applied sciences

MDPI

Article

Malicious URL Detection Model Based on Bidirectional Gated Recurrent Unit and Attention Mechanism

Tiefeng Wu, Miao Wang *, Yunfang Xi and Zhichao Zhao

School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266520, China
* Correspondence: wangmiao_qut@163.com

Abstract: With the rapid development of Internet technology, numerous malicious URLs have appeared, which bring a large number of security risks. Efficient detection of malicious URLs has become one of the keys for defense against cyber attacks. Deep learning methods bring new developments to the identification of malicious web pages. This paper proposes a malicious URL detection method based on a bidirectional gated recurrent unit (BiGRU) and attention mechanism. The method is based on the BiGRU model. A regularization operation called a dropout mechanism is added to the input layer to prevent the model from overfitting, and an attention mechanism is added to the middle layer to strengthen the feature learning of URLs. Finally, the deep learning network DA-BiGRU model is formed. The experimental results demonstrate that the proposed method can achieve better classification results in malicious URL detection, which has high significance for practical applications.

Keywords: attention mechanism; bidirectional gated recurrent unit; dropout mechanism; malicious URL detection

Citation: Wu, T.; Wang, M.; Xi, Y.; Zhao, Z. Malicious URL Detection Model Based on Bidirectional Gated Recurrent Unit and Attention Mechanism. *Appl. Sci.* **2022**, *12*, 12367. https://doi.org/10.3390/app122312367

Academic Editor: David Ruano Ordás

Received: 3 November 2022
Accepted: 30 November 2022
Published: 2 December 2022

1. Introduction

With the rapid development of Internet technology, the number of Internet pages has increased almost exponentially in recent years. Although the Internet brings us convenience, it provides opportunities for many criminals. Criminals can maliciously install computer viruses and garbage software and execute other attacks to achieve the purposes of stealing user identity information or network fraud. In order to effectively reduce illegal behavior on the internet, a large number of researchers have conducted in-depth research on malicious URL detection technology.

In the past, the common method for diagnosing and defending malicious URL attacks was to use the blacklisting technique [1], which combines the key information of known malicious URLs into a list. By accessing this list, we can accurately identify the confirmed malicious URLs. For example, Prakash et al. [1] proposed the identification and discovery of new phishing pages through URL decomposition and similarity calculations, extending the scope of use of blacklisting and helping identify some malicious pages that do not appear in the blacklist. The blacklist mechanism for browsers such as Google Safe Browsing is also a similar method [2]. Lin Hai-lun et al. [3] presented an efficient method for detecting malicious URLs based on segment pattern, which had good performance and scalability. Because the blacklist is artificially constructed, it requires considerable human resources in subsequent maintenance. Moreover, with the continuous updating of malicious URL attacks, malicious URL detection is limited by the blacklisting technique, which has been unable to meet the needs of network attack defense in today's society. Therefore, researchers began to seek more efficient defense technology. Later, honeypot technology emerged. For different types of attacks, honeypot technology gradually derived the honey net, distributed honeypot, honey field, and other technologies [4–6]. For example, Ref. [7] studied the network intrusion detection system (NIDS), which used the snooping agents on the Web and honeypot technology to prevent the activities of intruders. However, according to the research by Zhuge Jianwei et al. [8], the honeypot environment cannot

solve the contradiction between simulation and controllability, and the honeypot technology is mainly used for large-scale common security threats, whose defense ability against some special threats is far from sufficient.

With the research wave based on machine learning, the application of machine learning to malicious URL detection has attracted a lot of attention from researchers. For example, Vanhoenshoven et al. [9] used a multilayer perceptron (MLP) to detect malicious URLs, and found that for the same data set, the detection results of different feature sets may be different. Arivazhagi et al. [10] presented an efficient unsupervised feature construction method based on the linear support vector machine model, and the research results proved its effectiveness. Abed Sa'ed et al. [11] proposed a machine learning model combining a self-encoder with a class of support vector machines. The dimension of input data was reduced by the self-encoder, and the network events were classified by a class of support vector machines. Azeez et al. [12] used a naive Bayesian algorithm to detect malicious URLs based on the syntax, vocabulary, hosts, and other content of the URL embedded in the email. Laughter et al. [13] integrated the http request features in the process of visiting the website into the detection feature set. By extracting the content of each field in the request header and request body, classification methods such as decision tree and SVM were used to complete the research. Because traditional machine learning methods require complex feature selection, such methods usually have the problem of poor scalability, that is, a feature or a class of features that is currently extracted may have good results in a certain type of malicious web page recognition problem, but the performance is degraded in other types of malicious web page recognition problems.

Deep learning is an important upgrade to machine learning. In recent years, with the unique advantages of deep learning in natural language processing (NLP), speech recognition, image recognition, and other fields, it has also brought new developments to the detection of malicious web pages. For example, Zhang et al. [14] presented an automatic URL feature extraction method based on a convolutional neural network and verified the extraction results by combining random forests, support vector machines (SVM), and other classification methods, which proved the effectiveness of the deep learning method for URL feature extraction. However, convolutional neural networks can only learn the local features of URLs and cannot learn their context information. J.J. Christy Eunaicy et al. [15] detected malicious URLs based on artificial neural network (ANN), convolutional neural network (CNN) and recurrent neural network (RNN) models. In their results, the RNN model had the best performance, with an accuracy of 94%. However, RNNs can easily experience gradient disappearance or gradient explosion, which will affect the accuracy of malicious URL detection and classification. Afzal et al. [16] proposed a hybrid deep learning method called URLdeepDetect for click-time URL analysis and classification to detect malicious URLs. DasA et al. [17] used a character-level embedded coding format to represent URLs and designed CNN and LSTM models to detect malicious URLs. However, the accuracy of the model was only 93.59%, which could not detect malicious URLs well. Cui et al. [18] combined the HTTP parameters generated by user requests with the corresponding URLs and compared six vectorization methods in the data preprocessing stage.

