2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00133

Phishing Web Page Detection with HTML-Level Graph Neural Network

Linshu Ouyang and Yongzheng Zhang*
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
Email: {ouyanglinshu, zhangyongzheng}@iie.ac.cn

***Abstract*—Phishing web page is one of the most serious threats to the users of the Internet. Traditional phishing web page detection methods rely on manually designed features. Recently, deep learning-based methods using HTML as input have achieved significant detection performance improvement. They usually treat HTML codes as sequences of characters and utilize Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) for classification. However, CNN and RNN typically can only extract local features in the HTML code sequences while failing to model the long-range semantics that is crucial for phishing detection. In this paper, we propose a novel Graph Neural Network (GNN) based phishing web page detection method that can effectively utilize the inherent structural information of HTML to capture the long-range semantics. We first naturally represent an HTML as a graph according to its Document Object Model (DOM) and utilize RNN to extract the local features of node attributes. Then we adopt GNN to model the long-range relations between nodes based on these local features and the graph structure. Our proposed model combines the advantage of RNN and GNN to better understand the intention of HTML codes. Extensive experiments on a real-world dataset demonstrate that the accuracy of our method outperforms other state-of-the-art methods by a large margin.**

I. Introduction

Phishing is a kind of social engineering attack generally defined as stealing sensitive information from the target by impersonating trusted entities [1]. Although there exist several types of phishing, most attackers choose to set up fraudulent web pages to trick the victims [2], [3]. It continues to be one of the most popular network attack methods due to its low cost and high success rate [4].

The black-list and rule-based approaches are the most common and widely deployed phishing web page detection methods. The precision of them are high, but they usually suffer from limit abilities to detect new phishing variants since it's easy for attackers to make minor modifications of the phishing web page to escape from these methods.

To tackle these problems, plenty of machine learning-based methods have been proposed. They usually design a variety of features, then utilize classifiers, such as support vector machine, random forest, etc., to conduct classification [5], [6]. The features used in these methods can be categorized into three groups: URL, HTML, and other external information such as domain owner [7]–[9]. Using URL as input requires relatively less computation resource, at the cost of lower accuracy. Using HTML as input will significantly improve the accuracy consuming more computation resources. Using external information can further improve the accuracy, but will usually require time-consuming network communication with third party services. In this work, we focus on using HTML as input.

In recent years, several deep learning-based phishing web page detection methods have been proposed. The initial attempts focus on using URL as the input. Wang et al. [10] propose to use an improved recurrent convolutional network to better learn the local features with CNN and sequence information with RNN. Huang et al. [11] propose to use CNN and attention-based hierarchical RNN to focus on the important areas of the URL. Recently, Opara et al. [3] apply the deep learning method to the HTML input. They combine a character-level CNN and a word-level CNN to better classify the HTML.

While the above approach has achieved significant improvement, it is far from optimal to treat the HTML simply as sequences of characters or words. As is shown in Fig. 1, the co-occurrence of <form> and its child <input> implies that this is a login page. These correlations between tags are crucial in classifying the phishing web pages. However, such tag pairs may be far away in the raw HTML sequence, and it is difficult for the CNN or RNN to capture these long-range semantics. This inspired us to make use of the inherent tree structure information of HTML.

In this paper, we propose a novel graph neural network-based phishing web page detection method to solve the above problems. We leverage the inherent DOM tree structure of HTML to naturally represent it as a graph. Then we extract the local features of each node with RNN and distributed representations. Next, we utilize the graph neural network [12] to effectively capture the aforementioned long-range semantics between each pair of nodes. Finally, we obtain the graph level representation and classify it with a fully connected layer. With these designs, our proposed approach achieves significant accuracy improvement by combining the advantages of RNN in extracting local features and the advantages of GNN in capturing long-range semantics.

To summarize, our main contributions are as follows:

- We propose a novel graph neural network-based phishing web page detection method. Our proposed method

