# Related Text Discovery Through Consecutive Filtering and Supervised Learning

Daqing Wu and Jinwen Ma(✉)

Department of Information Science, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, People's Republic of China
wudq@pku.edu.cn, jwma@math.pku.edu.cn

**Abstract.** In a related or topic-based text discovery task, there are often a small number of related or positive texts in contrast to a large number of unrelated or negative texts. So, the related and unrelated classes of the texts can be strongly imbalanced so that the classification or detection is very difficult because the recall of positive class is very low. In order to overcome this difficulty, we propose a consecutive filtering and supervised learning method, i.e., consecutive supervised bagging. That is, in each consecutive learning stage, we firstly delete some negative texts with the higher degree of confidence via the classifier trained in the previous stage. We then train the classifier on the retained texts. We repeat this procedure until the ratio of the negative and positive texts becomes reasonable and finally obtain a tree-like filtering and recognition system. It is demonstrated by the experimental results on 20NewsGroups data (English data) and THUCNews (Chinese data) that our proposed method is much better than AdaBoost and Rocchio.

**Keywords:** Related text discovery · Text filtering · Imbalanced data Bagging · Logistic regression

## 1 Introduction

Related text discovery or filtering is a process of matching an incoming text stream to a topic or profile of user's interests to detect or recommend the texts according to that topic or profile [1]. So, it is just a binary text classification which divides the input texts into two categories 'related' and 'unrelated'. In fact, this text filtering problem has been investigated from two different communities: machine learning (ML) and information retrieval (IR) [5].

In the IR community, most studies are based on Rocchio algorithm [6], which was developed under the framework of the vector space model. If all the texts are ranked for a query to the topic or profile, an ideal query should rank all the related texts above all unrelated texts. In Rocchio algorithm, an optimal query is defined to maximize the difference between the average scores of the related

© IFIP International Federation for Information Processing 2018
Published by Springer Nature Switzerland AG 2018. All Rights Reserved
Z. Shi et al. (Eds.): ICIS 2018, IFIP AICT 539, pp. 211–220, 2018.
https://doi.org/10.1007/978-3-030-01313-4_22

and unrelated texts. In this way, an optimal query vector is the difference vector of the centroid vectors for the related and unrelated texts: $Q_{opt} = \frac{1}{R}\sum_{i \in Rel} t_i - \frac{1}{N-R}\sum_{i \notin Rel} t_i$, where $t_i$ denotes the weighted term vector for text $i$, $R = |Rel|$ is the number of related texts in $Rel$, and $N$ is the total number of texts in the data set. All the negative components of the resulting query are assigned a zero weight. According to the actual performance, the recall of Rocchio algorithm is high, but its precision is very low, which means that the retrieved texts do not contain many related texts.

In the ML community, the most successful method is the boosting algorithm, especially the AdaBoost algorithm [9]. Schapire [7] proposed the boosting method and further proved that a weak learner can be turned into a strong learner in the sense of probably approximately correct learning framework. Actually,

AdaBoost is the most representative algorithm for text classification.

In fact, AdaBoost uses the whole training data to train each classifier serially, but after each round, it brings about more impact to difficult instances, with the goal of correctly classifying examples in the next iteration that are incorrectly classified during the current iteration. Hence, it gives more impact to examples that are harder to classify, the quantity of impact is measured by a weight, which is initially equal for all instances. After each iteration, the weights of misclassified instances are increased; on the contrary, the weights of correctly classified instances are decreased. Furthermore, another weight is assigned to each individual classifier depending on its overall accuracy which is then used in the test phase, i.e. more confidence is given to more accurate classifiers. Finally, when a new instance is submitted, each classifier gives a weighted vote, and the class label is selected by majority [8].

From the actual performance of the Adabost algorithm, most of retrieved texts are related, but the number of retrieved texts is very small, that is, it loses many texts that the user wants. In recent years, Liu et al. [2] constructed some different classifiers with contextual features to complete the ensemble learning for the text filtering; Lu et al. [4] utilized the RNN with attention mechanism and combine it; Kang et al. [3] built the text-based hidden Markov models and aggregated such models to obtain a final classifier. However, these methods have not paid attention to the ratio of the negative and positive examples.

Generally speaking, we want to know the conditional probability of the form:

$$Pro(y = 1|\mathbf{w}, \mathbf{x}) \tag{1}$$

from a set of training examples $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$.