Deep neural networks imitate the human brain mechanism to interpret data; the brain does not simply process incoming information in a moment, but instead it promotes the perception of the scene through attention, expectation, and prior knowledge, extracting interested characteristics to understand information. Therefore, researchers presented the attention mechanism [19], which achieves the purpose of obtaining more effective information by giving higher weights to the concerned parts in deep learning and has achieved good results in machine translation, false news discrimination [20], and other tasks. Attention mechanisms were inspired by the fact that deep neural networks [15] do not distinguish information in the URLs when identifying malicious URLs, and the key information is not fully utilized as a result. The attention mechanism can effectively allocate the information's characteristics and improve the deep learning efficiency so as to obtain better recognition performance of malicious URLs.

Based on the above analysis, this paper proposes a malicious URL detection method based on a bidirectional GRU and an attention mechanism. This method applies Word2Vec to train the word vector of URLs, uses BiGRU to extract all the sequence information of URLs and learn the relationships between sequences, and introduces an attention mechanism to strengthen the model to learn useful information. In addition, to prevent overfitting during model training, a dropout mechanism is added to the input layer to improve the accuracy of malicious URL detection and classification.

2. Correlation Technique

2.1. Word2Vec

Word2Vec [21] is a word vector computing tool proposed by Google in 2013 that is commonly used in the field of natural language processing. Special features of Word2Vec are that it can conduct efficient training on large dictionaries and data sets, map words into real number vectors in vector space, and effectively measure the similarity between words. In this paper, the word vector is trained by the skip-gram model in Word2Vec, and the URLs are vectorized to facilitate subsequent feature extraction. The skip-gram model uses the current position word x_t to predict the probability of context words. The framework of skip-gram model is shown in Figure 1.

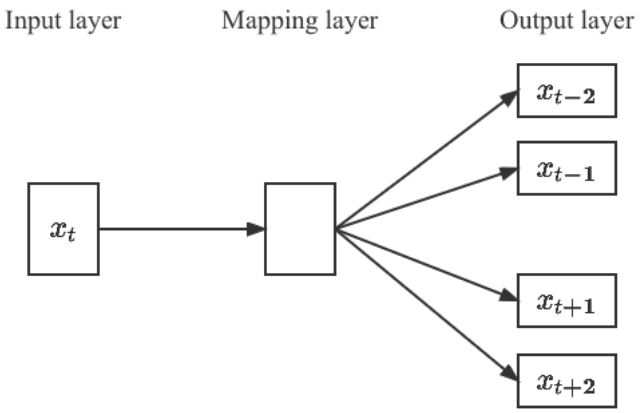


Figure 1. Skip-gram model framework.

2.2. BiGRU

In 2014, Cho et al. [22] presented a gated recurrent unit (GRU) neural network model. GRU has a gate control that supports the hidden state, which is used to determine the time of updating the hidden state and the method of resetting the hidden state. These mechanisms can be learned through training. For example, a word is extremely important for discrimination, and the model does not update the hidden state after the first learning. Similarly, the model can also learn to ignore some unimportant words. In addition, the model will also learn to reset the hidden state when needed. The following details introduce the working mechanism of GRU gate control.

2.2.1. Reset Gate and Update Gate

Figure 2 describes the structure of the reset gate R_t and the update gate Z_t in GRU. The input of the two gates is composed of the input of the current moment and the hidden state of the previous moment. The output of the two gates is calculated by the full connection layer containing a sigmoid activation function. The calculation formula is shown in (1) and (2).

$$R_t = \sigma(W_{xr} \times x_t + W_{hr} \times h_{t-1} + b_r). \quad (1)$$

$$Z_t = \sigma(W_{xz} \times x_t + W_{hz} \times h_{t-1} + b_z). \quad (2)$$

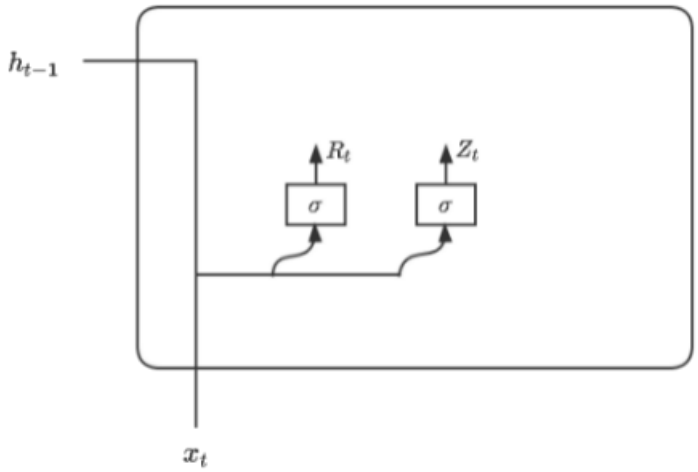


Figure 2. Reset gate and update gate structure of GRU.

The meaning of each symbol in the above formula is shown in Table 1.

Table 1. Symbolic meaning.

Symbolic	Meaning
σ	Activation function
$W_{xr}, W_{hr}, W_{xz}, W_{hz}$	weight parameter
b_r, b_z	offset parameters
h_{t-1}	Hidden state of the previous time
x_t	Current input

2.2.2. Candidate Hidden State

Figure 3 illustrates the structure of the candidate hidden state. The output of the reset gate R_t is multiplied by the hidden state of the previous moment h_{t-1} and mapped to the current input x_t. The data are constrained in $[-1, 1]$ using the tanh function to output the memory content of the candidate hidden state $\widetilde{h}_t$. The calculation formula is as follows.

$$\widetilde{h}_t = \tanh(W_{zh} \times x_t + W_{hh} \times (h_{t-1} \odot R_t) + b_h). \quad (3)$$

