Received: 20 July 2018 | Revised: 14 October 2018 | Accepted: 23 December 2018
DOI: 10.1002/cpe.5140

WILEY

SPECIAL ISSUE PAPER

Research on improved text classification method based on combined weighted model

Yongchang Wang | Ligu Zhu

Communication University of China, Beijing, China

Correspondence
Yongchang Wang, Communication University of China, No. 1 Dingfuzhuang East Street, Chaoyang District, Beijing 100024, China.
Email: 583696995@qq.com

Funding information
National Natural Science Foundation of China, Grant/Award Number: 6137006

Summary

Text classification is very important in information retrieval, but the traditional text classification model has many problems, such as the feature dimension disaster, the lack of semantic features, etc. Aiming at the problems, this paper proposes an improved TFIDF model combined with the Word2vec model for weighing word vectors. In view of the inability of the Word2vec model to distinguish the importance of words with the text, TFIDF is further introduced to weighing Word2vec word vectors to achieve a weighted Word2vec classification model. For data preprocessing, we optimized the traditional StringToWordVector algorithm. The main improvement of StringToWordVector is the introduction to a new algorithm of stem extraction. First, this paper gives a simple description of the basic steps and algorithms of traditional text classification, and then, the ideas and steps of the improved StringToWordVector algorithm are proposed. Finally, experimental results using our improved algorithm are tested for four different data sets (WEBO_SINA and three standard UCI data sets). The experimental results show that the improved StringToWordVector algorithm combined with the combined weighted model has higher classification accuracy, recall, and F1 values than the traditional text classification model only using the Word2vec model or using TFIDF. The experimental results are satisfactory.

KEYWORDS
data preprocessing, text classification, TFIDF, Word2vec

1 | INTRODUCTION

1.1 | Research background

Short text classification of a microblog has a great research value of mining users' interest, hot topic discovery, and a personalized recommendation system. Text categorization is a supervised learning process that defines a category for each document in document sets according to a predefined subject. With the arrival of the big data era, the number of electronic documents on the Internet has increased significantly. Text classification has become the key technology of information retrieval and management. Text classification needs to convert text into data format that a computer can process; traditional document representation methods use the information retrieval technology, such as continuous bag-of-words (shortly as CBOW) and term frequency–inverse document frequency (shortly as TFIDF). In the CBOW model, documents are regarded as disordered vocabulary sets, ignoring its grammar and word order. TFIDF is a popular weighting technique in the information retrieval field for measuring the importance of a word with a document or corpus. The vector space model is a popular model for document representation; it achieves better results from dealing with long text classification. However, compared with the long text, the short text has the characteristics of sparsity, real time, and nonstandard. The sparsity of VSM will reduce the accuracy of the short text classification. For the sparse problem of the vector space model, one way is to extend the short text with the aid of an external knowledge base (such as Wikipedia, WordNet, and HowNet). Lin et al used the upper and lower relations of the words of the net to extend the short text[1]; Joachims extended the short text classification with the help of the Wikipedia knowledge base to assist in short text classification[2]; Schapire and Singer used WordNet query to calculate short text similarity[3]; Park et al extracted high-frequency words from the knowledge base of the knowledge network to extend the short text.[4] Gao et al used the Word2vec model in checking and filtering the duplicate short text and provided a very effective method in dealing with the short text.[5] Because

the external knowledge base contains fewer fields and topics and the updating of its vocabulary is slow, it is difficult to apply it to short text categorization. Another way to extend the short text features is using external text, such as the result of a search engine. Kumar et al designed an adaptive filtering algorithm for the particular case of a multi-input–single-output static system affected by noise,[6] which is useful for dealing with text preprocessing. Zhang et al designed a data model based on fuzzy membership functions and a modeling algorithm for facilitating data classification.[7]

Huang et al inputted short texts to search engines and calculated the similarity in short texts according to the results returned.[8] Soleimani and Miller measured text similarity by an unmarked external background corpus.[9] In recent years, research on short text classification based on semantic information has made some progress. Li et al used the LDA model (latent Dirichlet allocation) to extract the topic of the corpus, expand the short text features, and classify the short text of the Web.[10] Müller and Unay used the LDA theme model to classify the text of a microblog.[11] These scholars have made very good achievements in the field of short text classification. Their research has high practical value and research significance in the field of public opinion analysis and automatic text analysis. Their research methods provide us with a good reference. Almost all these methods need to use the commonly used classification technology. We will introduce these technologies in the next section.

1.2 | Related work on text classification

1.2.1 | The process of text classification

The process of text classification generally includes the process of text expression, the selection and training of classifiers, and the evaluation and feedback of classification results.[12] Text expression can be subdivided into text preprocessing, indexing and statistics, feature extraction, and other steps. The common functional modules of the text classification system are as follows.

1) Preprocessing: formatting the original corpus into the same format, so as to facilitate the subsequent unified processing.
2) Index: decomposes the document into basic processing units and reduces the cost of subsequent processing.
3) Statistics: word frequency statistics and item and classification probability.
4) Feature extraction: extract features from the document that reflect the theme of the document.
5) Classifier: training of a classifier.
6) Evaluation: analysis of the test results of classifiers.

1.2.2 | The common methods of text classification

Traditional text classification methods include the following: decision tree, Rocchio, naïve Bayes classifier, SVM, KNN, maximum entropy, etc. An introduction to these methods is as follows.

Decision tree. This is a classification method based on tree structure.[13] The tree's decision-making process needs to start from the root. The data to be measured are compared with the feature node in the decision tree, and the next branch is selected according to the comparison result, until the leaf node gets the final decision result. The most popular algorithms of the decision tree include ID3 and C4.5.

The core idea of the ID3 algorithm is using information gain as the measure of attribute selection and splitting the attributes with the biggest information gain.[14] Information entropy is used to measure the expected value of a random variable. If the uncertainty of information is greater, the entropy is greater, and more situations will arise.

The C4.5 algorithm uses the information gain ratio to select the attribute, and it can prune the tree during the construction of the tree and accept incomplete data and nondiscrete data.[15]

Rocchio. The basic idea of the Rocchio algorithm is to get a new vector by averaging each item of a sample document in a category. The image is called "centroid," and the centroid becomes the most representative vector representation of this category. When there are new documents to judge, compare how similar the new document is to the center of mass, and you can determine that the new document does not belong to this class.

Naïve Bayes classifier. The Bias algorithm focuses on documents belonging to a certain class probability. The probability that a document belongs to a category is a composite expression of the probability that each word of the document belongs to that category. The probability of each word belonging to this category can be roughly estimated by the number of times (word frequency information) the word appears in the training document of this category, thus making the whole calculation process feasible.