* Corresponding author

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00133

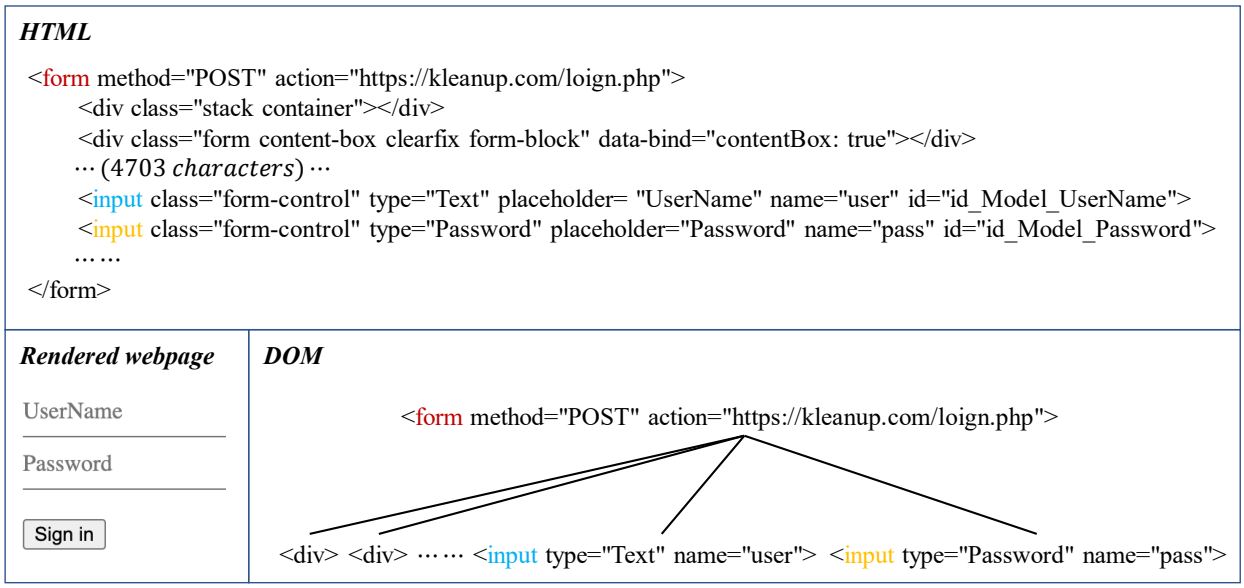

Fig. 1. A form in a phishing web page that collects information from the victims. The attributes of the <input> tag indicate that this form aims to collect user passwords. The attributes of the <form> tag indicate that this form sends information to a suspicious absolute URL. The co-occurrence of these two tags is a strong indication that this is a phishing web page. These two tags are naturally connected in the DOM but are far away in the raw HTML sequence.

effectively utilize the inherent tree structure of HTML to improve detection accuracy by combining the advantages of RNN in extracting local features and the advantages of GNN in capturing long-range semantics. To the best of our knowledge, this is the first study to model the HTML codes as graphs and detect phishing web pages by classifying the graphs using a graph neural network.

- We conduct extensive experiments on the real-world dataset to compare our method with other state-of-the-art phishing web page detection methods. Experimental results show the superiority of our method in terms of accuracy.

II. Related Work

The method we proposed is conceptually related to previous phishing web page detection methods, and graph neural network.

A. Traditional phishing web page detection

Traditional phishing web page detection methods can be broadly divided into three categories.

The first group is rule-based approaches. The black-list/whitelist methods rely on a constantly maintained list of known phishing web pages [13]. The heuristic-based approaches [14], [15] rely on experts designed rules to detect phishing web pages. These approaches are simple, precise, and fast, but have low recall and cannot detect new phishing attacks [16].

The second group is visual similarity-based approaches [17], [18], which detect phishing web pages by examining the visual similarity between phishing web pages and well-known non-phishing web pages. These methods can detect new phishing web pages, but at cost of high computation. Besides, they can only detect the phishing web pages that impersonate well-known targets.

The final group is machine learning-based approaches [7], [19], [20] . These methods typically rely on features designed by experts [4]–[6] and classification algorithms such as support vector machine [21], random forest [22], etc. This group of methods can detect new phishing attacks, but the performance is limited by the expressive power of manually designed features.

B. Deep learning based phishing web page detection

Recently, with the advancement of deep learning, several phishing web page detection methods based on deep text classification models have been proposed. They typically use URL as input, and there is quite limited work using HTML as the input.

Since the URL contains little structural information, most existing methods regarded it as sequences of characters and adopt natural language text classification model, including CNN [23], RNN [24], Attention [25], etc.

Wang et al. [10] proposed PDRCNN, an improved deep learning phishing web page detection method that uses URL as input only. Their method treats URL as sequences of characters and utilize a fused neural network architecture that combines recurrent neural network and convolutional neural network.

Huang et al. [11] proposed to use attention-based hierarchical RNN to improve the phishing detection accuracy. Their method also uses URL as input only, but their hierarchical RNN structure learns the features from both character level and word level. The Attention mechanism was adopted to let the neural network focus on the important area of the URL.

Recently, Opara et al. [3] made the first attempt to apply the deep learning method to HTML input. They combined the character-level CNN and word-level CNN to improve the detection accuracy. They achieved significant improvement compared to traditional machine learning-based approaches. However, they ignore the inherent structure information of HTML, and CNN has difficulties in capturing long-range semantics in HTML.

C. Graph Neural Network

Graph neural network has got great attention in recent years due to its superior power in modeling structural information [26], [27].

GNN is first proposed by [12]. Plenty of improved graph neural networks have been proposed since then [28]–[33]. GCN [34] is a simple but powerful graph neural network and is currently one of the most widely used and powerful graph neural networks.

