Received January 23, 2019, accepted February 7, 2019, date of publication February 28, 2019, date of current version March 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900462

Automatic Classification Method for Software Vulnerability Based on Deep Neural Network

GUOYAN HUANG[1], YAZHOU LI[1], QIAN WANG[1], JIADONG REN[1], YONGQIANG CHENG[2], AND XIAOLIN ZHAO[3]

[1]Computer Virtual Technology and System Integration Laboratory of Hebei Province, College of Information Science and Engineering, Yanshan University, Qinhuangdao 066000, China
[2]Computer Science, University of Hull, Hull HU6 7RX, U.K.
[3]Beijing Key Laboratory of Software Security Engineering Technology, School of computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Qian Wang (wangqianysu@163.com)

This work was supported in part by the National Key R&D Program of China under Grant 2016YFB0800700, in part by the National Natural Science Foundation of China under Grant 61472341, Grant 61772449, Grant 61572420, Grant 61807028, and Grant 61802332, and in part by the Natural Science Foundation of Hebei Province China under Grant F2016203330.

ABSTRACT Software vulnerabilities are the root causes of various security risks. Once a vulnerability is exploited by malicious attacks, it will greatly compromise the safety of the system, and may even cause catastrophic losses. Hence automatic classification methods are desirable to effectively manage the vulnerability in software, improve the security performance of the system, and reduce the risk of the system being attacked and damaged. In this paper, a new automatic vulnerability classification model (TFI-DNN) has been proposed. The model is built upon term frequency-inverse document frequency (TF-IDF), information gain (IG), and deep neural network (DNN): the TF-IDF is used to calculate the frequency and weight of each word from vulnerability description; the IG is used for feature selection to obtain an optimal set of feature word, and; the DNN neural network model is used to construct an automatic vulnerability classifier to achieve effective vulnerability classification. The National Vulnerability Database of the United States has been used to validate the effectiveness of the proposed model. Compared to SVM, Naive Bayes, and KNN, the TFI-DNN model has achieved better performance in multi-dimensional evaluation indexes including accuracy, recall rate, precision, and F1-score.

INDEX TERMS Deep neural network, information gain, software security, vulnerability classification.

I. INTRODUCTION

With the rapid development of information technology, the impacts brought to industries by application of the Internet and computers are twofold. They bring convenience but also huge risks and hidden dangers. Recently, with the improvement of the digitalization level of various industries, information security issues have become increasingly prominent. Vulnerabilities are defined as software and hardware defects of the system being illegally exploitable by unauthorized personnel. Once the vulnerability of information system is exploited by malicious attack, the security of information system will be at great risk and may cause inestimable consequences. For example, in 2017, hackers exploited Windows system vulnerabilities to expose 100,000 organizations worldwide to Bitcoin ransomware. In the same year, Microsoft released a total of 372 Office vulnerability patches. Hackers can use the office vulnerabilities to conduct Advanced Persistent Threat (APT) attacks, spread botnets, ransomware and so on. In recent years, the number and variety of vulnerabilities have gradually increased, so the management and analysis of software vulnerabilities has become more and more important.

If the vulnerability can be classified and managed effectively, it can not only improve the efficiency of vulnerability recovery and management, but also reduce the risk of system being attacked and damaged, which is vitally important for the security performance of the system. As software

The associate editor coordinating the review of this manuscript and approving it for publication was Aneel Rahim.

security vulnerabilities play an important role in cyber security attacks, more and more researches on vulnerability classification are conducted by relevant security researchers. The earlier vulnerability classification method RISOS [1], which is aimed at the operating system of the computer, mainly divides the operating system vulnerabilities into seven categories from the perspective of attack, and describes how to exploit the vulnerabilities instead of triggering these vulnerabilities' condition. The PA vulnerability classification method in [2] not only studied the vulnerabilities of the operating system, but also classified the vulnerabilities existing in the application. Andy Gray vulnerability classification method [3] proposed a vulnerability classification system consisting of ten categories according to the different analysis needs of the vulnerability. However, as the complexity of vulnerabilities increases, the limitations of traditional artificial vulnerability classification methods become more and more obvious. Therefore, researchers pay more attention to automatic classification of vulnerabilities.

A large number of machine learning methods have been recently reported in the field of text classification [4]. Classifying them by vulnerability description is also a kind of text classification. Therefore, the problem of automatic classification of vulnerabilities can also be solved using machine learning methods. The SVM classification method based on LDA model is applied in the domain of vulnerability classification by Shua *et al.* [5]. The SVM based on the topic model can make full use of the number of distributed vulnerabilities for classification. The experiment results indicated that SVM has achieved good results in vulnerability classification. Wijayasekara *et al.* [6] tested the Naïve Bayes classifier by using textual information from the error description. The analysis illustrates the feasibility of the Naïve Bayes classifier to classify textual information based on the vulnerability description. Sarang *et al.* [7] proposed a classification method for classifying CVE entries that could not provide enough information into vulnerability categories using the Naïve Bayes classifier. Gawron *et al.* [8] applied Naive Bayes algorithm and simplify artificial neural network algorithm to vulnerability classification, and made comparison on the same data set. The experimental results showed that the artificial neural network algorithm was superior to Naive Bayes algorithm in vulnerability classification.

