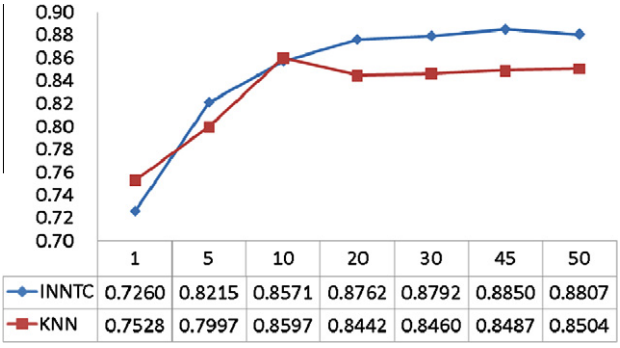

Fig. 1. The classification results with different k values in Reuters-21578.

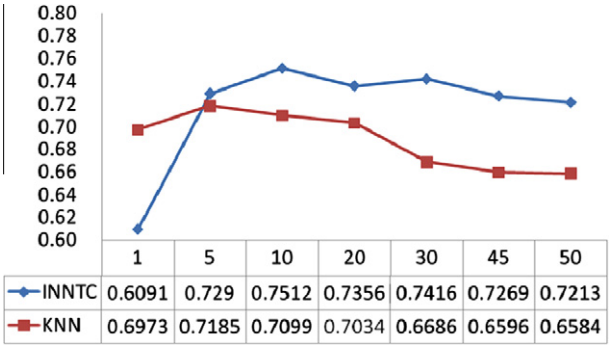

Fig. 2. The classification results with different k values in Fudan Univ. corpus.

the different K values are shown in Fig. 1 and 2. The comparisons with NB and SVM are shown in Tables 1 and 2. The results listed in Tables 1 and 2 and Figs. 1 and 2 are the best results we get for each algorithm in our experiments.

Fig. 1 shows that the INNTC classifier performance is much better than KNN classifier in most of the different K values. It has about 2.16% improvement in average performance. The KNN performance exceeds INNTC only when K is 1 and 10. However, K is not selected as 1 in KNN, since the classification result of KNN is easily affected by noisy samples, and the *Macro-F_1* value of KNN is slightly better than INNTC when K is 10. The INNTC and KNN achieve their best classification results when K is 45 and 10 respectively, in that case, the INNTC *Macro-F_1* value is 88.50%, 2.54% higher than KNN.

Fig. 2 demonstrates that INNTC classifier significantly outperforms KNN classifier in most of the different K values except K is 1. It has about 4.13% improvement in average performance. INNTC reaches the best result 75.12%, 3.27% higher than KNN.

In Table 1, the four classifiers all perform well in most individual categories and INNTC gets the largest *Macro-F_1* value. The results indicate that INNTC has the best performance over other three classifiers on the whole.

In Table 2, the results show that INNTC performs much better than KNN and NB on most individual categories and yields better

Table 3
The comparison of INNTC, *k*NNModel and Rocchio algorithms.

Category name	INNTC	Rocchio	kNNModel
Acq	0.9360	0.8403	0.8608
Corn	0.9126	0.8774	0.9185
Crude	0.7821	0.8451	0.8472
Earn	0.9643	0.9039	0.8945
Interest	0.9143	0.8084	0.8326
Ship	0.7364	0.8622	0.8673
Trade	0.8872	0.8164	0.8000
Macro *F*1	0.8850	0.8505	0.8601

performance than SVM on most minority categories, it indicates that INNTC has better generalization ability than SVM. Since NB extremely depends on the category prior probability, it has the worst performance in this skewed corpus. Note that INNTC attains the largest *Macro-*F_1 value, 3.27%, 29.22%, and 5.01% higher than KNN, NB and SVM.

In Tables 1 and 2, the "Cluster vectors" sub-column of INNTC column lists the number of clusters in classification model m_0 achieved by Step 1. "Compression ratio" sub-column lists the compression ratio between cluster vectors and training documents. The results illustrate that INNTC obtains small *F*1 value with low compression ratio on most minority categories. In large-scale corpus with imbalance multi-categories, such as Fudan Univ. text categorization corpus, INNTC has large movements in the compression ratios and *F*1 values of the categories While INNTC classifier has relatively stable performances on Reuter corpus.