Graph Convolutional Networks (GCN) has also been applied to text classification tasks in recent years. The first attempt is conducted by [35]. They generalize the traditional convolutional neural networks from grids data to graph structure data from the perspective of spectral graph theory.

Yao et al. [36] constructs a single large heterogeneous graph from an entire corpus, which contains words and documents as nodes. Then they use GCN to model the local correlation between nodes and jointly learn the word embedding and document embedding. Their method outperforms other state-of-the-art text classification methods.

Instead of building a large graph, Huang et al. [37] proposes to build a graph for each text sample, and utilize the message passing mechanism to learn the local features. By doing this, it consumes significantly fewer computation resources compared to Yao et al. [36].

III. Methods

In this section, we introduce our proposed method in detail. Fig. 2 shows the overall structure of our model. It is a classifier that takes an HTML document as input and outputs the possibility that this is a phishing web page. For each HTML, we constructed a small graph according to its inherent DOM structure. Then we utilize RNN and embedding method to encode the information of each node into a learnable vector. A graph neural network is then adopted to model the long-range semantics in the graph. Finally, we obtain the graph-level representation and classify it with a fully-connected layer.

A. Vertex Embedding and Graph Construction

We first convert raw HTML inputs into DOM trees. Each node in a DOM tree is an element, which is the building block of HTML pages. As is shown in Fig. 1, an element is enclosed by angle brackets, and consists of three main parts: Tag, Attribute, and Content.

Then we build a graph according to this DOM tree. There are two parts of building this graph: the structure, and the features of the nodes. The structure is relatively straightforward.

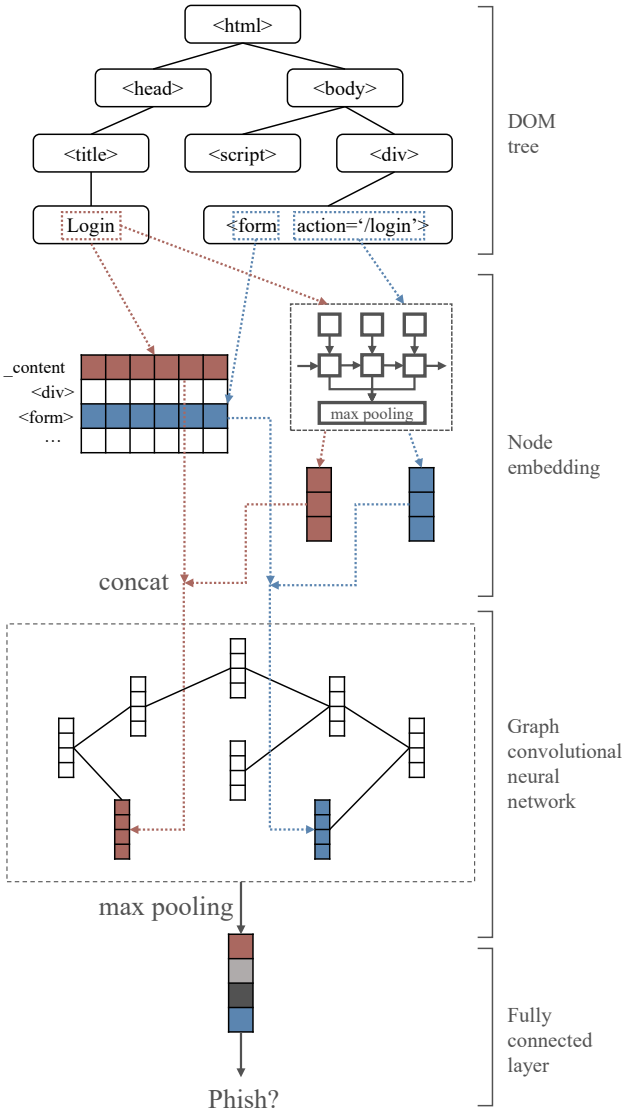

Fig. 2. The architecture of our proposed graph neural network-based phishing detection model. For each HTML file, we first parse it into a DOM-tree, then we build a graph according to this tree by embedding its nodes into learnable vectors. Next, we utilize the graph neural network to model the long-range semantics between each node pairs. Finally, we use a graph pooling layer to reduce the node representations into a single graph-level representation and output the score with a fully-connected layer.

Since the tree is a special kind of graph, we can naturally treat the structure of the DOM tree as a graph.

The key design choice is how to encode the information of each element into a learnable vector. Due to the different properties of the three parts of an element, we use different mechanisms to separately encode them into vector representations and combine them into one representation of the node.

