Postprint: Gated Memory Network Method for Network Attack Detection

Authors: Wang Jiabao, Xu Weiguang, Zhou Zhenji, Li Yang, Miao Zhuang

Date: 2018-05-24T00:00:00+00:00

Abstract

To address the challenge of large-scale network attack detection on the Internet, this paper proposes a gated memory network detection method that combines word vector feature representation with recurrent neural networks. The method first converts network request data into a low-dimensional real-valued vector sequence representation, then leverages the long-term memory capability of gated recurrent neural networks to extract features from the request data, and finally employs a logistic regression classifier to achieve automatic detection of network attacks. On the CSIC2010 public dataset, the proposed method achieves a 98.5% F1 score under 10-fold cross-validation. Compared with traditional methods, it substantially improves both the precision and recall of network attack detection. The proposed method enables automatic detection of network attacks and exhibits favorable detection performance.

Full Text

Gated Memory Network Approach for Web Attack Detection

Wang Jiabao, Xu Weiguang, Zhou Zhenji, Li Yang, Miao Zhuang
Army Engineering University of PLA, Nanjing 210007, China

Abstract

To address the challenge of large-scale network attack detection on the Internet, this paper proposes a gated memory network method that combines word vector feature representation with recurrent neural networks. The method first transforms network request data into a low-dimensional real-valued vector sequence representation, then extracts features from the request data using the long-term memory capabilities of a gated recurrent neural network, and finally implements automatic detection of network attacks using a logistic regression

classifier. On the CSIC2010 public dataset, the method achieves a 98.5% F1-score under 10-fold cross-validation. Compared with traditional methods, it significantly improves both the precision and recall rates for network attack detection. The proposed method can automatically detect network attacks and demonstrates good detection performance.

Key words: network attack detection; low-dimension real-value vector representation; gated recurrent neural networks

0 Introduction

As the scale of the Internet continues to expand, network attacks have become a major issue facing national security departments and enterprises worldwide. Due to the diverse and constantly evolving nature of attack methods, detecting network attacks presents enormous challenges. For large-scale network communication behaviors on the Internet, if we can automatically determine the maliciousness of communication activities, we can effectively prevent the damage caused by network attacks and potential secondary disasters. Currently, network attack detection methods are primarily divided into two categories: pattern matching-based detection and pattern classification-based detection [**?**]. The former represents the main approach used by most network security software today, which identifies abnormal behaviors in system or network logs—such as logging in from unexpected locations, accessing unauthorized files, abnormal network traffic, or anomalous program behavior—through pattern matching or statistical analysis [**?**]. These detection methods typically occur during or after an attack has taken place, making it impossible to predict and prevent attacks beforehand. The latter approach typically extracts features from communication content [**?**] and then uses classifiers for direct detection. This type of method can operate directly on network traffic data, processing or discarding anomalous communication data as it is discovered, thereby preventing attacks from impacting host systems.

Pattern classification-based detection is the focus of this research. Currently, this approach primarily borrows from text classification techniques by extracting descriptive features from content to transform the network attack detection problem into a pattern classification problem. Feature representation methods mainly include Bag-of-Words [**?**], TF-IDF [**?**], and n-gram representations [**?**], while classification detection methods encompass classical machine learning approaches such as Bayesian [**?**], decision tree [**?**], support vector machine [**?**, **?**], multilayer perceptron [**?**, **?**], and K-nearest neighbor [**?**].

In recent years, with the development of deep learning, both feature representation and classification research have achieved new progress. For text feature representation, Google' s research team proposed the word2vec model, which can transform traditional high-dimensional one-hot word vector representations into low-dimensional real-valued vector representations, changing how textual words are expressed and enabling distance (similarity) metrics between words

[?, ?]. This feature representation has been successfully applied in sentiment classification [?], though primarily limited to research related to textual word content. For network communications, protocol-based communication requests consist of string streams that can be converted into text word sequences for low-dimensional real-valued vector sequence representation. Meanwhile, for sequential data, recurrent neural networks (RNN) and gated recurrent neural networks demonstrate excellent modeling capabilities and have been widely applied in text classification [?], intrusion detection [?], and machine translation [?].