For a given classifier, the parameters are generally estimated by the principle of maximum likelihood estimation (MLE):

$$\max L = \max \Pi_{i=1}^{N} (Pro(y = 1|\mathbf{w}, \mathbf{x}))^{y_i} (1 - Pro(y = 1|\mathbf{w}, \mathbf{x}))^{1-y_i} \tag{2}$$

If the number of the negative texts is far greater than that of the positive texts, we will get a very weak classifier by MLE. Therefore, the key step is to find a good ratio of the negative and positive texts. We will solve this problem by the consecutive filtering procedure with the supervised learning process.

## 2 Methodology

The concept of bootstrap aggregating (bagging) [10] is useful to construct the ensemble. It consists in training different classifiers with bootstrapped replicas of the original training data. That is, a new data is formed to train each classifier by randomly drawing (with replacement) instances from the original data (usually, maintaining the original data size). Hence, the diversity is obtained with the resampling procedure by the usage of different data subsets. Finally, when an unknown instance is presented to each individual classifier, a majority or weighted vote is used to infer the class membership. However, this random undersampling method ignores the supervised information in the training data. Moreover, this is a fusion method based on certain local classifiers. Here, we try to use the supervised learning method to consecutively filter out the negative texts with the higher degree of confidence so that the ratio of the negative and positive texts decreases stably. In this way, we bag the texts with a better ratio of the negative and positive ones consecutively and finally leads to a tree-like bagging or filtering and recognition system.

Unlike Bagging's classifier $H_T(x) = sign(\sum_{t=1}^{T} h_t(x))$, where $h_t(x)$ is the trained classifier in the $t^{\text{th}}$ stage.

We retain a series of classifiers: $H_T = \{h_1, h_2, \cdots, h_T\}$ with the corresponding thresholds $\Delta_T = \{\delta_1, \delta_2, \cdots, \delta_T\}$. In each previous or bagging stage, the classifier can remove the examples that has been classified into the unrelated texts with the threshold. In the final stage, the classifier makes the final decision of the related text. Algorithm 1 shows the pseudocode of the consecutive supervised bagging (CSB) algorithm.

**Algorithm 1.** Consecutive Supervised Bagging

**Input:** A set of positive class examples $\mathcal{P}$.
A set of negative class examples $\mathcal{N}$.
The finally ratio of negative class and positive class, $\alpha$.
Sampling speed factor, $\lambda$(generally choose 0.1).

1: $i = 1$
2: **repeat**
3: $P_i = |\mathcal{P}|$, $N_i = |\mathcal{N}|$, $\alpha_i = \frac{P_i}{N_i}$
4: Learn classifier $h_i$ using $\mathcal{P}$ and $\mathcal{N}$
5: Calculate the number of examples that will be removed from $\mathcal{N}$:

$$n_i = \text{int}(\lambda \cdot \alpha_i \cdot P_i)$$

6: Through the classifier $h_i$, select the $n_i^{\text{th}}$ small $Pro(y = 1|\mathbf{x}_i \in \mathcal{N})$ as threshold $\delta_i$
7: Remove from $\mathcal{N}$ all examples that satisfy $Pro(y = 1|\mathbf{x}_i \in \mathcal{N}) \leq \delta_i$
8: $i \leftarrow i + 1$
9: **until** $\alpha_i \leq \alpha$

**Output:** An ensemble classifier $H_T = \{h_1, h_2, \cdots, h_T\}$with thresholds $\Delta_T = \{\delta_1, \delta_2, \cdots, \delta_T\}$($T$ is the number of iterations).

In Line 5, the number of removed examples $n_i$ is changed, and decreases in the direct proportion to the ratio of the negative and positive data $\alpha_i$ in $i^{\text{th}}$ iteration. In Line 6, we rank the prediction probabilities from small to large through the classifier $h_i$, and select the probability of $n_i^{\text{th}}$ small negative example. The reason for Line 6 is that the positive examples are so few, and the number of positive examples which prediction probabilities are lower than $\delta_i$ is almost 0. When select $\delta_i$ as threshold, almost positive examples can be retained. In Line 7, the number of removed examples may be higher than $n_i$. Because, all negative examples that have the same probabilities equal to $\delta_i$ will be removed in $i^{\text{th}}$ iteration. Through the CSB, we get $H_T = \{h_1, h_2, \cdots, h_T\}$ with thresholds $\Delta_T = \{\delta_1, \delta_2, \cdots, \delta_T\}$.

