symmetry

MDPI

*Article*

# Research on WebShell Detection Method Based on Regularized Neighborhood Component Analysis (RNCA)

**Aijun Zhou, Nurbol Luktarhan * and Zhuang Ai**

College of Information Science and Engineering, Xinjiang University, Urumqi 830000, China; zaj@stu.xju.edu.cn (A.Z.); az@stu.xju.edu.cn (Z.A.)
* Correspondence: nurbol@xju.edu.cn

**Abstract:** The variant, encryption, and confusion of WebShell results in problems in the detection method based on feature selection, such as poor detection effect and weak generalization ability. In order to solve this problem, a method of WebShell detection based on regularized neighborhood component analysis (RNCA) is proposed. The RNCA algorithm can effectively reduce the dimension of data while ensuring the accuracy of classification. In this paper, it is innovatively applied to a WebShell detection neighborhood, taking opcode behavior sequence features as the main research object, constructing vocabulary by using opcode sequence features with variable length, and effectively reducing the dimension of WebShell features from the perspective of feature selection. The opcode sequence selected by the algorithm is symmetrical with the source code file, which has great reference value for WebShell classification. On the issue of the single feature, this paper uses the fusion of behavior sequence features and text static features to construct a feature combination with stronger representation ability, which effectively improves the recognition rate of WebShell to a certain extent.

**Keywords:** WebShell; feature selection; RNCA; opcode; text statistical features

**Citation:** Zhou, A.; Luktarhan, N.; Ai, Z. Research on WebShell Detection Method Based on Regularized Neighborhood Component Analysis (RNCA). *Symmetry* **2021**, *13*, 1202. https://doi.org/10.3390/sym13071202

Academic Editor: Basil Papadopoulos

Received: 4 June 2021
Accepted: 28 June 2021
Published: 4 July 2021





## 1. Introduction

The integration, penetration, and driving effect of Internet technology on the development of the entire economy and society is becoming more and more obvious, and the risks and challenges that it brings are also increasing as cyberspace threats and risks are increasing. Abundant network applications expand the scope of network attacks, exposing sensitive information of individuals, companies, and even countries to risks. In 2020, the National Internet Emergency Center (CNCERT) released the "Analysis Report on my country's Internet Network Security Monitoring Data in the First Half of 2020", which pointed out that 35,900 websites in China have been maliciously implanted with backdoors. The number of websites increased by 36.9% from last year [1]. According to the Internet Live Status monitoring data in September 2019 [2], a large number of websites were attacked every day (from 25,000 websites being attacked every day in April 2015 to 61,750 websites every day in September 2019), and nearly 4.43 billion Internet sites were attacked every day. The user has had a direct and significant impact. If an attacker wants to implant advertisements, tamper with pages, and control web services, it can be achieved by implanting various WebShells in the website. Therefore, timely detection of WebShells is very necessary.

The WebShell detection method proposed by people from the perspective of statistical features and grammatical features has been able to identify Java WebShells well, but the improvement of WebShell detection capabilities is also faced with the hidden monster, that is, the PHP WebShell detection problem. Figure 1 shows the percentage of websites with various server-side programming languages calculated by W3Techs. Among them, 79.1% of the websites use PHP for server-side development, ranking first, far surpassing ASP.NET and Ruby, which are ranked second and third. Since most web pages are written in PHP and support many dynamic features, there are more and more ways to obfuscate

the WebShell in the PHP category. Therefore, this article mainly studies WebShell files written in PHP. For the convenience of presentation, it is referred to as WebShell in the text.

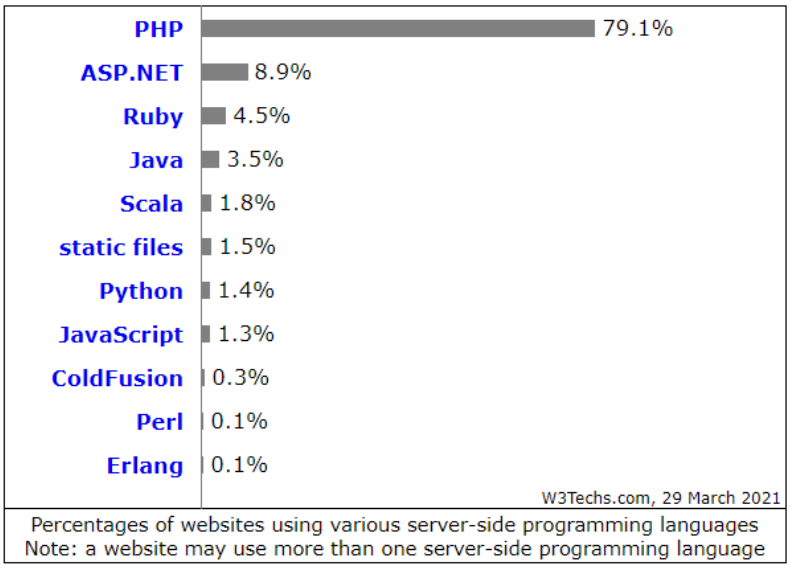



**Figure 1.** Percentage of websites written in server-side languages.

The characteristics of WebShell are very similar to normal Web pages, and they are symmetrical. It is relatively easy to hide them in normal source files and bypass the detection of traditional firewalls and anti-virus software. Various anti-detection feature obfuscation and concealment technologies are gradually applied to WebShell, resulting in poor detection effect of the feature code-based WebShell detection technology, and it is difficult to detect the new WebShell in time. In order to alleviate the impact of encryption, compression, and confusion on WebShell detection, in recent years, researchers have tended to conduct research from the perspective of the WebShell text layer, such as text-based statistical characteristics and text-based semantic characteristics. In order to preserve the completeness of the text information, and effectively use the text context information to identify more types of WebShell, the dimension of characterizing sample features will be very high, which will cause interference to subsequent model detection. Therefore, mining effective information in high-dimensional features becomes the key to further improving the detection performance of WebShell.

Learning distance measurement in feature space can greatly improve the performance of classifiers, which is of great significance in practical applications. Nearest neighbor is a simple and effective nonlinear decision rule, and regularized neighborhood component analysis is used to analyze the high-dimensional data of the pattern recognition problem. By finding a weight vector, the algorithm maximizes the leave-one-out (LOO) classification accuracy of random nearest neighbors in low-dimensional space. Theoretically, the most favorable data features can be found as the input of classification model. This paper innovatively applies RNCA to WebShell feature dimensionality reduction, which can improve the detection performance of classification model.

The main contributions of this paper are as follows:

(1) To study the WebShell written in PHP language, and construct the WebShell source code compiler layer. The opcode sequence features are taken as the research object, and the feature index model suitable for WebShell is constructed.
(2) To solve the problem of incomplete consideration of opcode sequence feature context, this paper proposes to preserve the context information of opcode by using non-fixed length sequence fragments.
(3) A WebShell feature selection method based on regularized neighborhood component analysis (RNCA) is proposed. The method solves two problems. The first one is

to select features from the feature set of the opcode sequence, which is beneficial to classification, and to reduce the dimension of features. Secondly, it solves the problem of limited WebShell detection performance caused by feature simplification. Method feature completeness was increased from the statistical point of view, and the combination of static statistical features and behavioral sequence features was used to judge whether the samples to be detected were WebShell or not.

## 2. Related Work

In recent years, the detection method of WebShell has been greatly improved compared with the traditional method based on feature library. The reason is that the research angle of WebShell is gradually widening, such as text similarity, text statistical characteristics, text semantic features, opcode features, etc.

### *2.1. Research Status of WebShell Detection*

Based on the text similarity detection method, Peter M. Wrench et al. [3] used four different similarity calculation methods to create a representative similarity matrix in 2015. By calculating the similarity between samples and WebShell sets for detection, this method can effectively identify unknown, deformed and encrypted samples, but there are still problems such as high false alarm rate.

