Expert Systems With Applications 168 (2021) 114438

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Modified DFS-based term weighting scheme for text classification

Long Chen [a], Liangxiao Jiang [a,b,*], Chaoqun Li [c,**]

[a] School 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] School of Mathematics and Physics, China University of Geosciences, Wuhan 430074, China

ARTICLE INFO

Keywords:
Text classification
Term weighting
Term frequency
Distinguishing feature selector

ABSTRACT

With the rapid growth of textual data on the Internet, text classification (TC) has attracted increasing attention. As a widely used text representation method, the vector space model (VSM) represents the content of a document as a vector composed of term frequency (TF) in the term space. Because different terms have different levels of importance in a document, designing an appropriate term weighting scheme is crucial to improve the performance of TC. In this study, we first conducted a comprehensive survey of the existing well-known term weighting schemes and found that they are not fully effective and that researchers are still focused on proposing new term weighting schemes. To further improve the performance of TC, we propose a new term weighting scheme based on the modified distinguishing feature selector (DFS), which we call TF–MDFS (modified DFS-based TF). Experimental results show that TF–MDFS is overall better than existing state-of-the-art term weighting schemes in terms of the classification accuracy of widely used base classifiers.

1. Introduction

With the rapid growth of textual data on the Internet, properly organizing, managing, and utilizing them is becoming a great challenge. Text classification (TC) is the task of automatically classifying a set of textual documents into different classes from a predefined set. As a first and vital step of TC, text representation converts the content of a textual document into a compact format so that the textual document can be classified by a classifier. Of the numerous text representation methods, the vector space model (VSM) (Salton & McGill, 1984) is widely used. In the VSM, the content of a textual document is represented as a term (feature) vector in the term space, where each term refers to a word occurring in the document (Jiang et al., 2013, 2016b; Wang et al., 2015b), and the term value corresponds to its weight, indicating the importance of the word in distinguishing document categories. One of the most common ways to indicate this weight is term frequency (TF). However, TF alone is insufficient because the terms that occur more often will have a very large weight in a document. Therefore, term weighting is an important factor in improving the effectiveness of TC by assigning appropriate weights to different terms (Jiang et al., 2016a, 2019a).

Based on whether class information in the document is used, related work can be broadly divided into two main categories: namely unsupervised term weighting and supervised term weighting (Dogan & Uysal, 2019). Unsupervised term weighting mainly includes TF–IDF (inverse document frequency-based TF) (Salton & Buckley, 1988) and its variants. Unsupervised term weighting ignores the available class information of training documents and is thus often ineffective for TC. To make use of the class information of training documents, the study of supervised term weighting schemes has attracted increasing attention. Debole and Sebastiani (2003) proposed TF–CHI (Chi-square statistic-based TF), TF–IG (information gain-based TF), and TF–GR (gain ratio-based TF); Lan et al. (2009) proposed TF–RF (relevance frequency-based TF); Liu et al. (2009) proposed TF–PB (probability-based TF); Wang and Zhang (2013) proposed TF–ICF (inverse class frequency-based TF); Ren and Sohrab (2013) proposed TF–IDF–ICF (inverse document frequency and inverse class frequency-based TF) and TF–IDF–ICSDF (inverse document frequency and inverse class space density frequency-based TF); Wang et al. (2015a) proposed TF–DC (distributional concentration-based TF) and TF–BDC (balanced distributional concentration-based TF); Chen et al. (2016) proposed TF–IGM (inverse gravity moment-based TF); and Dogan and Uysal (2019) proposed TF–IGM$_{imp}$ (improved inverse gravity moment-based TF).

Although there exist many term weighting schemes for TC, finding a more effective and practical term weighting scheme remains a great challenge. By analyzing the existing term weighting schemes, we found that most of them do not take full advantage of the distribution information of terms in all training documents. The distinguishing feature

[*] Corresponding author at: School of Computer Science, China University of Geosciences, Wuhan 430074, China.
[**] Corresponding author.
E-mail addresses: lchen@cug.edu.cn (L. Chen), ljiang@cug.edu.cn (L. Jiang), chqli@cug.edu.cn (C. Li).

https://doi.org/10.1016/j.eswa.2020.114438
Received 3 April 2020; Received in revised form 17 August 2020; Accepted 2 December 2020
Available online 7 December 2020

selector (DFS) (Uysal & Günal, 2012) is a well-accepted term (feature) selection method, which assigns a high score to a term that frequently occurs in a single class and does not occur in the other classes. This raises the question of whether using the term selection score of DFS directly as the term weight can provide better performance. To answer this question, we first adapted it as a term weighting scheme, namely TF–DFS (DFS-based TF), and found that there exist some defects in DFS when it comes to term weighting. To polish these defects, we propose a modified version of DFS (MDFS). Specifically, we first decomposed the term selection score of DFS into multiple (i.e., the number of classes) class-specific scores. Then, we calculated each class-specific score from "positive" and "negative" perspectives and defined the final class-specific score as their product. Finally, we assigned different weights to different class-specific scores and used the weighted sum across all class-specific scores as the whole term selection score of MDFS. Based on MDFS, we propose a new term weighting scheme simply called TF–MDFS. Extensive comparison results validate the advantages of TF–MDFS in terms of classification accuracy of widely used base classifiers, such as multinomial naive Bayes (MNB), support vector machines (SVM) and logistic regression (LR).

The remainder of this paper is organized as follows. We first present a comprehensive survey of existing well-known term weighting schemes in Section 2. Then, we propose our new term weighting scheme (TF–MDFS) in Section 3. The experiments and results are reported in Section 4. Finally, conclusions are drawn and some main directions for future work are outlined in Section 5.

2. Related work

In this section, we present a comprehensive survey of existing well-known term weighting schemes, as listed in Table 1.

The raw TF term weighting scheme is more reasonable and efficient than *binary*. However, TF may assign greater weights to common terms with weak text discriminating power, which results in a huge impact on the classification performance. To offset this deficiency, a global factor IDF (inverse document frequency) must be introduced to the TF scheme. The resulting term weighting scheme is TF–IDF (inverse document frequency-based TF) (Salton & Buckley, 1988). TF–IDF assigns high term weights to rare terms that occur in less documents. Because it ignores the available class information of training documents, it is a typical unsupervised term weighting scheme. After TF–IDF, many researchers have proposed variants (Deisy et al., 2010; Sabbah et al., 2017) and some other unsupervised term weighting schemes, such as document frequency (DF) (Yang & Pedersen, 1997), Glasgow weight (Al-Zaidy et al., 2011), and entropy (Krishnapuram et al., 2003).

One of the most straightforward ways to make use of the class information of training documents is to use term selection scores to define the term weights and then assign different weights to terms. Inspired by this idea, Debole and Sebastiani (2003) proposed TF–CHI (Chi-square statistic-based TF), TF–IG (information gain-based TF), and TF–GR (gain ratio-based TF) by replacing the IDF global factor in TF–IDF with term selection metrics: χ^2 statistic (CHI), information gain (IG), and gain ratio (GR), respectively. Because they make use of the class information of training documents, they are supervised term weighting schemes and are thus more promising and reasonable than TF–IDF.

TF–RF (relevance frequency-based TF) (Lan et al., 2009) was proposed to distinguish positive documents from negative documents. Experimental results showed that TF–RF improved the discriminating power of terms for TC tasks. In the same year, Liu et al. (2009) proposed a simple probability-based term weighting scheme called TF–PB (probability-based TF) to better distinguish documents in minority classes. TF–PB tackled the problem that classes with fewer instances are under-represented and the resulting classifiers often perform far below satisfactorily.

IDF measures the weight of a term on a single document, causing many terms in the collection to have the same IDF weight. Similarly, the specificity of a term to a class of documents should be reflected by its inverse class frequency (ICF). ICF is defined as the inverse ratio of the number of classes that the term occurs in training documents to the total number of classes. ICF means that the fewer the classes in which a term occurs, the more specific it is to those classes or the more it can distinguish between classes. Based on this premise, Wang and Zhang (2013) proposed TF–ICF (inverse class frequency-based TF) and its variant TF–ICF-Based, which have been proved to be better than TF–IDF for TC tasks.