In Baaging, when an unknown instance is presented to each individual classifier, a majority or weighted vote is used to infer the class. But, we take another way to predict an unknown instance in CSB. In each iteration, instances which have been predicted to positive class through classifier $h_{i-1}$ can be send to classifier $h_i$. So the finally results of instances is

$$y_i^{pred} = \begin{cases} 1, \text{ if } i \in I^{(T)} \text{ and } y_i^{(T)} = 1 \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{3}$$

where, $T$ is the number of all iterations, $I^{(T)}$ is the instances index set in the $T^{\text{th}}$ iteration and $y_i^{(T)}$ is prediction of $\mathbf{x}_i$ in the $T^{\text{th}}$ iteration. This criterion means that the last prediction which is positive can be predicted positive. Figure 1 shows the process of prediction.

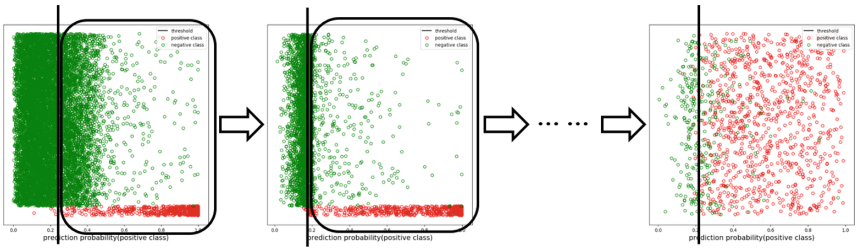


**Fig. 1.** The black vertical line represents the threshold in each classification. In the black box, all instances can be predicted by the next classifier.

Until now, we have not solved how to find a good ratio of the negative and positive examples. The Ratio Adjustment algorithm which is based on CSB can easily solve this problem. Algorithm 2 shows the pseudocode for the Ratio Adjustment.

**Algorithm 2.** Ratio Adjustment

**Input:** A set of positive class examples $\mathcal{P}$.
A set of negative class examples $\mathcal{N}$.
Sampling speed factor, $\lambda$(generally choose 0.1).
Adjust speed factor, $\mu$(generally choose 0.1).

1: $i = 1$, $\alpha_i = 1$, $\text{marker}_i = 0$
2: **repeat**
3: $H_{T_i}$, $\Delta_{T_i}$ = **Supervised Undersampling**$(\mathcal{P}, \mathcal{N}, \alpha_i)$
4: **for all** $j = 1$ to $T_i$ **do**
5: $f_j$ = **Prediction**$(\mathcal{P} \cup \mathcal{N}, H_{T_i}, \Delta_{T_i}, j)$
6: **end for**
7: **if** $\{f_1, f_2, \cdots, f_j\}$ firstly increase, then decrease **then**
8: $\alpha_i = \alpha_i + \mu$, $\text{marker}_i = 1$
9: **else**
10: $\alpha_i = \alpha_i - \mu$, $\text{marker}_i = -1$
11: **end if**
12: **until** $\text{marker}_i \cdot \text{marker}_{i-1} = -1$

**Output:** The best ratio of positive class and negative class(in training data), $\alpha^*$

In Line 5, $f_j$ is f-measure of positive class. In Line 7–10, we make sure that the trend of $\{f_1, f_2, \cdots, f_{T_i}\}$ has only two ways. One is that $f_j$ firstly increases, and then decreases. Another is that $f_j$ increases. Because, the ratio determines the number of iterations. The ratio decreases, the number of iterations increases, more positive instances are judged to negative. Figure 2 shows this phenomenon.