SVM. The support vector machine (SVM) method has a solid theoretical foundation. The essence of SVM training is to solve a quadratic programming problem and get the global optimal solution. This makes it have the advantage that other statistical learning techniques cannot compare.

KNN. K nearest neighbors (KNN) takes the nearest K training data according to the distance to the unclassified data by computing the distance between each training data to the unclassified data; the class of unclassified data will belong to the group that has the majority class attribute in the K training data.[16]

Maximum entropy. Maximum entropy (shortly as MaxEnt) is a criterion in probabilistic model learning. Its idea follows: when learning a probabilistic model, the model with the maximum entropy is the best model in all possible models; if the probabilistic model needs to satisfy

some constraints, then the maximum entropy principle is to select the maximum entropy model from the set of conditions satisfying the known constraints.

These text classification methods require a large number of external corpora; hence, the computational cost increases greatly, and they lack the flexibility in configuring text representation models.[17] Aiming at these problems, this paper proposes an improved TFIDF model combined with the Word2vec model for weighing word vectors. Relative to the traditional word weight model, our model can distinguish the importance of words of the text. In the process of data preprocessing, we optimized the traditional StringToWordVector algorithm by introducing a new algorithm of stem extraction. The research in this paper mainly includes three parts. Section 2 gives theoretical foundations of traditional text classification. Section 3 introduces the ideas and steps of the improved StringToWordVector algorithm. Section 4 selects four different data sets for experiment, designs the tests by integrating our model and our proposed method into the WEKA platform, and gives the analysis of the experiment.

2 | THE THEORETICAL FOUNDATIONS

Text classification methods have been applied to many scenarios after a long period of research, but the problem of short text classification started late, and there has been no general and effective method. The focus of short text classification is on feature processing and classification algorithm. There are two problems: 1) short text provides less words and limited information available; 2) the word frequency or feature matrix constructed according to the result of segmentation is usually very sparse, and most algorithms are not effective against dealing with sparse matrix problems. There are two common methods of short text processing: one is based on some rules to improve the classification process and optimize the improved model; the other is based on external semantic information to expand the amount of short text information, so as to improve the classification effect.

Existing research has proposed a large number of text classification methods, but the algorithms that are suitable for short text classification mainly include naïve Bayes classifiers, KNN, maximum entropy, SVM, decision tree, and some ensemble learning method. Each of these classification methods has a strict theoretical basis; hence, we will introduce the theories of this section.

2.1 | Naïve Bayes classifier

The basic idea of this algorithm is simple; the detail is as follows.

1. Use an *n*-dimensional eigenvector (indicated as X = {x1, x2, … , xn}) to describe the *n* measurements of the *n* attribute samples (A1, A2, … , An), and each data can be represented by the eigenvector.
2. Give *m* classes (namely as C1, C2, … , Cm) and an unknown data X without label.
3. Maximize P(Ci/X); the largest class Ci is called the maximum posterior hypothesis.
4. If the prior probabilities of classes are unknown, it is usually assumed that these classes are equal probabilities. Thus, it only needs to maximize P(X/Ci). Otherwise, we maximize P(X/Ci)P(Ci).
5. Given the class label of a sample, it is assumed that the attribute values are independent of each other; the P(X1/Ci), P(X2/Ci), … , P(Xn/Ci) can be estimated by training samples.
6. To classify X, we compute P(X/Ci)P(Ci) for each class Ci.

Then, X is assigned to the class Ci that has the largest value of P(X/Ci)P(Ci).

2.2 | KNN

The steps of the algorithm are as follows.

1. Redescripting the training text vectors based on feature item sets.
2. After the arrival of the new text, the vector representation of the new text is determined by word segmentation of the new text feature words.
3. Select the K texts nearest to the new text in the training set and compute the similarity in the two texts.

   There is no good method for determining the K value at present. Hence, we can give an initial value and adjust the K value according to the test result. The initial value of K is set to a number between a few hundreds to several thousands.
4. Compute the weight of each class of the K neighbors of the new text by the following formula:

$$p\left(x,\ C_j\right) = \sum_{\overline{t_i} \in KNN} Sim\left(\overline{x},\ \overline{t_i}\right) y\left(\overline{t_i},\ \overline{C_j}\right), \tag{1}$$

   where $\overline{x}$ is the feature vector of the new text, $Sim\left(\overline{x},\ \overline{t_i}\right)$ means the similarity, and $y\left(\overline{t_i},\ \overline{C_j}\right)$ is a class attribute function (if $\overline{t_i}$ belongs to class C_j, the function value is 1; otherwise, it is 0).
5. Compare the weight of the class, deciding the class attribute of the text according to the largest weight.

2.3 | Maximum entropy

The learning of the maximum entropy model is to solve the maximum entropy under certain constraints. For a given training data set T and eigenfunction f(), the learning of the maximum entropy model is equivalent to the constrained optimization problem. The mathematical form of the problem is

$$\max_{P \in C} \quad H(p) = -\sum_{x,y} \tilde{p}(x) p(y|x) \log p(y|x) \tag{2}$$

$$\text{s.t.} \quad Ep(fi) = E\tilde{p}(fi), \quad i = 1, 2, 3, \ldots, n \tag{3}$$

$$\sum_{y} p(y|x) = 1, \tag{4}$$

where $\tilde{p}(x)$ represents the empirical distribution.

2.4 | SVM

Support vector machine (SVM) is originally proposed for two classes of pattern classification problems with linearly separable conditions. Given a set of training samples T = {(x1, y1), (x2, y2), … , (xn,yn)}, where Xi ∈ Rn, Yi ∈{−1, 1} are class scales, the problem boils down to finding a linear decision function f(x) = <w·x + b > that can separate two types of data points, in which Xi ∈ Rn, b ∈ R, maximizes the interval if the constraint yi [w·x + b] > = ± 1 is satisfied.

The mathematical form of the problem is

$$U(w, N) = \frac{1}{2}(w, 1, w) + C\sum_{i=1}^{l} N \tag{5}$$

$$\text{s.t.} \quad yi((w, 1, xi) + b) \geq 1 - Ni, i = 1, \ldots, l$$
$$Ni \geq 0, i = 1, \ldots, l. \tag{6}$$

SVM can also use the kernel function to solve the problem of "dimension disaster" caused by the algorithm.

2.5 | Decision tree

The steps of the algorithm are as follows.

Algorithm 1. C4.5

Define att as attribute of data set

Input: A data set DS

Output: A tree

1) for all att ∈ DS do
2) Compute splitting criteria on att
3) end for
4) let ba = best attribute
5) Tree = a decision node tested by the root
6) let Isd = sub-data sets of DS tested by ba
7) for all Isd do
8) Trees = C4.5(Isd)
9) Add Trees to the related branch of Tree
10) end for

J48 is the implementation of the C4.5 algorithm in the WEKA platform. It provides the following interfaces to be invoked.