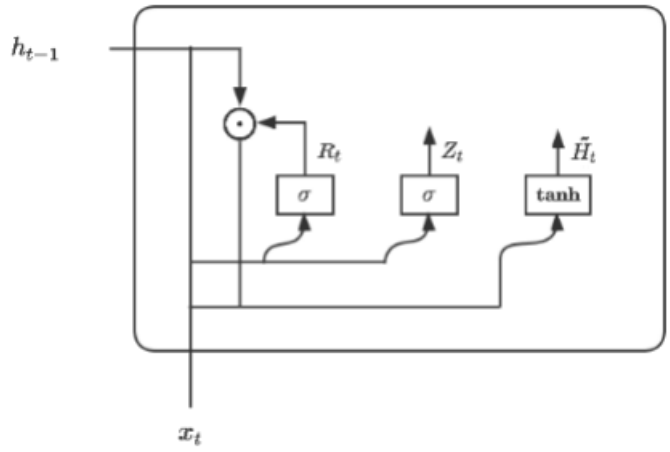


Figure 3. Structure of candidate hidden states.

2.2.3. Hidden State

After calculating the candidate hidden state $\widetilde{h}_t$, it must be combined with the update gate Z_t to further determine the weight of the hidden state at the previous moment h_{t-1}

and the new candidate hidden state $\widetilde{h_t}$ in the new hidden state h_t. The calculation formula is as follows.

$$h_t = h_{t-1} \odot Z_t + (1 - Z_t) \odot \widetilde{h_t}. \quad (4)$$

It can be seen from (4) that when the update gate Z_t approaches 1, most of the model retains the information of the hidden state at the previous moment h_{t-1}, and the information from x_t will be ignored. When the update gate Z_t approaches 0, most of the information in the candidate hidden states $\widetilde{h_t}$ is extracted from the model. This can solve the problem of gradient disappearance in recurrent neural networks. Figure 4 demonstrates the entire process framework for GRU.

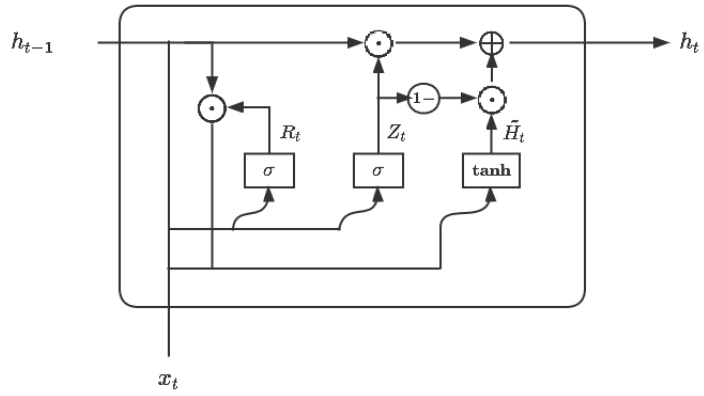


Figure 4. Entire process framework for GRU.

BiGRU is a variant of GRU. Its output depends on the dual effects of forward and backward states, which makes the final output more accurate. The structure of the BiGRU model is shown in Figure 5, where x_t denotes the input word at t time, e_t denotes the embedding layer word vector, $\overrightarrow{h_t}$ denotes the hidden state of the forward GRU, $\overleftarrow{h_t}$ denotes the hidden state of the reverse GRU, and h_t denotes the hidden state after the forward and reverse splicing.

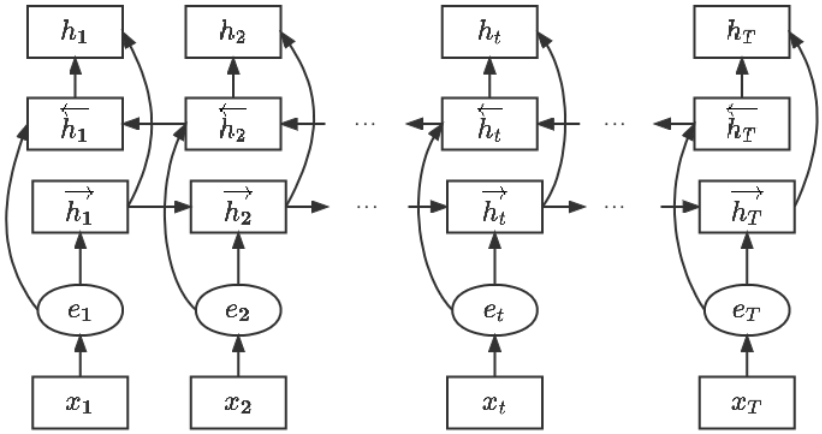


Figure 5. Structure of BiGRU.

2.3. Attention Mechanism

Different locations of URLs have different specification requirements. Therefore, for a URL, in order to fully learn the dependency between a word or a symbol and the words in other locations of the URL, this paper introduces an attention mechanism. This allows the model to focus more on the key information of URLs, while other irrelevant content will be ignored by the model. The introduction of an attention mechanism can facilitate more effective use of data, thereby improving the accuracy of the model. The specific calculation formula is as follows.

$$e_t = W^T \sigma(W_l * x_t). \tag{5}$$

$$q_t = \frac{exp(e_t)}{\sum_{t=1}^{T} exp(e_t)}. \tag{6}$$

$$x_t^* = \sum_{t=1}^{T} q_t * x_t. \tag{7}$$

In the formula, x_t is the input information at t time W_l, W^T is the learned weight, σ is the tanh activation function, and x_t^* is the output information after t time passes through the attention layer.

2.4. Dropout Mechanism

Dropout is a regularization method, which is used to randomly disable neural network units. In the forward propagation process of the model training stage, the activation values of some neurons are stopped with a certain probability, which can make the model more generalized. This method can be exempted from the dependence on other neurons, thereby enabling the network to learn independent correlation. This method can reduce the density of the network, as shown in Figure 6.