It is more likely that detection based on text statistical features can detect confused encrypted WebShell. Luczko P [4] compiled a NeoPi detector by using five statistical features: information entropy, longest word, coincidence index, feature code, and file compression ratio. This detector can detect confused and encrypted WebShell files through comprehensive analysis of the above features; however, this tool is slow to update, the feature library is relatively old, and the possibility of killing a new WebShell is low. Ying Zhou et al. [5] input the statistical features of Neopi tool into Apriori and FP-Growth algorithms to improve the detection accuracy.

Based on semantic feature detection methods, DENGLY et al.'s research [6] on WebShell detection methods mainly focuses on the lexical level of scripting language, judging by whether feature words are used. Zhang Haoyi [7] obtains function call sequences based on an abstract syntax tree through semantic analysis of precompiled script files, and uses a word set model to carry out feature vectorization of behavior characteristics of script files.

WebShell detection based on opcode features is a hot research object in recent years. Yong F [8] discussed the effectiveness of opcode sequences for WebShell detection, and proposed a combined algorithm FRF-WD based on FastText and random forest. Opcode sequences are used as an important feature of the prediction model. Experimental results show that the model provides a low false alarm rate, and the accuracy rate is as high as 99.23%. Hong Cui et al. [9] proposed a PHP-like WebShell detection method based on RF-GBDT (random forest gradient advancing decision tree) model. It is a combination of the random forest classifier and the GBDT classifier. It uses common statistical features of PHP files and text features of opcode sequences. The accuracy of the trained model is 99.169%, and the false alarm rate is 0.682.

### *2.2. Research Status of Nearest Neighbor Component Analysis*

Nearest neighbor is a simple and effective nonlinear decision rule. Nearest neighbor component analysis is a distance measure learning algorithm proposed by Goldberger et al. [10] in 2004, which is used for high-dimensional data analysis of pattern recognition problems. The algorithm maximizes the leave-one-out (LOO) classification accuracy of random nearest neighbors in low-dimensional space by finding a projection matrix. Theoretically, the data features that are most conducive to classification can be found as the input of the classification model. This classification method has been successfully applied to text classification, speech recognition, face recognition [11], and other fields.

Liu Congshan et al. [12] applied the nearest neighbor component analysis algorithm to the field of text classification and proposed a K nearest neighbor component analysis

algorithm K-NCA. H.V. Nguyen et al. [13] proposed a distance measure learning method for face recognition-indirect neighborhood component analysis (INCA). INCA is a combination of one-time similarity learning (OSS) algorithm and proximity component analysis (NCA). Experiments show that it can also achieve better results in very low dimension space. Ferdinando H. et al. [14] used nearest neighbor component analysis to reduce the dimension, which improved the recognition performance of ECG for three types of emotions. Natasha Singh-Miller et al. [15] learned the representation of high-dimensional acoustics in low-dimensional space through nearest neighbor component analysis method, optimized the features closely related to the classification accuracy of nearest neighbor classifiers, and improved the classification accuracy. Rizwan et al. [16] used NCA to reduce the dimension of feature space from 50 to 22 in speech recognition research, and verified the effectiveness of feature space. JIN M et al. [17] used nearest neighbor component analysis algorithm to select features, so that the prediction accuracy of the model can reach 56.25%. Y Zheng [18] proposed a convolutional nonlinear neighborhood component analysis model, which can automatically learn distance metrics and solve classification problems in time series problems. MALAN NS et al. [19] optimized the feature selection algorithm of nearest neighbor component analysis (NCA) by modifying regularization parameters, and applied it to high-dimensional EEG feature selection to improve the classification performance of moving image signals. Zhou H et al. [20] used the idea of an NCA feature extraction and hidden Markov algorithm CHMM for reference, and proposed an NCA-CHMM algorithm to fuse multi-source information, remove redundant features, and solve the problems of bearing fault diagnosis and fault severity classification. D. Wang et al. [21] extended the NCA method to a Bayesian framework for the first time and proposed the Bayesian neighborhood component analysis learning algorithm, which can be applied to model retrieval with uncertainty.

Drawing lessons from the effective application in other fields, this paper innovatively applies nearest neighbor component analysis to the field of WebShell detection, and proposes a WebShell feature selection method based on RNCA. From the point of view of feature selection, the problem of unsatisfactory detection performance caused by WebShell dimension disaster is dealt with. The obtained specific opcode sequence features have great reference value for WebShell recognition and are helpful for website managers to find the back door of web pages in time and quickly. In order to avoid feature singleness, this chapter proposes a combination of statistical features and opcode behavior sequence features to improve the detection accuracy of WebShell.

## 3. WebShell Feature Selection Method Based on RNCA

The flow of the RNCA-based feature selection method is shown in Figure 2, which mainly includes: feature selection and processing (text statistical feature extraction, opcode sequence extraction, opcode feature vectors), RNCA-based feature selection, and classification model prediction.

### *3.1. Feature Selection and Processing*

WebShell detection technology based on opcode opcode sequence can avoid the interference of useless annotations, has certain resistance to some encryption and coding confusion methods, and has certain detection ability to unknown types of WebShell [4]. Therefore, we choose the opcode operation sequence as the signature of WebShell sample. By using PHP's VLD extension, you can easily obtain the opcode sequence of PHP files. This section examines the effectiveness of opcode sequences in identifying WebShells. The construction of WebShell feature index model is mainly divided into two parts: opcode opcode text vectorization and text statistical feature extraction.

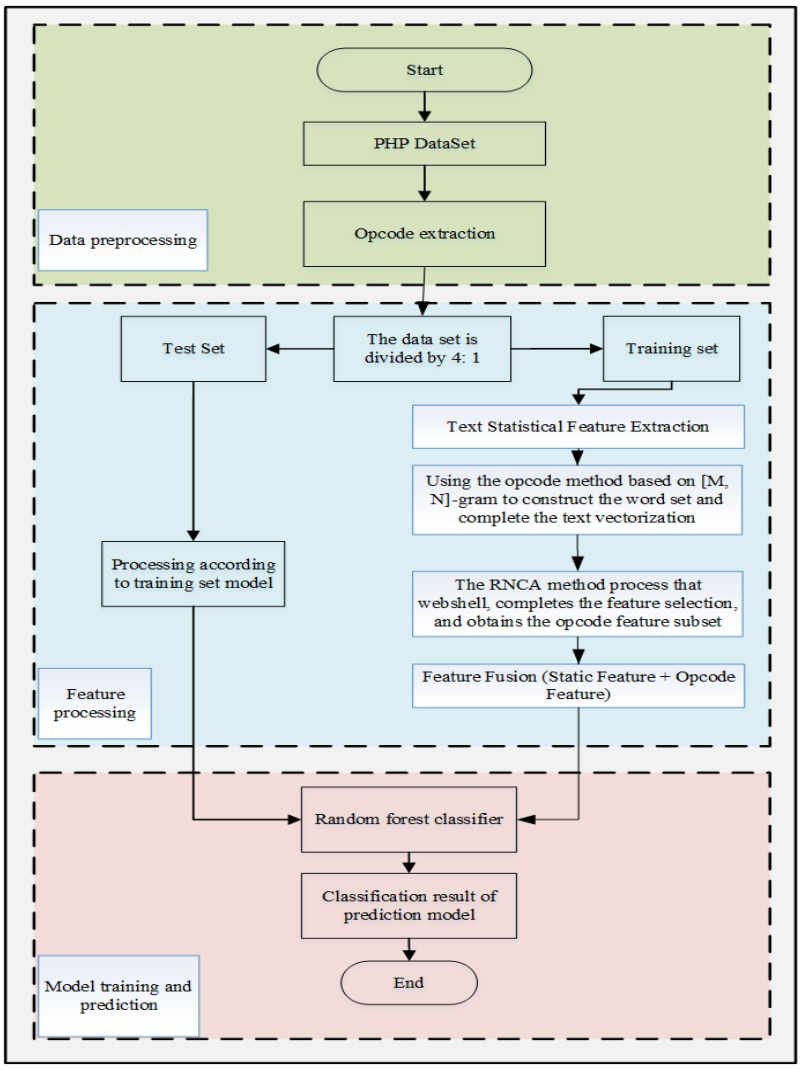



**Figure 2.** Flow Chart of WebShell Detection Based on RNCA.

#### 3.1.1. Text Statistical Characteristics

The static statistical features of the text layer refer to the detection features of NeoPi, and the specific static statistical features are described in Table 1. In the research work based on static feature detection, the longest character word is mainly used to identify coded and confused WebShell; the statistics of malicious characteristics can effectively evaluate the possibility of malicious behavior in script files. In WebShell, most people use the eval() function. When the frequency of using this function is abnormal, it is usually WebShell. The file compression ratio can effectively detect two types of WebShell, one-sentence Trojan horse with small storage occupation and the other-large Trojan horse with large storage occupation, by calculating the size of the file.