1) Instances object

An Instance represents a table. It can deal with an ARFF file or a CSV file, and the mean variance of a column can be taken through an Instances object; it is mainly a package of a number of rows recorded.

2) Classifier interface

Each classifier in WEKA inherits the interface. This interface provides a "build Classifier" method to import an Instances object for training.

3) J48 class

This is the main class of the classifier that is used to implement the Classifier interface.

4) ClassifierTree interface

It is a node in a tree for maintaining and composing the structure of a tree. The class C45PruneableClassifierTree and the class PruneableClassifierTree are used in J48.

5) ModelSelection interface

The interface is responsible for inferring and selecting the best attributes. The different Instances will be placed in different subsets according to the attribute, and the ClassifierTree interface uses ModelSelection to generate the structure of the tree. The implementation of the interface used in J48 is BinC45ModelSelection and C45ModelSelection, and the latter is to generate a standard C45 tree.

2.6 | Ensemble learning method

The ensemble learning method mainly includes bagging and lifting methods. Bagging is a method based on placing back random samples. The essence of bagging is that each weak classifier only trains to get a part of the knowledge. When new samples arrive, each weak classifier is trained to classify and synthesize the results of multiple classifiers as final classification results. The boosting method is weighted by the learned if classifier, and the final classifier is obtained. This method gradually combines weak rules into strong rules by focusing on the errors of weak rules. It is an error-driven method. AdaBoost can achieve high performance in spam filtering. If different depth decision trees are used as weak learners, the classification accuracy will be improved with the increase in decision tree depth. The training speed is slow.

3 | DESIGN OF THE IMPROVED TEXT CLASSIFICATION METHOD

Text categorization technology needs to convert text into a format that the computer can process, and traditional document representation technology uses information retrieval technology, such as CBOW (continuous bag-of-words) and TFIDF (term frequency–inverse document frequency). In the bag-of-words model, documents are regarded as disordered vocabulary sets ignoring the order for grammar and words. At present, most research is based on the VSM model (vector space model); it has achieved good result in dealing with traditional long text classification problems.

3.1 | Introduction to the combined weighted model

3.1.1 | TF*IDF model

TFIDF is a popular weighting technique in the information retrieval field for measuring the importance of a word of a document or corpus. The model believes that if a word is very important, it should appear many times in the article. Hence, we must count the term frequency (shortly as TF).[5] However, stop words are useless to results and must be filtered out. In statistical language, it is necessary to assign "importance" weights to each word based on word frequency. The most common words are given the smallest weight, giving smaller weights to the more common words and giving larger weights to less common words. We call the weight "inverse document frequency" (shortly as IDF), and its size is inversely proportional to the frequency of a word. Multiplying the value of TF and the value of IDF, we get the TF-IDF value of a word. The higher the importance of a word to an article, the greater its TF-IDF value. Therefore, some words with the top TF*IDF values are the key words with this article.

The details of the algorithm are as follows.

Step 1: Computing TF (term frequency). In view of the length of the article, in order to facilitate the comparison of different articles, the "word frequency" should be standardized as follows:

$$\mathrm{TF} = \mathrm{SN}/\mathrm{AN}, \tag{7}$$

where SN means the occurrence number of a word in a document, and AN means the total number of words.

Step 2: Computing IDF (inverse document frequency). IDF can be calculated by the following formula:

$$\mathrm{IDF} = \log\left(\mathrm{CN}/\mathrm{DN} + 1\right), \tag{8}$$

where CN means the document number in a corpus, and DN means the document number containing the word.

If the frequency of a word appears higher, the denominator is larger, and, hence, the value of IDF is closer to 0. In order to avoid the value denominator being 0 (that is, all documents do not contain the word), the denominator should add 1.

Step 3: Computing TF-IDF. The value of TF-IDF can be calculated by the following formula:

$$\mathrm{TF} - \mathrm{IDF} = \mathrm{TF}^{*}\mathrm{IDF}. \tag{9}$$

3.1.2 | The improved TF*IDF mode

Sometimes, important words may not appear frequently. The words that appear in the front or the words that appear in the back are all considered to have the same importance, which is not correct. It also does not consider the characteristic of the word. When there are two features with different lengths, longer features have a more obvious effect on theme expression than short features; thus, we should give longer weight distribution of larger weight coefficients.

We improved the weight formula of TF-IDF and normalized the formula as follows:

$$W(d) = \text{count}(d)\ s\ ^{*}\ \log(N/n + 0.1)\ ^{*}\ lw, \tag{10}$$

where count(d) is the occurrence number of word d in the article, N is the total number of the text of the whole corpus of the experiment, and n is the occurrence number that the word appears in the background corpus text. S is the size of the feature set, and lw is the length weight of the feature word.

The modified TF-IDF weight is based on the feature length; thus, we indirectly realize the original text extension by expanding the training set through the synonym library.

3.1.3 | The combined weighted model

Word2vec is a tool used by Google to train word vectors in 2013, and it provides a way of using distributed vectors for text representation. The Word2vec model has good advantages in dealing with short text compared with the traditional VSM model. In view of the inability of the Word2vec model to distinguish the importance of words with text, this paper further introduces the TFIDF model to compute the weight of word vectors of Word2vec and proposes a weighted Word2vec model. Finally, we combine the two models of weighted Word2vec and TFIDF and use the merged features to represent the text.

Word2vec is a probabilistic language model based on a neural network proposed by MIKOLOV; it can be used to compute the word vector of words. Compared with the traditional high-dimensional word vector, the dimension of the Word2vec word vector is usually between 100 and 300; it reduces the complexity of the computation and does not cause vector dimension disaster. In addition, the word vector of Word2vec is calculated according to the context in which the word is located; this fully captured the semantic information about the context, and it is easy to compute the similarity of the two words with it.

Word2vec contains two training models: continuous bag-of-words (shortly as CBOW) and Skip_gram.[18] The CBOW model predicts the probability of generating a given word through the context; the mathematical expression of CBOW is as follows:

$$P(Wt \mid \tau(Wt-k, Wt-k+1, \ldots, Wt+k-1, Wt+k)), \tag{11}$$

where Wt is a word in corpus; it predicted the probability of Wt appearances by means of words with the size of K adjacent to Wt. The operator τ represents the addition of word vectors of adjacent words in the context window.

Skip_gram predicts its context by means of the current word. That is to say, Skip_gram predicts the probability of vocabulary in adjacent window K through Wt. The mathematical representation of Skip_gram is as follows:

$$P(Wt-k, Wt-k1, \ldots, Wt+k-1, Wt+k \mid Wt). \tag{12}$$

Figure 1 shows a more direct comparison of CBOW and Skip-gram, and CBOW on the left of Figure 1 gives a word in the text.