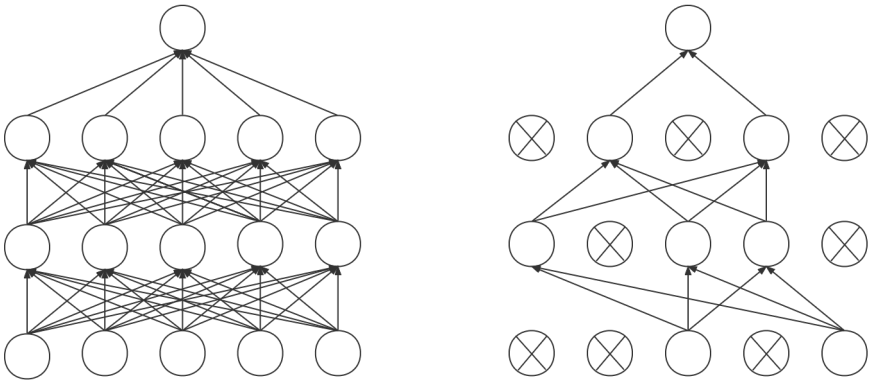


Figure 6. Dropout Mechanism.

3. Model Structure

URLs themselves have strong requirements for sequence order, and GRU is a model that can process sequence data. At the same time, BiGRU can deeply mine and make full use of the relevant information in the obtained URL data. Therefore, this paper uses BiGRU as the basic model structure. In order to prevent the neural network model from overfitting in the training process, that is, a situation where it performs well and has high accuracy on the training set, but it performs poorly and has low accuracy on the test set. In this paper, a layer of dropout mechanism is added before URL data enter the model, which avoids the excessive influence of some weights on the network model to a certain extent and reduces the model's deviation.

URLs are not complicated, but they have certain requirements for sequence relations. Therefore, this paper first uses a layer of BiGRU, and the number of neural nodes is set to 128 to learn the characteristics of URLs. Then, the learned information is outputted to the attention layer to improve the full use of key information. Finally, the full connection layer using the tanh function as the activation function is added, and the softmax function is used for the final classification. Thus, a dropout–attention bidirectional gated recurrent unit (DA-BiGRU) is formed. The DA-BiGRU model structure is shown in Figure 7.

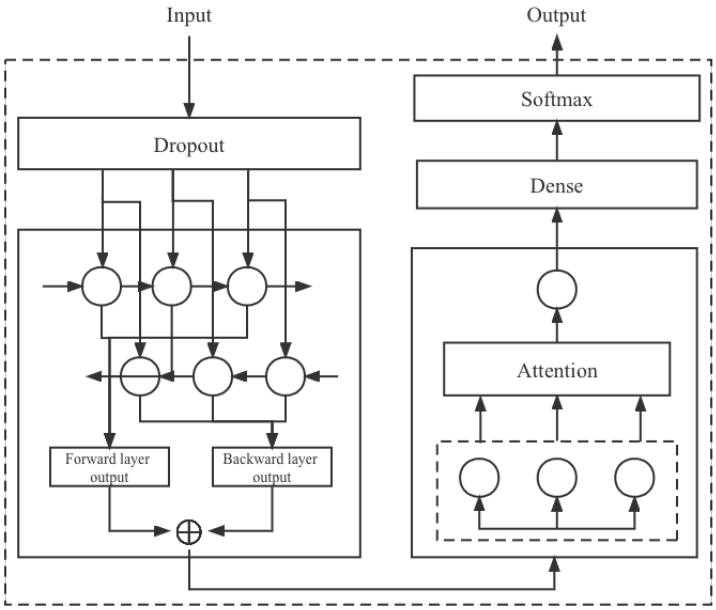


Figure 7. Model structure of DA-BiGRU.

This method firstly preprocesses the URL data set using Word2Vec to construct a word vector and segmenting the data set into a digital vector. Secondly, the preprocessed data set is constructed into the data format required by the model through embedding layer. Then, we train the proposed model DA-BiGRU. Finally, utilizing the best-performing model, we can perform the malicious URL detection task and obtain the classification result. The calculation equations are shown below.

The input at t time x_t becomes x_t' by dropout mechanism:

$$x_t' = dropout(x_t). \tag{8}$$

The hidden state output of forward propagation GRU and back propagation GRU at t time can be defined as:

$$\overrightarrow{h_t} = GRU(x_t', \overrightarrow{h_{t-1}}). \tag{9}$$

$$\overleftarrow{h_t} = GRU(x_t', \overleftarrow{h_{t-1}}). \tag{10}$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ denote the hidden state output of forward propagation GRU and back propagation GRU at $t-1$ time, respectively.

The hidden state output of BiGRU at t time is the weighted sum of forward and backward:

$$h_t = \overrightarrow{W_t} \cdot \overrightarrow{h_t} + \overleftarrow{W_t} \cdot \overleftarrow{h_t} + b_t. \tag{11}$$

where $\overrightarrow{W_t}$ is the weight of $\overrightarrow{h_t}$, $\overleftarrow{W_t}$ is the weight of $\overleftarrow{h_t}$, and b_t is the bias.

The output of hidden states through attention mechanism h_t^* can be expressed as:

$$h_t^* = attention(h_t). \tag{12}$$

After adding the full connection layer, the output y^* is calculated by $W_{\tanh}$ and h_t^*:

$$y^* = \tanh(W_{\tanh}, h_t^*). \tag{13}$$

Finally, entering the softmax layer, the classification result is obtained by:

$$y = softmax(W_s y^* + b_s). \tag{14}$$

where W_s and b_s represent the weight and bias of the softmax layer, respectively.

4. Experimental Results and Analysis

The flowchart of this experiment is shown in Figure 8. First, URL data are preprocessed. That is, the data are segmented first, and then Word2Vec is used to obtain the word vector. Secondly, the data after the embedding layer are input into the DA-BiGRU model for learning. Finally, URLs are classified using the DA-BiGRU model.