Although these machine learning classification algorithms have achieved promising results in many fields, due to the large amount of vulnerability data and short description, the generated word vector space presents the characteristics of high dimension and sparse. These machine learning algorithms are not very effective in dealing with high and sparse problems. At the same time, they ignore specific vulnerability information and the classification accuracy is not high. However, in recent years, deep learning has been applied in many fields and has achieved success, such as the field of speech and image recognition [9], [10], the error rate in speech recognition is reduced by 20% - 30% [11], and the error rate in the ImageNet evaluation task is reduced by 26% - 15% [12]. Deep learning also has a significant impact in the field of natural language [13], [14]. Jo *et al.* [15] studied the classification problem in the field of natural language, and applied convolutional neural networks (CNN) and recurrent neural networks (RNN) to the field of large-scale text classification and achieved success. Aziguli *et al.* [16] proposed a novel text classifier using DNN model to improve the computational performance of processing large text data with mixed outliers.

Therefore, in order to better deal with the high and sparse word vector space and take advantage of the automatic feature extraction by deep learning, this paper proposes an automatic vulnerability classification model TFI-DNN based on term frequency-reverse document frequency (TF-IDF), information gain (IG) and deep neural network (DNN). In the model, we first use TF-IDF-IG (TFI) algorithm to extract the feature of the description text and reduce the dimension of the generated high-dimensional word vector space, then construct a DNN neural network model based on deep learning. The TFI-DNN model was trained and tested using vulnerability data from the National Vulnerability Database (NVD). The test results show that the automatic vulnerability classification model in this paper effectively improves the performance of vulnerability classification.

The rest of this paper is arranged as follows. Section 2 introduces the definition of relevant algorithms. In section 3, the implementation details of our model are described. Experiment dataset and results are discussed in Section 4 with comparative analysis. Section 5 outlines the conclusions of this paper and potential future research.

II. RELATED DEFINITION

The automatic classification model of vulnerability (TFI-DNN) is constructed in this paper. The relevant definitions are as follows.

A. TF – IDF

TF-IDF is a common weighted technology based on statistical methods [17]. Assume there are a set of files and each file contains a number of words. We define the importance of the word i in file j as follows.

$$\mathrm{tf_{ij}} = \frac{\mathrm{n_{i,j}}}{\sum_{\mathrm{k}} \mathrm{n_{k,j}}} \tag{1}$$

where both i and j are positive integers, $\mathrm{n_{i,j}}$ denotes the frequency of word i in the file j

The formula for IDF is as follows.

$$\mathrm{idf_i} = \log \frac{|\mathrm{F}|}{|\{\mathrm{j} : \mathrm{t_i} \in \mathrm{d_j}|\}} \tag{2}$$

where |F| represents the total number of files in the corpus, $\mathrm{f_j}$ is the $\mathrm{j^{th}}$ file, and $|\{\mathrm{j} : \mathrm{t_i} \in \mathrm{f_j}|$ represents the number of files containing the word $\mathrm{t_i}$.

The formula for TF-IDF is as follows.

$$\mathrm{TF-IDF} = \mathrm{tf_{ij}} * \mathrm{idf_i} \tag{3}$$

TF-IDF is used to evaluate the importance of a word to a document in the document set or in a corpus. The importance of a word increases proportionally with the number of times it appears in the file, but it also decreases inversely with the frequency it appears in the corpus.

B. INFORMATION GAIN (IG)

IG [18] refers to that, if a feature X in class Y is known, the information uncertainty of class Y will decrease, and the reduced uncertainty degree reflects the importance of feature X to class Y. Set the training data set to D, |D| indicates the number of samples in D. Suppose there are K classes $\mathrm{C_k}$, k $= 1, 2, \ldots, \mathrm{K}|\mathrm{C_k}|$ is the number of samples belonging to class $\mathrm{C_k}, \sum_{\mathrm{C_k}=1}^{\mathrm{K}} |\mathrm{C_k}| = |\mathrm{D}|$. If feature A has n different values$\{a_1, a_2, \ldots, a_n\}$, D will be divided into n subsets according to the values in feature A, denoted as $\mathrm{D} = (\mathrm{D_1 D_2}, \ldots, \mathrm{D_n})$, where $|\mathrm{D_i}|$ is the number of samples in $\mathrm{D_i}$, $\sum_{\mathrm{i}=1}^{\mathrm{n}} |\mathrm{D_i}| = |\mathrm{D}|$. The set of samples belonging to class $\mathrm{C_k}$ in $\mathrm{D_i}$ is $\mathrm{D_{ik}}$, $\mathrm{D_{ik}} = \mathrm{D_i} \cap \mathrm{D_k}$, $|\mathrm{D_{ik}}|$ is the number of samples of $\mathrm{D_{ik}}$.