Thanks to the success of TF–ICF, Ren and Sohrab (2013) merged inverse class frequency (ICF) with TF–IDF to propose TF–IDF–ICF. TF–IDF–ICF provides positive discrimination on rare terms in the vector space, whereas it is biased against frequent terms for TC tasks. Therefore, Ren and Sohrab (2013) revised the ICF function and implemented a new inverse class space density frequency (ICSDF) to propose TF–IDF–ICSDF, which provides positive discrimination of infrequent and frequent terms. Although TF–IDF–ICSDF takes class information into account on the basis of TF–IDF, it ignores the fact that it gains large values when irrelevant terms are sparsely distributed across all classes.

Considering that class-specific terms are more useful to discriminate different classes, Wang et al. (2015a) proposed TF–DC (distributional concentration-based TF) and TF–BDC (balanced distributional concentration-based TF). TF–DC and TF–BDC measure the discriminating power of a term based on its global distributional concentration in the classes of a corpus. Recently, Chen et al. (2016) found that the more concentrated the distribution of a term is among classes, the shorter the distance from the center of gravity to the origin, and the less the sum of class-specific gravity moments is. Based on their observation, they proposed TF–IGM (inverse gravity moment-based TF). TF–IGM uses a new statistical model called inverse gravity moment (IGM) to accurately measure the class discrimination capability of terms. To adequately reflect the distinguishing power of terms under certain circumstances, Dogan and Uysal (2019) proposed TF–IGM$_{imp}$ (improved inverse gravity moment-based TF).

3. Proposed scheme

Our research started from the DFS (Uysal & Günal, 2012). Therefore, we first provide an overview of it here. The DFS argues that an ideal term (feature) selection method should assign high scores to distinctive terms while assigning lower scores to irrelevant ones. Specifically, four requirements must be satisfied in DFS:

1. If a term occurs frequently in a single class and does not occur in other classes, it is distinctive and must be assigned a high score.
2. If a term occurs frequently in all classes, it is irrelevant and must be assigned a low score.
3. If a term occurs rarely in a single class and does not occur in other classes, it is irrelevant and must be assigned a low score.
4. If a term occurs in some of the classes, it is relatively distinctive and must be assigned a medium score.

To meet the above four requirements, DFS defines the term selection score of each term t_i as follows:

$$DFS(t_i) = \sum_{j=1}^{q} \frac{P(c_j|t_i)}{P(\bar{t}_i|c_j) + P(t_i|\bar{c}_j) + 1}, \tag{1}$$

where q is the total number of classes, $P(c_j|t_i)$ is the conditional probability of class c_j given the presence of term t_i, $P(\bar{t}_i|c_j)$ is the conditional probability of the absence of term t_i given class c_j, and $P(t_i|\bar{c}_j)$ is the conditional probability of term t_i given the absence of class c_j.

DFS (Uysal & Günal, 2012) has been proved to be an effective and efficient term (feature) selection method. This raises the question of

Table 1
Weighting notations of existing term weighting schemes.

Schemes	Formulas	Descriptions
TF	f_i	f_i: the raw term frequency of term t_i.
TF–IDF (Salton & Buckley, 1988)	$f_i \cdot \log\left(\frac{n}{d(t_i)}\right)$	q: the total number of classes.
TF–CHI (Debole & Sebastiani, 2003)	$f_i \cdot \max_{j=1}^{q}\left(\frac{n\cdot\left(d(t_i,c_j)\cdot d(\overline{t_i},\overline{c_j})-d(\overline{t_i},c_j)\cdot d(t_i,\overline{c_j})\right)^2}{d(t_i)\cdot d(\overline{t_i})\cdot d(c_j)\cdot d(\overline{c}_j)}\right)$	n: the total number of documents.
TF–IG (Debole & Sebastiani, 2003)	$f_i \cdot \max_{j=1}^{q}\left(\sum_{t\in\{t_j,\overline{t_j}\}}\sum_{c\in\{c_j,\overline{c_j}\}}\frac{d(t,c)}{n}\log\frac{n\cdot d(t,c)}{d(t)\cdot d(c)}\right)$	$d(t_i)$: the number of documents containing term t_i.
TF–GR (Debole & Sebastiani, 2003)	$f_i \cdot \max_{j=1}^{q}\left(\frac{\sum_{t\in\{t_j,\overline{t_j}\}}\sum_{c\in\{c_j,\overline{c_j}\}}\frac{d(t,c)}{n}\log\frac{n\cdot d(t,c)}{d(t)\cdot d(c)}}{-\sum_{c\in\{c_j,\overline{c_j}\}}\frac{d(c)}{n}\cdot\log\frac{d(c)}{n}}\right)$	$d(\overline{t_i})$: the number of documents not containing term t_i.
TF–RF (Lan et al., 2009)	$f_i \cdot \max_{j=1}^{q}\log\left(2+\frac{d(t_i,c_j)}{\max\left(1,d(t_i,\overline{c_j})\right)}\right)$	$d(c_j)$: the number of documents belonging to class c_j .
TF–PB (Liu et al., 2009)	$f_i \cdot \max_{j=1}^{q}\log\left(1+\frac{d(t_i,c_j)}{d(\overline{t_i},c_j)}\cdot\frac{d(t_i,c_j)}{d(t_i,\overline{c_j})}\right)$	$d(\overline{c_j})$: the number of documents not belonging to class c_j .
TF–ICF (Wang & Zhang, 2013)	$f_i \cdot \log\left(1+\frac{q}{c(t_i)}\right)$	$d(t_i,c_j)$: the number of documents belonging to class c_j containing term t_i.
TF–ICF-Based (Wang & Zhang, 2013)	$f_i \cdot \log\left(2+\frac{d(t_i,c_j)}{\max\left(1,d(t_i,\overline{c_j})\right)}\cdot\frac{q}{c(t_i)}\right)$	$d(\overline{t_i},c_j)$: the number of documents belonging to class c_j not containing term t_i.
TF–IDF–ICF (Ren & Sohrab, 2013)	$f_i \cdot \left(1+\log\frac{n}{d(t_i)}\right)\cdot\left(1+\log\frac{q}{c(t_i)}\right)$	$d(t_i,\overline{c_j})$: the number of documents not belonging to class c_j containing term t_i.
TF–IDF–ICSDF (Ren & Sohrab, 2013)	$f_i \cdot \left(1+\log\frac{n}{d(t_i)}\right)\cdot\left(1+\log\frac{q}{\sum_{j=1}^{q}\frac{d(t_i,c_j)}{d(c_j)}}\right)$	$d(\overline{t_i},\overline{c_j})$: the number of documents not belonging to class c_j not containing term t_i.
TF–DC (Wang et al., 2015a)	$f_i \cdot \left(1+\frac{1}{\log q}\cdot\sum_{j=1}^{q}\frac{d(t_i,c_j)}{d(t_i)}\log\frac{d(t_i,c_j)}{d(t_i)}\right)$	$c(t_i)$: the number of classes containing term t_i.
TF–BDC (Wang et al., 2015a)	$f_i \cdot \left(1+\frac{1}{\log q}\cdot\sum_{j=1}^{q}\frac{\frac{d(t_i,c_j)}{d(c_j)}}{\sum_{j=1}^{q}\frac{d(t_i,c_j)}{d(c_j)}}\log\frac{\frac{d(t_i,c_j)}{d(c_j)}}{\sum_{j=1}^{q}\frac{d(t_i,c_j)}{d(c_j)}}\right)$	f_{ir}: the number of documents containing term t_i in the rth class, which are sorted in descending order with r being the rank.
TF–IGM (Chen et al., 2016)	$f_i \cdot \left(1+\lambda\frac{f_{i1}}{\sum_{r=1}^{q}f_{ir}\cdot r}\right)$	$D_{total}(t_{i-max})$: the number of total documents in the 1th class.
TF–IGM$_{\text{imp}}$ (Dogan & Uysal, 2019)	$f_i \cdot \left(1+\lambda\frac{f_{i1}}{\sum_{r=1}^{q}f_{ir}\cdot r+\log_{10}\left[\frac{D_{total}(t_{i-max})}{f_{i1}}\right]}\right)$	λ: an adjustable coefficient.

