Knowledge-Based Systems 100 (2016) 137–144

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Two feature weighting approaches for naive Bayes text classifiers

Lungan Zhang[a], Liangxiao Jiang[a,b,*], Chaoqun Li[c,**], Ganggang Kong[a]

[a] Department of Computer Science, China University of Geosciences, Wuhan 430074, China
[b] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China
[c] Department of Mathematics, China University of Geosciences, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 23 November 2015
Revised 22 February 2016
Accepted 23 February 2016
Available online 2 March 2016

Keywords:
Naive Bayes text classifiers
Feature weighting
Gain ratio
Decision tree

ABSTRACT

This paper works on feature weighting approaches for naive Bayes text classifiers. Almost all existing feature weighting approaches for naive Bayes text classifiers have some defects: limited improvement to classification performance of naive Bayes text classifiers or sacrificing the simplicity and execution time of the final models. In fact, feature weighting is not new for machine learning community, and many researchers have made fruitful efforts in the field of feature weighting. This paper reviews some simple and efficient feature weighting approaches designed for standard naive Bayes classifiers, and adapts them for naive Bayes text classifiers. As a result, this paper proposes two adaptive feature weighting approaches for naive Bayes text classifiers. Experimental results based on benchmark and real-world data show that, compared to their competitors, our feature weighting approaches show higher classification accuracy, yet at the same time maintain the simplicity and lower execution time of the final models.

1. Introduction

In recent years, the exponential growth of text documents on the Internet, digital libraries and other fields [26,32] has attracted the attention of many scholars. The task of automatic text classification is to assign text documents to pre-specified classes, which has been an important task in information retrieval [18,31]. Text classification presents unique challenges due to a large number of features, a large number of documents and strong dependencies among features [8,9].

To tackle text classification tasks, documents are characterized by the words that appear in them. Thus, one simplest way to apply machine learning to text classification is to treat each word as a Boolean variable. This is the first statistical language model called multi-variate Bernoulli naive Bayes (BNB) model [20]. BNB assumes that a document is represented by a vector of binary feature variables. The vector indicates which words occur or not in the document, and ignores the information of the number of times a word occurs in the document. To overcome this shortcoming confronting BNB, the multinomial naive Bayes (MNB) model [19] is proposed by capturing the information of the number of times a word occurs in a document. However, one systemic problem confronting MNB is that when one class has more training documents than others, MNB selects poor weights for the decision boundary. This is due to an under-studied bias effect that shrinks weights for classes with few training documents. To balance the amount of training documents used per estimate and deal with skewed training data, a complement class version of MNB called complement naive Bayes (CNB) is proposed [21]. The one-versus-all-but-one model (commonly misnamed one-versus-all, simply denoted by OVA) is a combination of MNB and CNB [21]. It is proved that OVA performs much better than MNB. Rennie et al. [21] attributed the improvement with OVA to the use of complement weights.

Although these naive Bayes text classifiers have already demonstrated remarkable classification accuracy, like naive Bayes classifiers, their conditional independence assumption is rarely true in reality. So, it is natural to improve naive Bayes text classifiers by relaxing the conditional independence assumption required by them. There are some approaches to do it such as structure extension [14], local learning [11,23], instance weighting [5,13], feature selection [2,10,27,30], and feature weighting [4,12,16,22], and so on.

This paper focuses on feature weighting approaches for naive Bayes text classifiers. To our knowledge, there exist some feature weighting approaches especially designed for naive Bayes text classifiers [16,22]. However, almost all of these existing approaches have some defects. The χ^2 statistic-based feature weighting approach [16] runs fast but the improvement to classification performance of naive Bayes text classifiers is limited. The CFS-based feature weighting algorithm [22] shows good classification accuracy but suffers from relative high execution time. So this paper tries to

* Corresponding author at: Department of Computer Science, China University of Geosciences, Wuhan 430074, China. Tel.: +86 2767883716.
** Corresponding author.
E-mail addresses: ljiang@cug.edu.cn (L. Jiang), chqli@cug.edu.cn (C. Li).

http://dx.doi.org/10.1016/j.knosys.2016.02.017

propose some feature weighting approaches which have good classification performance and simultaneously maintain the simplicity and low execution time of the final models.

Feature weighting is not new for machine learning community. Many feature weighting algorithms have been especially designed for standard naive Bayes classifiers. We hope to borrow from previous research achievements about feature weighting of standard naive Bayes classifiers to improve naive Bayes text classifiers. For this purpose, this paper reviews some feature weighting algorithms especially designed for standard naive Bayes classifiers and finds some simple and efficient algorithms. But directly applying them to naive Bayes text classifiers cannot get good results, and some actual problems have to be solved. We adapt these algorithms for improving naive Bayes text classifiers. As a result, this paper proposes two adaptive feature weighting approaches for naive Bayes text classifiers. Compared to their competitors, our feature weighting approaches show higher classification accuracy, yet at the same time maintain the simplicity and lower execution time of the final models.

The remainder of this paper is organized as follows. Section 2 reviews the related work with regard to this paper. Section 3 proposes two adaptive feature weighting approaches for naive Bayes text classifiers. Section 4 describes in detail the experimental setup and results. The last section draws conclusions and outlines main directions for our future work.

2. Related work

Given a test document d, represented by a word vector $< w_1, w_2, \ldots, w_m >$, MNB, CNB, and OVA classify d using Eqs. (1)–(3), respectively.

$$c(d) = \arg\max_{c \in C}[logP(c) + \sum_{i=1}^{m} f_i logP(w_i|c)] \tag{1}$$

$$c(d) = \arg\max_{c \in C}[-logP(\bar{c}) - \sum_{i=1}^{m} f_i logP(w_i|\bar{c})] \tag{2}$$

$$c(d) = \arg\max_{c \in C}[(logP(c) - logP(\bar{c})) + \sum_{i=1}^{m} f_i(logP(w_i|c) - logP(w_i|\bar{c}))] \tag{3}$$

where C is the set of all class labels, $\bar{c}$ is the complement classes of the class c (all classes except the class c), m is the vocabulary size in the text collection (the number of different words in all of the documents), $w_i (i = 1, 2, \ldots, m)$ is the ith word occurs in the document d, f_i is the frequency count of the word w_i in the document d. The prior probabilities $P(c)$ and $P(\bar{c})$ are generally estimated by Eqs. (4) and (5), respectively, and the conditional probabilities $P(w_i|c)$ and $P(w_i|\bar{c})$ are generally estimated by Eqs. (6) and (7), respectively.