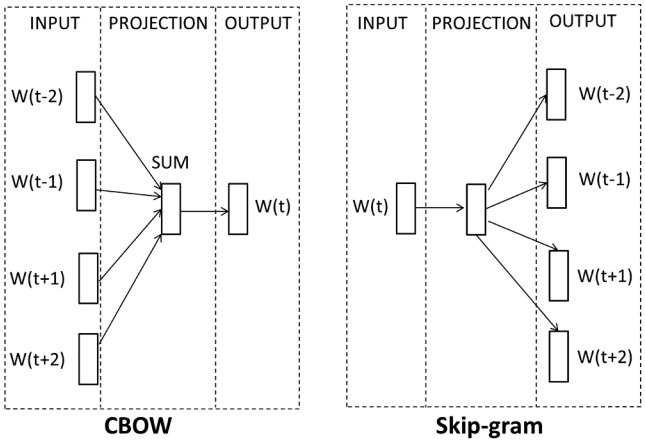

FIGURE 1 Comparison of CBOW and Skip-gram

Skip_gram has higher semantic accuracy than CBOW, but it has higher computational complexity and longer training time. Due to the limitation of the window size, the CBOW model cannot predict the relationship of words outside of the windows. The Skip_gram model constructs phrases by jumping words, avoiding the problem of losing semantic information due to window size constraints. Our experiment uses the Skip_gram model, and the improved TF-IDF model is introduced to calculate the lexical weight based on lexical importance. The weighted vector is added to the new vector of the document Di and is expressed by weight_R(dj), as shown in the following formula:

$$\text{weight\_R(dj)} = \sum \text{word2vec(t)} \times \text{wt}, \text{ where } \text{wt} = \text{tfidft}. \tag{13}$$

Finally, we combine the weighted Word2vec and TFIDF models to get the new document vector, which is indicated by C(di), as shown in the following formula:

$$\text{C}(\text{di}) = \text{concatenate}(\text{tfidf}(\text{di})\,\text{weight\_R}(\text{di})), \tag{14}$$

where tfidf(di) represents the tfidf vector representation of document di.

3.2 | The StringToWordVector algorithm

As we all know, WEKA is noncommercialized, open-source software based on the JAVA environment for data mining and machine learning. The algorithm StringToWordVector is a class in WEKA; it converts a given text document format to the content of the VSM model, which is a necessary module for text classification. The basic steps of the StringToWordVector algorithm include parameter setting, setting data format, processing data, stopwords processing, statistical computing (TF, IDF), normalization, outputting results, and so on, described in Algorithm 2.

Algorithm 2. StringToWordVector

Input: An original document containing only text

Output: File of ARFF format based on vector space model

1) Initialize and set related parameters of executing StringToWordVector class, such as attributeIndices, minTermFreq, outputWordCounts, etc;
2) Compute the index mapped for each word of a string, calculate word frequency, and save it in an array;
3) Select the related algorithms of tokenizier, stemmer, stopwords (the algorithm can also be customized);
4) Calculate or modify the specific TF*IDF formula;
5) Normalize the word frequency or TFIDF, eliminate the influence of different text lengths;
6) Output formatted ARFF documents that the classification algorithm needs

In the StringToWordVector class, the default tokenizer is the class of weka.core.tokenizers. The aim of tokenizing is traversing and dividing the characters for a long string. If you need to cut the sentence as a unit, you can create a new tokenizer by choosing delimiters ". ! \n?" or the ending sentence of Chinese sentences "。! ?".

In language morphology and information retrieval, stem extraction is the process of removing the affix and getting the root of the word.[19] For example, you can transform "cats," "catlike," "catty," and other words into the root "cat."

The popular stemming algorithms include the LookUp algorithm, the Affix algorithm, SnowballStemmer, and so on. The Potter stem algorithm is the most widely used medium-complex, suffix-based stem extraction algorithm; it is also called the Porter Stemmer. The hottest retrieval systems, such as Lucene, Whoosh, and other word stem filters, are all based on the Porter Stemmer algorithm.

The stemmed algorithms inside the StringToWordVector class include LovinsStemmer (written by Lovins in 1968) and SnowballStemmer (written by Martin Porter in 1980); the latter is recommended, but it cannot be used directly as you need to download the package. The built-in stemmer in WEKA is targeted at the English language and can be changed according to your need.

In this paper, we use the Porter2Stemming algorithm.[20]

3.3 | The improved StringToWordVector algorithm

Keeping the framework of the basic StringToWordVector algorithm unchanged, we designed a new StringToWordVector algorithm by introducing a modified vector space model and an improved Porter2Stemming algorithm. We call it the STWV2 algorithm, which is described as follows.

Algorithm 3. STWV2

Input: An original document containing only text

Output: File of ARFF format based on vector space model

1) Initialize and set related parameters of executing StringToWordVector class, such as attributeIndices, minTermFreq, outputWordCounts, etc;
2) Compute the index mapped for each word of a string, calculate word frequency, and save it in an array;
3) Select the customized algorithms of Porter2stemmer;
4) Calculate the modified combinatorial weighted model based on TF*IDF and Word2vec;
5) Normalize the word frequency or TFIDF, eliminate the influence of different text lengths;
6) Output formatted ARFF documents that the classification algorithm needs

4 | EXPERIMENT AND ANALYSIS

The general process of text classification includes data acquisition, data preprocessing (stopwords processing, statistical TFIDF calculation, stem extraction, and so on), text classification, result output, and analysis.[21] The StringToWordVector algorithm is a process of data preprocessing. When the data are preprocessed, we need to introduce a classification algorithm to classify the preprocessed text. In the open WEKA platform, data preprocessing and classification algorithms can be integrated. These classification algorithms are called base classifiers. Here, we respectively used naïve Bayes, KNN, SVM, maximum entropy, bagging, and J48 as the base classifiers to combine with the improved StringToWordVector algorithm and carried out the classification test on some public data sets in the WEKA platform.

4.1 | Experimental environment

The hardware environment of this experiment is a computer with the following specifications: Intel Core i7-4510@2.60GHZ CPU, 8GB memory, and 2 T hard disk capacity. The software environment includes the Windows 10 professional operating system and an open-source software called WEKA. WEKA is a noncommercial, open-source software that is written by JAVA for data mining and machine learning. In our experiment, we used WEKA 3.7.12.

4.2 | Experimental data sets

The data set in our experiment comes from the UCI standard data set and microblog data collected from the Sina microblog (we called it WEBO_SINA). The UCI data are called the News Popularity in Multiple Social Media Platforms Data Set. This is a large data set of news items and their respective social feedback on multiple platforms: Facebook, GooglePlus, and LinkedIn. The collected data relate to a period of eight months, between November 2015 and July 2016, accounting for about 100 000 news items on four different topics: economy, Microsoft, Obama, and Palestine. This data set is tailored for evaluative comparisons in predictive analytics tasks, although also allowing for tasks in other research areas such as topic detection and tracking, sentiment analysis in short text, first story detection, or news recommendation. Our experiments chose four of the data sets, including Facebook_Economy.csv, GooglePlus_Economy.csv, LinkedIn_Economy.csv, and News_Final.csv,.