whether using $DFS(t_i)$ directly as the weight of term t_i can provide better performance. To answer this question, we adapted it as a term weighting scheme, which we call TF–DFS (DFS-based TF). The detailed formula is

$$W_{\text{TF-DFS}}(t_i) = f_i \cdot DFS(t_i). \tag{2}$$

The experimental results in Section 4 show that the performance of TF–DFS, compared to the raw TF, does not produce an expected improvement. Why does TF–DFS perform so poorly? The fundamental reason is that DFS assigns scores to all terms between 0.5 and 1.0 according to their significance (Uysal & Günal, 2012). With the DFS evaluated in a small range of values, a natural question arises: "is the specificity of a term in a class adequately demonstrated?" To answer this question, a simple example could be helpful to get some intuitive feeling. Suppose there are 100 documents, 10 of which belong to class c_1 and 90 of which belong to class c_2. The number of documents containing term t_i in both classes is 5. It is obvious that the specificity score of term t_i in class c_1 should be much higher than that in class c_2. However, according to $DFS(t_i)$ calculated by Eq. (1), the specificity scores of term t_i for each class are relatively close, only 0.3214 and 0.2045, respectively.

To address this issue, we must find an appropriate evaluation criteria to more precisely judge the specificity of terms to a single class (Jiang et al., 2019b; Zhang et al., 2020). Therefore, in addition to the four requirements of DFS, we argue that an ideal term selection score should satisfy a fifth requirement: "For class c_j, $d(t_i,c_j)$ must be large and $d(\bar{t}_i,c_j)$ must be small; For class $\bar{c}_j$ (absence of class c_j), $d(\bar{t}_i,\bar{c}_j)$ must be large and $d(t_i,\bar{c}_j)$ must be small". However, the existing $DFS(t_i)$ does not satisfy this requirement. Therefore, we must modify the existing $DFS(t_i)$. Specifically, we first decomposed $DFS(t_i)$ into q class-specific scores $MDFS_{cs}(t_i,c_j)$. Then, we calculated each class-specific score $MDFS_{cs}(t_i,c_j)$ from both "positive" and "negative" perspectives and defined the final class-specific score as their product. The detailed formula is

$$MDFS_{cs}(t_i,c_j) = \frac{P(c_j|t_i)P(\bar{c}_j|\bar{t}_i)}{P(\bar{t}_i|c_j)+P(t_i|\bar{c}_j)+1}, \tag{3}$$

where $P(\bar{c}_j|\bar{t}_i)$ is the conditional probability of the absence of class c_j given the absence of t_i.

As can be seen from Eq. (3), the conditional probability $P(c_j|t_i)$ reflects the inter-class distribution of documents containing term t_i. The multiplication factor of $P(\bar{c}_j|\bar{t}_i)$ reflects the inter-class distribution of documents not containing term t_i. For each class c_j, this will make better use of the distribution information of term t_i in all training documents. Now, again for the above example, according to $MDFS_{cs}(t_i,c_j)$ calculated by Eq. (3), the specificity scores of term t_i for each class are widely spaced at 0.3036 and 0.0114, respectively. We can see that the specificity score of term t_i in class c_1 decreases only slightly, while the specificity score in class c_2 decreases to a very small value. This is the reason why $MDFS_{cs}(t_i,c_j)$ is more consistent with the evaluation criteria of the class-specific specificity of terms.

Besides, we argue that for each term, different class-specific scores should have different contributions (importance) to the whole term score. Therefore, we should assign different weights for different class-specific scores and then use the weighted sum across all class-specific scores as the whole term score. Based on this premise, we propose a modified DFS (MDFS). The detailed formula is

$$MDFS(t_i) = \sum_{j=1}^{q} w_{ij}\cdot MDFS_{cs}(t_i,c_j), \tag{4}$$

Table 2
The benchmark datasets used in our experiments.

Dataset	Source	#Documents	#Words	#Classes	min class size	max class size	avg class size
fbis	TREC	2463	2000	17	38	506	144.9
la1s	TREC	3204	31 472	6	273	943	534.0
la2s	TREC	3075	31 472	6	248	905	512.5
new3s	TREC	9558	26 832	44	104	696	217.2
oh0	OHSUMED-233445	1003	3182	10	51	194	100.3
oh10	OHSUMED-233445	1050	3238	10	52	165	105.0
oh15	OHSUMED-233445	913	3100	10	53	157	91.3
oh5	OHSUMED-233445	918	3012	10	59	149	91.8
ohscal	OHSUMED-233445	11 162	11 465	10	709	1621	1116.2
re0	Reuters-21578	1504	2886	13	11	608	115.7
re1	Reuters-21578	1657	3758	25	10	371	66.3
tr11	TREC	414	6429	9	6	132	46.0
tr12	TREC	313	5804	8	9	93	39.1
tr21	TREC	336	7902	6	4	231	56.0
tr23	TREC	204	5832	6	6	91	34.0
tr31	TREC	927	10 128	7	2	352	132.4
tr41	TREC	878	7454	10	9	243	87.8
tr45	TREC	690	8261	10	14	160	69.0
wap	WebACE	1560	8460	20	5	341	78.0

Table 3
Classification accuracy comparisons for TF–MDFS versus its competitors based on MNB.

Dataset	TF	TF–DC	TF–BDC	TF–IGM	TF–IGM$_{\text{imp}}$	TF–DFS	TF–MDFS
fbis	77.11 ± 2.49	79.45 ± 2.53	79.86 ± 2.28	79.02 ± 2.65	79.01 ± 2.65	77.86 ± 2.46	79.97 ± 2.57
la1s	88.41 ± 1.62	88.04 ± 1.66	88.11 ± 1.56	88.62 ± 1.70	88.86 ± 1.59	88.01 ± 1.60	88.80 ± 1.66
la2s	89.88 ± 1.55	88.94 ± 1.57	89.13 ± 1.60	90.43 ± 1.51	90.43 ± 1.54	89.79 ± 1.47	90.21 ± 1.52
new3s	79.28 ± 1.09	79.61 ± 1.26	79.77 ± 1.30	81.37 ± 1.22	81.48 ± 1.26	78.74 ± 1.14	82.55 ± 1.24
oh0	89.55 ± 2.82	93.08 ± 2.13	93.05 ± 2.29	92.40 ± 2.68	92.79 ± 2.55	90.91 ± 2.77	93.55 ± 2.20
oh10	80.60 ± 3.13	83.68 ± 2.92	83.47 ± 2.75	83.26 ± 2.88	83.61 ± 2.85	81.56 ± 2.94	84.60 ± 2.90
oh15	83.60 ± 3.13	86.24 ± 3.13	86.00 ± 3.28	85.65 ± 3.10	86.36 ± 2.97	84.45 ± 3.05	88.35 ± 2.68
oh5	86.63 ± 3.07	92.00 ± 2.75	92.22 ± 2.70	91.23 ± 2.83	91.63 ± 2.88	87.97 ± 3.04	93.06 ± 2.77
ohscal	74.70 ± 1.18	77.57 ± 1.25	77.54 ± 1.21	76.00 ± 1.25	76.75 ± 1.31	75.75 ± 1.17	79.20 ± 1.20
re0	80.02 ± 2.95	78.80 ± 2.62	78.93 ± 2.93	78.99 ± 2.95	79.44 ± 3.16	80.63 ± 2.56	82.71 ± 2.82
re1	83.31 ± 2.75	86.71 ± 2.12	87.51 ± 2.03	83.97 ± 2.67	84.98 ± 2.56	82.76 ± 2.43	88.18 ± 2.21
tr11	85.21 ± 4.90	86.70 ± 4.35	86.46 ± 4.32	86.36 ± 4.04	86.46 ± 4.27	85.01 ± 5.28	88.66 ± 4.24
tr12	80.99 ± 6.08	85.24 ± 5.41	84.02 ± 6.22	84.73 ± 5.69	85.34 ± 5.64	83.13 ± 5.79	86.67 ± 6.14
tr21	61.90 ± 8.78	64.95 ± 8.56	67.80 ± 8.50	66.13 ± 8.23	66.19 ± 8.47	66.92 ± 7.92	80.90 ± 6.44
tr23	71.15 ± 9.68	76.85 ± 8.53	82.73 ± 8.00	73.61 ± 9.33	73.90 ± 9.55	74.77 ± 9.54	89.98 ± 6.76
tr31	94.60 ± 2.41	95.43 ± 2.31	95.74 ± 2.28	95.63 ± 2.17	96.00 ± 2.00	94.73 ± 2.39	97.80 ± 1.82
tr41	94.65 ± 2.21	94.95 ± 2.21	94.82 ± 2.19	94.53 ± 2.27	94.71 ± 2.21	94.67 ± 2.36	95.96 ± 2.05
tr45	83.64 ± 4.33	89.62 ± 3.47	91.55 ± 3.36	90.32 ± 3.37	91.07 ± 3.27	84.88 ± 4.15	93.07 ± 3.11
wap	81.22 ± 2.59	79.46 ± 2.52	80.49 ± 2.69	81.69 ± 2.77	82.81 ± 2.54	79.01 ± 2.41	84.35 ± 2.58
Average	82.44	84.60	85.22	84.42	84.83	83.24	87.82
Ranking	6.1053	3.9474	3.5	4.4474	3.1579	5.6842	1.1579