The empirical entropy H (D) of data set D is calculated as follows.

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \tag{4}$$

The empirical conditional entropy H (D|A) of feature A for dataset D is calculated as follows.

$$H(D|A) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \tag{5}$$

The information gain calculation formula for each feature is as follows.

$$g(D, A) = H(D) - H(D|A) \tag{6}$$

According to the feature selection method of the information gain criterion, the information gain of each feature is calculated, and the features with larger information gain value are selected.

C. FORWARD PROPAGATION ALGORITHM

The forward propagation algorithm uses a number of weight coefficient matrices W, the offset vector b and the input value vector x to perform a series of linear operations and activation operations, starting from the input layer to the output layer to get the final output. The calculation formula is as follows.

$$x^l = \sigma\left(z^l\right) = \sigma(W^l x^{l-1} + b^l) \tag{7}$$

where W is the matrix corresponding to the hidden layer and the output layer, $\mathrm{b^l}$ is the offset vector, x is the input value vector, l is the number of layers of the neural network, and $\mathrm{x^l}$ is the output result.

D. ACTIVATION FUNCTION

softmax and tanh are commonly used activation functions in multiple classification problems they can map the output of multiple neurons to the range of [0,1]. The calculation formulas are as follows.

$$\mathrm{softmax}(\mathrm{y_i}) = \frac{\mathrm{e^{yi}}}{\sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{e^{yj}}} \tag{8}$$

$$\tanh(y_i) = \frac{1 - e^{-2y_i}}{1 + e^{-2y_i}} \tag{9}$$

where, $\mathrm{y_i}$ represents output of the neural network.

E. CROSS ENTROPY COST FUNCTION

Cross entropy describes the distance between two probability distributions, i.e. the distance between the actual output and the expected output. The calculation formula is as follows.

$$L = -\frac{1}{n} \sum_{n} |y \ln(a) + (1 - y) \ln(1 - a)| \tag{10}$$

where, y is the desired output, a is the actual output of the neuron and n is the number of categories.

The smaller the value of the cross entropy, the closer the two probability distributions are. It is a widely used loss function in the classification problem.

F. BATCH GRADIENT DESCENT ALGORITHM (BGD)

The gradient descent algorithm is used on all the parameters to make the loss function reach a smaller value, and finally all the parameters of the model with the minimum loss are obtained. The calculation formula is as follows.

$$\theta_i := \theta_i - \alpha \sum_{j=0}^{m} (h_\theta\left(x_0^{(j)}, x_1^{(j)}, \ldots x_n^{(j)}\right) - y_j) x_i^{(j)} \tag{11}$$

where α and θ_{i} represent the learning rate and the ith parameter, θ represents all parameters, m is the sample number, $\mathrm{y_j}$ is the actual category and $\mathrm{h}_\theta\left(\mathrm{x_0^{(j)}}, \mathrm{x_1^{(j)}}, \ldots \mathrm{x_n^{(j)}}\right)$ is the output of sample j on θ. $\mathrm{x_i^{(j)}}$ is ith feature of sample j.

III. THE TFI-DNN ALGORITHM

The vulnerability automatic classification model TFI-DNN is mainly composed of TFI and DNN. The original vulnerability data is preprocessed first, then TFI is used to extract the features of the vulnerability description text and reduce the dimensionality of the generated higher-dimensional word vector space, and then the DNN is constructed to realize the automatic training and classification of the vulnerability. TFI-DNN algorithm is shown in Figure 1.

A. FEATURE SELECTION USING TFI

TFI is mainly used to extract feature word set. The steps are in Algorithm 1.

B. OPTIMIZATIONS USING DNN

DNN consists of one input layer, multiple hidden layers and one output layer, whose input is the feature vector of instance and output is the category of instance. It mainly includes two propagation processes, forward propagation and back propagation. The propagation process is in Algorithm 2.