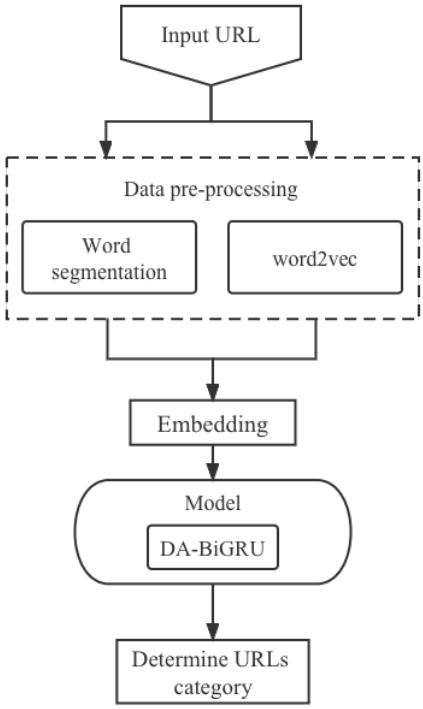


Figure 8. Experimental process.

4.1. Data Set

The experimental data set used in this paper is a collection malicious phish URLs. The malicious phish URLs (https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset) come from the Kaggle community and are from a collection of URL data collected by Manu Siddhartha across multiple communities. In this experiment, 65,536 benign URLs and 65,536 malicious URLs were randomly selected, for a total of 131,072 URLs. This article divides the URL data into a training set, a validation set, and a test set after randomly disrupting the order. The specific sample size is shown in Table 2.

Table 2. Number of samples in each data subset.

	Training Set	Validation Set	Test Set	Aggregate
Malicious URLs	45,956	5879	13,701	65,536
Benign URLs	45,795	5917	13,824	65,536
Aggregate	91,751	11,796	27,525	131,072

4.2. Data Preprocessing

The data preprocessing is divided into four parts: data cleaning, word segmentation, vocabulary creation, and length interception. The data preprocessing flowchart is shown in Figure 9.

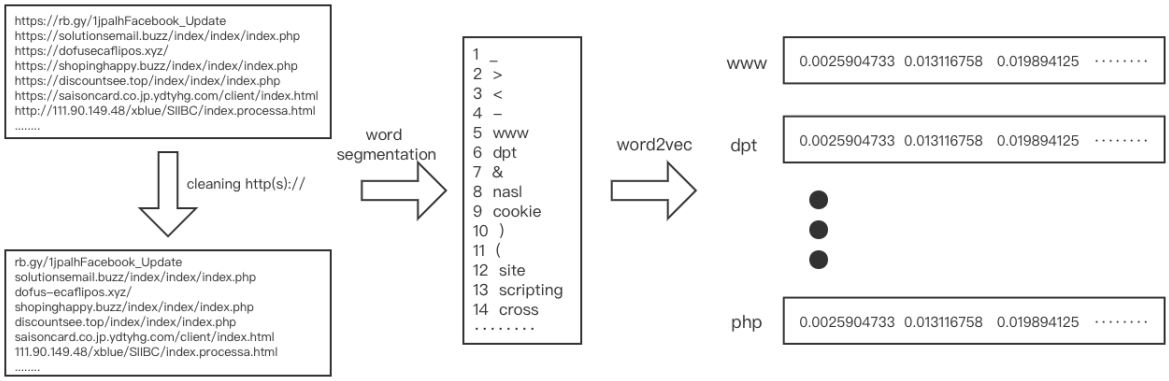


Figure 9. Flowchart of data preprocessing.

Data cleaning: because the URL protocol is generally http:// or https://, this part of the content has little effect on the identification of malicious URLs. Therefore, this paper conducts data cleaning on URLs containing http:// and https:// in the experimental data set to remove redundant information to reduce the waste of features.

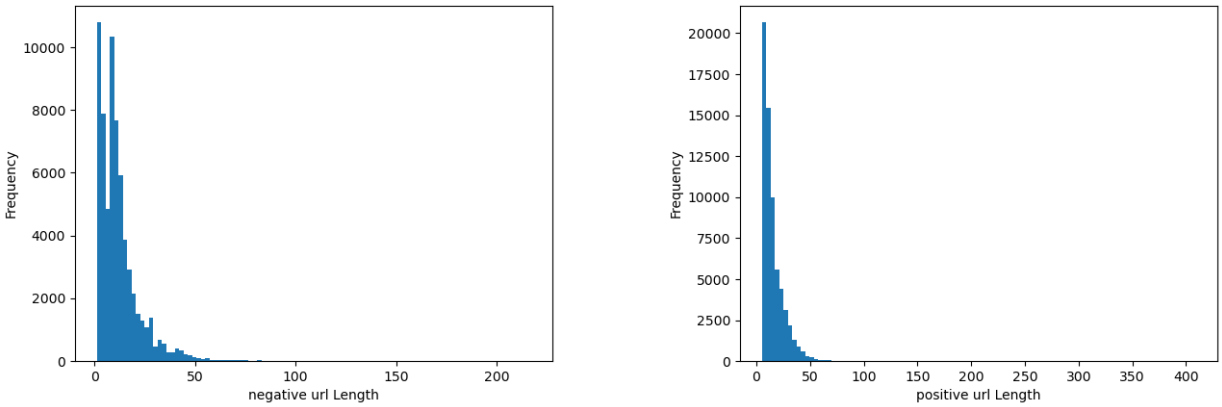
Word segmentation processing: the clean data are segmented by regular expressions. Because the special symbols in the URL contain important feature information, the special symbols that often appear in the URL are counted. The statistics show that the following 14 special symbols appear more frequently: '-', '_', '.', '=', '/', '?', '&', '#', '<','>', '(', ')', '+', and '@'. Finally, this paper sets the word segmentation rules according to the following 14 symbols: ' - _ . = / ? & # + < > ( ) @'.