where w_{ij} represents the specific weighting factor of term t_i for class c_j, which can be defined as

$$w_{ij} = \log\left(1 + \frac{d(t_i, c_j)}{\max\left(1, d(t_i, \overline{c_j})\right)} \cdot \frac{d(\overline{t_i}, \overline{c_j})}{\max\left(1, d(\overline{t_i}, c_j)\right)}\right), \tag{5}$$

where the constant 1 is used to avoid negative weights. In an extreme case, if $d(t_i, \overline{c_j}) = 0$ or $d(\overline{t_i}, c_j) = 0$, we set the minimal denominator to 1 to avoid the zero denominator problem.

Before presenting the detailed comparison results, it could be helpful to get some intuitive feeling on the advantages of MDFS through a simple example. Suppose there are five terms(t_1, t_2, t_3, t_4 and t_5) whose document frequencies in five classes are $\{9,0,0,0,0\}$, $\{9,8,0,0,0\}$, $\{9,8,7,0,0\}$, $\{9,8,7,6,0\}$, and $\{9,8,7,6,5\}$, respectively. Then, assume that each class has 10 documents. According to $DFS(t_i)$ calculated by Eq. (1), the term scores of t_1, t_2, t_3, t_4, and t_5 are 0.9091, 0.7375, 0.6317, 0.5614, and 0.5130, respectively. However, according to our $MDFS(t_i)$ calculated by Eq. (4), they are 7.5351, 3.2625, 1.8252, 1.0802, and 0.6490, respectively. It is obvious that the term scores by DFS are intensively distributed in the range of 0.5 to 1, whereas the distribution obtained by MDFS is more divergent.

Algorithm 1 TF–MDFS Learning (D)

```
Input: $D$-a training document set
Output: All term weights
for each term $t_i$ $(i = 1, 2, \cdots, m)$ and each class $c_j$ $(j = 1, 2, \cdots, q)$ do
    Calculate $MDFS_{cs}(t_i, c_j)$ by Eq. (3)
    Calculate $w_{ij}$ by Eq. (5)
end for
for each term $t_i$ $(i = 1, 2, \cdots, m)$ do
    Calculate $MDFS(t_i)$ by Eq. (4)
    Calculate $W_{\text{TF-MDFS}}(t_i)$ by Eq. (6)
end for
return $W_{\text{TF-MDFS}}(t_i)$ $(i = 1, 2, \cdots, m)$
```

Finally, based on $MDFS(t_i)$, we propose a new term weighting scheme called TF–MDFS (modified DFS-based TF). The detailed formula is

$$W_{\text{TF-MDFS}}(t_i) = f_i \cdot MDFS(t_i). \tag{6}$$

Now, the detailed learning algorithm for our TF–MDFS can be briefly summarized by Algorithm 1. As can be seen from Algorithm 1, the execution time for computing all these weights is proportional to $2mq + 2m$. If we only take the highest order term, the training time complexity for obtaining these weights is only $O(2mq)$, the same as for

Table 4
Statistical test comparisons for each pair of term weighting schemes based on MNB.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
21	TF vs. TF–MDFS	7.058829	0	0.002381
20	TF–DFS vs. TF–MDFS	6.458078	0	0.0025
19	TF–IGM vs. TF–MDFS	4.69337	0.000003	0.002632
18	TF vs. TF–IGM$_{imp}$	4.20526	0.000026	0.002778
17	TF–DC vs. TF–MDFS	3.979978	0.000069	0.002941
16	TF vs. TF–BDC	3.717149	0.000201	0.003125
15	TF–IGM$_{imp}$ vs. TF–DFS	3.604508	0.000313	0.003333
14	TF–BDC vs. TF–MDFS	3.34168	0.000833	0.003571
13	TF–BDC vs. TF–DFS	3.116398	0.001831	0.003846
12	TF vs. TF–DC	3.078851	0.002078	0.004167
11	TF–IGM$_{imp}$ vs. TF–MDFS	2.853569	0.004323	0.004545
10	TF–DC vs. TF–DFS	2.4781	0.013208	0.005
9	TF vs. TF–IGM	2.365459	0.018008	0.005556
8	TF–IGM vs. TF–IGM$_{imp}$	1.839801	0.065797	0.00625
7	TF–IGM vs. TF–DFS	1.764707	0.077613	0.007143
6	TF–BDC vs. TF–IGM	1.351691	0.176474	0.008333
5	TF–DC vs. TF–IGM$_{imp}$	1.126409	0.259992	0.01
4	TF–DC vs. TF–IGM	0.713392	0.475603	0.0125
3	TF–DC vs. TF–BDC	0.638298	0.523279	0.016667
2	TF vs. TF–DFS	0.600751	0.548006	0.025
1	TF–BDC vs. TF–IGM$_{imp}$	0.488111	0.625472	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.005:

- TF vs. TF–MDFS;
- TF–DFS vs. TF–MDFS;
- TF–IGM vs. TF–MDFS;
- TF vs. TF–IGM$_{imp}$;
- TF–DC vs. TF–MDFS;
- TF vs. TF–BDC;
- TF–IGM$_{imp}$ vs. TF–DFS;
- TF–BDC vs. TF–MDFS;
- TF–BDC vs. TF–DFS;
- TF vs. TF–DC;
- TF–IGM$_{imp}$ vs. TF–MDFS.

TF–DC and TF–BDC. Compared to the latest TF–IGM and TF–IGM$_{imp}$, which have training time complexities of $O(mq \log q)$, our TF–MDFS is slightly faster. In short, our TF–MDFS is simple and efficient, and the experimental results in the next section validate its effectiveness.

4. Experiments and results

4.1. Experiments on benchmark text datasets

The purpose of this section is to validate the effectiveness of our proposed TF–MDFS. Therefore, we compare our TF–MDFS with TF, TF–DFS, and four existing state-of-the-art term weighting schemes published in the last five years, including TF–DC, TF–BDC, TF–IGM, and TF–IGM$_{imp}$, in terms of the classification accuracy of widely used base classifiers, such as multinomial naive Bayes (MNB) (McCallum & Nigam, 1998), support vector machines (SVM) (Cortes & Vapnik, 1995) and logistic regression (LR) (Aseervatham et al., 2012; Zhang & Oles, 2001), respectively. We used the existing implementations of MNB, SVM (LibLINEAR with L2-regularized L2-loss support vector classification (dual) (Fan et al., 2008)) and LR (LibLINEAR with L2-regularized logistic regression (dual) (Fan et al., 2008)) in the WEKA platform (Witten et al., 2011) and implemented all term weighting schemes in the WEKA platform (Witten et al., 2011).