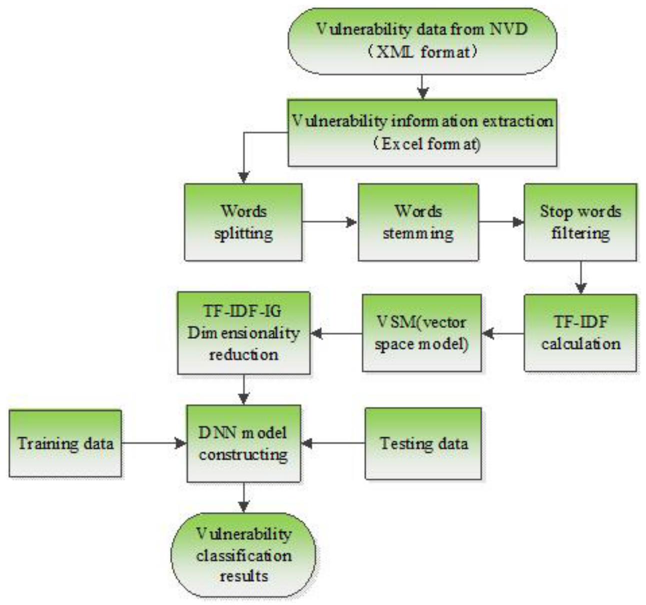
Vulnerability data from NVD (XML format)
Vulnerability information extraction (Excel format)
Words splitting
Words stemming
Stop words filtering
TF-IDF-IG Dimensionality reduction
VSM(vector space model)
TF-IDF calculation
Training data
DNN model constructing
Testing data
Vulnerability classification results

FIGURE 1. Framework of the method.

IV. EXPERIMENTAL ENVIRONMENT AND DATA SET

A. ENVIRONMENT

The experiment was conducted on PC with Intel(R) Core(TM) i5-4460 processor, 3.20 GHz and 8.00 GB memory, running Windows 7 operating system. Programming uses Pycharm 2017 on Anaconda3 version and the TensorFlow framework.

B. DATA SET OF EXPERIMENT

In order to verify the effectiveness, we use the internationally recognized National Vulnerability Database (NVD) [19] as experimental data. The source file of this dataset is a series of XML files, which contains comprehensive information about the vulnerability, such as CVE number, vulnerability release date, CVSS_version, CVSS_score, CVSS_vector, vulnerability text description, and so on. The annual vulnerability amount (2000-2016) Pof NVD vulnerability database is shown in Figure 2, and the total number of vulnerability records is 56,838.

By the end of 2016, the NVD vulnerability database contained 23 vulnerability categories and 1 "unknown" vulnerability category. The number of vulnerabilities records in each category is highly variable. For example, there are 11284 vulnerabilities records in the unknown" category whist there are only 5 vulnerabilities records in the "environment conditions" category. As the "unknown" category of vulnerability has no value for the model training, the vulnerability in "unknown" category is excluded from the experiment. The other 23 vulnerability categories with clearly identified categories are included. The distribution of vulnerability records in each category is shown in Figure 3.

The required vulnerability information are extracted from the XML vulnerability files using program codes written in Python, including CVE number, vulnerability text description, and vulnerability category. All the text information of vulnerabilities from 2000 to 2016 is collected for statistics, 10,000 of them were selected as the training set, and 1,000 of them were used as the test set. Samples of the data are shown in Table 1.

Algorithm 1

```
Input: word list (word_list) formed by word segmentation, lemmatiation, and stop word filtering.
Output: feature word set (feature_words).
1) Traversing each word in the word_list.
2)Word frequency statistics for word_list, stored in the doc_frequency list.
3) Traversing the word frequency list doc_frequency.
4) Calculate the TF value of each word according to (1) and store it in the word_tf dictionary.
5) Calculate the IDF value of each word according to (2) and store it in the word_idf dictionary.
6) Calculate the TF-IDF value of each word according to (3) and store it in the word_tf_idf dictionary.
7) The word set is sorted in descending according to the TF-IDF value.
8) Select the first n words as an important feature set.
9) Save important words in the feature list (features_vocabSet).
10) Traverse features_vocabSet, divide features_vocabSet and store the subset into the subDataSet.
11) Calculate probability of subDataSet.
12) Calculate the empirical conditional entropy of each word according to (4) and (5) and store it in newEntropy.
13) Calculate the IG value of each word according to (6).
14) Save each word and the corresponding IG value in the dictionary.
15) The word set is sorted descending by IG value.
16) Select the first m words as features and store them in the feature_words.
17) Return feature_words.
```

V. EXPERIMENTAL PROCEDURE

A. DATA PREPROCESSING

1) WORD SEGMENTATION

Word segmentation refers to the cutting of coherent vulnerability text information into one word, which transforms the entire vulnerability text information into the smallest semantic unit that can be counted by statistics. This is the first and most important step in the process of vulnerability text preprocessing. For the text of the vulnerability described in English, the word segmentation is very simple. You only need to identify the entire vulnerability description by dividing the space or punctuation between the texts.