Creating a vocabulary: although one-hot encoding is easy to construct, it cannot accurately express the arrangement of words, and when the number of words is large, one-hot encoding is prone to dimension disaster. Therefore, this paper selects the skip-gram model algorithm of Word2Vec to train the vocabulary and obtain 50-dimensional word vectors.The vocabulary example can be seen in Table 3.

Table 3. Vocabulary example.

The Vocabulary Example: songfacts.com/detail.php?id=13410	
Vocabulary	**Word Vector (50 Dimensional)**
songfacts	0.019443182 0.013166171 −0.01845449 . . . 0.0075058653 −0.014516649
com	−0.014362931 −0.012048096 0.0060079815 . . . 0.0001564598 0.012821173
detail	0.017598134 −0.0045375777 0.008738079 . . . −0.009549899 −0.012004385
php	−0.012267966 0.016383838 −0.012978291 . . . −0.0018558073 −0.009730892
id	−0.012249594 0.0065935757 −0.016826343 . . . 0.016014166 0.0075407173
13,410	0.01679577 −0.011711321 −0.00045675278 . . . 0.011311102 −0.013720703

Length truncation: Figure 10 shows the length statistics for the experimental data set URLs. Figure 10a shows that the length of malicious URLs is concentrated below 25 words in the experimental data set, and Figure 10b shows that the length of benign URLs is concentrated below 20 words. Therefore, in order to ensure the effective use of data, the cutoff length of URLs is set to 30 words, the URLs with less than 30 words are filled with 0s, and URLs with more than 30 words are cut to 30 words. Finally, the format of the input data is [64, 30, 50].



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

Figure 10. Length statistics of URLs.

4.3. Hyperparameters

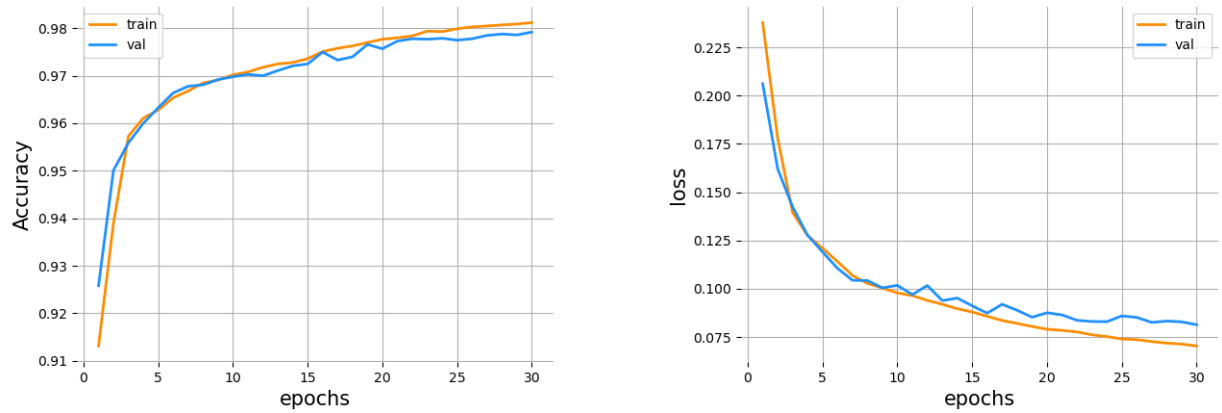
During the experiment, the hyperparameters are set as shown in Table 4.

Table 4. Hyper-parameter settings.

Hyper Parameter	
Maximum length of truncated URLs	30
Initial learning rate	0.0001
Total training times	30
Batch size	64
BiGRU layer dropout rate	0.2

4.4. Experimental Comparison

In the training process, the change in malicious URL recognition accuracy and loss value with the number of training iterations is shown in Figure 11, where orange represents the training curve and blue represents the test curve. It can be seen from Figure 11a that the accuracy rate increases rapidly at the beginning of training. After the fifth iteration, an inflection point begins to appear, and after the 15th iteration, the training curve begins to flatten. It can be seen in Figure 11b that the training loss curve and the validation loss curve begin to converge after the 15th iteration.



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

Figure 11. Accuracy curve and loss curve of the experimental model.

Figure 12 is the ROC curve of the experimental model. By observing the ROC curve, the area under curve (AUC) is close to 1, indicating that the DA-BiGRU model has a good classification effect.

MLP is the starting point for studying more complex deep learning methods. GRU is a variant of LSTM, which is simplified compared with the internal structure of LSTM, and the accuracy is also improved. Therefore, this paper selects four models of MLP, Att-BiLSTM, Att-BiGRU, and Dro-Att-BiLSTM for comparative testing to verify the classification effect of this model. Among them, the Att-BiLSTM model adds an attention mechanism to BiLSTM, the number of BiLSTM layers is set to 1 layer, and the number of nerve nodes is set to 128; the Att-BiGRU model adds an attention mechanism to BiGRU, the number of layers of BiGRU is set to 1 layer, and the number of neural nodes is set to 100; the Dro-Att-BiLSTM model adds a dropout mechanism to the Att-BiLSTM model, and the dropout rate is set to 0.2. The data of the comparison model are consistent with those of the model in this paper. The tanh function is used as the activation function and the softmax function is used for final classification.

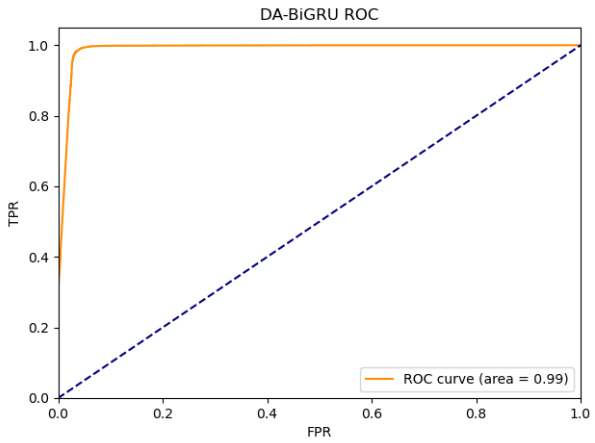


Figure 12. ROC curve of the experimental model.