We ran our experiments on 19 widely used multi-class text datasets[1] published on the main web site of the WEKA platform, which represent a wide range of domains and data characteristics. All these problems come from TREC, OHSUMED-233445, Reuters-21578, and WebACE. The data were originally converted to word counts by Han and Karypis (2000). Dataset "fbis" is from the Foreign Broadcast Information Service data of TREC-5. Datasets "la1s" and "la2s" are from the Los Angeles Times data of TREC-5. Datasets "tr11", "tr12", "tr21", "tr23", "tr31", "tr41", "tr45", and "new3s" are derived from TREC-5, TREC-6, and TREC-7 collections. Datasets "oh0", "oh5", "oh10", "oh15", and "ohscal" are from OHSUMED collection subset of MEDLINE database. Datasets "re0" and "re1" are from Reuters-21578 text categorization test collection Distribution 1.0. Dataset "wap" is from the WebACE project (WAP). Table 2 describes the detailed information of these 19 benchmark text datasets.

Tables 3, 5 and 7 show detailed comparison results in terms of the classification accuracy of MNB, SVM and LR, respectively. In our experiments, the classification accuracy is defined as the percentage of test instances correctly classified. All classification accuracy estimates were obtained by averaging the results from 10 separate runs of stratified 10-fold cross-validation. The average of each classifier across all datasets provides a gross indicator of the relative performance in addition to other statistics.

Then, we took advantage of the well-known KEEL Data-Mining Software Tool (Alcalá-Fdez et al., 2011)[2] to complete a Friedman test (Demsar, 2006; Garcia & Herrera, 2008; Zhang et al., 2016) to thoroughly compare each pair of term weighting schemes. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA (Demsar, 2006). The average rankings of the algorithms obtained by applying the Friedman test are also summarized at the bottom of Tables 3, 5 and 7, respectively. With 7 algorithms and 19 datasets, F_F is distributed according to the F distribution with 6 and 108 degrees of freedom: 25.85778, 7.80034 and 8.982649, respectively, which are all greater than the critical value of $F(6,108)$ for $\alpha = 0.05$ (The table of critical values can be found in any statistical books). So we reject the null hypotheses and proceed with the post-hoc Holm's test to further analyze which pairs of algorithms are significantly different. Tables 4, 6 and 8 report the detailed results achieved on post-hoc comparisons for $\alpha = 0.05$, and also indicate which pairs of algorithms are significantly different. From these comparisons, we can see the following:

1. In terms of MNB, the average rankings of them are TF–MDFS (1.1579), TF–IGM$_{imp}$ (3.1579), TF–BDC (3.5), TF–DC (3.9474), TF–IGM (4.4474), TF–DFS (5.6842) and TF (6.1053), respectively. According to the post-hoc Holm's test, TF–MDFS is notably better than all the other existing competitors.
2. In terms of SVM, the average rankings of them are TF–MDFS (2.5), TF–IGM$_{imp}$ (3.3158), TF–DC (3.4737), TF–BDC (3.8158), TF–IGM (4), TF–DFS (4.6579) and TF (6.2368), respectively. According to the post-hoc Holm's test, TF–MDFS is markedly better than TF and TF–DFS.
3. In terms of LR, the average rankings of them are TF–MDFS (2.1316), TF–DC (3.3421), TF–IGM$_{imp}$ (3.4211), TF–IGM (3.8947), TF–BDC (4.2895), TF–DFS (4.8684) and TF (6.0526), respectively. According to the post-hoc Holm's test, TF–MDFS is significantly better than TF, TF–DFS and TF–BDC.
4. In terms of MNB, SVM and LR, TF–DC, TF–IGM$_{imp}$ and our TF–MDFS, significantly outperform the raw TF. TF–BDC is markedly better than the raw TF in terms of MNB and SVM. TF–IGM is markedly better than the raw TF in terms of SVM and LR. This indicates that using TF alone is insufficient and term weighting is very important in improving the effectiveness of TC by assigning appropriate weights to different terms.
5. The improvements achieved by TF–DFS are very limited. The difference between TF–DFS and the raw TF is not significant. This indicates that using the existing term selection score of DFS directly as the term weight is not an appropriate term weighting scheme.
6. TF–MDFS significantly outperforms TF–DFS. This indicates that our modified DFS is notably better than the existing DFS when it comes to term weighting.
7. According to all above experimental results, we can roughly rank the performance of all these term weighting schemes as: TF–MDFS >TF–IGM$_{imp}$ >TF–DC >TF–BDC >TF–IGM >TF–DFS >TF.

[1] https://waikato.github.io/weka-wiki/datasets/

[2] http://sci2s.ugr.es/keel/

Table 5
Classification accuracy comparisons for TF–MDFS versus its competitors based on SVM.

Dataset	TF	TF–DC	TF–BDC	TF–IGM	TF–IGM$_{imp}$	TF–DFS	TF–MDFS
fbis	82.98 ± 1.99	84.51 ± 1.83	85.20 ± 1.79	83.95 ± 1.89	83.96 ± 1.96	83.52 ± 1.93	84.40 ± 1.73
la1s	88.30 ± 1.85	88.47 ± 1.88	87.98 ± 1.71	88.51 ± 1.94	88.46 ± 1.89	88.52 ± 1.89	88.30 ± 1.75
la2s	90.15 ± 1.43	89.36 ± 1.45	88.91 ± 1.59	89.99 ± 1.51	90.12 ± 1.55	90.27 ± 1.42	90.07 ± 1.69
new3s	87.33 ± 1.03	87.45 ± 0.95	86.91 ± 1.04	87.90 ± 1.06	87.99 ± 1.08	87.80 ± 1.02	88.32 ± 1.00
oh0	89.49 ± 2.56	92.24 ± 2.46	91.91 ± 2.67	92.44 ± 2.29	92.42 ± 2.30	90.32 ± 2.41	92.26 ± 2.47
oh10	80.36 ± 3.69	82.99 ± 3.23	82.69 ± 3.16	83.56 ± 3.02	83.39 ± 3.20	80.87 ± 3.76	83.28 ± 3.31
oh15	84.26 ± 3.65	85.51 ± 3.42	85.04 ± 3.56	86.31 ± 3.49	86.62 ± 3.42	84.92 ± 3.42	86.69 ± 3.27
oh5	89.77 ± 2.83	91.91 ± 2.85	91.79 ± 2.83	92.55 ± 2.62	92.40 ± 2.72	90.45 ± 2.98	92.21 ± 2.80
ohscal	75.68 ± 1.29	77.55 ± 1.16	77.49 ± 1.17	76.94 ± 1.20	77.07 ± 1.19	76.15 ± 1.23	77.32 ± 1.22
re0	84.91 ± 2.38	86.04 ± 2.48	85.44 ± 2.38	83.87 ± 2.78	84.09 ± 2.94	85.79 ± 2.41	84.96 ± 2.55
re1	85.88 ± 2.62	87.69 ± 2.54	87.95 ± 2.22	85.61 ± 2.72	85.78 ± 2.81	87.30 ± 2.49	87.14 ± 2.33
tr11	89.36 ± 4.14	90.37 ± 4.12	90.90 ± 4.24	90.22 ± 3.87	90.37 ± 3.81	90.13 ± 4.08	91.09 ± 3.77
tr12	88.85 ± 5.30	90.12 ± 4.79	89.64 ± 4.95	90.12 ± 5.37	90.25 ± 5.20	89.51 ± 5.38	91.30 ± 5.08
tr21	91.39 ± 4.30	92.76 ± 4.08	93.00 ± 4.29	92.31 ± 4.17	92.43 ± 4.02	92.76 ± 3.97	94.31 ± 4.05
tr23	90.50 ± 6.41	92.11 ± 6.08	93.34 ± 5.38	91.52 ± 5.92	92.11 ± 5.92	91.43 ± 5.80	94.81 ± 4.68
tr31	98.39 ± 1.36	98.65 ± 1.19	98.74 ± 1.17	98.60 ± 1.18	98.56 ± 1.16	98.51 ± 1.31	98.71 ± 1.17
tr41	96.59 ± 2.00	97.77 ± 1.54	97.74 ± 1.63	97.74 ± 1.52	97.78 ± 1.51	97.10 ± 1.73	97.26 ± 1.82
tr45	94.45 ± 2.77	95.59 ± 2.73	95.90 ± 2.43	95.71 ± 2.61	95.84 ± 2.53	95.26 ± 2.52	96.52 ± 2.15
wap	85.15 ± 2.39	84.72 ± 2.45	84.06 ± 2.52	84.76 ± 2.66	85.22 ± 2.58	85.58 ± 2.36	86.01 ± 2.29
Average	88.09	89.25	89.19	89.09	89.20	88.75	89.73
Ranking	6.2368	3.4737	3.8158	4	3.3158	4.6579	2.5