For example, the vulnerability text numbered CVE-2016-9990 is described as: "IBM iNotes 8.5 and 9.0 is vulnerable to cross-site scripting. This vulnerability allows

Algorithm 2

```
Input: training set after vectorization.
Output: coefficient matrix W and offset vector b of DNN.
1) Input training data set x, initialize weight coefficient matrix W and offset vector b.
2) Perform a linear calculation for each neuron xi, the calculation formula is $z = \sum_{i=1}^{m} W_i x_i + b$, use tanh for the activation operation, then calculate the final output result layer by layer according to (7). The result output is $a^l = \sigma(z^l) = \sigma(W^l a^{l-1} + b^l)$.
3) Softmax is used to assign probabilities to different objects for the final output, and the output of multiple neurons is mapped to the range of [0,1].
4) The cross entropy is selected as the loss function to measure the loss between the output calculated by the training sample and the actual training sample output.
5) Select the batch gradient descent algorithm as the optimization algorithm, minimize the loss function, set the batch size, iteration threshold and learning rate $\alpha(0 < \alpha \leq 1)$, the DNN model was trained iteratively.
6) Finally, output optimized DNN model parameters W and b.
```

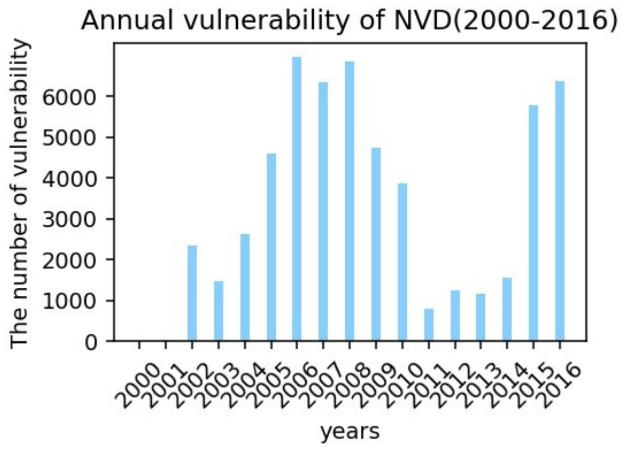

FIGURE 2. Annual vulnerability of NVD (2000-2016).

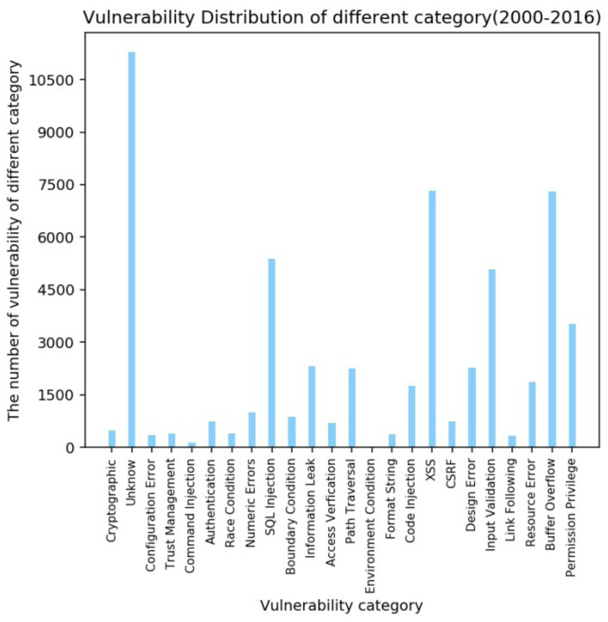

FIGURE 3. Vulnerability distribution of different category.

TABLE 1. Examples of datasets.

CVE_ID	Description	Vulnerability category
CVE-2016-9997	SPIP 3.1.x suffers from a Reflected Cross Site Scripting Vulnerability in involving the ⋯	XSS
CVE-2016-9993	IBM Kenexa LCMS Premier on Cloud 9.0, and 10.0.0 is vulnerable to SQL injection. A remote attacker could send specially-crafted SQL statements, which could allow the attacker to view, add, modify or delete information in the back-end database. IBM Reference #: 1992067.	SQL Injection
⋯	⋯	⋯

users …”. After the word segmentation, you can get the word set as [“IBM”, “iNotes”, “8.5”, “and”, “9.0”, “is”, “vulnerable”, “to”, “cross”, “—”, “site”, “scripting”, “.”, “This”, “vulnerability”, “allows”, “users”,…].

2) LEMMATIZATION

Lemmatization refers to the transformation of non-root form into root form in the word set, and that is the verb in English description changing according to the person into the verb prototype. Convert the plural form of a noun to the singular form, convert gerund form to verb prototype, etc. From the perspective of data mining, these words should belong to the same category of semantically similar words. For example, “attack”, “attacking”, and “attacked” can be divided into the same word, which can be expressed as “attack” in the root form.