There are, in total, 110 729 pieces of microblog data in WEBO_SINA, which covers 15 topics. Of the data, 80 000 are used for training, and 30 729 microblog data are used for testing. The 15 themes include IT, finance, animation, real estate, health, education, tourism, food, women, cars, fashion, sports, games, entertainment, and childcare. The number of microblogs under each topic is shown in Table 1.

4.3 | The evaluation measures of the experiment

The performance evaluation of the text classifier is mainly based on the specific experimental results on the corpus. Commonly used evaluation measures of the classification performance include precision, recall, and F-measure values.

1. Precision and recall

Precision and recall are the most widely used measures in text classification. Precision refers to the proportion of samples truly belonging to one class, ie, to the class which the classifier thinks it should belong to; it measures the accuracy of the classification. Recall refers to the ratio of the number of samples that the classifier correctly judges to the total number of samples belonging to this class; it mainly checks the completeness of the classification. Table 2 is a contingency table used for computing precision and recall.

TABLE 1 Sample number of different subjects in WEBO_SINA

Topic	Number
IT	6290
Finance	7355
Animation	6980
Real estate	6450
Health	7659
Education	6738
Tourism	7836
Food	7265
Women	7108
Cars	7653
Fashion	7432
Sports	7596
Games	8536
Entertainment	7822
Childcare	8009

TABLE 2 Contingency table used for computing precision and recall

	The number of texts that truly belong to this class	The number of texts that truly do not belong to this class
The number of texts that are judged to belong to this class	TP	FP
The number of texts that are judged not to belong to this class	FN	TN

Precision and recall are defined in formulas (15) and (16), respectively.

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

Precision and recall are not independent; they are often related to each other. That is to say, in order to get a higher recall, precision will decrease. Thus, many independent evaluation measures cannot really explain the classification performance of a method.

2. F-measure

The F-measure value proposed by Van Rijsbergen takes overall the precision and the recall measure into account to evaluate the classifier. The most commonly used one is the F1 measure; it can be described by the following formula:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{17}$$

The contingency table can only be used to evaluate the classification performance of a single class. If we want to evaluate the overall performance of the classification in a global sense, we need to consider all the categories. At present, two comprehensive indicators are commonly used, namely, micro-average F1 (MicroF1) and macro-average F1 (MacroF1), which are defined by the following formula:

$$MicroF1 = \frac{2 * MicroR * MicroP}{MicroP + MicroR} \tag{18}$$

$$MacroF1 = \frac{2 * MacroR * MacroP}{MacroP + MacroR}, \tag{19}$$

where MicroR, MicroP, MacroR, and MacroP are defined as

$$MicroR = \frac{\sum_{j=1}^{n} a_j}{\sum_{j=1}^{n} a_j + \sum_{j=1}^{n} c_j} \tag{20}$$

$$MicroP = \frac{\sum_{j=1}^{n} a_j}{\sum_{j=1}^{n} a_j + \sum_{j=1}^{n} b_j} \tag{21}$$

$$MacroR = \frac{\sum_{j=1}^{n} r_j}{n} \tag{22}$$

$$MacroP = \frac{\sum_{j=1}^{n} p_j}{n}. \tag{23}$$

Macro-average is the average of classes; hence, it is easy to be affected by small classes, whereas micro-average is considered as a whole, which emphasizes the impact on classification with large classes.

3. ROC curve

The receiver operating characteristic curve (shortly as ROC curve) is also called the sensitivity curve. The reason for this name is that the points on the curve reflect the same feeling; they are all responses to the same signal, but only under two different criteria. It is the curve drawn by the subjects under the specific stimulus conditions for different results obtained by different criteria of judgment. The values of TPR and FPR can be calculated by formulas (24) and (25), respectively, as follows:

$$TPR = \frac{TP}{TP + FN} \tag{24}$$

$$FPR = \frac{FP}{TN + FP}, \tag{25}$$

where TPR means true positive ratio, FRP means false positive ratio, and the definitions of TP, FP, TN, and FN are the same as those shown in Table 2.

The ROC curve can reflect both the sensitivity and the specificity of the continuous variable.[22] The relationship between sensitivity and specificity is revealed by the composition method. Specificity and sensitivity are computed by setting a number of different critical values for a continuous variable. The larger the area of the curve, the higher the diagnostic accuracy. The evaluation method of the ROC curve is different from the traditional evaluation method. It can be allowed to have an intermediate state according to the actual situation and can divide the test results into multiple ordered classifications. Therefore, the ROC curve evaluation method has a wider application.

4.4 | Experimental result and analysis

4.4.1 | Performance comparison and analysis of various classification methods

In the experiment, we chose four data sets that include WEBO_SINA, Facebook_Economy, GooglePlus_Economy, and News_Final; they are all files of standard CSV format. Then, we made classification tests on these data sets by using the traditional algorithm (including naïve Bayes, KNN, maximum entropy, SVM, and bagging) and the improved StringToWordVector algorithm in the WEKA platform. We get six classification performance values of these tests. The six performance values are precision, recall, F1, micro-F1, macro-F1, and roc_area. Tables 3-6 show the test results obtained by the six algorithms on the four data sets, respectively.

Figures 2-7 compare the performance of the six classification algorithms on the data set of WEBO_SINA in the form of a histogram. The comparison of performance on other data sets is similar.

The ROC curve is a comprehensive index that can reflect classification performance[23]; hence, in order to compare the performance of the six algorithms on different data sets, we just give two graphs of the ROC curve, which are based on the data sets of WEBO_SINA and GooglePlus_Economy, respectively, as shown in Figures 8 and 9. The comparison of performance on other data sets is similar.