Table 6
Statistical test comparisons for each pair of term weighting schemes based on SVM.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
21	TF vs. TF–MDFS	5.331669	0	0.002381
20	TF vs. TF–IGM$_{imp}$	4.167713	0.000031	0.0025
19	TF vs. TF–DC	3.942431	0.000081	0.002632
18	TF vs. TF–BDC	3.454321	0.000552	0.002778
17	TF vs. TF–IGM	3.191492	0.001415	0.002941
16	TF–DFS vs. TF–MDFS	3.078851	0.002078	0.003125
15	TF vs. TF–DFS	2.252818	0.024271	0.003333
14	TF–IGM vs. TF–MDFS	2.140177	0.03234	0.003571
13	TF–IGM$_{imp}$ vs. TF–DFS	1.914895	0.055506	0.003846
12	TF–BDC vs. TF–MDFS	1.877348	0.06047	0.004167
11	TF–DC vs. TF–DFS	1.689613	0.091102	0.004545
10	TF–DC vs. TF–MDFS	1.389238	0.164761	0.005
9	TF–BDC vs. TF–DFS	1.201503	0.229556	0.005556
8	TF–IGM$_{imp}$ vs. TF–MDFS	1.163956	0.244442	0.00625
7	TF–IGM vs. TF–IGM$_{imp}$	0.976221	0.328955	0.007143
6	TF–IGM vs. TF–DFS	0.938674	0.347898	0.008333
5	TF–DC vs. TF–IGM	0.750939	0.452689	0.01
4	TF–BDC vs. TF–IGM$_{imp}$	0.713392	0.475603	0.0125
3	TF–DC vs. TF–BDC	0.488111	0.625472	0.016667
2	TF–BDC vs. TF–IGM	0.262829	0.792683	0.025
1	TF–DC vs. TF–IGM$_{imp}$	0.225282	0.82176	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.003333:

- TF vs. TF–MDFS;
- TF vs. TF–IGM$_{imp}$;
- TF vs. TF–DC;
- TF vs. TF–BDC;
- TF vs. TF–IGM;
- TF–DFS vs. TF–MDFS.

4.2. Experiments on real-world text datasets

In the previous experiments, we focused on the term weighting schemes widely used in VSM. Although the text representation of VSM is simple, it ignores the context of each term, thus losing a lot of semantic information in the text. We add a word embedding approach as a comparison algorithm in the following experiment. The word embedding model maps each word to a continuous vector space where semantically similar words are mapped to adjacent regions. This facilitates the acquisition of semantic information in the document.

In our study, we choose word2vec (Mikolov et al., 2013) to produce the word vectors. After data preprocessing, we use 300-dimensional version in the word2vec model. Each document is represented by the average of all word vectors in the document. Since negative values will appear in the final word vector, MNB cannot be used as the base classifier. In the following experiments, we only show the comparative experimental results on SVM and LR.

In order to further explore the performance of different term weighting schemes and the word embedding approach on different types of datasets, we observe their performance on four real-world text classification datasets: Movie Review,[3] 20 Newsgroups,[4] Reuters-21578[5] and RCV1.[6]

Movie Review: We choose the polarity dataset v2.0, which is a binary dataset containing 1000 positive and 1000 negative processed reviews. Stop-word removal and stemming (Porter, 1980) are carried out in the preprocessing phase. To save training time, rare words which occur less than 10 times in the dataset, numbers, punctuation marks and other non-alphabetic characters are removed. At the same time, the letters are converted to lower case. The resulting corpus has a vocabulary of 7103 terms. Each document contains an average of 351.21 terms.

20 Newsgroups: This dataset contains 19,997 documents of newsgroup messages, which are divided into 20 classes. Except for 997 documents in one class, there are 1000 documents in each of the remaining 19 classes. By preprocessing in the same way as above for the Movie Review dataset, the resulting corpus has a vocabulary of 20746 terms. Each document contains an average of 244.55 terms.

Reuters-21578: The top-10 largest classes of the celebrated Reuters-21578 were selected in our experiment. Among the reduced 9980 documents, the largest class (earn) contains 3964 documents and the smallest class (corn) contains 237 documents. Unlike the above two balanced datasets, this is a very unbalanced dataset. After preprocessing, the resulting corpus has a vocabulary of 4854 terms and each document contains an average of 83.64 terms.

RCV1: This dataset contains feature characteristics of documents originally written in five different languages and their translations. We only choose the original English documents containing 18758 documents in our experiment. This dataset contains 6 classes: C15, CCAT, E21, ECAT, GCAT and M11. The number of documents for each class is 5102, 4331, 1234, 2055, 4829 and 1207, respectively. The corpus has a vocabulary of 21,531 terms and each document contains an average of 362.17 terms. Since each document in this dataset has been preprocessed and the context information of each term has been

[3] http://www.cs.cornell.edu/people/pabo/movie-review-data/

[4] http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

[5] http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization +Collection

[6] http://archive.ics.uci.edu/ml/datasets/Reuters+RCV1+RCV2+Multilingual, +Multiview+Text+Categorization+Test+collection#

Table 7
Classification accuracy comparisons for TF–MDFS versus its competitors based on LR.

Dataset	TF	TF–DC	TF–BDC	TF–IGM	TF–IGM$_{imp}$	TF–DFS	TF–MDFS
fbis	85.42 ± 1.89	86.82 ± 1.84	86.74 ± 1.63	85.78 ± 1.84	85.79 ± 1.81	86.09 ± 1.84	86.41 ± 1.71
la1s	89.83 ± 1.71	89.98 ± 1.64	89.42 ± 1.63	89.79 ± 1.80	89.73 ± 1.81	90.05 ± 1.65	89.77 ± 1.63
la2s	91.41 ± 1.47	91.27 ± 1.38	90.52 ± 1.71	91.31 ± 1.47	91.39 ± 1.49	91.57 ± 1.48	91.60 ± 1.56
new3s	88.75 ± 0.99	88.90 ± 1.01	88.36 ± 1.01	89.21 ± 0.98	89.22 ± 1.03	89.14 ± 1.02	89.51 ± 0.96
oh0	90.12 ± 2.43	92.85 ± 2.26	92.51 ± 2.39	93.17 ± 2.27	93.02 ± 2.27	90.29 ± 2.37	92.89 ± 2.35
oh10	82.05 ± 3.39	85.26 ± 2.93	84.98 ± 3.15	84.92 ± 2.98	84.64 ± 3.18	82.79 ± 3.37	84.91 ± 3.10
oh15	85.22 ± 3.48	87.32 ± 3.34	86.92 ± 3.33	87.43 ± 3.28	87.70 ± 3.24	86.07 ± 3.26	87.87 ± 3.18
oh5	90.81 ± 2.88	93.45 ± 2.57	93.15 ± 2.71	93.36 ± 2.42	93.37 ± 2.63	91.22 ± 2.71	92.89 ± 2.63
ohscal	78.28 ± 1.20	81.27 ± 1.13	81.27 ± 1.14	79.08 ± 1.13	79.16 ± 1.17	79.17 ± 1.13	79.96 ± 1.09
re0	86.87 ± 2.24	86.34 ± 2.19	84.31 ± 2.33	86.56 ± 2.39	86.47 ± 2.33	86.26 ± 2.20	86.98 ± 2.30
re1	88.04 ± 2.41	88.34 ± 2.16	87.89 ± 2.35	88.66 ± 2.30	88.82 ± 2.32	88.30 ± 2.29	89.59 ± 1.90
tr11	89.51 ± 3.89	90.59 ± 3.96	91.00 ± 4.12	90.25 ± 3.99	90.32 ± 3.97	89.80 ± 3.72	91.45 ± 3.80
tr12	89.64 ± 5.05	91.21 ± 4.75	90.92 ± 4.50	90.60 ± 5.21	90.73 ± 5.04	90.12 ± 5.05	91.59 ± 5.03
tr21	91.99 ± 4.05	92.73 ± 3.87	93.09 ± 3.58	92.94 ± 3.90	93.00 ± 3.76	92.58 ± 3.97	94.37 ± 3.66
tr23	90.50 ± 6.73	93.00 ± 5.28	94.81 ± 4.57	92.75 ± 5.66	92.90 ± 5.65	91.44 ± 6.15	95.11 ± 4.34
tr31	98.17 ± 1.39	98.77 ± 1.12	98.72 ± 1.19	98.62 ± 1.19	98.61 ± 1.19	98.25 ± 1.44	98.84 ± 1.11
tr41	96.61 ± 1.82	97.39 ± 1.63	97.27 ± 1.85	97.51 ± 1.53	97.54 ± 1.50	96.98 ± 1.72	96.98 ± 1.80
tr45	94.48 ± 2.59	95.14 ± 2.60	95.32 ± 2.42	95.67 ± 2.48	95.71 ± 2.51	95.16 ± 2.62	96.25 ± 1.96
wap	85.66 ± 2.37	85.53 ± 2.52	84.86 ± 2.58	85.38 ± 2.79	85.72 ± 2.60	86.24 ± 2.37	86.99 ± 2.16
Average	89.12	90.32	90.11	90.16	90.20	89.55	90.73
Ranking	6.0526	3.3421	4.2895	3.8947	3.4211	4.8684	2.1316