Inspired by low-dimensional real-valued word vector models [?, ?] and gated recurrent network models [?, ?], this paper proposes a gated memory network method for network attack detection. In this approach, HTTP network communication request content is represented as low-dimensional real-valued vector sequences, enhancing data feature representation capabilities. Simultaneously, a gated recurrent neural network model with long-term memory capabilities is used to model long-interval word relationships in sequences, effectively improving classification detection performance. Compared with traditional methods, the proposed gated memory network method significantly improves the precision and recall rates for network attack detection on the CSIC2010 public dataset.

1 Methodology

1.1 Framework of the Gated Memory Network Method

Figure 1 [Figure 1: see original paper] illustrates the flow framework of the proposed network attack detection method based on gated memory networks. In this framework, an attacker sends attack requests through Web protocols, and these request data serve as the raw data for network attack detection. The process involves first capturing and parsing packets from network switches; the parsed data then undergoes low-dimensional real-valued feature representation to become input for the gated memory network model; finally, a pre-trained gated memory network model predicts unknown input data to determine whether it is an attack request or a normal request. The parameters of the gated memory network model are learned in advance from training data that includes both normal and attack samples. After low-dimensional real-valued feature representation, the data is input to the model for training. The core components of this framework are the low-dimensional real-valued feature representation and the gated memory network model.

1.2 Low-Dimensional Real-Valued Feature Representation

Network attacks depend on network communications and typically appear as abnormal communication requests. Since communication requests can be viewed as command strings, and abnormal network communication request data often contains special command strings (such as “systemInfo” , “alert” , “SELECT” , etc.), network attacks can be detected by classifying the strings contained in

request data.

Figure 2 [Figure 2: see original paper] shows the low-dimensional real-valued feature representation process for HTTP protocol attack data, which mainly includes three steps: word tokenization, quantization, and dimensionality reduction.

1) Word Tokenization: For communication request strings, the content is constructed according to network protocol rules and thus follows certain formatting conventions that can be converted into text consisting of a series of "words" or symbols. For example, a large portion of access to web servers is based on HTTP protocol, and the parameter strings submitted in the payload portion of request data can be viewed as several segments separated by the symbol "&" , with each segment containing key and value parts connected by "=" . Therefore, request strings can be segmented into sequences of "words" or symbols, even if these "words" or symbols have no literal meaning. This tokenization provides the foundation for subsequent low-dimensional real-valued feature representation.

2) Quantization: After word tokenization, traditional text classification feature representation methods typically represent each word as a one-hot vector. A one-hot vector is a binary vector where the element at position i is set to 1 for the i-th word in the vocabulary, and all other elements are set to 0, uniquely representing a word. For a sequence consisting of L words, it can be represented as a one-hot sequence of length L. However, one-hot vectors are sparse high-dimensional binary vectors that are computationally expensive and make it difficult to measure distance (similarity) relationships between two words, such as synonyms with similar semantics.

3) Dimensionality Reduction: To overcome the limitations of one-hot vector representation, this paper projects a one-hot vector x_i into a low-dimensional real-valued vector $z_i \in \mathbb{R}^d$ (where d is the space dimension). This projection can be achieved by left-multiplying vector x_i with a projection matrix $M \in \mathbb{R}^{d \times |V|}$ (Equation (1)), where $|V|$ is the number of unique words in the dictionary.

$$z_i = Mx_i$$

The matrix M can be obtained through random assignment or learned through a network containing a hidden layer. Experiments show that the learned matrix M provides better low-dimensional representation. The network for learning matrix M takes a one-hot word vector as input and outputs the next one-hot word vector adjacent to it, thereby learning the relationship between two co-occurring words. After network training, the hidden output serves as the low-dimensional real-valued vector z_i. This dimensionality reduction representation transforms a high-dimensional sparse one-hot sequence $(x_1, x_2, ..., x_L)$ into a low-dimensional real-valued vector sequence $(z_1, z_2, ..., z_L)$ for input into the model.