**Table 1.** Description of statistical characteristics.

| Static Features | Description |
|---|---|
| Coincidence index | The low coincidence index indicates that the file code is potentially encrypted or mixed; |
| Entropy | The uncertainty of documents is measured by using ASCII code table; |
| Longest Word | The longest string may potentially be encoded or confused; |
| Signature_nasty | Searching for known malicious code string fragments in files (such as eval(), assert(), exec(), shell_exec()); |
| UsesEval | Use the function eval() to account for the proportion; |
| Compression | Compare the compression ratio of the file, the ratio of the uncompressed file size to the compressed file size. |

### 3.1.2. Opcode Vectorization

1. Extract opcode

   Opcode is an abbreviation for opcode. An instruction can contain one or more opcodes. After lexical analysis, syntax analysis, and static compilation, each PHP script finally obtains an opcode sequence. In this paper, plug-in VLD is used to convert source code and opcode. Figure 3 shows the whole process of opcode extraction. First read the data set, save the storage path of the PHP file, VLD commands such as: -d vld. active=1-d vld. execute=0 a. PHP outputs the compilation layer results of the corresponding file. As shown in Figure 4, the compilation result of a Trojan horse is the line number, instruction code, script start tag, end tag, opcode operation code, additional data associated with the opcode (such as the jump target opcode), return value, and instruction parameters. Regular expressions are used to align the column where the opcode is located, and each opcode is saved in sequence. Positive samples and negative samples respectively extract opcode sequences according to the process of Figure 3, and save the results under two files for subsequent data processing. Figure 5 shows the opcode sequence extracted from a PHP file.

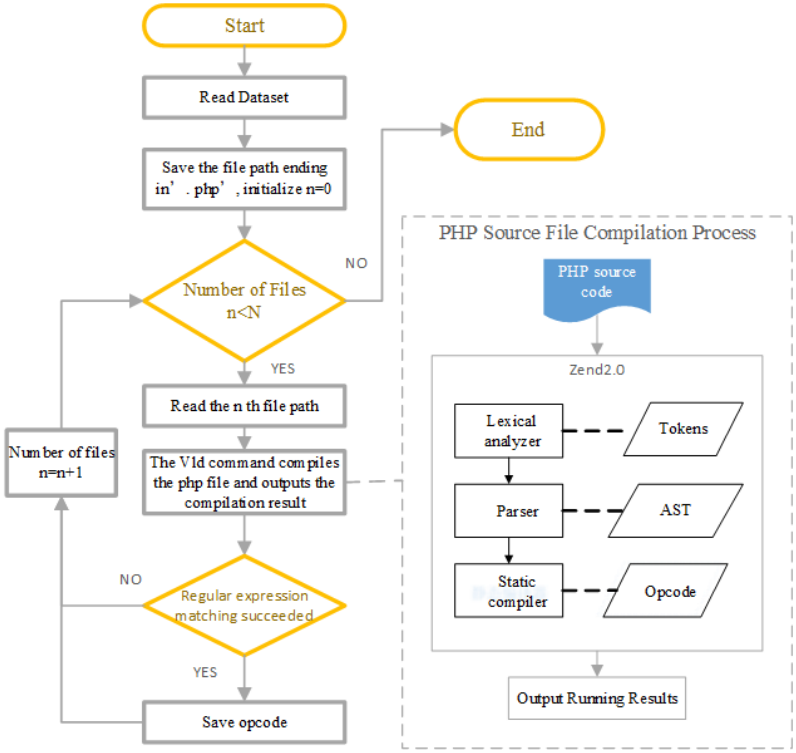



**Figure 3.** Opcode sequence extraction process.

```
<?php
@eval($_POST["value"]);
?>
```

```
line     # *  op                              fetch          ext  return  operands
-------------------------------------------------------------------------------
  2      0  >   BEGIN_SILENCE                                      ~0
         1      FETCH_R                         global           $1      '_POST'
         2      FETCH_DIM_R                                      $2      $1, 'value'
         3      INCLUDE_OR_EVAL                                          $2, EVAL
         4      END_SILENCE                                              ~0
  4      5  >   RETURN                                                   1
```

**Figure 4.** PHP file compilation layer results.

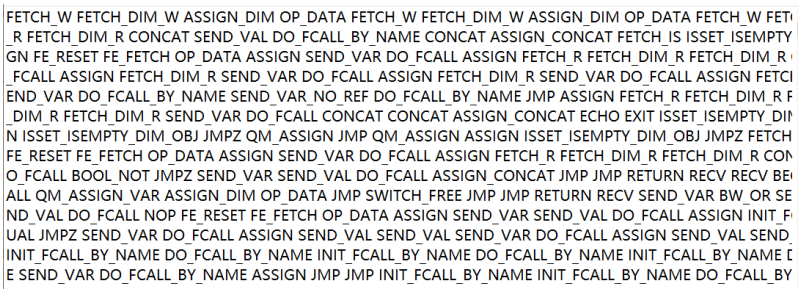



**Figure 5.** Opcode representation of a PHP file.

2. Opcode vectorization
Opcode text is ordered, and the context of an opcode is associated. The traditional N-gram algorithm divides all opcode sequences into fixed-length corpus fragments to form opcode word bags. However, this segmentation method ignores the influence of different opcode sequences on WebShell detection results, so expressing opcode sequences as different lengths for feature selection has a positive impact on detection. In this paper, in order to find out the opcode feature sequence that can effectively identify WebShell, this paper proposes an opcode feature processing method based on [M, N]-gram, which preserves the context environment of samples as much as possible. The algorithm flow is shown in Algorithm 1.

