## Application of Doc2vec and Stochastic Gradient Descent algorithms for Text Categorization

Zhongyue Hu, Mingliang Liu\*, Ping Yang, Maomao Chen

Electronic Engineering College Heilongjiang University Harbin, China \*Corresponding author: mll\_0608@163.com

Received May,2018; revised August,2018

ABSTRACT. In text categorization, text representation has become an important factor limiting the classification accuracy of classifiers. Ignoring the semantics, grammar, and location information of lexical items in the Vector Space Model (VSM) of a text leads to the loss of considerable feature information that could be useful for text categorization. Therefore, in this study, we propose text classification algorithm combining Doc2vec and Stochastic Gradient Descent algorithms. First, the Doc2vec algorithm is used to train the original corpus to generate the paragraph vectors of the text. Then, for each piece of text, all paragraph vectors are connected as eigenvectors of the text. Finally, the text is classified using the polynomial Naive Bayes and SGD classifier. The experimental results for the 20Newsgroup corpus indicate that our proposed algorithm can classify texts quickly and efficiently with an accuracy of more than 90%. **Keywords:** Doc2vec model; SGD algorithm; Text categorization

1. Introduction. The text categorization(TC)[1] involves the use of computers to automatically classify texts of unknown categories into one or more prior categorization systems based on the content of the texts; thus, it is obvious that that there is a one-tomany mapping of functions between texts and categories. However, in general, a text is classified into a category for which it has obtained the maximum probability from among the different probabilities of belonging to a particular category. Text categorization plays an important role in many applications, such as redundancy filtering, organization management, intelligent retrieval, information filtering, metadata extraction, and ambiguity elimination, among others. There are several key processes in text categorization, including text representation, text similarity computation, and evaluation of classification results. In particular, text representation is not only the basis of text classification, but also the most important process.

Raw text is unstructured data that the computer cannot directly process; thus, in order to ensure that raw text can be processed by the computer, text representation is performed, which primarily involves extracting the metadata representing the essential features of the text in a structured form and transforming the unstructured documents into representations suitable for machine learning algorithms and classification tasks. The text representation model provides a series of algorithms that translate the original text into learning algorithms and data formats required for the classification tasks. Among these text representation models, the Vector Space Model (VSM) model is widely used in the field of information retrieval and text classification. Although this model involves

1338

simple calculations, it suffers from various drawbacks; nevertheless, this model generates a high-dimensional sparse vector, wherein the dimension of the vector is consistent with the total number of terms in the text set. However, the number of entries in a realistic text categorization task is considerably large, which can lead to an unrealistically and unsuitably high-dimensional vector. Furthermore, the VSM assumes that the terms in the text are independent of each other and the frequency of entries, information gain, Chisquare statistic, and other statistics are considered as the basis for feature extraction and selection, while ignoring semantics, grammar, location, and other related characteristics between terms. Therefore, using VSM, it is easy to categorize texts with similar content into different categories; however, it might not be efficient for more complex texts. For example, in a scientific article, the term "Apple" is specified several times. In another text, there are many instances of the terms "iPhone," "iPad," and "Mac". Considering these two texts, a text representation algorithm based on the VSM approach will likely classify the two texts into different categories; in particular, the document containing the term "Apple" will probably be classified under the category of fruit, which is different from the second text. In fact, though both texts convey information about "Apple," and therefore, should have been grouped under the same category, they are not. The primary reason for this discrepancy is that the word "Apple" is a word polysemy, and can represent both fruit as well as Apple Inc., a company based in the United States. In summary, the VSM approach cannot differentiate in such cases with same or similar words with different meanings.

In contrast to the VSM approach, the Latent Semantic Indexing (LSI) [2] algorithm, which is based on the word document co-occurrence matrix, performs singular decomposition and dimensionality reduction, mapping words and documents to potential semantic spaces; the dimensionality of these semantic spaces is smaller than the corresponding feature vector. On the other hand, the Latent Dirichlet Allocation [3](LDA) algorithm builds a three-level Bayesian probabilistic model of words, topics, and documents to mine potential topics in the corpus and cluster relevant keywords. Although LSI and LDA [4] algorithms consider the semantic information of lexical entries, the selection of thematic information in the LSI model considerably affects the results. There are no definite algorithms for the selection of the number of topics, which can only be set manually. Furthermore, the LDA models ignore the relationship between the words themselves. Based on the above discussion, it is clear that these models have some limitations for practical applications.