TABLE 3 Performance values of different classification algorithms on the data set of WEBO_SINA

	Performance Values					
Methods	Precision	Recall	F1	micro-F1	macro-F1	roc_area
Naïve Bayes	0.93	0.94	0.93	0.94	0.92	0.94
KNN	0.92	0.93	0.94	0.94	0.94	0.94
Maximum entropy	0.92	0.94	0.93	0.94	0.92	0.94
SVM	0.94	0.94	0.94	0.94	0.93	0.95
Bagging	0.95	0.95	0.95	0.95	0.94	0.96
Our method	0.96	0.96	0.96	0.96	0.95	0.97

TABLE 4 Performance values of different classification algorithms on the data set of Facebook_Economy

	Performance Values					
Methods	Precision	Recall	F1	micro-F1	macro-F1	roc_area
Naïve Bayes	0.93	0.94	0.93	0.94	0.92	0.94
KNN	0.92	0.94	0.95	0.96	0.94	0.95
Maximum entropy	0.94	0.96	0.95	0.96	0.94	0.94
SVM	0.93	0.93	0.93	0.93	0.93	0.94
Bagging	0.95	0.95	0.95	0.95	0.94	0.96
Our method	0.94	0.96	0.95	0.96	0.95	0.96

TABLE 5 Performance values of different classification algorithms on the data set of GooglePlus_Economy

Methods	Performance Values					
	Precision	Recall	F1	micro-F1	macro-F1	roc_area
Naïve Bayes	0.91	0.92	0.93	0.93	0.92	0.93
KNN	0.92	0.94	0.93	0.94	0.93	0.94
Maximum entropy	0.92	0.94	0.93	0.94	0.92	0.94
SVM	0.93	0.93	0.93	0.93	0.92	0.94
Bagging	0.94	0.94	0.94	0.94	0.94	0.95
Our method	0.86	0.86	0.86	0.86	0.85	0.87

TABLE 6 Performance values of different classification algorithms on the data set of News_Final

Methods	Performance Values					
	Precision	Recall	F1	micro-F1	macro-F1	roc_area
Naïve Bayes	0.93	0.94	0.93	0.94	0.92	0.94
KNN	0.92	0.93	0.92	0.94	0.93	0.93
Maximum entropy	0.93	0.93	0.93	0.93	0.93	0.93
SVM	0.94	0.95	0.95	0.94	0.95	0.95
Bagging	0.95	0.95	0.95	0.95	0.94	0.96
Our method	0.96	0.96	0.96	0.96	0.95	0.97

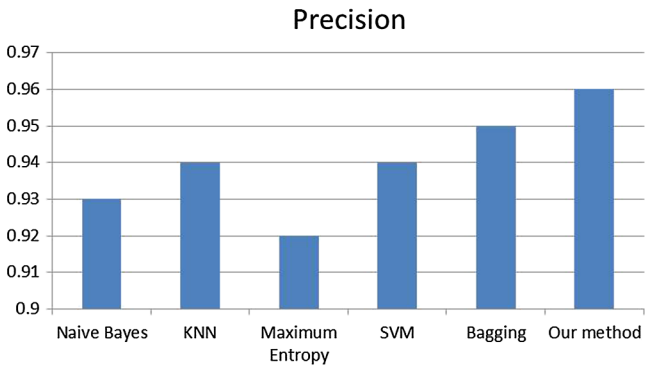

FIGURE 2 Precision of the six algorithms on the data set of WEBO_SINA

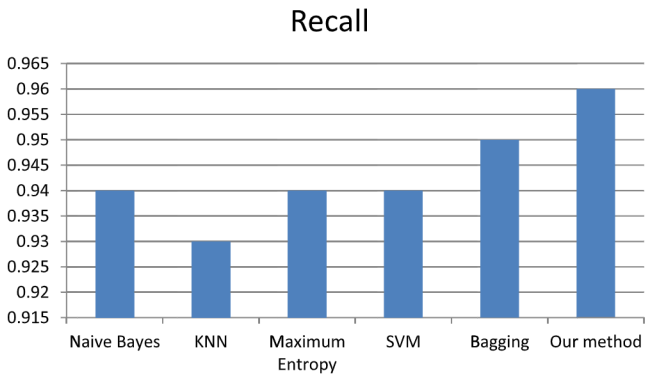

FIGURE 3 Recall of the six algorithms on the data set of WEBO_SINA

Tables 3 and 4 show that the naïve Bayesian, KNN, and maximum entropy methods have the same classification performance on the WEBO_SINA and Facebook_Economy data sets, but slightly different on the GooglePlus_Economy and News_Final data sets, but generally the same. The data in Tables 5 and 6 show that the classification performance of SVM and bagging methods is similar, but compared with Tables 3 and 4, the two methods are superior to the naïve Bayesian method and the maximum entropy method. However, this advantage comes at the expense of computational overhead, because the SVM method and the bagging method are much slower than the naïve Bayesian method and the maximum entropy method from the training model to the results. Tables 3, 4, and 5 show that our classification model is superior to the other