Therefore, the set of words in the vulnerability text numbered CVE-2016-9990 is converted by lemmatization: [“IBM”, “iNote”, “8.5”, “and”, “9.0”, “is”, “vulnerable”, “to”, “cross”, “-”, “site”, “scripting”, “.”, “This”, “vulnerability”, “allow”, “user”,…]. It can be seen that the following words in the original description information, “iNotes”, “allows”, “users” are restored to “iNote”, “allow”, “user”, etc.

3) STOP WORD FILTERING

Stop word filtering refers to words that appear frequently in text and contribute little or no contribution to the content or classification of text information, it includes both public stop words and professional stop words. We can download public stop word list from the Internet [20], such as common prepositions, articles, auxiliary words, modal verbs, pronouns, and conjunctions, etc. In addition, some words such as "information", "security", and "vulnerability" are meaningless to the classification of vulnerability so these words should be filtered out. In this paper, each word of the vulnerability text is arranged in descending order by word frequency, and we set a threshold. Based on our experience, we select words higher than the threshold value and meaningless for classification of vulnerabilities as stop word and add them into the stop word list to form a professional stop word list.

B. FEATURE EXTRACTION

The candidate feature set composed of 17,935 words was obtained from all experimental data through the above data preprocessing operation. First, the weight of each word in the vulnerability candidate feature set is calculated by TF-IDF, and the results are arranged in descending order. The number of feature words of the extracted feature subset is 1024. Part of the arrangement is shown in Table 2.

TABLE 2. Feature words based on TF-IDF.

Feature	overflow	SQL	XSS	buffer	traversal	…
TF-IDF	0.0215	0.0206	0.0198	0.0110	0.0109	…

TABLE 3. Feature words based on IG.

Feature	overflow	traversal	SQL	XSS	buffer	…
IG	0.6110	0.5168	0.4572	0.4225	0.3039	…

According to the 1024 words selected in 1), IG is used to calculate the information gain value of each word in the word set, and the results are arranged in descending order. The number of feature words extracted from the feature subset is 900. Part of the arrangement is shown in Table 3.

C. THE WORD VECTOR SPACE ESTABLISHMENT

In the application of this paper, each vulnerability description is expressed as an m-dimensional vector (m is the number of feature words in the feature word set) [21]. After all the vulnerability text samples are vectorized through the m-dimensional space, the vector representation of the vulnerability is established. Each vulnerability sample can be regarded as a point in the high-dimensional space, where the representation of one vulnerability sample v_i is $v_i = t_1, t_2, \ldots, t_n$. Among this formula, m is the number of feature words, which is consistent with the dimension of the vector space of vulnerability.

D. DNN MODEL CONSTRUCTION

After the vectorization representation of vulnerability description, the vulnerability text data described by natural language is transformed into the data structure that can be recognized by the machine and expressed through statistical learning. We use the deep learning framework TensorFlow to

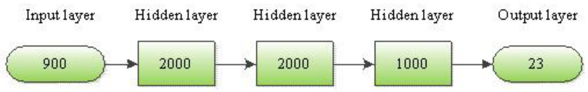

FIGURE 4. The model of DNN.

TABLE 4. Multi-class confusion matrix.

True category	Forecast category		
	Buffer Overflow	Configuration Errors	…
Buffer Overflow	True Positive(TP_i)	False Negative(FP_i)	…
Configuration Errors	False Positive(FN_i)	True Negative(TN_i)	…
…	…	…	…

construct the DNN model. Various configuration parameters of the DNN model were tested, including the number of hidden layers, the number of neurons in each layer, the learning rate of the neural network and the number of iterations. Finally, the parameters are set as empirical values from experiment. The DNN model is shown in Figure 4.

The parameters of the DNN model are set as follows.

1) Initialize the weight of DNN model with normal distribution function with standard deviation of 0.1.
2) The offset values of the input layer and hidden layer are set to 0.1.
3) The forward propagation of the input layer and hidden layer uses *tanh* as the activation function, while the output layer uses *softmax* as the activation function.
4) Use the dropout function in TensorFlow to prevent overfitting.
5) The cross entropy function is used as the loss function to measure the loss between the calculated output of the training sample and the actual output of the training sample.
6) Optimize the DNN model using a batch gradient descent algorithm with a batch size set to 100 and a learning rate set to 0.2.
7) The number of iterations of DNN model training is 20.

Next, the training sample of the NVD data set is used to train the TFI-DNN vulnerability automatic classification model, and then the vulnerability test set is used to evaluate the model performance.

VI. RESULT AND COMPARISONS

A. EVALUATION INDEX

In order to evaluate the performance of automatic classification model (TFI-DNN) in this paper and to compare the performance between different algorithms, unified evaluation criteria are needed. According to the general method of data mining theory, the most widely used classification method for multi-class problem evaluation model is multi-class confusion matrix. As shown in Table 4.