$$P(c) = \frac{\sum_{j=1}^{n} \delta(c_j, c) + 1}{n + l} \tag{4}$$

$$P(\bar{c}) = \frac{\sum_{j=1}^{n} \delta(c_j, \bar{c}) + 1}{n + l} \tag{5}$$

$$P(w_i|c) = \frac{\sum_{j=1}^{n} f_{ji}\delta(c_j, c) + 1}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ji}\delta(c_j, c) + m} \tag{6}$$

$$P(w_i|\bar{c}) = \frac{\sum_{j=1}^{n} f_{ji}\delta(c_j, \bar{c}) + 1}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ji}\delta(c_j, \bar{c}) + m} \tag{7}$$

where n is the number of training documents, l is the number of classes, c_j is the class label of the jth training document, f_{ji} is the frequency count of the word w_i in the jth training document, $\delta(c_j, c)$ and $\delta(c_j, \bar{c})$ are two binary functions, which can be defined as:

$$\delta(c_j, c) = \begin{cases} 1, & if\ c_j = c \\ 0, & otherwise \end{cases} \tag{8}$$

$$\delta(c_j, \bar{c}) = \begin{cases} 1, & if\ c_j \in \bar{c}, namely\ c_j \neq c \\ 0, & otherwise \end{cases} \tag{9}$$

Of numerous approaches to improve above naive Bayes text classifiers by relaxing their conditional independence assumption, feature weighting has received some attention from researchers. The resulting improved models classify d using Eqs. (10)–(12), respectively.

$$c(d) = \arg\max_{c \in C}[logP(c) + \sum_{i=1}^{m} W_i f_i logP(w_i|c)] \tag{10}$$

$$c(d) = \arg\max_{c \in C}[-logP(\bar{c}) - \sum_{i=1}^{m} W_i f_i logP(w_i|\bar{c})] \tag{11}$$

$$c(d) = \arg\max_{c \in C}[(logP(c) - logP(\bar{c})) + \sum_{i=1}^{m} W_i f_i(logP(w_i|c) - logP(w_i|\bar{c}))] \tag{12}$$

where W_i is the weight of the word w_i.

Obviously, how to learn each feature's weight $W_i (i = 1, 2, \ldots, m)$ is crucial in improving naive Bayes text classifiers by feature weighting. In order to learn the weights of features (words), [16] proposed a χ^2 statistic-based feature weighting approach, simply denoted by $R_{w,c}$. The weighted naive Bayes classifier using $R_{w,c}$ improves the text classification performance of basic naive Bayes classifier by measuring positive term-class dependency accurately at the training phase. Note that this feature weighting approach is originally proposed to improve standard naive Bayes for text classification, and thus the improvement to above naive Bayes text classifiers, including MNB, CNB, and OVA, is proved to be very limited [22]. To scale up the classification performance of above naive Bayes text classifiers, Wang et al. [22] proposed a CFS-based feature weighting approach, which firstly conducts a correlation-based feature selection (CFS) [6] process to select a best feature subset from the whole feature space and then assigns larger weights to the features in the selected feature subset and smaller weights to others. Their experimental results on a large suite of benchmark datasets show that the CFS-based feature weighting approach can dramatically improve the classification accuracy of above naive Bayes text classifiers. However, this feature weighting approach needs to employ a best first heuristic search to find the best feature subset, which incurs an approximately quadratic time complexity and affects its application in high-dimensional text data classification tasks.

Although there are not many feature weighting approaches especially designed for naive Bayes text classifiers, previous works have presented many feature weighting algorithms for standard naive Bayes classifier. Zhang and Sheng [29] proposed a gain ratio-based feature weighting approach for standard naive Bayes, in which a feature with higher gain ratio is assigned higher weight. Hall [7] proposed a decision tree-based feature weighting approach for standard naive Bayes. The decision tree-based feature weighting approach weights predictive features according to the degree to which they depend on other features' values and assigns lower weights to those features that have many dependencies. To estimate the degree to which a feature depends on others, an unpruned decision tree is built from a training data and the minimum depth d at which the feature is tested in the built tree is recorded, and then the weight for this feature is set to $1/\sqrt{d}$. If a feature does not appear in the built tree, then it receives a weight of zero. In order to stabilize the estimated weights, multiple decision trees are built by using bagging and the weights across the ensemble are averaged.

Besides, Wu and Cai [25] proposed a differential evolution algorithm-based feature weighting approach, which utilizes sophisticated differential evolution algorithms to refine feature weights. Zaidi et al. [28] proposed to select feature weights to minimize either the negative conditional log likelihood or the mean squared error objective functions rather than select feature weights based on measures of predictiveness. Recently, Lee [15] proposed another new paradigm of assigning weights called value weighting method, which assigns different weights to the values of each feature.

Above feature weighting approaches all show better performance to improve standard naive Bayes. Among them, the gain ratio-based feature weighting approach and the decision tree-based feature weighting approach are relatively simple and run fast. But when we directly apply the two approaches to naive Bayes text classifiers, they don't show good classification performance and some actual problems have to be solved. The next section adapts these two feature weighting approaches for naive Bayes text classifiers.

3. Two adaptive feature weighting approaches

3.1. Adapting gain ratio-based feature weighting for text classification

The gain ratio-based feature weighting approach has already been studied for standard naive Bayes [29], here we adapt it for naive Bayes text classifiers. When the gain ratio-based feature weighting approach comes to text classification data, a key issue needed to be addressed is how to define the gain ratio of each feature (word) partitioning a collection of training documents. The feature values handled by standard naive Bayes are generally nominal, while the feature values in text classification data are integral. A standard text classification dataset is a collection of documents, where each document is represented as a word vector with the frequency count of each word occurring in the document. The text classification data is often a sparse matrix because the vocabulary of the words bag is quite vast. Each feature value in this matrix is zero or a positive integer. Moreover, most of these feature values are zero and the values greater than one are quite few. Therefore, in our definition of the information gain ratio, we assume that all features only have two values of zero and nonzero.