1.3 Gated Memory Network Model

The gated memory network model consists of a gated recurrent neural network and a logistic regression classifier. The gated recurrent neural network models sequential data to extract long-term memory feature representations, while the logistic regression classifier completes the binary classification prediction.

1.3.1 Gated Recurrent Neural Network In recent years, recurrent neural networks have achieved superior results in speech recognition. Their structure is primarily composed of a recurrent process:

$$h_t = \varphi(W_h z_t + R_h h_{t-1} + b_h)$$

where the activation h_t at time t is determined by the input z_t at time t and the activation h_{t-1} at time $t-1$, as shown on the left side of Figure 3 [Figure 3: see original paper]. Recurrent neural networks enable long-term memory of information through iterative computation and exhibit excellent modeling capabilities for sequential data. However, classical recurrent neural networks face the gradient vanishing problem during model training [**?**]. To overcome this issue, the Long Short-Term Memory (LSTM) unit structure [**?**, **?**] was introduced into recurrent neural networks. This structure consists of gate units such as input gates, forget gates, and output gates to maintain information memory for long sequences. Since the network comprises a series of gate units, it is also called a gated recurrent neural network.

Inspired by gated recurrent neural networks [**?**, **?**], the gated memory network unit structure for network attack detection is shown on the right side of Figure 3 [Figure 3: see original paper]. Here, z_t is the input vector of low-dimensional real-valued features; h_{t-1} is the hidden state of the network at time $t-1$; h_t is the hidden state output by the network at time t; and h_t serves not only as input for time t but also as input for calculating the final output result. In the dashed box, arrows indicate data flow; $\odot$ represents element-wise multiplication; $\oplus$ represents vector addition; σ represents the sigmoid activation operation; and tanh represents the tanh activation operation.

The gated memory network unit structure shown in Figure 3 is primarily composed of a reset gate and an update gate, formalized as follows:

$$r_t = \sigma(W_r z_t + R_r h_{t-1} + b_r)$$
$$u_t = \sigma(W_u z_t + R_u h_{t-1} + b_u)$$

where W_r, W_u and R_r, R_u are weights for input z_t and hidden state h_{t-1}, respectively; b_r, b_u are corresponding biases; and $\sigma(\cdot)$ is the sigmoid function. The final output is:

$$\tilde{h}_t = \tanh(W_h z_t + r_t \odot (R_h h_{t-1}) + b_h)$$
$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t$$

where $\odot$ denotes element-wise multiplication; tanh is the tanh function; and W_h, R_h are weights for input and hidden states, with b_h as the corresponding bias. The calculation result of $\tilde{h}_t$ is controlled by the reset gate. If $r_t = 0$, the historical information h_{t-1} at time $t-1$ does not affect the input z_t at time t, meaning the hidden information h_{t-1} is forgotten and reset by the input information z_t at time t. The final output h_t is obtained by controlling the addition of the hidden information $\tilde{h}_t$ at time t and the hidden information h_{t-1} at time $t-1$.

Compared with LSTM, the gated memory network unit structure integrates the input gate and forget gate into an update gate to balance the activation h_{t-1} at time $t-1$ and the updated activation $\tilde{h}_t$ at time t, while the reset gate determines whether to forget the activation h_{t-1} at time $t-1$ through element-wise multiplication. The final output for sequential data is the hidden state h_L with long-term memory characteristics, which serves as input to the classifier.

1.3.2 Logistic Regression Classifier For the binary classification problem of network attack detection, the input is network protocol request data, which is converted through low-dimensional real-valued feature representation into a vector sequence of length L, $(z_1, z_2, ..., z_L)$. This sequence is then processed by the gated recurrent neural network to extract long-term memory features h_L, which finally pass through a binary classifier for classification output.