Where i represents a category of a vulnerability. According to the confusion matrix, the accuracy, recall rate, precision rate and F1-score are used as evaluation indexes of TFI-DNN model. The calculation formulas of each index are as follows.

Accuracy(*i*) indicates the ratio of the number of test instances correctly classified by vulnerability *i* to the total number of test instances. The calculation formula is below.

$$\text{Accuracy}(i) = \frac{TP_i + TN_i}{TP_i + FP_i + FN_i + TN_i}$$

Recall(*i*) indicates the ratio of the number of the positive examples correctly classified by vulnerability *i* to the number of actual positive examples. The calculation formula is below.

$$\text{Recall(i)} = \frac{\text{TP}_\text{i}}{\text{TP}_\text{i} + \text{FN}_\text{i}}$$

Precision(*i*) represents the ratio of the number of positive examples correctly classified by vulnerability *i* to the number of instances classified as positive examples. The calculation formula is below.

$$\text{Precision(i)} = \frac{\text{TP}_\text{i}}{\text{TP}_\text{i} + \text{FP}_\text{i}}$$

F1-score(*i*) combines the recall rate and the precision rate. The calculation formula is below.

$$\text{F1} - \text{score(i)} = \frac{2 * \text{Recall(i)} * \text{Precision(i)}}{\text{Recall(i)} + \text{Precision(i)}}$$

The value of F1-score is between 0 and 1, and the higher the value, the better the performance of the vulnerability classification model.

B. COMPARISONS BETWEEM TF-DIF-DNN AND TFI-DNN

In order to evaluate the effect of TFI for feature word selection, we compare it with the method of TF-IDF without IG, the same configured DNN model was iterated 20 times. The evaluation results of the vulnerability classification in the multidimensional evaluation index are shown in Figure 5.

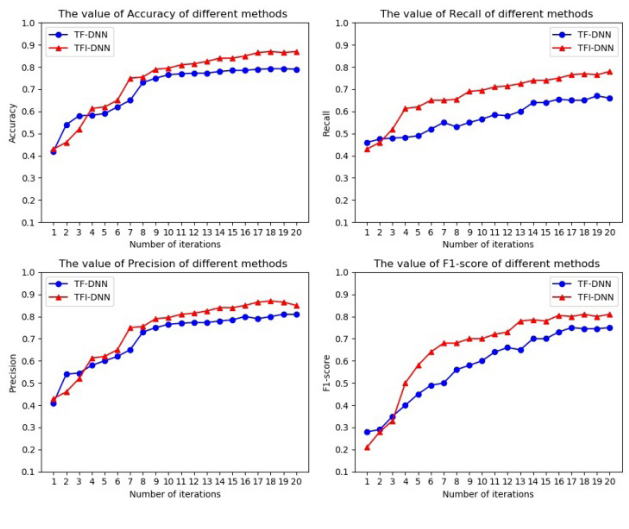

FIGURE 5. The value of evaluation indexes by different methods.

Figure 5 shows that the DNN model based on TFI at the same number of iterations is superior to the DNN model based on TF-IDF in accuracy, recall rate, precision and F1-socre. Because TF-IDF is not comprehensive and it just considers the word frequency to measure the importance of words, and sometimes important words may not appear many times. TFI not only considers the importance of word frequency, but also takes into account the importance of words to the category of vulnerability, so that a better word set can be further extracted according to the IG value, which makes better performance in various evaluation indexes.

C. COMPARISONS AMONG CLASSIFICATION METHORDS

Many machine learning algorithms, such as SVM [22], Naive Bayes(NB) [8], KNN [23] and so on have been widely used in the field of text classification. However, due to the large amount of vulnerability data and short description, the generated word vector space presents the characteristics of high dimension and sparse. But these machine learning algorithms are not very effective in dealing with high and sparse problems. Therefore, this paper proposes a vulnerability automatic classification model TFI-DNN based on deep learning. In order to evaluate the performance of TFI-DNN in vulnerability classification, we compare and evaluate it with the traditional machine learning algorithm based on TFI in terms of accuracy, recall, precision, and F1-score. The experimental results are shown in Table 5.

TABLE 5. Results of different classification methods.

Model	Accuracy	Recall	Precision	F1-score
TFI-SVM	0.68	0.68	0.61	0.63
TFI-NB	0.76	0.78	0.72	0.75
TFI-KNN	0.81	0.78	0.76	0.77
TFI-DNN	**0.87**	**0.82**	**0.85**	**0.81**

According to the experimental comparison results, we can see that the TFI-DNN vulnerability automatic classification model achieved an overall better performance than the other three algorithms. Specifically TFI-DNN is 19%, 11% and 6% higher in classification accuracy, and 18%, 6% and 4% higher in F1-score comprehensive evaluation indexes compared to SVM, Naive Bayes and KNN respectively. The experiment results above have validated the effectiveness of the TFI-DNN model in the automatic classification of vulnerabilities.