**Algorithm 1:** Opcode representation of a PHP file

**Input:** opcode sequence, sequence length range $M, N$, where $\leq N, N \geq 2$
**Output:** opcode sequence of length $n$
An Opcode Feature Extraction Algorithm Based on $[M, N] -$ gram
**BEGIN**
1.Traversing opcode sequence;
2.Sliding the sequence with the size of n opcodes to generate a sub-sequence with a length of $n$;
3.Save each subsequence feature to a list;
4.Calculate the TF-IDF value for each opcode subsequence;
5.Take TF-IDF as the eigenvalue of the subsequence;
**END**

### *3.2. Feature Selection Method Based on RNCA*

#### 3.2.1. Rnca Algorithm

RNCA, as a feature selection technology, is a feature weighting scheme. Its goal is to learn a feature weight vector by maximizing the classification accuracy of the LOO method [22]. In RNCA, a reference sample point $x_j$ is randomly selected for sample $x_i$ from all PHP samples, and the probability of selection depends on the distance between the two samples. The distance $D_w$ between samples is defined as follows:

$$D_w(x_i, x_j) = \sum_{m=1}^{r} w_m^2 |x_{im} - x_{jm}| \tag{1}$$

$w_m$ denotes the weight of the mth opcode sequence feature. The correlation between probability $p_{ij}$ and weighted distance $D_w$ is established by introducing kernel function $k$, that is, the smaller $D_w$, the larger $k$.

$$P(ref(x_i) = x_j|S) \propto k(D_w(x_i, x_j)) \tag{2}$$

$$k(z) = exp(-\frac{z}{\sigma}) \tag{3}$$

In order to make the sum of all probabilities exactly equal to Equation (1), the probability $p_{ij}$ that $x_j$ is selected as the reference point is defined as

$$p_{ij} = P(ref(x_i) = x_j|S^{-i}) = \frac{k(D_w(x_i, x_j))}{\sum_{t=1,t\neq i}^{n} k(D_w(x_i, x_t))} \quad (4)$$

If $i = j$, then $p_{ij} = 0$. $x_t$ Represents all samples except $x_i$. Parameter $\sigma$ in kernel function $k$ is kernel width, which affects the probability that sample $x_j$ is selected as reference point. For $x_i$, the points of all $S^{-i}$ are calculated, and the average probability that $x_i$ is correctly classified at this time can be expressed as

$$p_i = \sum_{j=1,j\neq i}^{n} P(ref(x_i) = x_j|S^{-i}) I(y_i = y_j) = \sum_{j=1,j\neq i}^{n} p_{ij} y_{ij} \quad (5)$$

where $y_i = y_j$, $y_{ij} = 1$, otherwise $y_{ij} = 0$. Then the average LOO correct rate of the whole classification is the sum of $p_i$ in all samples divided by the total number of samples, i.e., Equation (6), which can be regarded as the objective function that needs to be maximized.

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} p_i = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} p_{ij} y_{ij} \quad (6)$$

In the process of learning weight vectors, the algorithm does not involve any explicit complexity control of vector elements. Therefore, there will be over-learning in the learning process. In order to reduce over-fitting and perform feature selection, a regularization term is further introduced, thus obtaining the following objective function:

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} p_i - \lambda \sum_{m=1}^{r} w_m^2 = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{j=1,j\neq i}^{n} p_{ij} y_{ij} - \lambda \sum_{m=1}^{r} w_m^2 \right) \quad (7)$$

3.2.2. Feature Selection Model Construction

The value of regularization parameters greatly affects the calculation of feature weights. If the value is too large, the weights of all features are close to zero, resulting in all features being irrelevant features. If the value is too small, there will be more redundant features. The value is directly linked to the calculation of minimizing classification loss, so the selection is very important. Selecting regularization parameters according to experience is one of the commonly used methods, but this method has great uncertainty and instability. For feature selection of WebShell, we adopt a learning-based method to adjust regularization parameters. The specific steps of the RNCA-based WebShell feature selection model are as follows:

Step 1: Feature selection is applied to WebShell data set, training set $S = \{(x_i, y_i), i = 1, 2, \ldots N\}$, where $x_i$ represents the ith sample, $N$ is the number of samples in the training set, $y_i$ represents the sample label, $y_i \in \{0, 1\}$. In WebShell data set, $x_i$ consists of 1000 sequence features, while $y_i$ only contains white list and black list.

Step 2: Perform five-fold cross-validation on the training set S (4/5 is used as the training set and 1/5 is used as the verification set), and evaluate the generalization error err. err is defined as Equation (8), $k_i$ represents the predicted sample label, and $t_i$ represents the actual label of the sample. When $k_i$ and $t_i$ are not equal, $I(x)$ returns 1, otherwise $I(x)$ returns 0.

$$err = \frac{1}{n} \sum_{i=1}^{N} I(k_i \neq t_i) \quad (8)$$

Step 2 is used to check whether feature selection is valid, fit RNCA model, and calculate generalization error $\text{err}_{\lambda=0}$ when $\lambda = 0$. If the generalization error after fitting NCA model is $\text{err}_{\lambda\neq 0} < \text{err}_{\lambda=0}$, feature selection is required.

Step 3: Adjust the regularization parameter $\lambda$ to find the minimum classification loss value. In order to find the best $\lambda_{best}$, we generate a evenly distributed array $\lambda_{val}$ with length L, train NCA model with each $\lambda$ in $\lambda_{val}$, and store the generalization error in the array. The parameter adjustment process is shown in Figure 6.

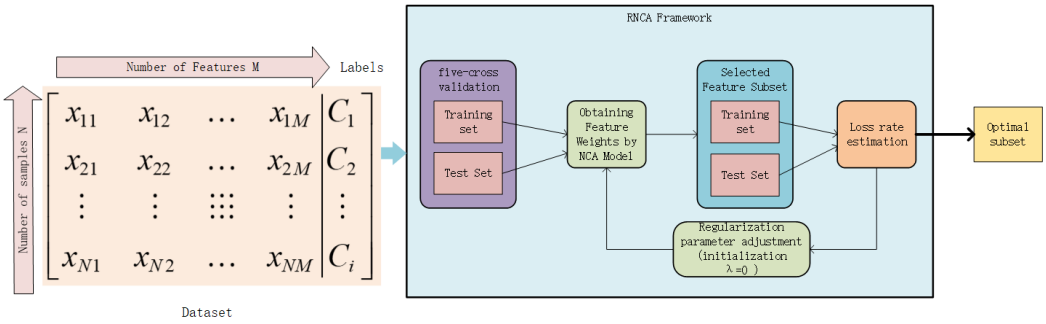



**Figure 6.** RNCA parameter selection.

Step 4: Repeatedly execute Step 2, and calculate the classification loss by five cross-validation for each $\lambda$. At the same time, the average classification loss of each $\lambda$ is recorded. At the same time, the average classification loss of each $\lambda$ is recorded, and then the $\lambda$ corresponding to the smallest average loss is selected as the $\lambda_{best}$ .

Step 5: $\lambda_{best}$ is used to train NCA model on the whole training set, and the weight of each feature is calculated. Finally, the feature whose weight value is greater than 0.05 of the maximum weight value is selected as the final classification feature, and the selected data set is represented as $X_{op}^{d}$, and the feature weight value is represented as $w_d$ [18].

Step 6: The static features of the whole data set are extracted, and the opcode sequence feature $X_{op}^{d}$ and the static feature $X_s$ are combined by feature weighting, that is, $X_{\text{new}} = \left[ w_d X_{op}^{d} X_s \right]$.

Step 7: Use combined features to train random forest classifiers, complete WebShell prediction, and calculate relevant evaluation indexes.

## 4. Experiment and Result Analysis

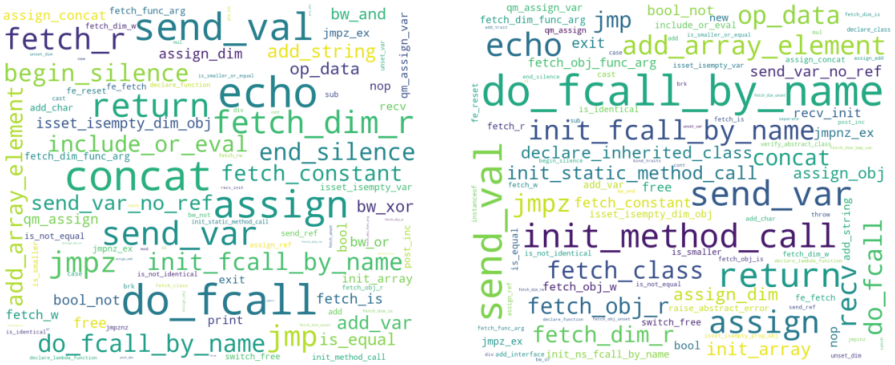
### *4.1. Experimental Environment and Data Processing*

The computer configuration used in this experiment is as follows: Intel (R) Core (TM) i3-3227U CPU @ 1.9 GHz, 8.00 GB memory, the software development environment is Windows 10, MATLAB019a is used to complete the feature selection of RNCA, and Python 3.7 is used to complete the extraction of statistical features and the construction of classification models. Sklearn, a machine learning library in Python 3.7, is mainly used. The data used in the experiment are all from the Github website, including PHP files of open-source projects such as PHPCMS and WordPress, among which, 5394 normal samples and 634 malicious samples were collected. In order to avoid the influence of redundant files on the experiment, the collected PHP files were deduplicated by MD5. The final experimental data set consists of 562 malicious WebShell samples and 5384 normal PHP samples. The source of the experimental data is indicated in Table 2. In the experiment, 80% samples were randomly selected as training sets to generate detection models, and the remaining data were used as test sets to detect the effect of models.