Table 8
Statistical test comparisons for each pair of term weighting schemes based on LR.

i	Algorithms	$z = (R_0 - R_i)/SE$	p	Holm
21	TF vs. TF–MDFS	5.594497	0	0.002381
20	TF–DFS vs. TF–MDFS	3.904884	0.000094	0.0025
19	TF vs. TF–DC	3.867337	0.00011	0.002632
18	TF vs. TF–IGM$_{imp}$	3.754696	0.000174	0.002778
17	TF–BDC vs. TF–MDFS	3.078851	0.002078	0.002941
16	TF vs. TF–IGM	3.078851	0.002078	0.003125
15	TF–IGM vs. TF–MDFS	2.515647	0.011881	0.003333
14	TF vs. TF–BDC	2.515647	0.011881	0.003571
13	TF–DC vs. TF–DFS	2.177724	0.029427	0.003846
12	TF–IGM$_{imp}$ vs. TF–DFS	2.065083	0.038915	0.004167
11	TF–IGM$_{imp}$ vs. TF–MDFS	1.839801	0.065797	0.004545
10	TF–DC vs. TF–MDFS	1.72716	0.084139	0.005
9	TF vs. TF–DFS	1.689613	0.091102	0.005556
8	TF–IGM vs. TF–DFS	1.389238	0.164761	0.00625
7	TF–DC vs. TF–BDC	1.351691	0.176474	0.007143
6	TF–BDC vs. TF–IGM$_{imp}$	1.23905	0.215327	0.008333
5	TF–BDC vs. TF–DFS	0.826033	0.408785	0.01
4	TF–DC vs. TF–IGM	0.788486	0.430412	0.0125
3	TF–IGM vs. TF–IGM$_{imp}$	0.675845	0.499139	0.016667
2	TF–BDC vs. TF–IGM	0.563204	0.573296	0.025
1	TF–DC vs. TF–IGM$_{imp}$	0.112641	0.910315	0.05

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.003333:
• TF vs. TF–MDFS; • TF–DFS vs. TF–MDFS; • TF vs. TF–DC;
• TF vs. TF–IGM$_{imp}$; • TF–BDC vs. TF–MDFS; • TF vs. TF–IGM.

Table 9
The real-world datasets used in our experiments.

Dataset	#Documents	#Words	#Classes	#Average length
Movie Review	2000	7103	2	351.21
20 Newsgroups	19 997	20 746	20	244.55
Reuters-21578	9980	4854	10	83.64
RCV1	18 758	21 531	6	362.17

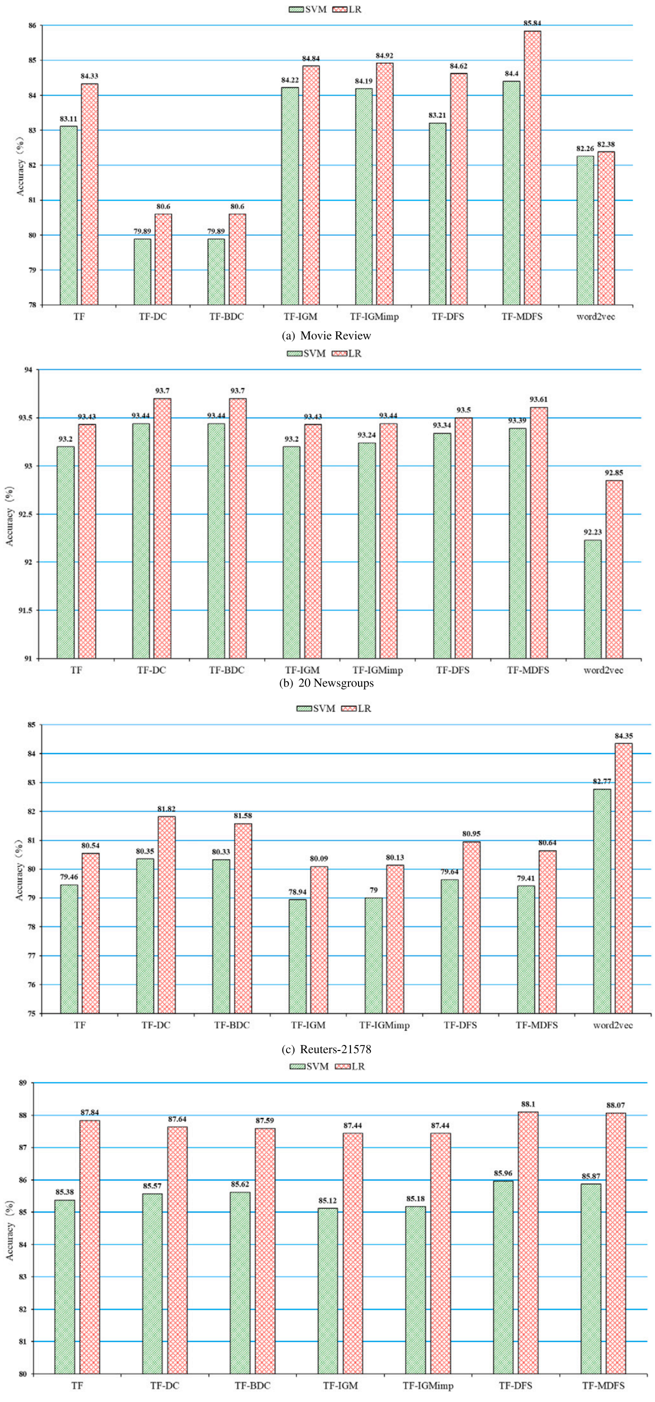
lost, we did not include word2vec into our comparisons on this group of experiments.