In 2013, Mikolov et al. proposed an efficient text word vector representation model called Word2vec[5, 6]; This model contains two training algorithms, including Continuous Bag-of-Words (CBOW) and Skip-Gram, to generate the word vector. Furthermore, it should be noted that their model is based on neural networks; however, while calculating the weights between different neurons, the model does not adopt forward feedback or back-feed algorithms in the neural networks. On the contrary, it uses the Huffman tree instead of hidden layers as well as neurons of the output layer; thus, in this model, the leaf nodes of the binary tree and output neurons have similar functions. Moreover, the number of leaf nodes is equal to the counts of terms in the corpus; then, the Huffman tree is constructed based on the word frequencies appearing in the corpus. With Huffman encoding, the code length corresponding to the high frequency words is longer, whereas the code length of the low frequency words is shorter; this ensures the shortest path for the weighted ownership of the tree. Compared with the LSI algorithm, Word2vec preserves the linear relationship between different words; furthermore, it trains faster and retains the memory relationship between different words compared with the LDA algorithm.

However, Word2vec does not adequately represent the semantic features of the text in the case of short texts.

Le and Mikolov proposed the Doc2vec[7] algorithm, which can handle sentences, paragraphs, and documents with different lengths. By constructing paragraph vectors, this algorithm can learn fixed-length features from corpora of different lengths[8]. In addition, the paragraph vectors make word vectors more effective for semantic texts, especially short texts.

Considering this, in this study, we use the Doc2vec algorithm to extract the feature vectors of texts; then, we classify the texts using the Stochastic Gradient Descent (SGD)[10] classifier. Testing our proposed method on the 20Newsgroups datasets, a classification accuracy of over 92% was achieved for each category, which verifies that our text classification model is efficient and feasible.

## 2. Text representation model.

2.1. **Doc2vec model.** The primary tasks in the Doc2vec algorithm involve the learning and generation of paragraph vectors; these paragraph vectors can learn fixed-length features from input sequences with different lengths[9]. Compared with the traditional methods, paragraph vectors do not require a specific word weight function and do not rely on complex parsing tree structures; therefore, their efficiency is relatively high. The Doc2vec algorithm includes two models, namely the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag-of-Words Version of Paragraph Vector (PV-DBOW). PV-DM can be used to predict center words when the paragraph vector and context are given, whereas PV-DBOW can be used to predict the probability of a group of words given only a paragraph vector.

In the PV-DM model, the paragraph vector can be used to predict the next word in the text when sufficient contextual information is available; in this model, all the paragraph vectors form a matrix and each paragraph vector corresponds to one column in the matrix D, which is a matrix of all the words in the same text and each word corresponds to a column in the matrix. In general, paragraph vectors are linked to word vectors or used to predict the next word in the text. The framework of the PV-DM model is shown in figure 1. where "baidu," "is," and "internet" are words in the context and "company" is the word to be predicted; all these words correspond to a certain column in the matrix. Further, "paragraph id" is the paragraph vector.



FIGURE 1. Framework of the PV-DM model

In the PV-DBOW model, when the paragraph vector is given, it can be used to predict the probability of occurrence of a group of words in a paragraph. The basic principle of this model is that in the iterative process of the SGA, a text window (the size of the text window is set by the user) is sampled at each iteration and then, one of the words is selected from the text window using a classifier. The framework of the PV-DBOW model is shown in figure 2. where D is the paragraph vector matrix, "Paragraph id" is a paragraph vector and "id" is the index of the column in the paragraph vector D.