Given a training document set D, The information gain ratio $IGR(C, w_i)$ of the word w_i partitioning D is defined as follows.

$$IGR(C, w_i) = \frac{IG(C, w_i)}{H(w_i)} \tag{13}$$

where C is the target (class) variable, $IG(C, w_i)$ is the information gain of the word w_i partitioning D, and $H(w_i)$ is the split information of the word w_i partitioning D. Based on above observation and assumption for text data, $IG(C, w_i)$ is defined as

$$IG(C, w_i) = H(C) - H(C|w_i) \tag{14}$$

where $H(C)$ is the entropy of D and $H(C|w_i)$ is the conditional entropy of D given the value of w_i, which can be calculated by Eqs. (15) and (16)respectively.

$$H(C) = -\sum_{c} P(c)\log_2 P(c) \tag{15}$$

$$H(C|w_i) = -\sum_{v} \frac{|D_v|}{|D|} \sum_{c} P(c|v)\log_2 P(c|v) \tag{16}$$

where $P(c)$ is the probability of class c in D, $|D_v|$ is the number of documents whose value of w_i is v (here $v \in \{0, \bar{0}\}$).

$H(w_i)$ is actually the entropy of D with respect to the values of the word w_i, and is defines as

$$H(w_i) = -\sum_{v} \frac{|D_v|}{|D|}\log_2\frac{|D_v|}{|D|} \tag{17}$$

Once the gain ratio $IGR(C, w_i)$ of each word $w_i (i = 1, 2, \ldots, m)$ is acquired, we can calculate the sum of all words' information gain ratios and define the weight W_i of each word w_i $(i = 1, 2, \ldots, m)$ as

$$W_i = \frac{IGR(C, w_i) \times m}{\sum_{i=1}^{m} IGR(C, w_i)} \tag{18}$$

After getting the weight value W_i of each word $w_i (i = 1, 2, \ldots, m)$ using Eq. (18), we follow the gain ratio-based feature weighting approach for standard naive Bayes [29] and apply these weights to Eqs. (10)–(12) to improve the classification performance of MNB, CNB and OVA. But we find that the improvement of classification performance is not ideal. The work of Wang et al. [22] has shown that if features' weights are incorporated not only into classification formulas of naive Bayes text classifiers but also into their conditional probability estimates, that will significantly enhance the classifiers' performance. So we also follow them to incorporate the learned feature weights into both the classification formulas and their conditional probability estimates. Consequently, the conditional probabilities $P(w_i|c)$ and $P(w_i|\bar{c})$ estimated by Eqs. (6) and (7) previously are now modified to the following Eqs. (19) and (20), respectively.

$$P(w_i|c) = \frac{\sum_{j=1}^{n} W_i f_{ji}\delta(c_j, c) + 1}{\sum_{i=1}^{m}\sum_{j=1}^{n} W_i f_{ji}\delta(c_j, c) + m} \tag{19}$$

$$P(w_i|\bar{c}) = \frac{\sum_{j=1}^{n} W_i f_{ji}\delta(c_j, \bar{c}) + 1}{\sum_{i=1}^{m}\sum_{j=1}^{n} W_i f_{ji}\delta(c_j, \bar{c}) + m} \tag{20}$$

When we apply the feature weighting approach to naive Bayes text classifiers, we call the resulting models as gain ratio weighted naive Bayes text classifiers. When base classifiers are MNB , CNB, and OVA, respectively, the resulting models are simply denoted as GRWMNB, GRWCNB, GRWOVA. Taking GRWMNB for example, the detailed learning process is described as Algorithm 1.

Algorithm 1. GRWMNB (D, d)
Input: a training document set D, a test document d
Output: the class value $c(d)$ of the test document d

1. For each word w_i $(i = 1, 2, \ldots, m)$ from D
   Calculate $IGR(C, w_i)$ using Eq. (13)
2. Calculate the sum of all words' gain ratios
3. For each word w_i $(i = 1, 2, \ldots, m)$ from D
   Calculate its weight W_i using Eq. (18)
4. For the test document d
   (a) Calculate $P(c)$ using Eq. (4)
   (b) Calculate $P(w_i|c)$ using Eq. (19)
   (c) Predict its class value $c(d)$ using Eq. (10)
5. Return the class value $c(d)$ of d

3.2. Adapting decision tree-based feature weighting for text classification

To improve standard naive Bayes, Hall [7] proposed a decision tree-based feature weighting approach, in which the weight of a feature is set to $1/\sqrt{d}$ if the minimum depth at which the feature is tested in the built tree is d (The root node of the tree has depth 1), and 0 if the feature does not appear in the built tree. Directly applying it to naive Bayes text classifiers cannot get obvious performance improvement. Here we also adapt it for naive Bayes text classifiers. Different from the approach presented by Hall [7], we set the weight of a feature to $1+\lambda/\sqrt{d}$ (λ is a user-provided positive integer that determines how heavily to weight $1/\sqrt{d}$.) if the minimum depth at which the feature is tested in the built tree is d, and 1 if the feature does not appear in the built tree. Thus, we define the weight W_i of each word w_i $(i=1,2,\ldots,m)$ as

$$W_i = \begin{cases} 1+\lambda/\sqrt{d_i}, & if\ w_i\ is\ tested\ in\ the\ built\ tree \\ 1, & otherwise \end{cases} \tag{21}$$

where d_i is the minimum depth of w_i tested in the tree.

Our adaption is based on two reasons. Firstly, we try to get a feature weighting algorithm instead of a feature selection algorithm. The work of Hall [7] sets the weights of those features which do not appear in the built tree to be 0, this is actually a feature selection algorithm. Our feature weighting algorithm hopes every feature could contribute to the target classifiers. Secondly, different classification models have different biases. Those features which are useless for building decision trees are likely not to be useless for other learning models. So it may be a little extreme to set the weights of those features which do not appear in the built tree to be 0 . Based on these two considerations, we set the weights of those features which do not appear in the built tree to be 1. And then we adjust the weights of those features which are tested in the built tree to be $1+5/\sqrt{d}$. In our experiments, we test a mass of values of λ such as 1, 2, 3, 4, 5, and 10, and find that our approach is not particularly sensitive to the choice of λ as long as it is greater than 3. For saving space, we do not present detailed experimental results here.