This paper adopts the logistic regression classifier. During training, given a set of binary labeled samples $\{(h_L^{(i)}, y^{(i)}) : i = 1, ..., N\}$, where $h_L^{(i)}$ is the feature vector of the i-th sample and $y^{(i)} \in \{1, 0\}$ is its corresponding true class label (1 indicates an attack sample, 0 indicates a normal sample), and N is the number of training samples. According to the principle of maximum likelihood estimation, the likelihood of all samples is calculated as:

$$\mathcal{L}(\theta) = \prod_{i=1}^{N} p(y^{(i)}|h_L^{(i)}) = \prod_{i=1}^{N} \left(\frac{1}{1 + \exp(-\theta^T h_L^{(i)})} \right)^{y^{(i)}} \left(1 - \frac{1}{1 + \exp(-\theta^T h_L^{(i)})} \right)^{1-y^{(i)}}$$

where θ is the parameter vector to be learned. For convenience in optimization, maximizing $\mathcal{L}(\theta)$ is transformed into minimizing the negative log-likelihood:

$$J(\theta) = -\log \mathcal{L}(\theta) = -\sum_{i=1}^{N} \left[y^{(i)} \log \left(\frac{1}{1 + \exp(-\theta^T h_L^{(i)})} \right) + (1 - y^{(i)}) \log \left(1 - \frac{1}{1 + \exp(-\theta^T h_L^{(i)})} \right) \right]$$

For a single training sample, only one term in the summation of Equation (7) is non-zero (depending on whether the label $y^{(i)}$ is 1 or 0). When $y^{(i)} = 1$, minimizing the optimization objective requires making $\frac{1}{1+\exp(-\theta^T h_L^{(i)})}$ approach 1, which means making $\theta^T h_L^{(i)}$ large.

In practice, the logistic regression classifier can be connected to the gated memory network for joint training. The hidden state h_L output by the gated memory network is passed through a fully connected layer to compute the output o, which is then connected to a sigmoid layer to implement the calculation of Equation (6). Finally, the loss layer computes the result of Equation (7). During optimization, the stochastic gradient descent algorithm is used for parameter updates. Partial derivatives are first calculated from back to front, and then the back-propagation algorithm is used to sequentially compute the partial derivatives of each layer to achieve network parameter learning.

During testing, the loss function is replaced with a logistic function $\sigma(\theta^T h_L^{(i)}) = \frac{1}{1+\exp(-\theta^T h_L^{(i)})}$. To predict whether a new sample belongs to class "1" or "0", we can compare the magnitude of $\sigma(\theta^T h_L^{(i)})$ with 0.5. If $\sigma(\theta^T h_L^{(i)}) > 0.5$, it is classified as "1"; otherwise, it is classified as "0".

1.4 Implementation Details of the Gated Memory Network Method

For the CSIC2010 dataset, communication request data is first extracted. For GET requests, URI information is directly extracted; for PUT and POST requests, URI and payload data are extracted and concatenated as input for low-dimensional real-valued feature representation, which converts them into a word vector sequence. In specific implementation, the word vector sequence consists of a word index sequence and a mapping matrix from word indices to word vectors. Each index in the word index sequence corresponds to a pre-trained word vector, which can be found in the mapping matrix from word index to word vector. This representation significantly saves space. The mapping from word index to word vector is implemented by a network embedding layer. Notably, word vector sequences are uniformly truncated or padded to a length of 56 based on the overall characteristics of the dataset, enabling batch input of data for training.

The gated memory network method adopts a deep learning architecture, with its specific framework shown in Table 1 .

Table 1: Framework of the Gated Memory Network Method

Layer	Description
Input Layer	Inputs word index sequence and outputs it without processing

Layer	Description
Embedding Layer	Inputs word index sequence and mapping matrix from word index to word vector, converting word indices to word vector outputs
Gated Memory Layer	Inputs word vector sequence, using the last hidden state calculated by gated memory units as output
Dropout Layer	Inputs hidden state and outputs it after dropping weights with a certain probability
Fully Connected Layer	Maps to two values through fully connected layers
Softmax Layer	Outputs two values normalized as two probability values

In the table, the input layer and embedding layer complete the low-dimensional real-valued feature representation; the Dropout layer is a primary means of suppressing overfitting in deep learning, and using this strategy can achieve better test accuracy; the fully connected layer and Softmax layer implement logistic regression, corresponding to the calculation of $\theta^T h_L$ in Equation (6). The fully connected layer completes the linear mapping calculation, and Softmax normalizes the two mapped values to obtain probabilities (p_0, p_1). During training, the network computes loss and performs gradient backpropagation and parameter updates; during testing, the network computes the F1-score to determine class labels.