Paragraph vectors can be used to preserve the information of a paragraph, such as the subject of the paragraph. In addition, the paragraph vectors are unique and cannot be shared between different paragraphs; however, word vectors between different paragraphs can be shared. Paragraph and word vectors are obtained using a back-propagation algorithm and trained using an SGA classifier. In particular, at each step of the random gradient, a certain length of text is sampled from random paragraphs one at a time, then the gradient error of the network is calculated, and the parameters of the model are updated based on the gradient. The paragraph and word vectors can be obtained after the sampling process is stopped when the error gradient reaches the user-set threshold.



FIGURE 2. Framework of the PV-DBOW model.

2.2. Text representation and feature extraction. In this study, the 20Newsgroups news corpus is selected as an experimental sample; this corpus is an international standard dataset for text categorization, text mining, and information retrieval. It contains a sample set for testing and training. The training corpus contains 20 categories with a total of 11,033 texts, while the test corpus also consists of 20 text categories, but with a total of 7,532 texts.

In the corpus, the content of text consists of "from," "subject," "ines," "text body," and some optional fields. The "from," "subject," and "lines" fields identify the source, theme, and length of the text, respectively. The "from" and "lines" fields have no effect on the topic expressed in the text. Therefore, only the contents of the "subject" and "text body" fields are retained for the preprocessing of the text.

In general, the punctuation, special symbols, and high-frequency words contained in the original corpus convey only little information regarding the text category; on the contrary, high-frequency words may cause some interference for rare words and may even increase the dimension of the feature vector. Therefore, in this study, we remove the punctuation and special symbols from the original corpus using regular expressions; furthermore, we remove the high-frequency words appearing in the text using stop words. In addition, in order to express certain tenses, grammatical structures, and parts-of-speech information in English, there are several forms of certain words, such as the prototype of a verb, its past tense, and its past participle, which are considered in this study. From the perspective of text categorization, there is no considerable difference among the categories alluded to by the different forms of the same word. Furthermore, we used the Porter Stemmer

algorithm in our study; this algorithm was first introduced in 1979 by Dr. Martin Porter. The Porter Stemmer algorithm unifies the different forms of a verb into a verb prototype, nouns are unified into the singular form, and the adjective and its adverb are unified into the adjective form. After stemming and word shape reduction, the number of words in the text is further reduced; this reduces the time required by the feature extraction algorithm.

After the processing of the original corpus, the noise information contained in the text is eliminated. Next, we use the Doc2vec algorithm on the training set to obtain the word vector of the text. In this study, we use the Doc2vec model provided by Gensim, which is a library in the Python programming language, to represent the text in a corpus. Gensim initially included a variety of Python scripts that were used in a Czech digital math library, to categorize similar texts; in addition, in the past, the library's functions were used for "potential semantic analysis"; unfortunately, the obtained results were not good. However, with continuous improvement, the efficiency and robustness of the Doc2vec algorithm has been considerably improved. Thus far, Gensim includes scripts for topic modeling, document similarity search on a large-scale corpus, as well as natural language processing and information retrieval functions.

In corpus training, we set the parameters  $min\_count$  to 1 window to 10, size to 200 worker to 2,  $dm\_concat$  to 1, and dm to 1, which is based on the training text used in the PV-DM training model. During the training process, all the words that appear only once in the corpus are ignored, then, the next word is forecasted using 10 contextually adjacent words, connecting all the paragraph vectors of peer text as a vector of the word vector form and the dimension of the feature vector is set to 200. In Gensim, the function parameters of the Doc2vec functions and their significance are listed in Table 1.

Parameter name	Meaning	Value
Dm	Training model 1 when using the PV-DM model 0 when using the PV-DBOW model	Default 1 Set the value to 1
size	Feature dimension	Set the value to 200
window	Maximum distance between the	Set the value to 200
	predicted and text words	
min_count	Minimum reserved word frequency	Set the value to 1
dm_concat	Vector connection mode	Set the value to 1
	1 Connection Mode	
	0 Non-connected Mode	
dm_mean	Algorithm	Set the value to 1
	0 means Summation	
	1 means Average	

TABLE 1. Parameters name	s of Doc2vec function	and their meanings
--------------------------	-----------------------	--------------------

Finally, all the word vectors in each document are connected together as a vector representation of the word.