Table 9 summarizes the characteristics of these four datasets. The same as the experiments on benchmark text datasets, all comparison results were also obtained by 10 separate runs of stratified 10-fold cross-validation. Fig. 1 shows the detailed comparison results based on SVM and LR, respectively. From these comparisons, we can see that:

1. From the experimental results on the Movie Review dataset, TF–MDFS is notably better than all the other existing competitors.
2. From the experimental results on the 20 Newsgroups dataset, the performance of TF–MDFS is very close to TF–DC and TF–BDC, and is also better than other competitors.
3. By comparing the results on the Movie Review dataset with that on the 20 Newsgroups dataset, we can find that the performance improvement of term weighting schemes on the low-dimensional data is higher than that on the high-dimensional data. This indicates that the dimensionality of the data may affect the performance of the term weighting schemes.
4. From the experimental results on the Reuters-21578 dataset, the advantages of TF–MDFS disappeared and even that TF–MDFS is only comparative with TF, TF–IGM, TF–IGM$_{imp}$ and TF–DFS. This is because our proposed TF–MDFS focuses too much on distinctive terms in small categories in this unbalanced Reuters-21578 dataset. To improve the performance of TF–MDFS in the class-imbalanced classification situations, re-balancing methods (Jiang et al., 2015), such as sampling, weighting and cloning, are alternative. This could be an interesting topic for our future work.
5. On the Reuters-21578 dataset, word2vec performs significantly better than all the term weighting schemes. But the performance of word2vec on the 20 Newsgroups dataset and Movie Review dataset with relatively long document length is not satisfactory. The main reason is that part of the semantic information may be lost in the process of generating the document vector based on the averaged word vectors. In a long text document, it is not easy to accurately obtain the semantic information of the entire document.
6. From the experimental results on the RCV1 dataset, the performance of TF–MDFS is very close to TF–DFS and is also overall better than other four existing term weighting schemes.
7. On the whole, our proposed TF–MDFS has overall better stability on different types of datasets and better classification performance in most cases.

5. Conclusion and future work

Although there exist many term weighting schemes for TC, finding an appropriate term weighting scheme remains a great challenge. In this study, we first conducted a comprehensive survey of existing well-known term weighting schemes and found that most of them do not



(a) Movie Review

(b) 20 Newsgroups

(c) Reuters-21578

(d) RCV1

Fig. 1. Classification accuracy comparisons on four real-world datasets.

take full advantage of the distribution information of terms in all training documents. Based on the analysis and modification of the well-accepted distinguishing feature selector (DFS), we then proposed a modified DFS (MDFS) and a new term weighting scheme called TF–MDFS. Considering its simplicity, effectiveness, and efficiency, TF–MDFS could be a promising term weighting scheme in many real-world applications.

Currently, we use the simplest vector space model (VSM) for text representation. We believe that the use of more sophisticated text representation methods, such as word embedding methods, could improve the performance of the current TF–MDFS and make its advantage stronger. This will be a major direction for our future work. In addition, applying more sophisticated optimization methods (Gong & Cai, 2013; Gong et al., 2020; Lu et al., 2018; Yan et al., 2018), to search term weights is another interesting topic for future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work was partially supported by National Natural Science Foundation of China (U1711267) and Fundamental Research Funds for the Central Universities, China (CUGGC03).

References

Al-Zaidy, R., Fung, B. C. M., & Youssef, A. M. (2011). Towards discovering criminal communities from textual data. In *Proceedings of the 2011 ACM symposium on applied computing*. (pp. 172–177).

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., & García, S. (2011). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, *17*(2–3), 255–287.

Aseervatham, S., Gaussier, É., Antoniadis, A., Burlet, M., & Denneulin, Y. (2012). Logistic regression and text classification. In *Textual information access: Statistical models* (pp. 61–84).

Chen, K., Zhang, Z., Long, J., & Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications*, *66*, 245–260.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Debole, F., & Sebastiani, F. (2003). Supervised term weighting for automated text categorization. In *Proceedings of the 2003 ACM symposium on applied computing*. (pp. 784–788).

Deisy, C., Gowri, M., Baskar, S., Kalaiarasi, S., & Ramraj, N. (2010). A novel term weighting scheme midf for text categorization. *Journal of Engineering Science and Technology*, *5*(1), 94–107.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Dogan, T., & Uysal, A. K. (2019). Improved inverse gravity moment term weighting for text classification. *Expert Systems with Applications*, *130*, 45–59.

Fan, R., Chang, K., Hsieh, C., Wang, X., & Lin, C. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Garcia, S., & Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Gong, W., & Cai, Z. (2013). Parameter extraction of solar cell models using repaired adaptive differential evolution. *Solar Energy*, *94*, 209–220.

Gong, W., Wang, Y., Cai, Z., & Wang, L. (2020). Finding multiple roots of nonlinear equation systems via a Repulsion-based adaptive differential evolution. *IEEE Transactions on Systems Man and Cybernetics Systems*, *50*(4), 1499–1513.

Han, E., & Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In *Proceedings of the 4th european conference on principles of data mining and knowledge discovery*. (pp. 424–431).

Jiang, L., Cai, Z., Zhang, H., & Wang, D. (2013). Naive Bayes text classifiers: a locally weighted learning approach. *Journal of Experimental and Theoretical Artificial Intelligence*, *25*(2), 273–286.

Jiang, L., Li, C., Wang, S., & Zhang, L. (2016). Deep feature weighting for naive Bayes and its application to text classification. *Engineering Applications of Artificial Intelligence*, *52*, 26–39.

Jiang, L., Qiu, C., & Li, C. (2015). A novel minority cloning technique for cost-sensitive learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *29*(4), 1551004:1–1551004:18.

Jiang, L., Wang, S., Li, C., & Zhang, L. (2016). Structure extended multinomial naive Bayes. *Information Sciences*, *329*, 346–356.

Jiang, L., Zhang, L., Li, C., & Wu, J. (2019). A correlation-based feature weighting filter for naive Bayes. *IEEE Transactions on Knowledge and Data Engineering*, *31*(2), 201–213.

Jiang, L., Zhang, L., Yu, L., & Wang, D. (2019). Class-specific attribute weighted naive Bayes. *Pattern Recognition*, *88*, 321–330.

Krishnapuram, R., Chitrapura, K. P., & Joshi, S. (2003). Classification of text documents based on minimum system entropy. In *Proceedings of the 20th international conference on machine learning*. (pp. 384–391).

Lan, M., Tan, C. L., Su, J., & Lu, Y. (2009). Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(4), 721–735.

Liu, Y., Loh, H. T., & Sun, A. (2009). Imbalanced text classification: A term weighting approach. *Expert Systems with Applications*, *36*(1), 690–701.

Lu, C., Gao, L., & Yi, J. (2018). Grey wolf optimizer with cellular topological structure. *Expert Systems with Applications*, *107*, 89–114.

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization (vol. 752) (no. 1)*. (pp. 41–48).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of workshop at ICLR (vol. 2013)*.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Ren, F., & Sohrab, M. G. (2013). Class-indexing-based term weighting for automatic text classification. *Information Sciences*, *236*, 109–125.

Sabbah, T., Selamat, A., Selamat, M. H., Al-Anzi, F. S., Herrera-Viedma, E., Krejcar, O., & Fujita, H. (2017). Modified frequency-based term weighting schemes for text classification. *Applied Soft Computing*, *58*, 193–206.

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

Salton, G., & McGill, M. (1984). *Introduction to modern information retrieval*. McGraw-Hill Book Company.

Uysal, A. K., & Günal, S. (2012). A novel probabilistic feature selection method for text classification. *Knowledge-Based Systems*, *36*, 226–235.

Wang, T., Cai, Y., Leung, H., Cai, Z., & Min, H. Entropy-based term weighting schemes for text categorization in VSM. In *Proceedings of the 27th IEEE international conference on tools with artificial intelligence*. (pp. 325–332).

Wang, S., Jiang, L., & Li, C. (2015). Adapting naive Bayes tree for text classification. *Knowledge and Information Systems*, *44*(1), 77–89.

Wang, D., & Zhang, H. (2013). Inverse-category-frequency based supervised term weighting schemes for text categorization. *Journal of Information Science and Engineering*, *29*(2), 209–225.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: practical machine learning tools and techniques* (3rd ed.). Morgan Kaufmann, Elsevier.

Yan, X., Yang, K., Hu, C., & Gong, W. (2018). Pollution source positioning in a water supply network based on expensive optimization. *Desalination and Water Treatment*, *110*, 308–318.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the 14th international conference on machine learning*. (pp. 412–420).

Zhang, L., Jiang, L., Li, C., & Kong, G. (2016). Two feature weighting approaches for naive Bayes text classifiers. *Knowledge-Based Systems*, *100*, 137–144.

Zhang, H., Jiang, L., & Yu, L. (2020). Class-specific attribute value weighting for Naive Bayes. *Information Sciences*, *508*, 260–274.

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, *4*(1), 5–31.