**Table 2.** Dataset distribution.

| Dataset | Download URL |
|---|---|
| WebShell | https://github.com/tennc/WebShell (accessed on 1 March 2019)<br>https://github.com/tanjiti/WebShellSample (accessed on 1 March 2019)<br>https://github.com/JohnTroony/php-WebShells (accessed on 1 March 2019) |
| PHP samples | https://github.com/johnshen/phpcms (accessed on 1 March 2019)<br>https://github.com/WordPress (accessed on 1 March 2019)<br>https://github.com/learnstartup/4tweb (accessed on 1 March 2019)<br>https://github.com/yiisoft/yii2 (accessed on 1 March 2019) |

The compiled bytecode data have obvious characteristics, and the data compiled by WebShell and the data of normal samples show obvious differences in many dimensions. We use word cloud tools to show the overall visibility of the data and analyze it. As shown in Figure 7, it can be seen that the distribution of hot words in the two samples is very different. The positive sample set is distributed in words such as "do_fcall_by_name", "send_var", and "init_method_call", while the WebShell sample set is distributed in "do_fcall", "echo", and "concat". The different distribution proves that the two have different emphases on function.




(**a**) (**b**)

**Figure 7.** Comparison of WebShell (**a**) and normal samples of (**b**) word clouds.

### *4.2. Experimental Design*

#### 4.2.1. Evaluation Criteria

In the experiment, this paper sets the label of WebShell to 1 and the normal sample to 0. In this paper, accuracy, recall rate, and *F*1 value are used to evaluate the effectiveness of feature dimension reduction method.

(a) Accuracy (*acc*): the ratio of the number of test samples judged correctly by all categories to the total number of test samples, i.e., Equation (9).

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

(b) Recall (*rec*): the ratio of the number of samples correctly judged as WebShells to the total number of actual WebShell samples, i.e., Equation (10).

$$rec = \frac{TP}{TP + FN} \tag{10}$$

(c) *F*1 value: A measure of classification problem is the harmonic average of good accuracy and recall rate, which can reflect the comprehensive effect of detection, i.e., Equation (11), $pre = \frac{TP}{TP+FP}$

$$F1 = \frac{2 * pre * rec}{pre + rec} \tag{11}$$

4.2.2. Parameter Selection

In this paper, the PHP files obtained after duplication are encoded by opcode, and each PHP file is encoded to obtain an opcode encoded string. Opcode call sequence can reflect the behavior characteristics of files. This paper uses the opcode feature extraction method based on [*M*, *N*]-gram to obtain the numerical representation of WebShell and use it as a feature vector. *M* and *N* are important parameters that affect the feature dimension. In order to ensure the local information of the code, we will study the length characteristics of opcode sequences between 1 *N* and 2 *N*. In order to prevent the excessive value of *N* from affecting the correlation between code features and find out the sequence combination with the most recognition ability, this paper sets the range of *N* to [2, 7]. Through 50% cross-validation, it is found that when the length of opcode sequence fragment, i.e., $M = 2$ and $N = 4$, the classification performance is the best, and the accuracy rate and recall rate exceed 95%, as shown in Figure 8.

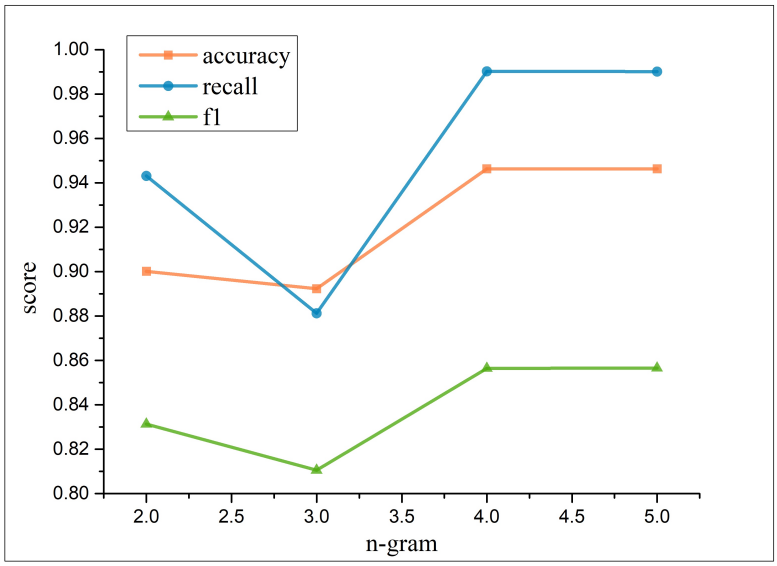



**Figure 8.** Parameter selection of grammatical model.

In the RNCA feature selection model, the weight value is used to evaluate the advantages and disadvantages of features, so it is meaningless to judge the importance of features according to the weight value when the feature scales differ too much. At this time, the whole data set needs to be standardized so that the scale range of each feature is the same. When using TFIDF to process opcode sequences, setting max_df=1 constrains all values in the [0, 1] interval, thus the input data of RNCA model is constructed. It can be seen from the formula that the parameters affecting RNCA model are mainly regularization parameters, learning rate, iteration times, and kernel width. The kernel width of the kernel function is set to 1 by referring to the conclusion of article [22]. For other parameters, we use the method of 50% cross-validation to optimize the parameters of the model on the verification set. The following is the specific process of parameter optimization:

For the optimization of regularization parameters, the values adopted in this paper are 20 points [23] at equal intervals from 0 to 20/n (50-fold cross-validation, 4/5 sample data as training set). The step size for the values are shown in Table 3. Figure 9 shows the average loss rate corresponding to the 20 values. When the values gradually increase, the average loss rate shows an upward trend, proving that the regularization parameter calculation based on learning is effective. When $\lambda = 0.00021$, the average loss rate is the smallest, which is best applied to the subsequent RNCA model training.