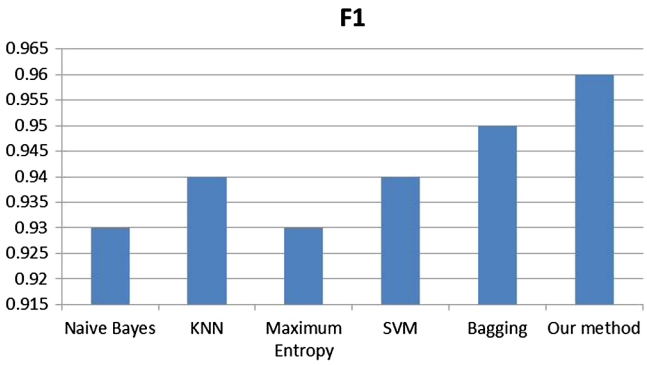

FIGURE 4 F1 of the six algorithms on the data set of WEBO_SINA

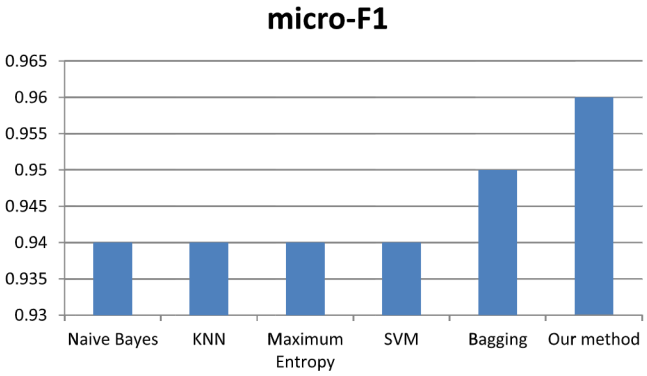

FIGURE 5 Micro-F1 of the six algorithms on the data set of WEBO_SINA

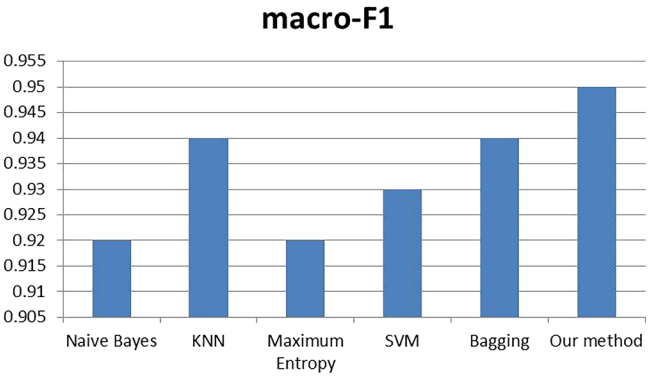

FIGURE 6 Macro-F1 of the six algorithms on the data set of WEBO_SINA

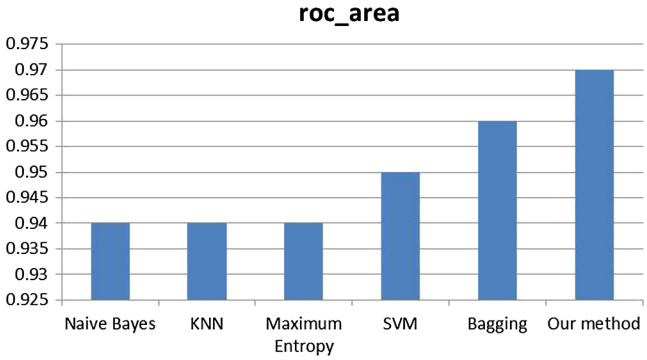

FIGURE 7 The roc_area of the six algorithms on the data set of WEBO_SINA

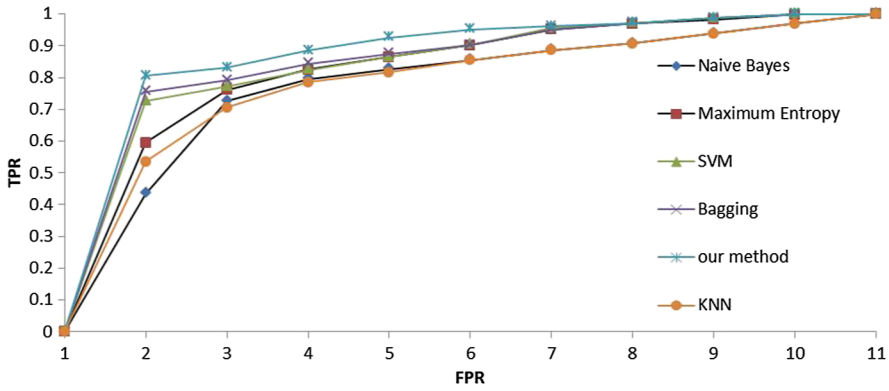

FIGURE 8 The ROC curves of the six algorithms tested on the data set of WEBO_SINA

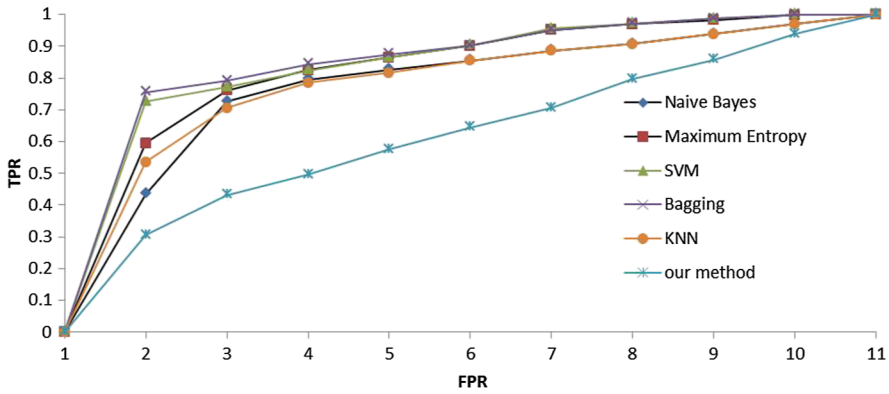

FIGURE 9 The ROC curves of the six algorithms tested on the data set of GooglePlus_Economy

classification methods. At the same time, we adopt a distributed computing model based on MapReduce, which greatly reduces the computational overhead and can, at the same time, provide a more efficient classification performance. Therefore, our algorithm has better performance in the experiments.

4.4.2 | Expandability analysis of the methods

Table 5 and Figure 9 show that our method performs poorly on the data set of GooglePlus_Economy. For this situation, we believe that for the same classification model, different parameter settings may result in performance differences for different data sets. Therefore, we carried out parameter adjustment and optimization on the WEKA platform.

In the classifier option of the WEKA platform, we choose the SMO algorithm, which is an efficient algorithm developed for solving the Lagrange dual problem of the SVM problem. We change the value of epsilon to 2.0, numFold to 2, tolerance parameter to 0.003, kernel function to select RBF kernel, and then restart the training process. The performance results are shown in Table 7.

The results in Table 7 show that our algorithm model has improved in six performance metrics on the GooglePlus_Economy data set after parameter optimization. In order to visually compare the performance differences of these algorithms, we draw ROC curves from the classification results of these algorithms on the GooglePlus_Economy data set through the WEKA platform, as shown in Figure 10.

From the ROC curve given in Figure 10, we can see that our classification algorithm model has the largest area under the ROC curve compared with other traditional algorithms, which shows that it has achieved good results. At the same time, it can be seen that after parameter tuning, the original performance of the data set on GooglePlus_Economy has been greatly improved, which shows that the algorithm model has good adaptability and scalability, and the experimental results are satisfactory.

TABLE 7 Performance of our method after parameter optimization on the data set of GooglePlus_Economy

	Performance Values					
Methods	Precision	Recall	F1	micro-F1	macro-F1	roc_area
Our method	0.95	0.95	0.96	0.95	0.95	0.96

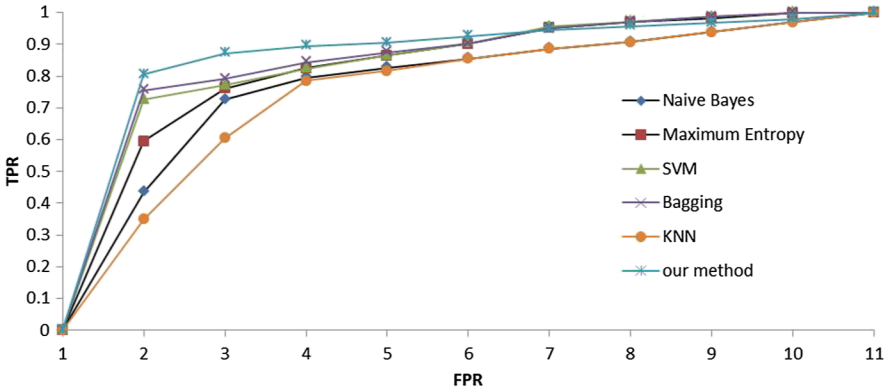

FIGURE 10 The ROC curves on GooglePlus_Economy after parameter tuning

TABLE 8 An example of classification

	Actual Class	
Predictive Class	A	B
A	a	b
B	c	d

4.4.3 | Statistical analysis tests

In order to verify the validity and consistency of our method, we used the KAPPA analysis and its validation method.