## 3. Text classification model.

3.1. **Polynomial Naïve Bayes classifier.** The Naïve Bayes algorithm is a classification method proposed by Maron and Kuhns based on the Bayes theorem and assumes that feature conditions are independent; this algorithm is widely used in text classification and

spam filtering. There are two common variants of the Naïve Bayesian algorithm, including the multinomial Naïve Bayes and multivariate Bernoulli models.

The multivariate Bernoulli model is a generation model in which the number of occurrences of entries, the occurrences of entries, and the relevance of entries are ignored, and each text can be represented as  $d = \{d_1, d_2, ..., d_n\}$ , and  $b_i \in \{0, 1\}$ . The selected features include *n* words, and  $b_i$  indicates whether the corresponding word appears in the text; in particular, 0 indicates that the word does not appear in the text, whereas 1 indicates that it does. The decision rule of the model is given by the equations(1).

$$c_{\max} = \underset{c \in C}{\arg\max} \hat{p}(c) \prod_{i \in V} \hat{p}(b_i | c)$$
(1)

where  $c_{\text{max}}$  is the document d most likely to belong to the category.

The polynomial Naïve Bayes model is also a generation model. The probability of a document belonging to a certain category can be obtained using equations(2).

$$p(c|d) = k * p(c) \prod_{1 \le i \le C_d} p(t_i|c)$$

$$\tag{2}$$

where p(c) is a priori probability of document d belonging to category c. Document d can be represented as  $d = \{t_1, t_2, ..., t_{C_d}\}$ , where  $t_i$  represents a term in d. Furthermore,  $C_d$  is a total number of terms in the d; k is a scale factor;  $p(t_i|c)$  represents the probability of a word  $t_i$  occurring under a certain category; p(c) and  $p(t_i|c)$  can be obtained from the corpus. Thus, the Naïve Bayes algorithm divides the document into the largest probability category, that is, the maximum posterior probability is used to decide the category of the text. This can be mathematically represented through equations(3) and (4) as follows:

$$C_{map} = \underset{c \in C}{\arg\max} p(c|d) = \underset{c \in C_i}{\arg\max} \hat{p}(c) \prod_{1 \le i \le C_d} p(t_i|c)$$
(3)

$$c_{map} = \underset{c \in C}{\operatorname{arg\,max}} p(c|d) = \underset{c \in C}{\operatorname{arg\,max}} [\log \widehat{p(c)} + \sum_{1 \le i \le C_d} \log p(\widehat{t_i}|c)]$$
(4)

Where  $\arg \max_{c \in C} p(c|d)$  indicates that document d belongs to the maximum probability category, which is usually the estimated maximum posteriori probability using the modified equations (3), equations (2) floating-point may occur under the limit overflow.

In contrast to the two models discussed above, the polynomial model considers not only whether the terms that appear in the text, but also the terms' frequency. Furthermore, considering classification performance, the classification accuracy of the polynomial model is better than that of the multivariate Bernoulli model when multiple features are involved in classification. In addition, polynomial models are more robust against noise. Therefore, taking into account the characteristics of text, in this study, we select the polynomial model as the text classifier.

Sklearn is a Python module that implements the classical machine learning algorithms and incorporates numpy, scipy, and matplotlib third-party modules for scientific computing. In particular, the Sklearn module provides classification, regression analysis, clustering, dimension reduction, model selection, feature extraction, and a series of other common functions. We use the Multinomial function in the naive\_bayes module under Sklearn to construct a polynomial naïve Bayesian classifier; this function has three key parameters, including *alpha*, *fit\_prior*, and *class\_prior*. Here, *alpha* indicates whether the training sample needs to be smoothened; in our study, we set it to 1.0 to obtain a smooth Laplace or lidstone. Further, for *fit\_prior* and *class\_prior*, we use the model default values,

1342

i.e., the probability of learning a class and the class's priority probability are automatically adjusted based on the data.