Based on the Windows 7 operating system (3.5 GHz CPU, 8 GB memory) and TensorFlow platform, the gated memory network method is implemented. All training and testing are completed using CPU. The network input is a sequence of d-dimensional vectors. The hidden state dimension of the single-layer gated memory unit is 128, followed by a Dropout layer with a dropout rate of 0.9. The 128-dimensional output is then fully connected and mapped to a 2-dimensional result output. During network training, the batch size is set to 128, the learning rate is set to 0.001, the number of training epochs is 10, and the Adam optimizer is used for training.

2 Experimental Results and Analysis

The CSIC2010 dataset contains tens of thousands of automatically generated HTTP protocol requests, primarily used for testing network attack protection systems. It was created by the Information Security Institute of the Spanish National Research Council (CSIC). The dataset targets an e-commerce Web application, with published data divided into training (normal only) and testing (both anomalous and normal) sets. In our experiments, we use over 36,000 normal requests and over 25,000 anomalous requests from the test set. The

attack requests in this dataset include various network attacks such as SQL injection, buffer overflow, information gathering, file disclosure, CRLF injection, cross-site scripting, and parameter tampering. Requests for hidden (or unavailable) resources are also considered anomalous [**?**, **?**]. Currently, due to privacy protection and other reasons, very few publicly available datasets exist for Web attack detection problems. Many attacks in datasets like DARPA KDD99 are outdated and do not include many new types of attacks.

For the raw request data, the experiments mainly extract GET, POST, and PUT request data for detection. After extraction, the data undergoes tokenization (string segmentation) based on HTTP request characteristics, primarily involving URL character decoding, separation of parameter items, key-value pairs, and special symbols. Tokenizing request data provides the foundation for subsequent low-dimensional real-valued feature representation.

2.1 Method Comparison

To verify the effectiveness of the proposed method, we conduct comparative analysis with both traditional methods and deep learning approaches. Traditional methods include Naïve Bayes (NB), Linear Support Vector Machine (LSVM), Neural Network (NN), K-Nearest Neighbor (KNN), and Decision Tree (DT). Deep learning methods include LSTM.

For feature representation, traditional methods uniformly adopt TF-IDF feature vectors. Deep learning methods use the low-dimensional data representation described in Section 1.2. For classification detection, traditional method parameters are set as follows: NB with constant parameter 0.01; LSVM using a linear classifier; NN with two hidden layers of 50 and 10 units respectively; KNN with neighbor parameter set to 3; DT using minimum entropy with minimum samples in leaf set to 3.

All methods are evaluated using 10-fold cross-validation. The dataset is randomly divided into 10 parts, with one part used for testing and the rest for training. Results from 10 runs are averaged to obtain overall performance. Test results are evaluated using accuracy, recall, and F1-score as metrics. Experimental results are shown in Table 2 .

Table 2: Experimental Training and Test Data

Method	Precision (%)	Recall (%)	F1-Score (%)
NB	85.20	84.30	84.70
LSVM	89.40	89.10	89.20
NN	90.10	90.20	90.10
KNN	87.60	87.90	87.70
DT	86.50	85.80	86.10
LSTM	97.70	97.90	97.80
Gated Memory Network	**98.40**	**98.50**	**98.40**

As shown in Table 2, the gated memory network method significantly outperforms other methods, achieving accuracy and recall rates of 98.4% and 98.5% respectively–0.7% higher in accuracy and 0.6% higher in recall compared to LSTM, and far exceeding traditional methods. These results demonstrate that the proposed method has excellent attack detection effectiveness.

2.2 Hidden Variable Parameter Analysis

In the gated memory network, the hidden state dimension is a key parameter affecting network capability. To further analyze how this parameter variation impacts performance, we fix the Dropout parameter at 0.9 and the low-dimensional real-valued word vector dimension at 40, then test the comprehensive F1-score performance of the model detection method under different dimensions. To further demonstrate the effectiveness of our method, we also compare the F1-scores of the LSTM method under different parameter conditions. Experimental results are shown in Figure 4 [Figure 4: see original paper]. Additionally, model training time under different parameter conditions is shown in Table 3 .