After getting the weights of each word, like the gain ratio weighted naive Bayes text classifiers, we also incorporate the learned weights not only into the classification formulas of naive Bayes text classifiers but also into their conditional probability estimates. Moreover, previous work [7] used bagging to build multiple decision trees in order to stabilize the estimated weights, but our feature weighting approach only builds a tree for estimating the feature weights. One of the reasons for this is to maintain the simplicity of feature weighting approach. Another is that we have set the weights of those features which do not appear in the built tree to be 1 instead of 0. By this way, those relative unimportant features also can contribute to the target classifiers.

When we apply the feature weighting approach to naive Bayes text classifiers, we call the resulting models as decision tree weighted naive Bayes text classifiers. When base classifiers are MNB , CNB, and OVA, respectively, the resulting models are simply denoted as DTWMNB, DTWCNB, and DTWOVA. Taking DTWMNB for example, the detailed learning process is described as Algorithm 2.

Algorithm 2. DTWMNB (D, d)
Input: a training document set D, a test document d
Output: the class value $c(d)$ of the test document d

1. Build an unpruned binary tree (each word's value is divided into zero and nonzero) using the gain ratio defined by Eq. (13) as the splitting criterion
2. Traverse the built tree to record the minimum depth d_i of each word w_i $(i=1,2,\ldots,m)$ tested in the tree
3. For each word w_i $(i=1,2,\ldots,m)$ from D
   Calculate its weight W_i using Eq. (21)
4. for the test document d
   (a) Calculate $P(c)$ using Eq. (4)
   (b) Calculate $P(w_i|c)$ using Eq. (19)
   (c) Predict its class value $c(d)$ using Eq. (10)
5. Return the class value $c(d)$ of d

Table 1
Benchmark datasets used in our experiments.

Dataset	Documents number	Words number	Classes number
fbis	2463	2000	17
oh0	1003	3182	10
oh10	1050	3238	10
oh15	913	3100	10
oh5	918	3012	10
re0	1657	3758	25
re1	1504	2886	13
tr11	414	6429	9
tr12	313	5804	8
tr21	336	7902	6
tr23	204	5832	6
tr31	927	10128	7
tr41	878	7454	10
tr45	690	8261	10
wap	1560	8460	20

4. Experiments and results

4.1. Experimental setting and benchmark data

The purpose of these experiments is to validate the classification performance of naive Bayes text classifiers employing our adaptive feature weighting approaches. We choose MNB, CNB, and OVA, respectively, to be the base naive Bayes text classifiers and show the performance of different feature weighting approaches. Take MNB for example, we introduce the comparison algorithms and their abbreviations.

- MNB: Multinomial naive Bayes model [19] .
- $R_{w,c}$MNB: MNB model employing the χ^2 statistic-based feature weighting approach [16].
- FWMNB: MNB model employing the CFS-based feature weighting approach [22].
- GRWMNB: MNB model employing our gain ratio weighting approach.
- DTWMNB: MNB model employing our decision tree weighting approach.

We design three groups of experiments to compare our adaptive feature weighting approaches with their competitors in terms of classification accuracy. The first group of experiments compare MNB with $R_{w,c}$MNB, FWMNB, GRWMNB, and DTWMNB. The second group of experiments compare CNB with $R_{w,c}$CNB, FWCNB, GRWCNB, and DTWCNB. The third group of experiments compare OVA with $R_{w,c}$OVA, FWOVA, GRWOVA, and DTWOVA. We use the existing implementations of MNB and CNB in the WEKA platform [24] and implement all the other algorithms on the WEKA platform [24].

We run our experiments on 15 widely used text classification benchmark datasets published on the main web site of WEKA [24], which represent a wide range of domains and data characteristics. The detailed description of these 15 datasets is shown in Table 1. For saving the time and memory in running experiments, we don't include four very large datasets: "la1s", "la2s", "new3s", and "ohscal" in our experiments.

4.2. Experimental results and analysis

Tables 2, 4, and 6 show the detailed classification accuracy of each algorithm on each dataset obtained via ten runs of ten-fold cross-validation, respectively. The averages are summarized at the bottom of the tables. The average (arithmetic mean) across all datasets provides a gross indication of relative performance in addition to other statistics.

Table 2
Classification accuracy comparisons with regard to MNB.

Dataset	MNB	$R_{w,c}$MNB	FWMNB	GRWMNB	DTWMNB
fbis	77.1094	79.8705	78.6850	79.7042	79.4478
oh0	89.5517	89.0525	91.4651	93.0603	92.2723
oh10	80.6000	80.4095	82.2476	84.0476	82.7048
oh15	83.6032	83.6141	85.6302	87.8097	86.3616
oh5	86.6312	86.4578	89.3219	92.3411	90.9778
re0	80.0210	77.0673	80.9323	82.3872	81.4509
re1	83.3137	82.7212	85.3774	88.7325	86.1683
tr11	85.2079	85.4419	86.8275	88.5650	86.6800
tr12	80.9919	84.7591	82.6179	86.5444	84.9194
tr21	61.9011	69.6283	65.1248	77.3298	62.4064
tr23	71.1500	73.8190	73.4048	86.4619	78.5619
tr31	94.5968	94.1983	95.5353	96.7555	95.6537
tr41	94.6472	93.0516	95.6144	95.5465	95.2388
tr45	83.6377	88.8841	86.5942	90.5942	89.0725
wap	81.2179	76.3269	82.5321	83.5641	82.4231
Average	82.2787	83.0201	84.1274	**87.5629**	**84.9559**
Average ranking	4.4667	4	3	**1.1333**	**2.4**

Table 3
P-values with regard to MNB.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
10	MNB vs. GRWMNB	5.773503	0	0.005
9	$R_{w,c}$MNB vs. GRWMNB	4.965212	0.000001	0.005556
8	MNB vs. DTWMNB	3.579572	0.000344	0.00625
7	FWMNB vs. GRWMNB	3.233162	0.001224	0.007143
6	$R_{w,c}$MNB vs. DTWMNB	2.771281	0.005584	0.008333
5	MNB vs. FWMNB	2.540341	0.011074	0.01
4	GRWMNB vs. DTWMNB	2.193931	0.02824	0.0125
3	$R_{w,c}$MNB vs. FWMNB	1.732051	0.083265	0.016667
2	FWMNB vs. DTWMNB	1.03923	0.298698	0.025
1	MNB vs. $R_{w,c}$MNB	0.80829	0.418923	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.01: • MNB vs. GRWMNB; • $R_{w,c}$MNB vs. GRWMNB; • MNB vs. DTWMNB; • FWMNB vs. GRWMNB; • $R_{w,c}$MNB vs. DTWMNB.