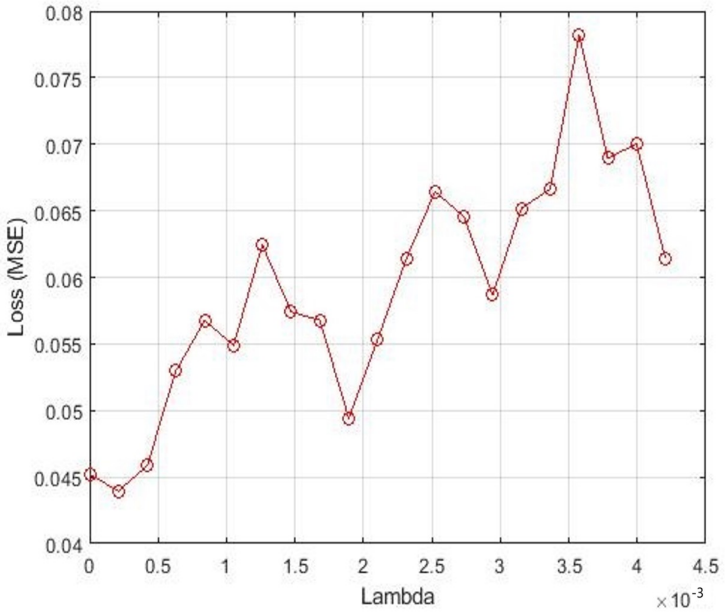



**Figure 9.** Estimation of Regularization parameter $\lambda$ under minimum loss value.

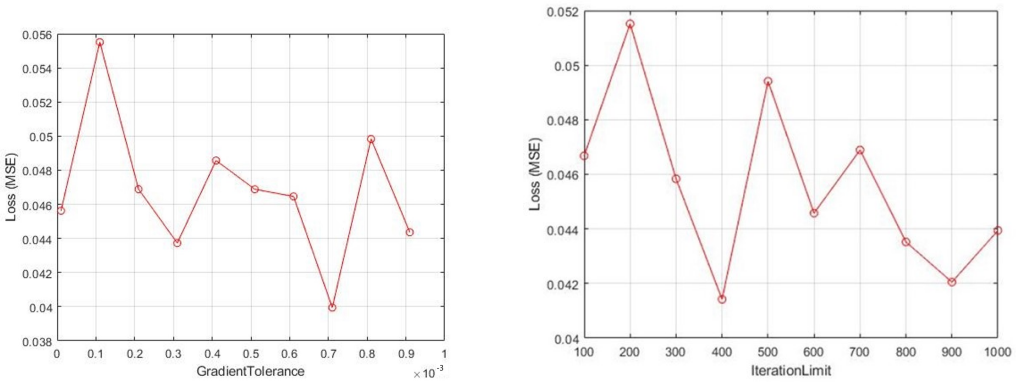
**Table 3.** Select regularization parameter $\lambda$.

| Number | $\lambda$ | Number | $\lambda$ |
|---|---|---|---|
| 1 | 0.000210261 | 11 | 0.002312868 |
| 2 | 0.000420521 | 12 | 0.002523129 |
| 3 | 0.000630782 | 13 | 0.002733389 |
| 4 | 0.000841043 | 14 | 0.00294365 |
| 5 | 0.001051304 | 15 | 0.003153911 |
| 6 | 0.001261564 | 16 | 0.003364172 |
| 7 | 0.001471825 | 17 | 0.003574432 |
| 8 | 0.001682086 | 18 | 0.003784693 |
| 9 | 0.001892347 | 19 | 0.003994954 |
| 10 | 0.002102607 | 20 | 0.004205214 |

In the training process, it is necessary to adjust the learning rate to control the accuracy rate to reach the maximum. According to the principle of invariant factors, we divide the learning rate into 10 ranges, (0–0.001). Experiments show that when the learning rate $\beta = 0.0007$, the loss rate is the smallest, as shown in Figure 10a. Using the same method, we found the number of iterations in the range of (1, 1000) to minimize the loss rate on the verification set, as shown in Figure 10b.

### *4.3. Experimental Results and Analysis*

According to the weight of each feature calculated by RNCA, As shown in Figure 11, we select the features whose weight value is greater than 5% of the maximum weight value as the final classification features, and finally select 31 valid detection features, of which the values of the first five features are shown in Table 4. Among them, send_val do_fcall, do_fcall, end_silence, and send_val do_fcall_by_name init_fcall_by_name sequences are often used to call new functions to realize certain functions of scripts, concat concat and assign concat are often used for string splicing. These behaviors conform to the characteristics of conventional WebShell construction. For the test results using the model, three indicators are used as standards, namely accuracy, recall rate, and *F*1-score.




(**a**) (**b**)

**Figure 10.** The RNCA parameter selection (**a**) represents that learn rate parameter $\beta$ selection; (**b**) represents the iteration number m parameter selection.

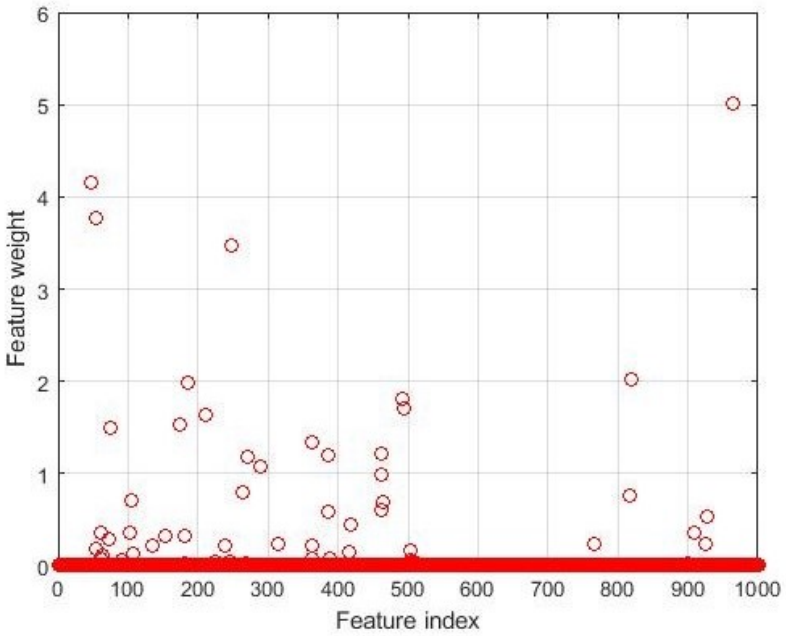



**Figure 11.** Feature weight graph.

**Table 4.** Feature selection results.

| RNCA Top 5 | Fisher Top 5 | ReliefF Top 5 |
|---|---|---|
| send_var send_val do_fcall | do_fcall end_silence | send_val do_fcall_by_name init_fcall_by_name |
| assign begin_silence | concat concat | send_val do_fcall bool_not jmpz |
| assign concat | begin_silence | fetch_r init_method_call fetch_obj_func_arg |
| do_fcall end_silence | assign concat | concat concat send_var do_fcall |
| send_val do_fcall assign | do_fcall concat concat | concat concat send_var |

In order to test the effectiveness of the selected features, we performed several groups of control experiments, as shown in Tables 3 and 4. In the control experiment, the typical representatives of filtering feature selection method, ReliefF and Fisher ratio, and the typical representatives of wrapping feature selection method, recursive elimination and random forest, are selected. These feature selection methods all use the same data set. RNCA mainly uses the nearest neighbor idea to complete feature selection, and ReliefF is also suitable for feature selection of nearest neighbor samples. Using Fisher ratio, ReliefF, random forest, recursive elimination, RNCA, and statistical features alone to input the

selected features into the random forest classification algorithm, it is found that the recall rate and *F*1 value obtained by using these six methods alone are not ideal. Among them, ReliefF algorithm assigns weights to each feature by comparing the distance between classes and the distance within classes of adjacent samples, and its classification accuracy in WebShell detection experiment is 93%. The Fisher ratio and RNCA are also some of the feature selection methods based on distance measurement, which has the characteristics of strong operability, less computation and high efficiency, and has a good effect on dimension reduction and classification of high-dimensional data. In view of the fact that the ReliefF and Fisher do not consider the classification effect of classifiers in the feature ranking process, their performance in accuracy and recall rate is slightly worse. Random forest combines the voting results of the classifier, and its overall performance is better than the two filtering feature selection methods, but the training time is longer. During the experiment, we found that when the statistical features are combined with the opcode sequence features of the five feature selection algorithms, the overall performance of the classifier is significantly improved, the accuracy rate is over 97%, and the recall rate and *F*1 value are over 85%. This phenomenon shows that the problem of low accuracy of WebShell detection caused by a single feature has been alleviated. Secondly, statistical features and behavioral sequence features can represent WebShell samples more comprehensively. Table 5 shows that the combined features formed by combining opcode sequence features and statistical features obtained by RNCA can effectively improve the accuracy, recall rate, and *F*1 value of random forest classification, further proving that the proposed method has good accuracy, generalization, and detection capability for unknown types of samples.