Table 3: Model Training Time (minutes)

Method	32	64	128	256	512
LSTM	45	89	178	356	712
Gated Memory Network	38	75	150	300	600

From Figure 4 and Table 3, we can draw the following conclusions:

a) Model performance continuously improves as parameter dimension increases, but the rate of improvement gradually diminishes;
b) Compared with the LSTM model, our method achieves better results under the same parameter conditions;
c) Model computation time increases with dimension, so practical applications require a trade-off between effectiveness and performance.

2.3 Dropout Strategy Analysis

Given the moderate scale of the CSIC2010 dataset, the Dropout strategy is incorporated during model training to suppress overfitting. The Dropout rate in this strategy determines model effectiveness. Therefore, we investigate the impact of different Dropout rates under fixed hidden variable dimension of 128 and low-dimensional real-valued word vector dimension of 40. Figure 5 [Figure 5: see original paper] shows the results for different Dropout rates.

From Figure 5, we can observe:

a) Model generalization performance continuously improves as Dropout rate increases, but the rate of improvement gradually diminishes;

b) The Dropout strategy (with larger Dropout rates) effectively suppresses overfitting and achieves good performance. At a Dropout rate of 0.9, the F1-score reaches 98.40%.

Low-dimensional real-valued word vectors have excellent feature representation capabilities and can measure similarity relationships between different words. Words with similar relationships or identical attributes are closer in distance in vector space and tend to cluster together. To illustrate the representation capability of low-dimensional real-valued word vectors, we select some vectors and embed them into a two-dimensional plane using the t-SNE method [**?**] to display relationships between words, as shown in Figure 6 [Figure 6: see original paper].

Figure 6 shows the embedding display of 13 low-dimensional real-valued word vectors. Among them, "select" , "like" , "script" , "alert" , and "waitfor" are keywords for SQL injection attacks. These words all cluster together in the embedding space, while normal words such as "login" and "password" show random distribution characteristics. This clustering of similar words makes it easier for the classifier to learn patterns of network attack behavior.

Additionally, different low-dimensional real-valued word vector dimensions also affect model accuracy. To study the impact of word vector dimension on algorithm performance, we select dimensions of 10, 20, 40, 80, 160, and 320, measuring both attack detection effectiveness and time cost. Figure 7 [Figure 7: see original paper] shows the corresponding test results.

From Figure 7, we can see:

a) As the dimension of low-dimensional real-valued word vectors increases, the detection effectiveness of our method also continuously improves, reaching an F1-score of 99.10% at dimension 160;
b) Meanwhile, the time cost of our method increases linearly with the dimension of low-dimensional real-valued word vectors;
c) In practical applications, a certain dimension of low-dimensional real-valued word vectors can be selected based on application requirements to achieve a trade-off between speed and effectiveness.

3 Conclusion

To address the problem of network attack detection, this paper proposes a gated memory network method that achieves an F1-score of 98.40% on the CSIC2010 dataset, surpassing both traditional methods and LSTM-based detection approaches. The method is simple to implement and highly effective, while also achieving real-time detection speeds. However, since learning low-dimensional data representations typically requires a certain scale of data support, our method needs a large volume of network request data for training.

Additionally, our method currently employs only a single-layer gated recurrent neural network structure. Future work will explore multi-layer network struc-

tures to further improve detection accuracy. Considering the time cost of multi-layer networks, two or three-layer structures can be constructed to trade off between detection precision and time consumption. Furthermore, to validate the effectiveness of our method, subsequent testing will be conducted in real network environments to verify practical applicability.

References

[1] Elbachirelmoussaid N, Toumanari A. Web application attacks detection: a survey and classification [J]. International Journal of Computer Applications, 2014, 103 (12): 1-6.

[2] Xu Zhoubo, Zhang Yongchao, Gu Tianlong, et al. Research on pattern matching algorithm in intrusion detection system [J]. Computer Science, 2017, 44 (9): 125-130.