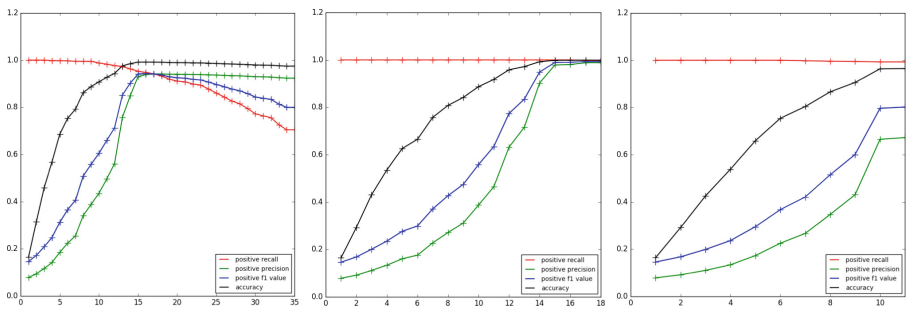


**Fig. 2.** The abscissa axis is the number of iterations $T$. Blue line is the positive class f-measure, red line is the positive class recall, green line is the positive class precision and black line is the accuracy. The left figure is $\alpha = 0.4$ and $T = 35$: The middle figure is $\alpha = 2.5$ and $T = 18$; The right figure is $\alpha = 5$ and $T = 11$. $\alpha = 2.5$ is appropriate, $\alpha = 5$ makes overly filtering and $\alpha = 0.5$ is just on the contrary.

# 3 Experimental Results

## 3.1 Data Sets

We use two standard text categorization test collections or data sets for our experiments. One is English texts–20NewsGroups, while another is Chinese texts–THUCNews.

1. **20NewsGroups** consists of roughly equal-sized samples of postings to 20 Usenet newsgroups. Some postings may belong to more than 1 newsgroup, and we treat the data as specifying 20 binary classification problems(One class as positive class, the others as negative class). We use the 'by date' training/test split (see http://people.csail.mit.edu/people/jrennie/20Newsgroups/) giving 11,314 training documents and 7,532 test documents.

2. **THUCNews** consists of 14 corpus. We treat the data as specifying 14 binary classification problems.

Table 1 shows the detail information of data sets.

**Table 1.** Basic information of 20NewsGroups and THUCNews

| | 20NewsGroups | | | THUCNews | | |
|---|---|---|---|---|---|---|
| class | name | train | test | name | train | test |
| 1 | alt.atheism | 480 | 319 | sports | 1000 | 500 |
| 2 | comp.graphics | 584 | 389 | entertainment | 1000 | 500 |
| 3 | comp.os.ms-windows.misc | 591 | 394 | home | 1000 | 500 |
| 4 | comp.sys.ibm.pc.hardware | 590 | 392 | lottery | 1000 | 500 |
| 5 | comp.sys.mac.hardware | 578 | 385 | house property | 1000 | 500 |
| 6 | comp.windows.x | 593 | 395 | education | 1000 | 500 |
| 7 | misc.forsale | 585 | 390 | fashion | 1000 | 500 |
| 8 | rec.autos | 594 | 396 | politics | 1000 | 500 |
| 9 | rec.motorcycles | 598 | 398 | constellation | 1000 | 500 |
| 10 | rec.sport.baseball | 597 | 397 | games | 1000 | 500 |
| 11 | rec.sport.hockey | 600 | 399 | society | 1000 | 500 |
| 12 | sci.crypt | 595 | 396 | technology | 1000 | 500 |
| 13 | sci.electronics | 591 | 393 | stock | 1000 | 500 |
| 14 | sci.med | 594 | 396 | finance and economics | 1000 | 500 |
| 15 | sci.space | 593 | 394 | | | |
| 16 | soc.religion.christian | 599 | 398 | | | |
| 17 | talk.politics.guns | 546 | 364 | | | |
| 18 | talk.politics.mideast | 564 | 376 | | | |
| 19 | talk.politics.misc | 465 | 310 | | | |
| 20 | talk.religion.misc | 377 | 251 | | | |

## 3.2 Text Representation

First, we break texts into tokens, that is, individual terms, such as words or phrases. Then, we tabulate the number of occurrences of each distinct term $t$ in a text $i$ and across texts. Finally, we compute a numeric weight for each term with respect to each text.

Now, we represent each text as a vector of such weights. We use a form of TF-IDF (term frequency times inverse document frequency)[11] term weighting.

$$TF_{ij} = \frac{n_{i,j}}{\sum_{i=1}^{|V|} n_{i,j}} \quad IDF_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

$$TFIDF_{i,j} = TF_{i,j} \times IDF_i$$

where, $n_{i.j}$ is the number of occurrences of term $i$ (term $i$ is in vocabulary $V$.) in a text $j$, $|V|$ is the total number of vocabulary $V$ (the length of vocabulary can be choose with feature selection.), $|D|$ is the number of training texts, $\{j : t_i \in d_j\}$ is the number of texts that contains term $i$.