In this paper, the DA-BiGRU model is compared with the other four models. The results of experimental comparison are shown in Figure 13.

Figure 13a is the loss function curve of the training set. It can be seen from the graph that the DA-BiGRU model curve is smoother, and the loss rate is the smallest when compared with the comparison models. Figure 13b is the loss function curve of the validation set, and the Dro-Att-BiLSTM model has the largest oscillation amplitude. Figure 13c is the precision curve of the validation set. It can be seen in the graph that the Att-BiGRU model and the DA-BiGRU model have higher accuracies. Although the curve of Att-BiGRU is higher than that of DA-BiGRU in the intermediate verification process, DA-BiGRU is greater than Att-BiGRU in the later verification, which shows that the fitting effect of the model is better after adding the dropout mechanism. Figure 13d is the accuracy curve of the validation set. It can be seen in the graph that the DA-BiGRU model in this paper achieves a good classification effect during the verification process, and the accuracy reaches 0.9792. Overall, the model curves based on BiLSTM are lower than those based on BiGRU, indicating that the GRU model is more suitable for classification tasks with a strong sequence. Figure 13e is the recall curve of the validation set. It can be seen from the graph that DA-BiGRU has a good malicious URL detection ability.

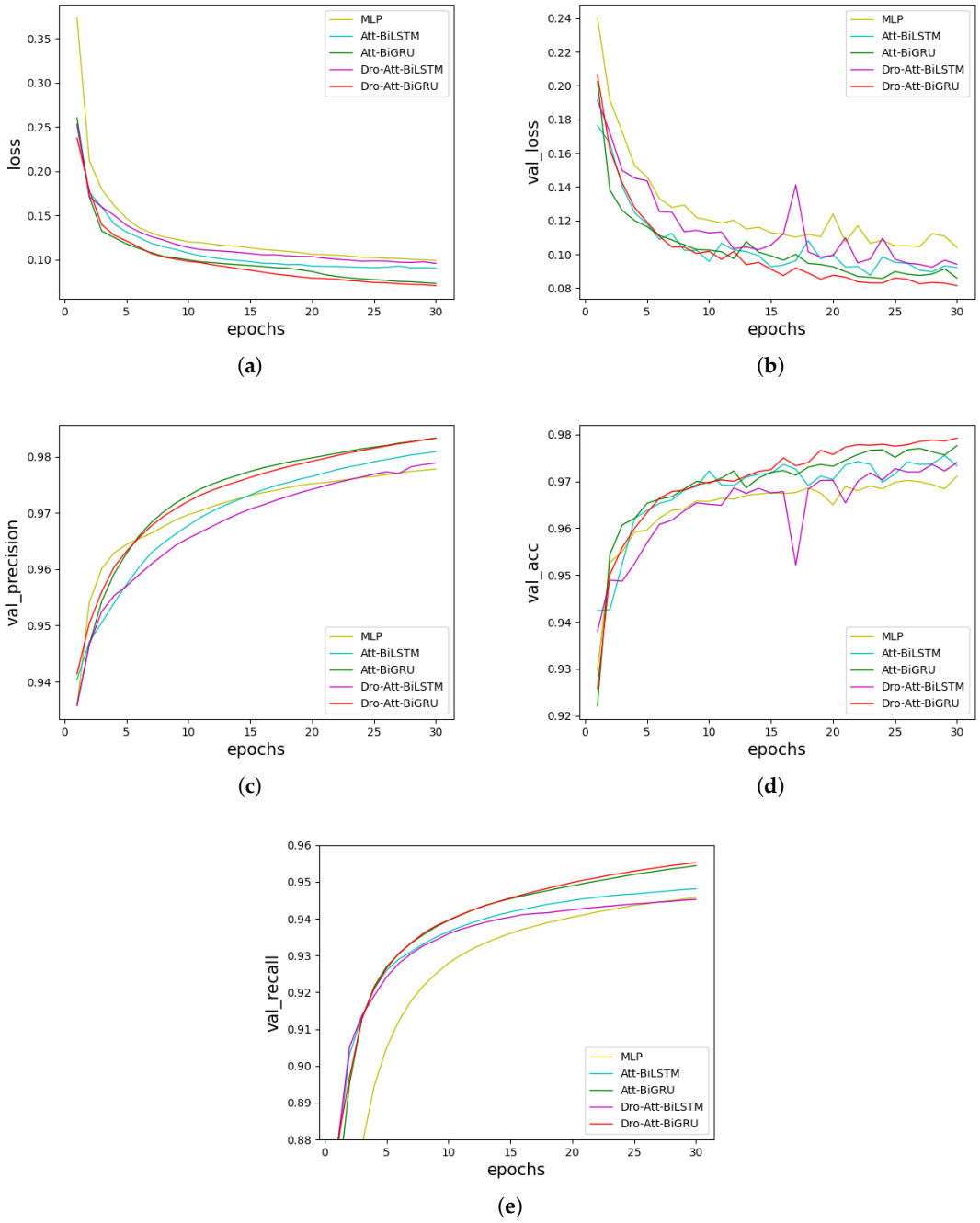


Figure 13. Comparison of experimental results of different methods.

The experimental results of different models on the test set are shown in Table 5.

Table 5. Comparison of results of different models on test sets.