Table 4
Classification accuracy comparisons with regard to CNB.

Dataset	CNB	$R_{w,c}$CNB	FWCNB	GRWCNB	DTWCNB
fbis	76.7765	78.2747	77.1704	76.8781	76.4681
oh0	92.3115	92.491	93.6168	93.7075	93.936
oh10	81.7619	82.2	83.2571	84.4571	83.5048
oh15	84.381	85.3242	86.1	86.768	86.5704
oh5	90.5775	90.9589	92.147	91.9943	92.7459
re0	82.3659	80.7377	83.4706	81.8873	83.5299
re1	84.9909	86.1562	84.816	86.4813	85.4556
tr11	82.6388	82.1864	83.2695	86.4808	84.0621
tr12	86.3165	86.5726	87.8821	88.0333	87.879
tr21	85.9421	86.3939	87.6747	90.8102	86.8093
tr23	70.5905	72.469	77.0619	90.2167	85.0357
tr31	94.6726	95.0936	96.0213	96.949	96.4099
tr41	94.2251	94.9079	94.9082	95.5692	94.9991
tr45	87.2029	89.1304	89.0725	91.8696	91.5507
wap	77.5256	78.1026	78.4103	78.3974	79.7308
Average	84.8186	85.3999	86.3252	**88.0333**	**87.2458**
Average ranking	4.6667	3.7333	2.8	**1.6667**	**2.1333**

Then, we employ a Friedman test for comparison of multiple algorithms over multiple datasets [1,3]. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA [3]. The average rankings of the algorithms obtained by applying the Friedman test are also summarized at the bottom of Tables 2, 4 and 6, respectively. With 5 algorithms and 15 datasets, F_F is distributed according to the F distribution with 4 and 56 degrees of freedom: 32.597633, 20.017279, and 111, respectively, which are all greater than the critical value of $F(4, 56)$ for $a = 0.05$ (The table of critical values can be found in any statistical books). So we reject the null

Table 5
P-values with regard to CNB.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
10	CNB vs. GRWCNB	5.196152	0	0.005
9	CNB vs. DTWCNB	4.387862	0.000011	0.005556
8	$R_{w,c}$CNB vs. GRWCNB	3.579572	0.000344	0.00625
7	CNB vs. FWCNB	3.233162	0.001224	0.007143
6	$R_{w,c}$CNB vs. DTWCNB	2.771281	0.005584	0.008333
5	FWCNB vs. GRWCNB	1.962991	0.049647	0.01
4	CNB vs. $R_{w,c}$CNB	1.616581	0.105969	0.0125
3	$R_{w,c}$CNB vs. FWCNB	1.616581	0.105969	0.016667
2	FWCNB vs. DTWCNB	1.154701	0.248213	0.025
1	GRWCNB vs. DTWCNB	0.80829	0.418923	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.01: • CNB vs. GRWCNB; • CNB vs. DTWCNB; • $R_{w,c}$CNB vs. GRWCNB; • CNB vs. FWCNB; • $R_{w,c}$CNB vs. DTWCNB.

Table 6
Classification accuracy comparisons with regard to OVA.

Dataset	OVA	$R_{w,c}$OVA	FWOVA	GRWOVA	DTWOVA
fbis	80.9386	80.8045	81.3603	82.8183	82.6921
oh0	91.4852	90.1192	92.8409	93.9968	93.6779
oh10	81.8571	81.5143	83.6	84.8952	83.7333
oh15	84.3912	84.5006	86.2539	88.0622	87.0301
oh5	89.4417	88.3098	90.9558	93.0393	92.0787
re0	81.5367	78.8098	82.4479	82.9595	82.7938
re1	84.7733	85.3651	85.9928	88.7747	86.7233
tr11	85.9303	86.1196	86.313	89.0976	88.1272
tr12	84.1462	86.0071	86.3175	88.8105	86.8871
tr21	71.3369	76.5847	82.7059	86.0704	72.7068
tr23	71.4405	73.8548	76.3048	90.1714	81.9571
tr31	94.6828	94.522	96.1179	97.0468	96.0414
tr41	94.9429	93.8264	95.7277	95.9905	95.6489
tr45	86.4493	89.2319	89.8116	92.087	91.8986
wap	80.6538	77.2051	81.7949	82.7885	82.1923
Average	84.2671	84.4517	86.5697	**89.1072**	**86.9459**
Average ranking	4.4667	4.4667	2.8	**1**	**2.2667**

Table 7
P-values with regard to OVA.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
10	OVA vs. GRWOVA	6.004443	0	0.005
9	$R_{w,c}$OVA vs. GRWOVA	6.004443	0	0.005556
8	OVA vs. DTWOVA	3.810512	0.000139	0.00625
7	$R_{w,c}$OVA vs. DTWOVA	3.810512	0.000139	0.007143
6	FWOVA vs. GRWOVA	3.117691	0.001823	0.008333
5	OVA vs. FWOVA	2.886751	0.003892	0.01
4	$R_{w,c}$OVA vs. FWOVA	2.886751	0.003892	0.0125
3	GRWOVA vs. DTWOVA	2.193931	0.02824	0.016667
2	FWOVA vs. DTWOVA	0.92376	0.355611	0.025
1	OVA vs. $R_{w,c}$OVA	0	1	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.016667: • OVA vs. GRWOVA • $R_{w,c}$OVA vs. GRWOVA • OVA vs. DTWOVA • $R_{w,c}$OVA vs. DTWOVA • FWOVA vs. GRWOVA • OVA vs. FWOVA • $R_{w,c}$OVA vs. FWOVA

hypotheses and proceed with a post-hoc Holm's test to further analyze which pairs of algorithms are significantly different. Tables 3, 5 and 7 report the obtained z-values and p-values, and also indicate which pairs of algorithms are significantly different.