Table 3 demonstrates the comparison of INNTC, Rocchio and *k*NNModel algorithms. *k*NNModel is an improved KNN text categorization by integrating standard KNN and Rocchio algorithm (Guo et al., 2004). The results of Rocchio and *k*NNModel are from (Guo et al., 2004), while its experiment is based on balance corpus (it chooses 200 documents for each category of seven most frequent Reuters categories to constitute their experimental corpus). We can find that INNTC has much better performance than *k*NNModel and Rocchio. Note that INNTC gets this classification result upon extreme imbalance corpus with respect to the balance corpus used by Guo.

In Table 4, the total time cost of INNTC classification upon Fudan Univ. corpus is approximately 66% of KNN. INNTC total classification time on Reuter corpus exceeds KNN for the following reason: the test documents scale of Reuter corpus is very small with respect to its relative large-scale training documents, so that INNTC takes much more time to build the classification model than the classification time. However, we can build the classification model off-line, while the classification stage requires rigorously in its efficiency in real-world applications. Comparing with KNN, INNTC just takes about 59% and 46% time averagely to classify the test documents of Reuter corpus and Fudan Univ. corpus with the different *K* values. The results illustrate that INNTC is more efficient in classify stage and has much better applicability and scalability than KNN.

3.3. Performance analysis on Ling-Spam benchmark

The number of messages in Ling-Spam is small compared to Reuters-21578 and Fudan Univ. text categorization corpus. Here, we use 10-fold stratified cross validation (Kohavi, 1995) to increase the confidence of experimental results. Our experiments show that INNTC achieves the best results when *K* is 10, so we have a comparison with INNTC (*K* = 10), Random Forest (RF), Decision Trees (DT), KNN (TiMBL), NB, SVM, Stacking and Outlook Patterns spam filters on Ling-Spam email spam corpus.

Table 5
The details about the Ling-Spam corpus and its filtering results.

Category name	Origin messages			Cluster vectors	Compact ratio	*F*1
	Train	Test	Total			
Legitimate messages	2171	241	2412	936	56.89%	0.9960
Spam messages	433	48	481	141	67.44%	0.9647
Total	2604	289	2893	1097	57.87%	0.9876

Table 6
The 10-cross validation classification results of eight classifiers on Ling-Spam[1].

Algorithm	SR	SP	SF1	Acc.	TCR
INNTC	0.9647	0.9958	0.9800	0.9934	25.33
NB	0.8235	0.9902	0.8992	0.9693	5.41
Outlook Patterns	0.5301	0.8793	0.6614	0.9098	1.84
TiMBL (10-NN)	0.3454	0.9964	0.5130	0.8908	1.52
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
Stacking	0.9170	0.9650	0.9404	N/A	8.44
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	–	–	–	0.8337	1.00

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

Table 5 lists the constrained one pass clustering and classification results on Ling-Spam. The total compression ratio and macro average *F*1 reach to 57.87% and 0.9876 respectively. The results illustrate that INNTC performs steadily with relative stable compression ratios and *F*1 values.

Table 6 demonstrates that INNTC classifier achieves impressive SR and SP, so that it significantly outperforms Outlook patterns, Stacking, RF, DT, KNN, NB, SVM classifiers over SF_1, accuracy and TCR.

To measure the classification performance of INNTC incremental modeling and its applicability, we build the classification model incrementally on Ling-Spam benchmark and also use 10-fold stratified cross validation to evaluate it. The details about the incremental modeling experiment are described as follows:

Step 1: Choose randomly one unused fold messages of the corpus (ten folds) as the test message samples and the rest nine folds messages are used as the training message samples.
Step 2: If nine folds training message samples are all used for training, go to Step 5.
Step 3: Select one unused fold of the training message samples to train (build or update) the classification model.

Table 4
The efficiency test of INNTC and KNN.

Corpus	Reuters-21578		Fudan Univ. corpus	
Time cost (minutes)	Modeling time + Classification time	Total	Modeling time + Classification time	Total
INNTC	3.18 + 2.82	6.00	66.75 + 155.8	222.55
KNN	0.00 + 4.86	4.86	0.00 + 339.33	339.33

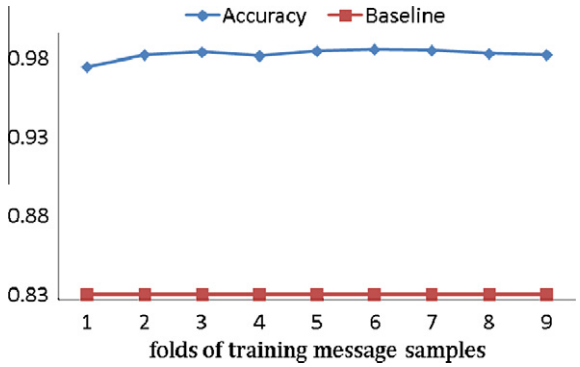

Fig. 3. The INNTC classification accuracy of incremental modeling on Ling-Spam.