**Table 5.** Comparison data of feature selection methods.

| Method | Dimension | Accuracy | Recall | *F*1 Value |
|---|---|---|---|---|
| Fisher ratio | 31 | 0.9580 | 0.6058 | 0.7159 |
| ReliefF | 31 | 0.9327 | 0.3365 | 0.4667 |
| RF | 31 | 0.9622 | 0.6538 | 0.7514 |
| Recursive elimination | 31 | 0.9361 | 0.3462 | 0.4865 |
| RNCA | 31 | 0.9655 | 0.6538 | 0.7771 |
| Statistical characteristics(SC) | 6 | 0.9815 | 0.8654 | 0.8911 |
| Fisher ratio+SC | 37 | 0.9824 | 0.8558 | 0.8945 |
| ReliefF+SC | 37 | 0.9857 | 0.875 | 0.9146 |
| RF+SC | 37 | 0.9908 | 0.9327 | 0.9463 |
| Recursive elimination+SC | 37 | 0.9773 | 0.8942 | 0.8732 |
| RNCA+SC | 37 | 0.9924 | 0.9327 | 0.9557 |

Compared with the well-known WebShell detection tools, we chose three popular WebShell detection tools, namely D Shield, WEBDIR +, and SHELLPUB. D Shield has a powerful rule base, including dangerous functions and WebShell libraries, which can be used to check and kill hidden back doors; WEBDIR + uses dynamic monitoring technology and zero rule checking to solve the problems of high false alarm rate and low killing rate existing in traditional regular expressions; SHELLPUB uses traditional features and cloud big data dual engines to realize killing. In this paper, these three tools are used to scan and detect all samples in our test set, and the results shown in Table 1 are obtained.

The two detection methods based on depth learning use multi-input and single-output to numerically represent the input of samples, including opcode sequence features and static statistical features. Three-layer LSTM is used for model training. When the loss value tends to be stable, the model stops training, and the results are shown in Table 6. SC in the table represents static statistical characteristics. On the premise that the Opcode feature dimension is controlled at 1000 dimensions, the two depth learning methods take a small amount of time to complete, but the recall rate and *F*1 value are low, which may be due to the model's sensitivity to parameters and unstable effects. Compared with the WebShell feature extraction method based on NCA_ReliefF proposed in Section 3, its accuracy rate exceeds 96% and recall rate exceeds 95%. If it is necessary to quickly build

a model to complete the prediction, choosing this method can retrieve a large number of WebShell samples. The detection model based on RNCA takes a long training time due to the learning of regularization parameters. If the accuracy of sample recognition is required to be high, the training time can be ignored. This method can complete the detection task with high quality.

**Table 6.** WebShell detection method based on contrast deep learning.

| Detection Method | Accuracy | Recall | *F*1 | Train | Test |
|---|---|---|---|---|---|
| Word2vec+sc+LSTM | 0.9756 | 0.8269 | 0.8557 | 28.64 s | 0.52 s |
| Doc2vec+sc+LSTM | 0.9403 | 0.6154 | 0.6432 | 23.31 s | 0.53 s |
| NCA_ReliefF | 0.9675 | 0.9527 | 0.8474 | 422.41 s | 0.60 s |
| Detection model based on RNCA | 0.9924 | 0.9327 | 0.9327 | 1304.00 s | 0.19 s |

As can be seen from Table 7, the accuracy rate of D Shield can be as high as 98.05%, and the recall rate is as high as 93.24%, which is the best performance among the three testing tools. The accuracy and recall rate of the model detection proposed in this chapter can reach 99.24% and 93.27%. Obviously, the WebShell detection model based on RNCA is superior to these WebShell detection tools. The results also prove the effectiveness of combining static statistical features with opcode sequence features, and the method can significantly improve the accuracy of WebShell detection.

**Table 7.** WebShell detection tool evaluation results.

| Testing Tool | Version | Accuracy | Recall |
|---|---|---|---|
| D shield | V2.1. 5.4 | 0.9805 | 0.9324 |
| WEBDIR + | V2019-0326-0800 | 0.9717 | 0.6756 |
| SHELLPUB | V1.6. 0 | 0.9482 | 0.5135 |
| Detection Model Based on RNCA | / | 0.9924 | 0.9327 |

In order to verify the relationship between the selected features and categories, we use covariance to judge the correlation between features and display them in the form of thermal maps. As shown in Figure 12, the top ten features with the highest importance obtained by five feature selection methods and the correlation between tags are included. Number 10 represents class tags. In the heat map, the darker the color, the lower the correlation. RNCA, ReliefF, and Fisher have stronger correlation with categories than random forest and recursive elimination. Among them, there are more dark regions between various features of RNCA, indicating that the correlation between features is not strong and redundant features are less.

We counted the frequency of the selected features in malicious samples and normal samples, respectively. Figure 13 show that the frequency of these opcode sequence features in malicious samples is relatively high, indicating that the selected features have strong WebShell discrimination ability.