[3] Dai Yuanfei, Chen Xing, Chen Hong, et al. Feature selection based approach to network intrusion detection [J]. Application Research of Computers, 2017, 34 (8): 2429-2433.

[4] Ma Linjin, Wan Liang, Ma Shaoju, et al. Distributed denial of service attack recognition based on bag of words model [J]. Journal of Computer Application, 2017, 37 (6): 1644-1649.

[5] Mao Chinghao, Lee H M, Liu Ensi, et al. Web mimicry attacks detection using HTTP token causal correlation [J]. International Journal of Innovative Computing Information & Control, 2011, 7 (7): 4347-4362.

[6] Oza A, Ross K, Low R M, et al. HTTP attack detection using n-gram analysis [J]. Computers & Security, 2014, 45 (3): 242-254.

[7] Singh K J, De T. An approach of DDOS attack detection using classifiers [M]// Emerging Research in Computing, Information, Communication and Applications. New Delhi: Springer, 2015: 429-437.

[8] García V H, Monroy R, Quintana M. Web attack detection using ID3 [M]// Professional Practice in Artificial Intelligence. Boston, MA: Springer, 2006: 315-324.

[9] Rawat R, Kumar Shrivastav S. SQL injection attack detection using SVM [J]. International Journal of Computer Applications, 2012, 42 (13): 1-4.

[10] Wu Shaohua, Cheng Shubao, Hu Yong. Web attack detection method based on support vector machines [J]. Computer Science, 2015, 42 (6A): 362-364.

[11] Silva L D S, Santos A C F D, Silva J D S D, et al. A neural network application for attack detection in computer networks [C]// Proc of IEEE International Joint Conference on Neural Networks. Budapest, Hungary: IEEE Press, 2004: 1569-1574.

[12] Li Jin, Liu Yong, Gu Lin. DDoS attack detection based on neural network [C]// Proc of IEEE International Symposium on Aware Computing. [S. l.]: IEEE Press, 2010: 196-199.

[13] Xiao Fu, Ma Junqing, Huang Xunsong, et al. DDoS attack detection based on KNN in software defined networks [J]. Journal of Nanjing University of Posts and Telecommunications: Natural Science Edition, 2015, 35 (1): 84-88.

[14] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space [J/OL]. CoRR, 2013, abs/1301.3781. https://arxiv.org/abs/1301.3781.

[15] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality [C]// Proc of 27th Annual Conference on Neural Information Processing Systems. Lake Tahoe, Nevada: Curran Associates, Inc, 2013: 3111-3119.

[16] Nowak J, Taspinar A, Scherer R. LSTM recurrent neural networks for short text and sentiment classification [C]// Proc of International Conference on Artificial Intelligence and Soft Computing. Cham: Springer, 2017: 553-562.

[17] Liu Pengfei, Qiu Xipeng, Huang Xuanjing. Recurrent neural network for text classification with multi-task learning [C]// Proc of International Joint Conference on Artificial Intelligence. New York, NY: IJCAI/AAAI Press, 2016: 2873-2879.

[18] Kim J, Kim J, Thu H L T, et al. Long short term memory recurrent neural network classifier for intrusion detection [C]// Proc of IEEE International Conference on Platform Technology and Service. Jeju, South Korea: IEEE Press, 2016: 1-5.

[19] Cho K, Merrienboer B V, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation [J/OL]. CoRR, 2014, abs/1406.1078. https://arxiv.org/abs/1406.1078.

[20] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling [J/OL]. CoRR, 2014, abs/1412.3555. https://arxiv.org/abs/1412.3555.

[21] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks [C]// Proc of International Conference on Machine Learning. Altanta, Georgia: PMLR, 2013: 1310-1318.

[22] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Computation, 1997, 9 (8): 1735-1780.

[23] Maaten L V D, Hinton G. Visualizing data using t-SNE [J]. Journal of Machine Learning Research, 2008, 9 (2605): 2579-2605.

Note: Figure translations are in progress. See original paper for figures.