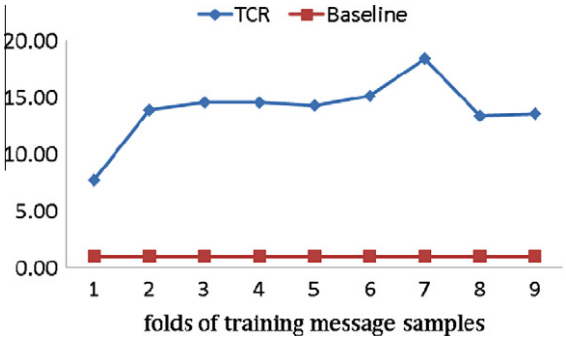

Fig. 4. The INNTC TCR value of incremental modeling on Ling-Spam corpus.

Step 4: Use the test message samples to evaluate the classification model, go to Step 2.
Step 5: If one fold of the corpus were unused as the test message samples, go to Step 1.
Step 6: Stop.

Figs. 3 and 4 list the accuracy and TCR for each phase in the incremental modeling. The results demonstrate that the accuracy and TCR of INNTC classifier significantly exceed the baseline accuracy and TCR in each phase.

Fig. 5 shows that the number of clusters in the classification model steadily increases with the growth of the scale of training message samples. On the contrary, the amount of new appended clusters gradually decreases on the whole. The results show that the incremental[5] modeling strategy of INNTC is reasonable and effective, which is valuable in practical applications.

4. Discussion and conclusions

KNN is an effective but less efficient sample-based lazy learning approach. In this paper, we propose a novel, effective and efficient improved KNN algorithm for text categorization INNTC. As categories vary significantly in terms of scope, which means that some categories are essentially a combination of one or more sub-topics or sub-categories, and clustering is a great tool to compress and discover the complex distribution of the training texts. INNTC uses constrained one pass clustering algorithm to capture sub-categories and categories relationship by the constrained condition (each cluster only contains one label). Therefore the clusters obtained by

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

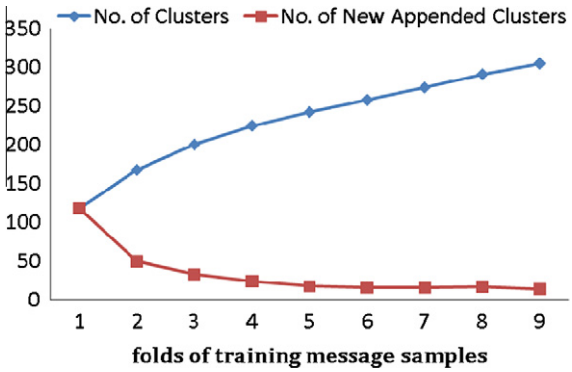

Fig. 5. The variation of the number of clusters and new appended clusters in incremental modeling.

constrainedly clustering are pure clusters, which are regarded as sub-categories, can better reflect the complex distributions of the training texts than original text samples. INNTC classifies test documents based on the cluster vectors instead of original text samples by using KNN approach, this classification way boosts up the effectiveness of KNN approach (in most cases, we achieve higher $F1$ value of the category with large compression ratio). On the other hand, the cluster vectors are the compact representations of the training samples. In our experiments, the compression ratios between the cluster vectors and the training documents fluctuate around 65%. When INNTC classifies test documents based on the cluster vectors, it can significantly enhance the classification efficiency of KNN approach.