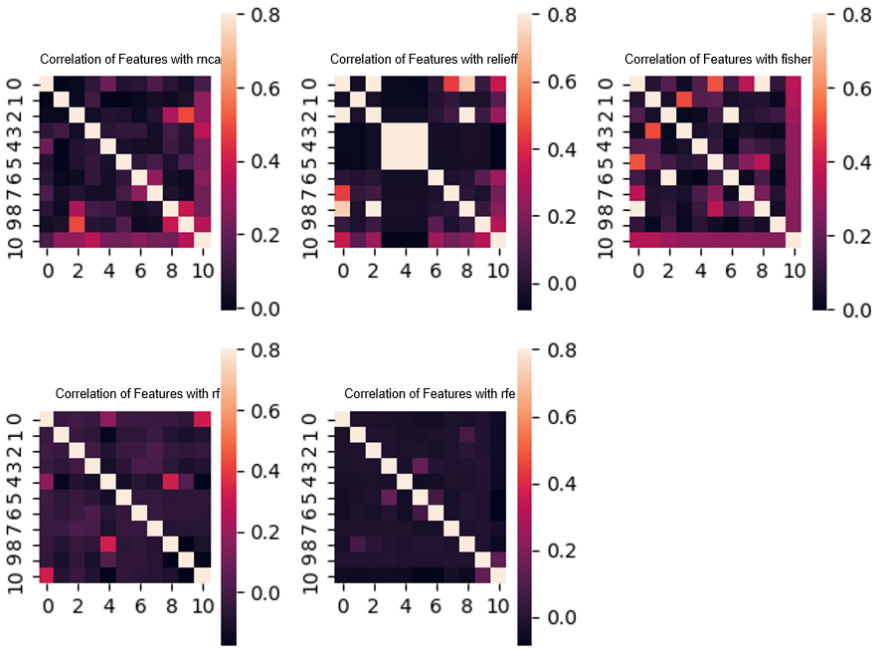



**Figure 12.** Characteristic thermodynamic map of selected opcode sequence.

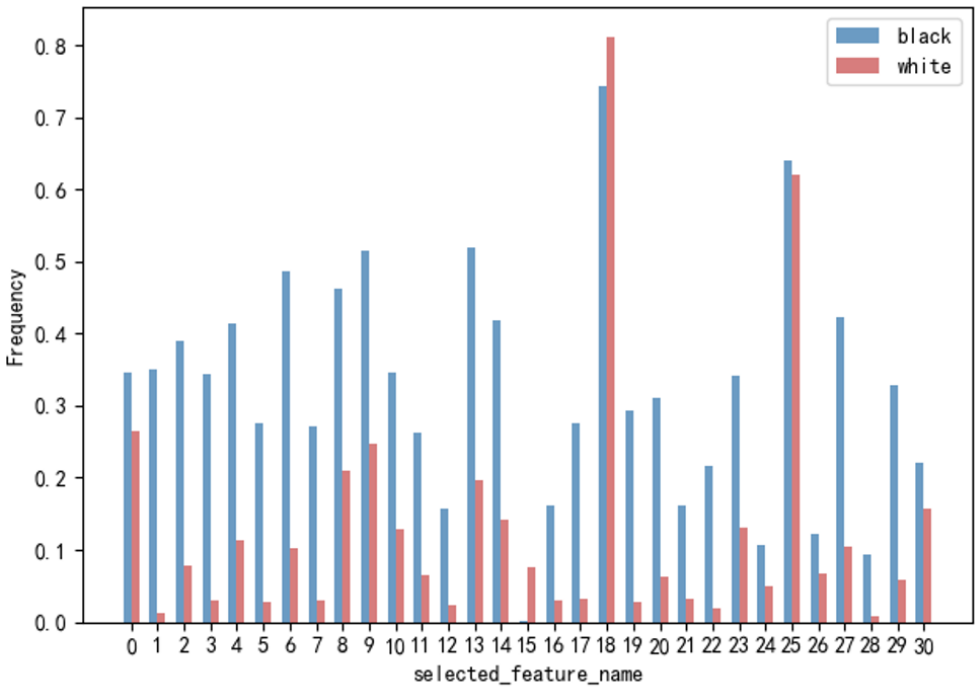



**Figure 13.** Frequency of occurrence of selected opcode sequence features.

## 5. Conclusions

In order to solve the problem that the feature dimension of WebShell samples is too high and the detection effect is poor, this paper deals with a WebShell detection method based on RNCA. The algorithm adopts the opcode feature processing method based on [M, N] to preserve the opcode context environment, and proposes the combination of opcode sequence features and text statistical features to improve the comprehensiveness of detection. Experiments show that the opcode sequence features screened out based on RNCA algorithm have stronger sample recognition ability and have a positive impact on WebShell sample detection. Its detection cost is low, which mainly depends on the size of

data set n and the number of retained features m, that is, $O(mn \log n)$. The research object of this paper is the WebShell written in PHP. How to apply this method to more types of WebShell detection needs further research.

**Author Contributions:** Conceptualization, A.Z. and N.L.; methodology, A.Z., N.L., and Z.A.; writing—original draft preparation, A.Z.; writing—review and editing, A.Z.; project administration, N.L.; funding acquisition, N.L. All authors read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61433012 and in part by the Innovation Environment Construction Special Project of Xinjiang Uygur Autonomous Region under Grant PT1811.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available in Table 2.

**Conflicts of Interest:** The authors declare that they have no conflict of interest to report.

## References

1. National Internet Emergency Center. Analysis Report of China's Internet Network Security Monitoring Data in the First Half of 2020. Available online: https://www.cert.org.cn/publish/main/46/index.html (accessed on 26 September 2020).
2. Internet Live Stats. Internet Usage and Social Media Statistics. 2019. Available online: https://www.internetlivestats.com/ (accessed on 26 September 2020).
3. Wrench, P.M.; Irwin, B.V.W. Towards a PHP WebShell taxonomy using deobfuscation-assisted similarity analysis. In Proceedings of the 2015 Information Security for South Africa (ISSA), Johannesburg, South Africa, 12–13 August 2015; pp. 1–8.
4. Luczko, P.; Thornton, J. PHP Shell Detector. Available online: https://github.com/emposha/PHP-Shell-Detector (accessed on 12 June 2012).
5. Zhou, Y.; Yong, H.U. Research on WebShell detection method based on correlation analysis. *Inf. Secur. Res.* **2018**, *3*, 252–255.
6. Deng, L.Y.; Dong, L.L.; Chen, Y.H.; Yann, L.X. Lexical Analysis for the WebShell Attacks. In Proceedings of the International Symposium on Computer, Consumer and Control, Xi'an, China, 4–6 July 2016; pp. 579–582.
7. Zhang, H.Y. WebShell detection method based on semantic analysis and neural network. *Cyberspace Secur.* **2019**, *10*, 17–23.
8. Fang, Y.; Qiu, Y.; Liu, L.; Huang, C. Detecting WebShell Based on Random Forest with FastText. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence (ICCAI 2018), Chengdu, China, 12–14 March 2018; pp. 52–56.
9. Cui, H.; Huang, D.; Yong, F.; Liu, L.; Huang, C. WebShell Detection Based on Random Forest–Gradient Boosting Decision Tree Algorithm. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018.
10. Goldberger, J.; Roweis, S.; Hinton, G.; Salakhutdinov, R. Neighbourhood components analysis. In Proceedings of the Advances Neural Information Processing Systems, Vancover, BC, Canada, 13–16 December 2004; Volume 17, pp. 513–520. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.449.1850&rep=rep1&type=pdf (accessed on 2 June 2021).
11. Zuo, W.; Zhang, H.; Wang, K. Fast neighbourhood component analysis with spatially smooth regulariser for robust noisy face recognition.*IET Biom.***2014**, *3*, 278–290.
12. Liu, C.; Li, X.; Yang, Y. A text classification algorithm based on nearest neighbor analysis. *Comput. Eng.* **2012**, *38*, 139–141.
13. Nguyen, H.V.; Bai, L. Face Verification Using Indirect Neighbourhood Components Analysis. In Proceedings of the Advances in Visual Computing 6th International Symposium, ISVC 2010, Las Vegas, NV, USA, 29 November–1 December 2010.
14. Ferdinando, H.; Seppänen, T.; Alasaarela, E. Emotion Recognition Using Neighborhood Components Analysis and ECG HRV-Based Features. In Proceedings of the International Conference on Pattern Recognition Applications & Methods, Porto, Portugal, 24–26 February 2017.
15. Singh-Miller, N.; Collins, M.; Hazen, T.J. Dimension Reduction for Speech Recognition Using Neighborhood Components Analysis. In Proceedings of the 8th Annual Conference of the International Speech Communication Association, Antwerp, Belgium, 27–31 August 2007.
16. Rizwan, M.; Anderson, D.V. Speaker similarity score based fast phoneme classification by using neighborhood components analysis. In Proceedings of the IEEE Global Conference on Signal and Information Processing (Global-SIP), Washington, DC, USA, 7–9 December 2016.
17. Jin, M.; Deng, W. Predication of different stages of Alz-heimer's disease using neighborhood component analysis and ensemble decision tree. *J. Neurosci. Methods* **2015**, *302*, 35–41. [CrossRef] [PubMed]
18. Zheng, Y.; Liu, Q.; Chen, E.; Zhao, J.L.; He, L.; Lv, G. Convolutional Nonlinear Neighbourhood Components Analysis for Time Series Classification. In Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Ho Chi Minh City, Vietnam, 19–22 May 2015.

19. Malan, N.S.; Sharma, S. Feature selection using regu-larized neighborhood component analysis to enhance the classification performance of motor image signals. *Comput. Biol. Med.* **2019**, *107*, 118–126. [CrossRef] [PubMed]
20. Zhang, H.; Xue, Z.; Shi, Y. Improved Method of Detecting WebShell based on Multi-Layer Perceptions. *Commun. Technol.* **2019**, *52*, 179–183.
21. Wang, D.; Tan, X. Bayesian Neighborhood Component Analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 3140–3151. [PubMed]
22. Yang, W.; Wang, K.; Zuo, W. Neighborhood component feature selection for highdimensional data. *J. Clin. Pharm.* **2012**, *7*, 161–168.
23. Zhao, Y.; Liu, Y.; Huang, W. Classification prediction of HBV reactivation after precise radiotherapy for primary liver cancer based on nearest neighbor component analysis algorithm. *Bioinformatics* **2018**, *16*, 163–169.