The Kappa coefficient is an index to measure the accuracy of classification. It is the result of summing up all categories by multiplying the total number of pixels in all real surface classifications by the sum of the confusion matrix diagonal, subtracting the sum of the total number of pixels and the total number of pixels classified in this category, and then dividing by the square of the total number of pixels minus the total number of pixels and the sum of all categories. Take Table 8 as an example.

The Kappa coefficient can be computed as follows:

$$k = \frac{p_o - p_e}{1 - p_e}, \tag{26}$$

where $p_o = (a + d)/n$, whereas $p_e = (a + b)(a + c) + (c + d)(b + d)/n^2$; n is the total number of samples. Usually, the value of Kappa that is higher than 0.75 means good consistency.

Take the WEBO_SINA data set as an example. The total number of samples is 110 729. The experimental results obtained by our method show that the number of TP is 55 005, TN is 44 995, FN is 5926, and FP is 4803; hence, the Kappa value of our method is 0.774 after computing. The result shows that our method has good consistency on the WEBO_SINA data set, and a similar performance is found on other data sets.

5 | CONCLUSION

In this paper, we have studied microblog short text classification based on the Word2vec model. In view of the fact that the Word2vec model cannot identify the weight of words in the text, in this paper, we have thus introduced an improved TFIDF model to compute the weight of the Word2vec word vector and proposed a weighted Word2vec model. Finally, we combined the weighted Word2vec model and the improved TFIDF for classifying the microblog short text. The experimental results show that the classification accuracy of the merged model is higher than that of the traditional text classification model. In view of the general performance of this method on the GooglePlus_Economy data set, we have optimized the parameters and re-tested; the results show that the performance indicators have been greatly improved, indicating that our method has good adaptability and scalability. The lesser the concurrence words in the text among the class, the higher the classification accuracy of the model. In the next study, we will study how to solve the problem of losing semantic information in short text word vectors by using word vector accumulation. The algorithm based on word vector distance will be considered for short text classification by calculating the distance between Word2vec words in short text.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under grant 6137006.

ORCID

Yongchang Wang https://orcid.org/0000-0003-1444-2558

REFERENCES

1. Lin Y-S, Jiang J-Y, Lee S-J. A similarity measure for text classification and clustering. *IEEE Trans Knowl Data Eng*. 2014;26(7):1575-1590.
2. Joachims T. Text categorization with support vector machines: learning with many relevant features. In: *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21-23, 1998 Proceedings*. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 1998:137-142.
3. Schapire RE, Singer Y. BoosTexter: a boosting-based system for text categorization. *Mach Learn*. 2000;39(2-3):135-168.
4. Park D, Kim S, Lee J, Choo J, Diakopoulos N, Elmqvist N. ConceptVector: Text visual analytics via interactive lexicon building using word embedding. *IEEE Trans Vis Comput Graph*. 2018;24(1):361-370.
5. Gao J, He Y, Zhang X, Xia Y. Duplicate short text detection based on word2vec. Paper presented at: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS); 2017; Beijing, China.
6. Kumar M, Mao Y, Wang Y, Qiu T, Chenggen Y, Zhang W. Fuzzy theoretic approach to signals and systems: static systems. *Inf Sci*. 2017;418-419:668-702.
7. Zhang W, Yang J, Fang Y, Chen H, Mao Y, Kumar M. Analytical fuzzy approach to biological data analysis. *Saudi J Biol Sci*. 2017;24(3):563-573.
8. Huang J, Li G, Huang Q, Wu X. Joint feature selection and classification for multilabel learning. *IEEE Trans Cybern*. 2018;48(3):876-889.
9. Soleimani H, Miller DJ. Semisupervised, multilabel, multi-instance learning for structured data. *Neural Comput*. 2017;29(4):1053-1102.
10. Li X, Rao Y, Xie H, Lau RYK, Yin J, Wang FL. Bootstrapping social emotion classification with semantically rich hybrid neural networks. *IEEE Trans Affect Comput*. 2017;8(4):428-442.
11. Müller H, Unay D. Retrieval from and understanding of large-scale multi-modal medical datasets: a review. *IEEE Trans Multimed*. 2017;19(9):2093-2104.
12. Ranjani J. Bi-level thresholding for binarisation of handwritten and printed documents. *IET Comput Vis*. 2015;9(1):41-50.
13. Fu Z, Wu X, Wang Q, Ren K. Enabling central keyword-based semantic extension search over encrypted outsourced data. *IEEE Trans Inf Forensics Secur*. 2017;12(12):2986-2997.
14. García-Plaza AP, Fresno V, Unanue RM, Zubiaga A. Using fuzzy logic to leverage HTML markup for web page representation. *IEEE Trans Fuzzy Syst*. 2017;25(4):919-933.
15. Whitehead NP, Scherer WT, Smith MC. Use of natural language processing to discover evidence of systems thinking. *IEEE Syst J*. 2017;11(4):2140-2149.
16. Zhang Y, Zhang Y, Jiang Y, Huang G. Multi-feature-based subjective-sentence classification method for Chinese micro-blogs. *Chin J Electron*. 2017;26(6):1111-1117.
17. Audhkhasi K, Rosenberg A, Saon G, et al. Recent progress in deep end-to-end models for spoken language processing. *IBM J Res Dev*. 2017;61(4/5):123-149.
18. Kharazmi MA, Kharazmi MZ. Text coherence new method using word2vec sentence vectors and most likely n-grams. Paper presented at: 2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS); 2017; Shahrood, Iran. https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8306037
19. Ali M, Khalid S, Aslam MH. Pattern based comprehensive Urdu stemmer and short text classification. *IEEE Access*. 2018;6:7374-7389.
20. The English (Porter2) stemming algorithm. http://snowball.tartarus.org/algorithms/english/stemmer.html
21. Wang Y, Li J, Li Y, Wang R, Yang X. Confidence interval for F_1 measure of algorithm performance based on blocked 3×2 cross-validation. *IEEE Trans Knowl Data Eng*. 2015;27(3):651-659.
22. Yu Z, Wang H, Lin X, Wang M. Understanding short texts through semantic enrichment and hashing. *IEEE Trans Knowl Data Eng*. 2016;28(2):566-579.
23. Liu M, Lang B, Gu Z, Zeeshan A. Measuring similarity of academic articles with semantic profile and joint word embedding. *Tsinghua Sci Technol*. 2017;22(6):619-632.

How to cite this article: Wang Y, Zhu L. Research on improved text classification method based on combined weighted model. *Concurrency Computat Pract Exper*. 2020;32:e5140. https://doi.org/10.1002/cpe.5140