### 3.3 Feature Selection

The chi-squared measure [12] chooses the features that are least independent from the class label and is widely used in text categorization. This measure is based on a $2 \times 2$ contingency table between a predictor term $t$ and a predicted category label $s$. Let $a$ be the number of training texts in $s$ and containing $t$, let $b$ be the number in $s$ but not containing $t$, let $c$ be the number not in $s$ but containing $t$, and let $d$ be the number neither in $s$ nor containing $t$. Let $n = a + b + c + d$ be the total number of training texts. Then the chi-squared measure is

$$\chi^2 = \frac{n(ad - bc)^2}{(a + b)(c + d)(a + c)(b + d)}$$

**Table 2.** Confusion matrix

| | Positive class | Negative class |
|---|---|---|
| Retrieval | True positive (TP) | False positive (FP) |
| Not retrieval | False negative (FN) | True negative (TN) |

### 3.4 Performance Measures

From confusion matrix (Table 2), we have following measures:

$$recall = \frac{TP}{TP + FN} \quad precision = \frac{TP}{TP + FP}$$

$$f\text{-}measure = \frac{2 \times recall \times precision}{recall + precision}$$

The recall measures the ratio of correctly classified positive instances and all positive instances. The precision measures the ratio of correctly classified positive instances and all instances which are classified as positive. The f-measure

metric combines precision and recall as a single measurement to indicate the effectiveness of a classifier [13].

For multiple classification, we usually use *macro-recall*, *macro-precision* and *macro-f*:

$$macro\text{-}recall = \frac{1}{n}\sum_{i=1}^{n} recall_i \quad macro\text{-}precision = \frac{1}{n}\sum_{i=1}^{n} precision_i$$

$$macro\text{-}f = \frac{2 \times macro\text{-}recall \times macro\text{-}precision}{macro\text{-}recall + macro\text{-}precision}$$

### 3.5 Parameter Setting

In our experiments, we choose logistic regression as basic classifier with log loss function and L1 penalty. In Algorithms 1 and 2, Sampling speed factor $\lambda = 0.1$ and Adjust speed factor $\mu = 0.1$. In vocabulary, we choose 5000 terms with high chi-squared measure.

### 3.6 Performances and Comparisons

From Table 3, we can see that our method is faster than Rocchio and AdaBoost in training and testing. Because, as the number of iterations increases, the number of retained data which will used to train classifier decreases.

The experimental results well reflect the properties of Rocchio and AdaBoost: Rocchio method's recall is high, but the precision is very low. AdaBoost method's precision is high, but the recall is low. In Table 4, the results show that our proposed model's macro-recall is a little lower than Rocchio, but the macro-precision is significantly lager than Rocchio in 20NewsGroups. Although, our model is better than Rocchio and AdaBoost in 20NewsGroups, there are still some classes that are not very good, like class 2,12,19,20. we guess some instances (postings) can belong to more than one class (1 newsgroup). In THUCNews, we found that our proposed model outperformed the others in the index of macro-recall, macro-precision and macro-f.

**Table 3.** The comparison of running time(second)

| | Training | | | Testing | | |
|---|---|---|---|---|---|---|
| Data name | Rocchio | Adaboost | Ours | Rocchio | Adaboost | Ours |
| 20NewsGroups | 975.92 | 1003.52 | **891.30** | 82.43 | 81.72 | **80.43** |
| THUCNews | 853.81 | 946.80 | **846.71** | 56.84 | 55.70 | **49.74** |