From these experimental results, we can see that our proposed feature weighting approaches rarely degrade the quality of original naive Bayes text classifiers and, in many cases, improve them remarkably. Besides, our proposed gain ratio-based feature weighting approach significantly outperforms all the other existing approaches. Now, we summarize the highlights as follows:

1. With regard to MNB, the average rankings of them are GRWMNB (1.1333), DTWMNB (2.4), FWMNB (3), $R_{w,c}$MNB (4),

Table 8
Elapsed training time comparisons with regard to MNB.

Dataset	MNB	$R_{w,c}$MNB	FWMNB	GRWMNB	DTWMNB
fbis	0.0036	0.0052	29.2272	0.9767	32.0989
oh0	0.0013	0.0025	27.4044	0.4183	9.5247
oh10	0.0016	0.0025	9.6859	0.4544	14.1205
oh15	0.0013	0.0023	12.5358	0.3745	12.1997
oh5	0.0014	0.002	7.5007	0.3488	6.6721
re0	0.0015	0.0026	21.6742	0.654	30.9513
re1	0.0032	0.0056	21.9635	0.941	34.5116
tr11	0.0019	0.0035	52.0251	0.357	4.3279
tr12	0.0016	0.0028	22.5121	0.2111	1.6043
tr21	0.0019	0.0031	91.2295	0.2959	4.5321
tr23	0.0012	0.002	13.1064	0.1427	0.5931
tr31	0.0027	0.0055	139.1323	1.4889	10.4585
tr41	0.0029	0.005	75.5843	1.0746	11.8081
tr45	0.0031	0.0054	83.4671	0.8899	5.2536
wap	0.0065	0.0112	410.6416	2.2177	184.9941
Average	0.0024	0.0041	67.846	0.723	24.2434

and MNB (4.4667), respectively. GRWMNB is notably better than all the other existing competitors: MNB, $R_{w,c}$MNB, and FWMNB. DTWMNB is markedly better than MNB and $R_{w,c}$MNB.
2. With regard to CNB, the average rankings of them are GRWCNB (1.6667), DTWCNB (2.1333), FWCNB (2.8), $R_{w,c}$CNB (3.7333), and CNB (eq4.6667), respectively. GRWCNB and DTWCNB all significantly outperform CNB and $R_{w,c}$CNB.
3. With regard to OVA, the average rankings of them are GRWOVA (1), DTWOVA (2.2667), FWOVA (2.8), $R_{w,c}$OVA (4.4667), and OVA (4.4667), respectively. GRWOVA is notably better than all the other existing competitors: OVA, $R_{w,c}$OVA, and FWOVA. DTWOVA is markedly better than OVA and $R_{w,c}$OVA.
4. Our proposed decision tree-based feature weighting approach is much better than the existing χ^2 statistic-based feature weighting approach ($R_{w,c}$) and is a little better than the existing CFS-based feature weighting approach.
5. Our proposed gain ratio-based feature weighting approach significantly outperforms all the other competitors: the χ^2 statistic-based feature weighting approach and the CFS-based feature weighting approach. Besides, our proposed gain ratio-based feature weighting approach is generally a little better than our proposed decision tree-based feature weighting approach.

Above experimental results show that our adaptive feature weighting approaches are obviously better than their competitors in terms of classification accuracy. In our another group of experiments below, we compare our feature weighting approaches to the CFS-based feature weighting approach in terms of elapsed training time in seconds. Our experiments are performed on a desktop PC with 64-bit Microsoft Windows 7 with Intel(R) Core(TM) i5-4570 Quad core CPU @ 3.20 GHz and 8GB RAM. The detailed comparison results are shown in Tables 8, 9, and 10. From these comparison results, we can see that:

1. The χ^2 statistic-based feature weighting approach is the fastest among all of the compared approaches.
2. Our gain ratio-based feature weighting approach runs slower than the χ^2 statistic-based feature weighting approach, while significantly faster than the rest feature weighting approaches. But its improvement to classification performance of naive Bayes text classifiers is remarkably better than its competitors. In a word, our gain ratio-based feature weighting approach gets the best balance between classification accuracy and execution time.

Table 9
Elapsed training time comparisons with regard to CNB.

Dataset	CNB	$R_{w,c}$CNB	FWCNB	GRWCNB	DTWCNB
fbis	0.0041	0.0051	26.9626	0.9669	33.0005
oh0	0.0013	0.002	27.3416	0.4265	9.9152
oh10	0.0012	0.0027	9.1418	0.4637	14.5999
oh15	0.0013	0.0018	12.2599	0.3832	12.7074
oh5	0.0012	0.0019	7.1493	0.3563	6.8543
re0	0.0014	0.0023	20.8333	0.6718	32.6682
re1	0.0031	0.0057	20.3502	0.9721	36.6334
tr11	0.0021	0.0033	51.9711	0.3685	4.5252
tr12	0.0016	0.0027	22.4469	0.2158	1.6674
tr21	0.0018	0.003	90.4464	0.2971	4.5689
tr23	0.0013	0.002	11.9679	0.1466	0.6294
tr31	0.003	0.005	142.2128	1.5396	10.8332
tr41	0.0028	0.0048	77.3283	1.1162	12.2998
tr45	0.0031	0.0052	86.4184	0.9193	5.509
wap	0.0059	0.0098	425.1851	2.288	196.7461
Average	0.0023	0.0038	68.801	0.7421	25.5439

Table 10
Elapsed training time comparisons with regard to OVA.