As integrating the advantages of the constrained one pass clustering and KNN approach, INNTC has significant performance comparing with KNN, NB, SVM and other classifiers in dealing with large-scale, high-dimensional and imbalance text data. The conclusions are drawn as follows:

(1) In the experiments of English and Chinese text categorization benchmarks, INNTC classifier is much more effective and efficient than KNN. Meanwhile, INNTC has much better performance and good generalization ability than NB and SVM, especially in the Fudan Univ. text categorization corpus.
(2) Many researches show that Stacking, RF, DT, KNN, NB and SVM classifiers are favorable for email spam filtering and have been widely used in this area. In our experiment on Ling-Spam corpus, INNTC consistently outperforms Outlook patterns, Stacking, RF, DT, KNN, NB, SVM classifiers over SF_1, accuracy and TCR, especially the TCR metrics. It indicates that INNTC has significant superiority over other classifiers in email spam filtering.
(3) In the real world, many text categorization applications require the classification model can be updated incrementally, such as email spam filtering, topic tracking, etc. The INNTC classification model can be incrementally updated, which receives a better performance on Ling-Spam corpus. This character greatly raises its applicability and scalability in real-world applications.
(4) Classification on imbalance data is now a major challenge in classification analysis. In the experiments on the three imbalance corpora INNTC gains significant performance, which has better performance and is much more robust than the state-of-the-art KNN, NB and SVM classifiers, especially on the minority categories.
(5) INNTC builds and updates the classification model using constrained one pass clustering algorithm. One pass clustering is a kind of incremental clustering algorithm, so the classification model can be built and updated in nearly linear time complexity.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 60673191), the Natural Science Research Programs of Guangdong Province's Institutes of Higher Education (No. 06Z012) and the National Natural Science Foundation of Guangdong Province of China (No. 9151026005000002).

References

Aghbari, Z. A. (2005). Array-index: A plug & search *K* nearest neighbor's method for high-dimensional data. *Data & Knowledge Engineering, 52*(3), 33–352.

Androutsopoulos, I., Koutsias, J., & Chandrinos, K. V. (2000a). An evaluation of Naive Bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona* (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).

Apte, Chidanand., Damerau, Fred., Sholom, M., & Weiss (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS), 12*(3), 233–251.

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

Chen, R. C., & Hsieh, C. H. (2006). Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications, 31*(2), 427–435.

Esuli, A., Fagni, T., & Sebastiani, F. (2008). Boosting multi-label hierarchical text categorization. *Information Retrieval, 11*(4), 287–313.

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

Guo, G., Wang, H., & Bell, D. (2004). kNN model-based approach and its application in text categorization. *Computational Linguistics and Intelligent Text Processing, LNCS, 2945*, 559–570.

Hao, P. Y., Chiang, J. H., & Tu, Y. K. (2007). SVM classification based on support vector clustering method and its application to document categorization. *Expert Systems with Applications, 33*(3), 627–635.

Hao, Xiulan., Tao, Xiaopeng., & Xu, Hexiang. (2009). A strategy to class imbalance problem for kNN text classifier. *Journal of Computer Research and Development, 46*(1), 52–61.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys, 31*(3), 264–323.

Jiang ShengYi., (2006). Efficient Classification Method for Large Dataset. In *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian* (pp. 1190–1194).

Jiang, ShengYi, Song, Xiaoyu, & Hui, Wang (2006). A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letters, 27*(5), 802–810.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann* (pp. 1137–1143).

Koprinska, I., Poon, J., & Clark, J. (2007). Learning to Classify E-Mail. *Information Sciences, 177*, 2167–2187.

Lewis, D. D., Schapire, R. E., & Callan, J. P. (1996). Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 298–306).

Li, RongLu, & Hu, YunFa (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.

Rocchio, J. (1971). Relevance feedback in information retrieval. *The SMART System*, 67–88.

Ruiz, M. E., & Srinivasan, P. (2002). Hierarchical text categorization using neural networks. *Information Retrieval, 5*(1), 87–118.

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).

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM, 18*(11), 613–620.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 24*(5), 513–523.

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

Su, JinShu, Zhang, BoFeng, & Xu, Xin (2006). Advances in machine learning based text categoriza-tion. *Journal of Software, 17*(9), 1848–1859.

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

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 k-NN search on vertically decomposed data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data, Madison, WI, USA* (pp. 322–333).

Wang Yu., & Wang ZhengOu. (2007). A Fast KNN Algorithm for text categorization. In *Proceedings of the 6th International Conference on Machine Learning and Cybernetics, Hong Kong* (pp. 3436–3441).

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), Berkeley* (pp. 42–49).

Ye, Q., Zhang, Z., & Law, Rob. (2009). Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Systems with Applications, 36*(3), 6527–6535.