**Table 4.** The testing results of 20NewsGroups

| | Recall | | | Precision | | | F-measure | | |
|---|---|---|---|---|---|---|---|---|---|
| Class | Rocchio | Adaboost | Ours | Rocchio | Adaboost | Ours | Rocchio | Adaboost | Ours |
| *20NewsGroups* | | | | | | | | | |
| 1 | 0.7492 | 0.4953 | **0.8433** | 0.4121 | 0.7822 | **0.8054** | 0.5317 | 0.6065 | **0.8239** |
| 2 | **0.8740** | 0.4139 | 0.7789 | 0.2794 | 0.6518 | **0.7519** | 0.4234 | 0.5063 | **0.7652** |
| 3 | 0.9213 | 0.4137 | **0.9264** | 0.1899 | 0.6443 | **0.9288** | 0.3148 | 0.5039 | **0.9276** |
| 4 | **0.8801** | 0.3776 | 0.8316 | 0.3200 | 0.6091 | **0.7837** | 0.4694 | 0.4661 | **0.8069** |
| 5 | **0.9013** | 0.5351 | 0.8883 | 0.3788 | 0.8110 | **0.8486** | 0.5334 | 0.6448 | **0.8680** |
| 6 | **0.9316** | 0.5544 | 0.9266 | 0.2471 | 0.7110 | **0.9196** | 0.3907 | 0.6230 | **0.9231** |
| 7 | **0.9308** | 0.5513 | 0.9282 | 0.4421 | 0.8238 | **0.9258** | 0.5995 | 0.6605 | **0.9270** |
| 8 | **0.9141** | 0.5000 | 0.9040 | 0.5552 | 0.7586 | **0.8585** | 0.6908 | 0.6027 | **0.8807** |
| 9 | 0.9347 | 0.7312 | **0.9397** | 0.7686 | **0.9268** | 0.9078 | 0.8435 | 0.8174 | **0.9235** |
| 10 | **0.8992** | 0.6574 | 0.8615 | 0.5107 | 0.7885 | **0.8301** | 0.6515 | 0.7170 | **0.8455** |
| 11 | 0.9674 | 0.7794 | **0.9975** | 0.6380 | 0.9482 | **0.9925** | 0.7689 | 0.8556 | **0.9950** |
| 12 | **0.8283** | 0.3232 | 0.5903 | 0.2965 | 0.5100 | **0.5370** | 0.4251 | 0.3956 | **0.5624** |
| 14 | 0.8359 | 0.5581 | **0.8510** | 0.4117 | **0.8435** | 0.8425 | 0.5517 | 0.6717 | **0.8467** |
| 15 | 0.9086 | 0.7107 | **0.9365** | 0.5967 | 0.9003 | **0.9111** | 0.7203 | 0.7943 | **0.9237** |
| 16 | 0.8970 | 0.7638 | **0.9548** | 0.5518 | 0.8889 | **0.9429** | 0.6833 | 0.8216 | **0.9488** |
| 17 | 0.8819 | 0.5687 | **0.9341** | 0.3929 | 0.6656 | **0.8924** | 0.5436 | 0.6133 | **0.9128** |
| 18 | 0.7846 | 0.6569 | **0.9096** | 0.8967 | **0.9216** | 0.9096 | 0.8369 | 0.7671 | **0.9096** |
| 19 | **0.7516** | 0.4194 | 0.6484 | 0.3807 | 0.6075 | **0.6401** | 0.5054 | 0.4962 | **0.6442** |
| 20 | 0.7092 | 0.2869 | **0.7211** | 0.2132 | 0.4865 | **0.5710** | 0.3278 | 0.3609 | **0.6373** |
| macro- | **0.8626** | 0.5505 | 0.8610 | 0.4584 | 0.7599 | **0.8314** | 0.5987 | 0.6384 | **0.8460** |
| *THUCNews* | | | | | | | | | |
| 1 | 0.9360 | 0.9680 | **1.0000** | 0.8357 | 0.9878 | **0.9980** | 0.8830 | 0.9778 | **0.9990** |
| 2 | 0.9580 | 0.8900 | **0.9980** | 0.8523 | 0.9488 | **0.9980** | 0.9021 | 0.9185 | **0.9980** |
| 3 | **0.9260** | 0.6680 | 0.9160 | 0.6044 | 0.8743 | **0.9034** | 0.7314 | 0.7574 | **0.9160** |
| 4 | 0.9060 | 0.9400 | **0.9920** | 0.9340 | 0.9792 | **0.9861** | 0.9198 | 0.9592 | **0.9890** |
| 5 | 0.9160 | 0.9460 | **1.0000** | 0.6050 | 0.9854 | **1.0000** | 0.7287 | 0.9653 | **1.0000** |
| 6 | 0.5980 | 0.6580 | **0.8920** | 0.8214 | **0.9139** | 0.8124 | 0.6921 | 0.7651 | **0.8503** |
| 7 | 0.9680 | 0.9080 | **0.9820** | 0.5538 | 0.9153 | **0.9840** | 0.7045 | 0.9116 | **0.9830** |
| 8 | 0.9420 | 0.6980 | **0.9640** | 0.6937 | 0.9160 | **0.9659** | 0.7990 | 0.7923 | **0.9650** |
| 9 | 0.9980 | 0.9960 | **1.0000** | 0.9559 | 0.9746 | **1.0000** | 0.9765 | 0.9852 | **1.0000** |
| 10 | 0.9040 | 0.8880 | **0.9860** | 0.8794 | 0.9548 | **0.9554** | 0.8915 | 0.9202 | **0.9705** |
| 11 | 0.9420 | 0.6740 | **0.9700** | 0.5621 | 0.8686 | **0.9700** | 0.7040 | 0.7590 | **0.9700** |
| 12 | **0.9640** | 0.8180 | 0.9600 | 0.6667 | 0.9191 | **0.9449** | 0.7882 | 0.8656 | **0.9524** |
| 13 | 0.9740 | 0.8620 | **0.9980** | 0.5650 | 0.8620 | **0.9881** | 0.7151 | 0.8620 | **0.9930** |
| 14 | 0.7980 | 0.6980 | **0.9060** | 0.7151 | **0.8596** | 0.8118 | 0.7543 | 0.7704 | **0.8563** |
| macro- | 0.9093 | 0.8294 | **0.9689** | 0.7318 | 0.9257 | **0.9513** | 0.8109 | 0.8749 | **0.9600** |