Dataset	OVA	$R_{w,c}$OVA	FWOVA	GRWOVA	DTWOVA
fbis	0.0051	0.0065	26.8168	0.9762	32.6951
oh0	0.0022	0.0029	27.3016	0.4271	9.7719
oh10	0.0023	0.003	9.1218	0.463	14.6751
oh15	0.0022	0.0029	12.2421	0.3849	12.4262
oh5	0.002	0.0027	7.1326	0.3621	6.9243
re0	0.0028	0.0036	20.7872	0.6743	32.4062
re1	0.0062	0.0083	20.3315	0.9756	36.0891
tr11	0.0038	0.0051	51.9138	0.3718	4.5441
tr12	0.003	0.0049	22.4475	0.2199	1.6736
tr21	0.0039	0.0045	90.3369	0.2975	4.7368
tr23	0.0023	0.003	11.9592	0.1479	0.6284
tr31	0.0053	0.0074	142.3888	1.5302	10.9571
tr41	0.006	0.007	76.9749	1.1225	12.3238
tr45	0.0057	0.0077	86.2313	0.9226	5.5284
wap	0.0115	0.0155	425.5217	2.2979	194.2246
Average	0.0043	0.0057	68.7672	0.7449	25.307

3. Our tree-based feature weighting approach runs slower than our gain ratio-based feature weighting approach, but it is also almost three times faster than the existing CFS-based feature weighting approach.
4. The CFS-based feature weighting approach runs most slowly, because it uses a best first heuristic search to find a best feature subset from the whole feature space, which incurs an approximately quadratic time complexity.

4.3. Experiments on real-world text data

To further prove the effectiveness of our adaptive feature weighting approaches, we observe their performance on a real-world text classification dataset [17] derived from the customer reviews in Amazon Commerce Website for authorship identification. This database contains 1500 instances described by 10000 words. The aim is to identify 50 of the most active customers who frequently posted reviews in these newsgroups. The number of the collected reviews for each customer is 30. In our experiments, we remove a mass of too sparse words from the dataset for saving the execution time in running experiments. If the percentage of the documents in which a word occurs is less than 5%, we speak of this word as being “too sparse”. Besides, we employ same experimental settings as the previous experiments. Fig. 1 shows the detailed comparison results on classification accuracy. We can see that our adaptive feature weighting approaches also perform much better than their competitors on the real-world application.

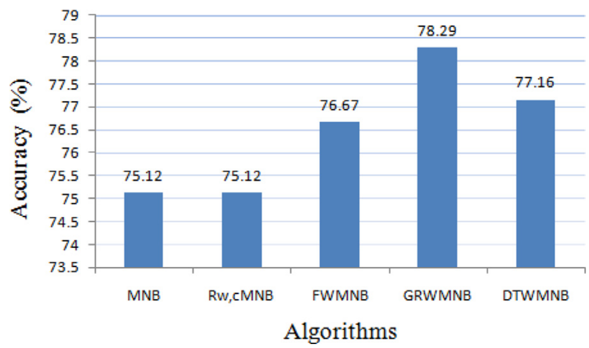

Fig. 1. Classification accuracy comparisons on the Amazon Commerce reviews dataset.

Table 11
Classification accuracy comparisons with regard to KNN.

Dataset	KNN	MIWKNN	GRWKNN	DTWKNN
fbis	80.8084	79.298	81.4865	80.9224
oh0	81.265	82.8617	91.0254	87.5952
oh10	72.9714	75.1714	81.9714	76.8
oh15	75.9265	72.0951	82.7138	78.6291
oh5	78.311	82.6136	89.5742	86.0124
re0	83.3836	80.9775	84.081	82.1861
re1	80.5307	83.7107	88.6717	84.8582
tr11	85.8287	88.5627	90.1109	87.378
tr12	81.4355	87.4012	89.6411	87.0202
tr21	86.4421	89.4715	88.3048	86.0499
tr23	80.2	88.7095	89.7952	83.9976
tr31	92.4061	96.3441	97.1964	95.2872
tr41	91.9694	92.9961	93.485	93.9289
tr45	87.6087	88.2609	91.6232	92.8986
wap	73.6154	74.8974	76.5	73.7821
Average	82.1802	84.2248	**87.7454**	**85.1564**
Average ranking	3.6667	2.7333	**1.2**	**2.4**

Table 12
P-values with regard to KNN.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
6	KNN vs. GRWKNN	5.23259	0	0.008333
5	MIWKNN vs. GRWKNN	3.252691	0.001143	0.01
4	KNN vs. DTWKNN	2.687006	0.00721	0.0125
3	GRWKNN vs. DTWKNN	2.545584	0.010909	0.016667
2	KNN vs. MIWKNN	1.979899	0.047715	0.025
1	MIWKNN vs. DTWKNN	0.707107	0.4795	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value $\leq$ 0.025: • KNN vs. GRWKNN; • MIWKNN vs. GRWKNN; • KNN vs. DTWKNN; • GRWKNN vs. DTWKNN.

4.4. Discussion

In this section, we discuss the novelty and contribution of our proposed feature weighting approaches to some other related state-of-the-art text classification models. Since the learning process of feature weighting is independent of the used naive Bayes text classification models, our feature weighting approaches are actually two meta-learning methods and can be used to improve some other text classification models such as KNN (the k-nearest neighbor algorithm) [9]. For simplicity, we call the resulting models as gain ratio weighted KNN (GRWKNN) and decision tree weighted KNN (DTWKNN), respectively. We design a group of experiments to validate the effectiveness of our proposed feature weighting approaches. Tables 11 and 12 show the detailed comparison results. From these results, we can see that our feature weighted versions significantly outperform KNN and are even much better than mutual information weighted KNN (MIWKNN) [9].

5. Conclusions and future work

The main purpose of this paper is to borrow from the research achievements about feature weighting algorithms of standard naive Bayes classifiers to improve naive Bayes text classifiers. In this paper, we adapt two simple, efficient, and effective feature weighting approaches to naive Bayes text classifiers. One is the gain ratio-based feature weighting approach, and the other is the decision tree-based feature weighting approach. The experimental results on a large number of text classification datasets validate their effectiveness and efficiency in terms of classification accuracy and elapsed training time in seconds, respectively.