VII. CONCLUSION

In order to better analyze and manage vulnerabilities according to their belonging classes, improve the security performance of the system, and reduce the risk of the system being attacked and damaged, this paper applied deep neural network to software vulnerability classification. The analysis of the method and construction process of TFI and DNN are discussed in detail. We compared the vulnerability classification model TFI-DNN to TFI-SVM, TFI-Naïve Bayes and TFI-KNN on the NVD dataset. The results show that the proposed TFI-DNN model outperforms others in accuracy, precision and F1-score and performs well in recall rate. And it is superior to SVM, Naïve Bayes and KNN on comprehensive evaluation indexes. The work in this paper shows the effectiveness of TFI-DNN in vulnerability classification, and provides a basis for our future research using the benchmark vulnerability dataset.

ACKNOWLEDGEMENT

The authors are grateful to valuable comments and suggestions of the reviewers.

REFERENCES

[1] R. P. Abbott, J. S. Chin, J. E. Donnelley, W. L. Konigsford, S. Tokubo, and D. A. Webb, *Security Analysis and Enhancements of Computer Operating Systems*. Washington, DC, USA: US Department of Commerce, 1976.

[2] I. R. Bisbey and D. Hollingworth, *Protection Analysis: Final Report*. Marina Del Rey, CA, USA: Univ. of Southern California, 1978.

[3] A. Gray, "An historical perspective of software vulnerability management," *Inf. Secur. Tech. Rep.*, vol. 8, no. 4, pp. 34–44, 2003.

[4] P. J. Kim, "An analytical study on automatic classification of domestic journal articles based on machine learning," *J. Korean Soc. Inf. Manage.*, vol. 35, no. 2, pp. 37–62, 2018.

[5] B. Shua, H. Li, M. Li, Q. Zhang, and C. Tang, "Automatic classification for vulnerability based on machine learning," in *Proc. IEEE Int. Conf. Inf. Automat. (ICIA)*, Aug. 2013, pp. 312–318.

[6] D. Wijayasekara, M. Manic, and M. McQueen, "Vulnerability identification and classification via text mining bug databases," in *Proc. 40th Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2014, pp. 3612–3618.

[7] S. Na, T. Kim, and H. Kim, "A study on the classification of common vulnerabilities and exposures using Naïve Bayes," in *Proc. Int. Conf. Broadband Wireless Comput., Commun. Appl.* Cham, Switzerland: Springer, 2016, pp. 657–662.

[8] M. Gawron, F. Cheng, and C. Meinel, "Automatic vulnerability classification using machine learning," *Proc. Int. Conf. Risks Secur. Internet Syst.* Cham, Springer, 2017, pp. 3–17.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[10] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.* vol. 115, no. 3, pp. 211–252, 2015.

[11] W. Xiong *et al.*, "Toward human parity in conversational speech recognition," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 12. PP. 2410–2423, Dec. 2017.

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Proc. Syst.*, 2012, pp. 1097–1105.

[13] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature* vol. 529, pp. 484–489, Jan. 2016.

[14] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daum, "Deep unordered composition rivals syntactic methods for text classification," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 1681–1691.

[15] H. Jo, J.-H. Kim, K.-M. Kim, J.-Ho Chang, J.-H. Eom, and B-T. Zhang, "Large-scale text classification with deep neural networks," *Comput. Congnit.*, vol. 23, no. 5, pp. 322–327, 2016.

[16] W. Aziguli *et al.*, "A robust text classifier based on denoising deep neural network in the analysis of big data," *Sci. Program.*, vol. 2017, 2017, pp. 1–10.

[17] H. Zhang, K. Lv, and C. Hu, "An automatic vulnerabilities classification method based on their relevance," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2017, pp. 475–485.

[18] Berrar, Daniel, and W. Dubitzky, *Information Gain*. New York, NY, USA: Springer, 2013.

[19] *National Vulnerability Database*. Accessed: 2018. [Online]. Available: https://nvd.nist.gov/

[20] *Stop Word List*. Accessed: 2018. [Online]. Available: https://pypi.org/project/stop-words/

[21] P. D. Turney and P. Pantel, "From frequency to meaning: vector space models of semantics," AI Access Found., Tech. Rep., 2010.

[22] A. Sun, E.-P. Lim, and Y. Liu, "On strategies for imbalanced text classification using SVM: A comparative study," *Decis. Support Syst.* vol. 48, no. 1, pp. 191–201, Dec. 2009.

[23] K. Shi, L. Li, H. Liu, J. He, N. Zhang, and W. Song, "An improved KNN text classification algorithm based on density," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, Sep. 2017, pp. 113–117.

Authors' photographs and biographies not available at the time of publication.