3.2. Support Vector Machine algorithms. Support Vector Machine (SVM) [11] algorithms typically include two types of classification models. The basic model involves defining the largest linear classifier in the feature space, which can be transformed into a nonlinear classifier through the choice of kernel function. The learning strategy of SVM is to maximize the interval, and the problem to be solved is one of convex quadratic programming. The commonly used SVM model can be divided into linear SVM in the linearly separable case, linear SVM, and non-linear SVM. When the features of the training samples are approximately linearly separable, a linear classifier is obtained by soft margin maximization. In contrast, when the characteristics of the training samples are linearly separable, a linear algorithm is obtained using the hard margin maximization algorithm classifier. Lastly, when the training data is linearly inseparable, a nonlinear SVM classifier is obtained through kernel trick and soft interval maximization.

Assuming the training dataset in the feature space is:

$$t = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$
(5)

where  $x_i$  is the eigenvector of the  $i_{th}$  sample,  $y_i$  is the category label of the  $i_{th}$  sample, and for the dichotomous question  $y_i \in \{0, 1\}, \{x_i, y_i\}$  represents a sample in the training set. The primary goal of the algorithm is to obtain a hyperplane using training intensive learning, which can correctly classify most of the training samples into the corresponding categories. The separated hyperplane satisfies the constraint  $\omega x + b = 0$ , where  $\omega$  is the normal vector of the hyperplane. Figure 4. illustrates a simple dichotomous problem, where  $h_1$  is a separation hyperplane. It is evident that, for the linearly separable dichotomous problem, we can find numerous discrete hyperplanes.





The corresponding classification decision function for the test sample is:

$$f(x) = sign(\omega \cdot x + b) \tag{6}$$

where f(x) is decision function; when its value is 1, the test sample is classified as "positive class", whereas when its value is -1, the sample is classified as "negative class". For linear separable problems, there are numerous hyperplanes that can correctly separate the two types of problems. The linear SVM can obtain the optimal separation hyperplane using interval maximization; this obtained plane is unique. Point *b* in equations(6) represents a sample point. The distance from this point to the hyperplane  $h_1$  is the length of the line ||ab|| (geometric interval), which can be calculated using equations (7). Thus, the geometric interval of the training set is defined as the minimum value of the geometric interval values of all the samples.

$$\gamma_i = y_i \left( \frac{\omega}{||\omega||} \cdot x_i + \frac{b}{||w||} \right)$$
  
$$\gamma = \min_{i=1,2,\dots,n} \gamma_i$$
(7)

where  $||\omega||$  is the L2 norm of  $\omega$ .



FIGURE 4. Geometric interval of samples

The largest interval is the separation hyperplane obtained in training. The hyperplane not only separates the positive and negative cases in the sample, but also correctly classifies the points closest to the hyperplane into the corresponding categories. The hyperplane with the largest geometric interval satisfies the following constraints:

$$\max_{\omega,b} \gamma$$

$$\gamma_i(\frac{\omega}{||\omega||} \cdot x_i + \frac{b}{||\omega||}) \ge \gamma$$
(8)

Equations (8) shows that the geometric interval between each training sample in the training set and hyperplane is at least  $\gamma$  and it is a convex quadratic programming problem. Figure 5. shows the maximum interval of the classified hyperplane;  $h_1$  and  $h_2$  are the boundaries of the interval, while  $h_3$  is the optimal classification interval hyperplane, and  $||h_1h_2||$  is the interval with a value of 2/||w||.



FIGURE 5. Maximum interval classification hyperplane

1344

When the training dataset is linearly inseparable, it contains some specific points that cannot satisfy the constraint that the interval of the function be greater than or equal to one. In order to solve this problem, a relaxation variable  $\xi_i \geq 0$  is introduced for each sample point  $(x_i, y_i)$ , and the constraint condition is defined as follows:

$$y_i(\omega \cdot x_i + b) \ge 1 - \xi_i \tag{9}$$

Accordingly, the cost function is modified to:

$$\frac{1}{2}||\omega||^2 + c\sum_{i=1}^n \xi_i \tag{10}$$

where c is the penalty factor and  $c \ge 0$ ; in particular, a larger value for c denotes increased penalty for misclassification, and vice versa. Thus, by adjusting c, the interval can be maximized and consequently, the number of misclassification points is minimized. For nonlinear problems, a non-linear transformation is typically adopted to transform the hypersurface model in input space  $\mathbb{R}^n$  into a hyperplane model in feature space H. Let  $\chi$ be an input space and H be a feature space; then, there is a mapping from  $\chi$  to H:

$$\phi(x): \chi \to \mathbf{H} \tag{11}$$

For all  $x, z \in \chi$  the function  $K(\chi, z)$  satisfies the following condition:

$$k(\chi, z) = \phi(\chi) \cdot \phi(z) \tag{12}$$

where  $k(\chi, z)$  is the kernel function,  $\phi(\chi)$  is the mapping function, and  $\phi(\chi) \cdot \phi(z)$  represents the inner product of  $\phi(x)$  and  $\phi(z)$ . Commonly used kernel functions include the polynomial kernel function, Gaussian kernel function, and string kernel function. The string kernel function is widely used in text classification, information retrieval, bioinformatics, and other fields, and its corresponding mapping relationship is as shown in equations (13). Assuming that S is a collection of no less than n strings in length, and s is an element in S; in addition,  $H_m$  is the feature space. Then, the mapping relationship between the character s and the feature space can be obtained by calculating  $\phi_n(s)$ 

$$[\phi_n(s)]_u = \sum_{i:s(i)=u} \lambda^{l(i)}$$
(13)

where  $\lambda$  is a attenuation parameter and  $0 < \lambda \leq 1$ , l(i) represents the length of the string i, and i: s(i) = u represents the same substring as u in s.

3.3. **SGD algorithm.** The so-called gradient involves calculating the partial derivative of the multivariate function; from the geometrical perspective, the direction of the function increases the fastest along the gradient direction. On the contrary, the opposite direction along the gradient is the direction in which the function decreases the fastest. Therefore, in machine learning algorithms, the minimized loss function is typically solved iteratively using the gradient descent algorithm.

For the linear regression problem, a sample is denoted as  $y_i = (x_1, x_2, ..., x_n)$ , assuming that the function and loss function are defined as follows:

$$h_{\theta}(x_1, x_2, ..., x_n) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$$
  
$$J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m} \sum_{i=0}^m (h_{\theta}(x_0, x_1, ..., x_n) - y_i)^2$$
(14)

where  $\theta_i$  is the parameter of the model and  $x_i$  is the *n*th eigenvalue of each sample. The steps of the gradient descent algorithm are as follows:

1) Determine the gradient of the current loss function. For each  $\theta_i$ , the gradient is:

$$\frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, ..., \theta_n), \tag{15}$$

2) Determine the distance of descent:

$$\alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{m} \sum_{j=0}^m \left( h_\theta(x_0^j, x_1^j, \dots, x_n^j) - y_j \right) x_i^j, \tag{16}$$

- 3) Determine whether all  $\theta_i$  are less than  $\varepsilon$ ; if true, the algorithm is terminated. Otherwise, Step 4 is executed.
- 4) Update all  $\theta$ , update the expression for  $\theta_i$  as follows, and return to Step 1 after the update.

$$\theta_i = \theta_i - \alpha \frac{1}{m} \sum_{j=0}^m \left( h_\theta(x_0^j, x_1^j, ..., x_n^j) - y_j \right) x_i^j, \tag{17}$$

In practical applications, because the ranges of different feature values of samples are different, the iterations in the algorithm might be considerably slow. In order to reduce the influence of feature values, we can normalize them, after obtaining their expectation  $\bar{x}$  and standard deviation sd(x) Then, the final sample is normalized using the following formula:

$$x' = \frac{x - \bar{x}}{sd(x)} \tag{18}$$

Compared with the common gradient descent algorithm, the SGD [12, 13, 14] algorithm only calculates the gradient of one sample at a time. Thus, the corresponding update formula is modified as follows:

$$\theta_i = \theta_i - \alpha (h_\theta(x_0^j, x_1^j, ..., x_n^j) - y_j) x_i^j$$
(19)