## 4 Conclusion

We have established a consecutive supervised bagging method to related or topic-based text discovery that is adaptive for the dataset without any human intervention. Related or topic-based text discovery is generally a strongly imbalanced

classification problem which is very challenging in both machine learning and information retrieval. Our proposed consecutive supervised bagging method can effectively solve this problem from the consecutively decreasing the ratio of the negative and positive texts with the supervised learning method. The experimental results on 20NewsGroups data (English data) and THUCNews (Chinese data) demonstrate that our proposed method is much better than AdaBoost and Rocchio.

**Acknowledgment.** This work is supported by the Natural Science Foundation of China for Grant 61171138. We also acknowledge Zhengzhou Shuneng Science and Technology Limited Company for the contribution of the data set THUCNews.

## References

1. Soboroff, I., Nicholas, C.: Combining content and collaboration in text filtering. In: IJCAI 1999 Workshop: Machine Learning for Information Filtering, pp. 86–91 (1999)
2. Liu, Y., Jiang, C., Zhao, H.: Using contextual features and multi-view ensemble learning in product defect identification from online discussion forums. Decis. Support. Syst. **105**, 1–12 (2018)
3. Kang, M., Ahn, J., Lee, K.: Opinion mining using ensemble text Hidden Markov Models for text classification. Expert. Syst. Appl. **94**, 218–227 (2018)
4. Lu, Z., Liu, W., Zhou, Y., et al.: An effective approach for Chinese news headline classification based on multi-representation mixed model with attention and ensemble learning. In: National CCF Conference on Natural Language Processing and Chinese Computing, pp. 339–350 (2017)
5. Schapire, R.E., Singer, Y., Singhal, A.: Boosting and Rocchio applied to text filtering. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 215–223 (1998)
6. Rocchio, J.J.: The SMART Retrieval System: Experiments in Automatic Document Processing. Relevance Feedback in Information Retrieval, pp. 313–323 (1971)
7. Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**, 197–227 (1990)
8. Galar, M., Fernandez, A., Barrenechea, E., et al.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans. Syst., Man, Cybern. Part C (Appl. Rev.) **42**(4), 463–484 (2012)
9. Freund, Y., Shapire, R.E.: Experiments with a new boosting algorithm. In: 13th ICML, pp. 148–156 (1996)
10. Breiman, L.: Bagging predictors. Mach. Learn. **24**, 123–140 (1996)
11. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. **24**(5), 513–523 (1988)
12. Zheng, Z., Wu, X., Srihari, R.: Feature selection for text categorization on imbalanced data. ACM SIGKDD Explor. Newslett. **6**(1), 80–89 (2004)
13. Rong, T., Gong, H., Ng, W.W.Y.: Stochastic sensitivity oversampling technique for imbalanced data. In: Wang, X., Pedrycz, W., Chan, P., He, Q. (eds.) ICMLC 2014. CCIS, vol. 481, pp. 161–171. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45652-1_18