	Loss	Accuracy	F1_Score	Precision	Recall
MLP	0.1041	0.9711	0.9616	0.9778	0.9458
Att-BiLSTM	0.0922	0.9733	0.9643	0.9809	0.9482
Att-BiGRU	0.0859	0.9776	0.9687	0.9833	0.9544
DA-BiLSTM	0.094	0.9740	0.9618	0.9790	0.9453
DA-BiGRU	**0.0814**	**0.9792**	**0.9691**	**0.9834**	**0.9553**

From Table 5, we can see that the DA-BiGRU model in this paper is better than other models in each result parameter on the test set, and the accuracy is 0.9792. The MLP

model is lower than other models in all parameters. This may be related to the MLP model itself, without taking into account the sequence of URLs and some deeper key information, so the classification effect is not good. By longitudinal comparison of the Att-BiLSTM, Att-BiGRU, DA-BiLSTM, and DA-BiGRU models, the model based on BiGRU is better than that based on BiLSTM, which shows that GRU is more suitable for URLs with strong sequences. Through the comparison between Att-BiGRU and DA-BiGRU, it is found that Att-BiGRU with dropout mechanism improves the fitting ability of the model, extracts useful features more accurately, and obtains the sequence information of the whole URLs, thereby improving the classification accuracy.

5. Conclusions

This paper proposes a malicious URL detection method based on a bidirectional GRU and an attention mechanism. Word2Vec is used to train the word vector of URLs, and a regularization mechanism is added to the input layer to reduce the overfitting of the model. BiGRU is used to extract all the sequence information of URLs, and an attention mechanism is introduced to learn the correlation between sequences to strengthen the learning of key useful information of the model. Through comparative experiments, the experimental results show that the proposed method can achieve good classification results in malicious URL detection tasks.

The analysis of URL word formation features essentially belongs to the field of NLP. The word vector based on Word2vec training and the BiGRU model with a regularization mechanism and an attention mechanism have proven to have a great positive effect in this paper. Therefore, the next research focus is to optimize the model so that it can be applied in real-life situations. At the same time, the application of these methods in other fields can also be investigated as a follow-up research direction.

Author Contributions: Validation, Y.X.; Writing—original draft, M.W.; Writing—review & editing, T.W.; Supervision, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Prakash, P.; Kumar, M.; Kompella, R.R.; Gupta, M. PhishNet: Predictive blacklisting to detect phishing attacks. In Proceedings of the 29th IEEE International Conference on Computer Communications, Honolulu, HI, USA, 3–6 August 2020; pp. 346–350.
2. Likarish, P.; Jung, E. *Leveraging Google Safe Browsing to Characterize Web-Based Attacks*; Association for Computing Machinery: New York, NY, USA, 2009.
3. Lin, H.L.; Li, Y.; Wang, W.; Yue, Y.; Lin, Z. Efficient segment pattern based method for malicious URL detection. *J. Commun.* **2015**, *36*, 141–148. [CrossRef]
4. Stoll, C.; Connolly, J.W.D. The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage. *Phys. Today* **1990**, *43*, 75–76. [CrossRef]
5. Project, T.H. *Know Your Enemy: Learning about Security Threats*; Addison-Wesley Professional: Boston, MA, USA, 2004.
6. Spitzner, L. Honeypot Farms. 2012. Available online: http://www.symantec.com/connect/articles/honeypot-farms (accessed on 2 November 2022).
7. Gulshan, K.; Rahul, S.; Mandeep, S.; Mritunjay, R.K. Optimized Packet Filtering Honeypot with Snooping Agents in Intrusion Detection System for WLAN. *Int. J. Inf. Secur. Priv.* **2018**, *12*, 53–62. [CrossRef]
8. Jianwei, Z.; Yong, T.; Xinhui, H.; Haixin, D. Advances in Research and Application of Honeypot Technology. *J. Softw.* **2013**, *24*, 825–842. [CrossRef]
9. Vanhoenshoven, F.; Nápoles, G.; Falcon, R.; Vanhoof, K.; Köppen, M. Detecting Malicious URLs Using Machine Learning Techniques. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, Athens, Greece, 6–9 December 2016.
10. Arivazhagi, A.; Kumar, S.R. An Efficient Stabbing Based Intrusion Detection Framework for Sensor Networks. *Comput. Syst. Sci. Eng.* **2022**, *43*, 141–157. [CrossRef]
11. Alshayeji, M.H.; AlSulaimi, M.; Jaffal, R. Network Intrusion Detection with Auto-Encoder and One-Class Support Vector Machine. *Int. J. Inf. Secur. Priv.* **2022**, *16*, 1–18. [CrossRef]
12. Azeez, N.A.; Salaudeen, B.B.; Misra, S.; Damaševičius, R.; Maskeliūnas, R. Identifying phishing attacks in communication networks using URL consistency features. *Int. J. Electron. Secur. Digit. Forensics* **2020**, *12*, 200–213. [CrossRef]

13. Laughter, A.; Omari, S.; Szczurek, P.; Perry, J. Detection of malicious http requests using header and url features. In Proceedings of the Future Technologies Conference, Vancouver, BC, Canada, 5–6 November 2020; pp. 446–449.
14. Zhang, H.; Qian, L.; Wang, L.; Yuan, C.; Zhang, T. Malicious URLs detection based on CNN and multi-classifier. *Comput. Eng. Des.* **2019**, *40*, 2991–2995, 3019.
15. Eunaicy, J.J.C.; Suguna, S. Web attack detection using deep learning models. *Mater. Today Proc.* **2022**, *62*, 4806–4813. [CrossRef]
16. Afzal, S.; Asim, M.; Javed, A.R.; Beg, M.O.; Baker, T. URLdeepdetect: A deep learning approach for detecting malicious URLs using semantic vector models. *J. Netw. Syst. Manag.* **2021**, *29*, 21. [CrossRef]
17. Das, A.; Das, A.; Datta, A.; Si, S.; Barman, S. Deep approaches on malicious URL classification. In Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1–3 July 2020; pp. 1–6.
18. Cui, Y.; Liu, M.; Hu, J. Cnn-based malicious web request detection technology. *Comput. Sci.* **2020**, *47*, 281–286
19. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
20. Trueman, T.E.; Kumar, A.; Narayanasamy, P.; Vidya, J. Attention-based C-BiLSTM for fake news detection. *Appl. Soft Comput.* **2021**, *110*, 107600. [CrossRef]
21. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
22. Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.