Further, in comparison, the random gradient algorithm uses one sample to iterate at a time, and thus, the training speed is considerably fast. In particular, when the number of training samples is large, the training speed of the sample can be accelerated using random gradient iteration. Because the SGD algorithm only iterates one sample at a time, the convergence rate is slow, and it is possible to obtain the local optimal value instead of the global optimal value. The mini-batch gradient descent algorithm can not only guarantee a faster training speed, but also accelerates the convergence speed of the model. The primary principle of the mini-batch gradient descent algorithm is that for the training set with n samples, m samples are randomly selected for iteration, and the corresponding update formula is as follows:

$$\theta_i = \theta_i - \alpha \sum_{j=t}^{t+m-1} (h_\theta(x_0^j, x_1^j, ..., x_n^j) - y_j) x_i^j$$
(20)

In this study, we select the SGD classifier class included in Sklearn to construct the text classifier; this class implements a simple SGD algorithm and allows small batch gradient learning<sup>14</sup>. The classifier supports different loss<sup>15</sup> and penalty functions<sup>16</sup>, which improves the performance of the classification model. The parameters of the SGD classifier function and their meanings are listed in the Table 2.

Other parameters of the SGD classifier function, such as *learning\_rate*, *max\_iter*, *alpha*, etc., are set to their default values; these default values can lead to a better classification result.

1346

Parameter Name	Parameter Meaning	Value of Parameters
Los	Optional parameters for the loss: "hing," "log," "modified_huber,"	Loss is set to 'log'
Penalty	"regression," and so on Type of penalty function: "none," "l2," "elasticnet," and so on	Penalty is set to 'l1'

TABLE 2. SGD classifier parameters names and meaning

4. Experiments and Results. The experimental corpus for the text uses the 20Newsgroups dataset as the sample. 20Newsgroups is an international standard dataset for text classification, text retrieval, and information retrieval research. This dataset contains 20 text categories; further, the training set contains 11,033 texts, while the test set contains 7,532 texts. Table 3 lists the information of some of the categories in the experimental dataset.

Category	Training Set Samples	Test Set Samples
alt.atheism	480	319
comp.graphics	584	389
rec.autos	594	396
misc.forsale	585	390
sci.space	593	394

TABLE 3. Experimental corpus information for some categories

In the experiment, the contents of a specific field are selected from the original corpus, and then the contents of the filter are preprocessed (including removal of stop words and restoration of the word steam). Then, the preprocessed text is used as the input parameter for the Doc2vec model, and the word vector of the text is trained. Then, all the word vectors are connected together to obtain the feature of the text. Finally, the feature is used as the input parameter of the SGD classifier to train and generate the corresponding text classification model, which is used to classify the test text.

In order to verify the performance of our classifier, the recall rate, accuracy rate and F1-score are used as evaluation parameters of the classification model; these are defined as follows:

$$r = \frac{tp}{tp + fn} \tag{21}$$

$$a = \frac{tp+tn}{tp+tn+fp+fn} \tag{22}$$

$$F_1 = \frac{tp}{(2*tp+fn)} \tag{23}$$

where tp indicates that the positive samples are still positive samples after classification; tn indicates that the negative samples are still negative samples after classification; fp indicates that the negative samples are wrongly classified as positive samples; and fn indicates that the positive samples are wrongly classified as negative samples. Some of the test results are listed in Table 3.

Based on the experimental results, the probability of correct classification of the text for each category is over 90%, with a mean value of 0.92. In addition, the mean values for recall rate and F1-score are 0.86 and 0.89, respectively. Thus, it can be predicted that the classification accuracy of classifiers will be further improved with the increase in the



FIGURE 6. Model evaluation parameters

Category	Precision	Recall	F1-score
misc.forsale	0.93	0.69	0.79
rec.motorcycles	0.93	0.93	0.93
rec.sport.baseball	0.92	0.93	0.91
sci.med	0.91	0.9	0.82
talk.politics.misc	0.92	0.74	0.91
comp.graphics	0.91	0.89	0.94
rec.sport.hockey	0.90	0.96	0.92
avg	0.92	0.86	0.89

TABLE 4. Partial test results

number of training texts in each category. Therefore, the classification algorithm based on Doc2vec and SGD proposed in this study is effective for text classification.