Many researchers have made fruitful efforts in the field of feature weighting, in spite that most works are not specially designed for text classification. We think that borrowing from previous research achievements to improve text classifiers is a natural way. The good performance of our adaptive feature weighting approaches proves the feasibility of this idea. In the future, we will try to do more works following the idea. Moreover, we will pay attention to more text classifiers instead of only naive Bayes text classifiers.

Acknowledgments

The work was partially supported by the National Natural Science Foundation of China (61203287), the Program for New Century Excellent Talents in University (NCET-12-0953), and the Chenguang Program of Science and Technology of Wuhan (2015070404010202).

References

[1] J. Alcalá-Fdez, L. Sánchez, S. García, M.J.d. Jesus, S.J. Ventura, M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, Keel: A software tool to assess evolutionary algorithms to data mining problems, Soft Comput. 13 (2009) 307–318.
[2] P. Bermejo, J.A. Gámez, J.M. Puerta, Speeding up incremental wrapper feature subset selection with naive Bayes classifier, Knowl. Based Syst. 55 (2014) 140–147.
[3] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[4] H.J. Escalante, M.A. García-Limón, A. Morales-Reyes, M. Graff, M. Montes-y Gómez, E.F. Morales, J. Martínez-Carranza, Term-weighting learning via genetic programming for text classification, Knowl. Based Syst. 83 (2015) 176–189.
[5] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufmann, 1996, pp. 148–156.
[6] M. Hall, Correlation-based feature selection for discrete and numeric class machine learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, 2000, pp. 359–366.
[7] M. Hall, A decision tree-based attribute weighting filter for naive Bayes, Knowl. Based Syst. 20 (2) (2007) 120–126.
[8] E. Han, G. Karypis, Centroid-based document classification: Analysis and experimental results, in: Proceedings of the Fourth European Conference on the Principles of Data Mining and Knowledge Discovery, Springer, 2000, pp. 424–431.
[9] E. Han, G. Karypis, V. Kumar, Text categorization using weight adjusted k-nearest neighbor classification, in: Proceedings of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2001, pp. 53–65.
[10] K. Javed, S. Maruf, A. Babri Haroon, A two-stage Markov blanket based feature selection algorithm for text classification, Neurocomputing 157 (2015) 91–104.
[11] L. Jiang, Z. Cai, H. Zhang, D. Wang, Naive Bayes text classifiers: A locally weighted learning approach, J. Exp. Theor. Artif. Intell. 25 (2) (2013) 273–286.
[12] L. Jiang, C. Li, S. Wang, L. Zhang, Deep feature weighting for naive Bayes and its application to text classification, Eng. Appl. Artif. Intell. 52 (2016) 26–39.
[13] L. Jiang, D. Wang, Z. Cai, Discriminatively weighted naive Bayes and its application in text classification, Int. J. Artif. Intell. Tools 21 (1) (2012) 1250007:1–1250007:19.
[14] L. Jiang, S. Wang, C. Li, L. Zhang, Structure extended multinomial naive Bayes, Inf. Sci. 329 (2016) 346–356.
[15] C.H. Lee, A gradient approach for value weighted classification learning in naive Bayes, Knowl. Based Syst. 85 (2015) 71–79.

[16] Y. Li, C. Luo, S.M. Chung, Weighted naive Bayes for text classification using positive term-class dependency, Int. J. Artif. Intell. Tools 21 (01) (2012) 1250008.
[17] S. Liu, Z. Liu, J. Sun, L. Liu, Application of synergetic neural network in online writeprint identification, Int. J. Digit. Content Technol. Appl. 5 (3) (2011) 126–135.
[18] D.E. Losada, L. Azzopardi, Assessing multivariate Bernoulli models for information retrieval, ACM Trans. Inf. Syst. 26 (3) (2008) 17:1–17:46.
[19] A. McCallum, K.A. Nigam, A comparison of event models for naive Bayes text classification, in: Proceedings of the AAAI/ICML Workshop on Learning for Text Categorization on Working Notes of the 1998, AAAI Press, 1998, pp. 41–48.
[20] J.M. Ponte, W.B. Croft, A language modeling approach to information retrieval, in: Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1998, pp. 275–281.
[21] J.D. Rennie, L. Shih, J. Teevan, D.R. Karger, Tackling the poor assumptions of naive Bayes text classifiers, in: Proceedings of the Twentieth International Conference on Machine Learning, Morgan Kaufmann, 2003, pp. 616–623.
[22] S. Wang, L. Jiang, C. Li, A CFS-based feature weighting approach to naive Bayes text classifiers, in: Proceedings of the Twenty-fourth International Conference on Artificial Neural Networks, Springer, 2014, pp. 555–562.
[23] S. Wang, L. Jiang, C. Li, Adapting naive Bayes tree for text classification, Knowl. Inf. Syst. 44 (1) (2015) 77–89.
[24] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, third ed., Morgan Kaufmann, 2011.
[25] J. Wu, Z. Cai, Attribute weighting via differential evolution algorithm for attribute weighted naive Bayes (wnb), J. Comput. Inf. Syst. 7 (5) (2011) 1672–1679.
[26] J. Yan, X. Gao, Detection and recognition of text superimposed in images base on layered method, Neurocomputing 134 (2014) 3–14.
[27] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 412–420.
[28] N.A. Zaidi, J. Cerquides, M.J. Carman, G.I. Webb, Alleviating naive Bayes attribute independence assumption by attribute weighting, J. Mach. Learn. Res. 14 (2013) 1947–1988.
[29] H. Zhang, S. Sheng, Learning weighted naive Bayes with accurate ranking, in: Proceedings of the Fourth International Conference on Data Mining, IEEE, 2004, pp. 567–570.
[30] L. Zhang, L. Jiang, C. Li, A new feature selection approach to naive Bayes text classifiers, Int. J. Pattern Recognit. Artif. Intell. 30 (2) (2016) 1650003:1–1650003:17.
[31] T. Zhang, F.J. Oles, Text categorization based on regularized linear classification methods, Inf. Retr. 4 (1) (2001) 5–31.
[32] W. Zhang, X. Tang, T. Yoshida, Tesc: An approach to text classification using semi-supervised clustering, Knowl. Based Syst. 75 (2015) 152–160.