5. Conclusion. In this study, we use the text vector generated using the Doc2vec model to express the characteristics of the text; then, an SVM classifier (training algorithm using SGD) was used to train the text classification model. Compared with the traditional feature extraction algorithms, the word vector generated using the Doc2vec model contains rich semantic, grammatical, and positional information. Therefore, the word vector is more representative of the characteristic information of the text. The training algorithm of classifier adopts the mini-batch gradient descent algorithm; because the algorithm only selects some samples for gradient update at each iteration, it not only accelerates the convergence speed of the classification model, but also improves the classification accuracy of the model. Our test results on the 20Newsgroup corpus show that our proposed method is efficient.

## REFERENCES

 R. H. W. Pinheiro, G. D.C. Cavalcanti, I. R. Tsang, "Combining dissimilarity spaces for text categorization" J. Information Sciences, vol.37, no. 407, pp.87-101, 2017.

- [2] R. M.Silva, T. C. Alberto, T. A.Almeida, Akebo Yamakami "Towards filtering undesired short text messages using an online learning approach with semantic indexing" J. Expert Systems with Applications, vol.83, no. 26, pp314-325, 2017.
- [3] M. Pavlinek, V. Podgorelec, "Text classification method based on self-training and LDA topic models" J. Expert Systems with Applications, vol.80, no. 8, pp83-93, 2015.
- [4] M. Hajjem, Chiraz Latiri, "Combining IR and LDA Topic Modeling for Filtering Microblogs" J. Procedia Computer Science, vol.112, no. 77, pp761-770, 2017.
- [5] T. Mikolov, K. Chen, G. Corrado, et al. "Efficient estimation of word representations in vector space," J. Computer Science, arXiv:1301.3781v3, 2013.
- [6] Q. V. Le, T. Mikolov "Distributed Representation of Sentences and Document," J.Computer Science, vol.2, no. 4, pp1188-1196, 2014.
- [7] B. Pan, Q. C. Zhang, C. C. Yu, S. Cao, "Application of Doc2vec on job salary prediction," J. Application Research of Computer, vol.35, no. 1, pp155-157, 2018.
- [8] Y. Sha, Z. Shi, R. Li, Q. Liang, B. Wang. "Resolving Entity Morphs based on Character-Word Embedding" J. Procedia Computer Science, vol.108, no. 6, pp45-57, 2017.
- M. Kamkarhaghighi, M. Makrehchi. "Content Tree Word Embedding for document representation" J. Expert Systems With Applications, vol.90, no. 19, pp241-249, 2014.
- [10] L. Wang, Y. Yang, R. Min, S. Chakradhar. "Accelerating deep neural network training with inconsistent stochastic gradient descent" J. Neural Networks, vol.93, no. 24, pp219-229, 2017.
- [11] Y. Aytar, A. Zisserman, "Part level transfer regularization for enhancing exemplar SVMs" J. Computer Vision and Image Understanding, vol.138, no. 12, pp114-123, 2015.
- [12] Y. Ming, Yawei Zhao, Chengkun Wu, Kuan Li, "Distributed and asynchronous Stochastic Gradient Descent with variance reduction" J. Neurocomputing, vol.281, no. 4, pp420-431, 2017.
- [13] K. Sopyła, Paweł Drozda. "Stochastic Gradient Descent with Barzilai Borwein update step for SVM" J. Information Sciences, vol.316, no. 15, pp218-223, 2015.
- [14] S. Amari. "Backpropagation and stochastic gradient descent method" J. Neurocomputing, vol.5, no. 4, pp185-196, 1993.
- [15] X. M. Dong, Di-Rong Chen. "Learning rates of gradient descent algorithm for classification" J. Journal of Computational and Applied Mathematics, vol.224, no.19, pp182-192, 2009.
- [16] T. Huang, B. C. Li, D. Shen, J. Cao, B. Mao. "Analysis of the grain loss in harvest based on logistic regression" J. Procedia Computer Science, vol.112, no. 45, pp698-705, 2017.