1) Tag: A Tag represents the type of an element. The complete set of HTML tags is defined by the World Wide Web Consortium [38] and there are about 110 types of tag in total. Since each node has exactly one Tag, and there are

relatively few types of Tags, we embed it by a mapping relationship C that maps each Tag u to a K_{tag}-dimensional vector $C(u) \in R^{K_{tag}}$. This mapping relationship is represented by a $(M * K_{tag})$ matrix of free parameters, where M is the number of Tag types, and K_{tag} is the embedding dimension. In this way, the Tag of the i-th node in the graph will be mapped into a K_{tag}-dimensional vector $\mathbf{t_i}$. Notice that a type of Tag may appear multiple times in an HTML, they will be embedded into the same vector, but in different positions of the graph, and will interact with different Attributes. The parameters of the embedding matrix are randomly initialized.

2) Attribute: Attributes contain meta-information describing the element. It is essentially a set of key-value pairs, and its content is much more flexible comparing to the Tags. The number of Attributes for each node is uncertain. Simply using a mapping relation with fixed vocabulary to embed it will lead to sub-optimal performance since the size of the vocabulary will be very large, and most of them will appear only a few times in the dataset, which will result in insufficient training of parameters. Therefore, we use RNN to embed the Attribute and get its vector representation. Specifically, consider the attribute as a sequence of character $(c_1, c_2, \cdots, c_m)$, we first embed these characters with a embedding layer E that maps each character c to a k-dimensional vector $E(c) \in R^k$:

$$(\mathbf{e_1}, \mathbf{e_2}, \cdots, \mathbf{e_m}) = (E(c_1), E(c_2), \cdots, E(c_m)) \tag{1}$$

Then, we fed the above sequence of embedded vectors into an RNN to extract their context features:

$$(\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_m}) = \text{RNN}(\mathbf{e_1}, \mathbf{e_2}, \cdots, \mathbf{e_m}) \tag{2}$$

The output of the above RNN is a sequence of vectors. Since the length of the Attribute of each node is variable, the lenghth of the output is also variable. To obtain a fixed-length vector representation of a node, we apply Maxpooling on the output of the RNN:

$$\mathbf{a_i} = \text{Maxpooling}(\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_m}) \tag{3}$$

3) Content: Content is the information that is finally shown to the users of the web pages. In the DOM tree, the Content of non-leaf nodes is other child nodes, and the Content of leaf nodes is a vanilla string. We regard the Content of the leaf node as a special node, which has a special Tag: "content". The Content string is treated as the Attribute of the node. In this way, this special node can be directly fed into the above two embedding modules like other nodes in the graph. The reason behind this is that only a few nodes in the DOM tree contain Content. If we employ a separate embedding method for it, then the vector representations will be zero for most nodes.

Through the above steps, for each node i in the graph, two fixed-length vector representations t_i and a_i will be generated, representing the information that comes from its Tag, Attribute, and Content. We use the concatenation operation to combine these two vectors into one embedding vector that represents all the information of the node:

$$\mathbf{r_i} = \text{Concat}(\mathbf{t_i}, \mathbf{a_i}) \tag{4}$$

B. Graph convolutional layer

With the graph and vector representation of the vertexes, we utilize the graph convolutional network to capture the long-range semantics between nodes in the graph. The core idea of the graph convolutional network is to communicate the information between neighbors by message passing. Specifically, consider a node in the graph, a graph convolutional layer typically collects information from its neighbors, then utilize a fully-connected layer to model the relation between this node and its neighbors.

Among the many variants of graph convolutional networks, we choose a simple but powerful model Topology Adaptive Graph Convolutional Networks (TAGCN) [31]. Instead of aggregating information from direct neighbors, TAGCN obtains information from different hops with different weights. This gives the model more flexibility to model direct correlation and indirect correlation separately.

We represent the structure of a DOM tree as $G = (V, E)$, where $V = \{v_1, \cdots, v_N\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges between nodes. We use a square $|V| \times |V|$ matrix $\mathbf{A}$ to denote the adjacency matrix. An element A_{ij} in the adjacency matrix is 1 when there is an edge from node v_i to node v_j and 0 other wise. $\mathbf{D}$ is the degree matrix corresponding to $\mathbf{A}$:

$$D_{i,j} := \begin{cases} \deg(v_i) & , \ i = j \\ 0 & , \text{ other} \end{cases} \tag{5}$$

where $\deg(v_i) = \sum_j A_{ij}$ denotes the number of neighbors of node v_i.

We stack the vectors $\mathbf{r_i}$ of each node obtained from the last section to form a feature matrix:

$$\mathbf{F} = \begin{bmatrix} — & r_1^T & — \\ — & r_2^T & — \\ & \vdots & \\ — & r_n^T & — \end{bmatrix} \tag{6}$$

With the adjacency matrix, the degree matrix and the feature matrix $\mathbf{F}$, a layer of TAGCN can be defined as:

$$\mathbf{H} = \sum_{b=0}^{B} (\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^b \mathbf{F}\mathbf{W}_b \tag{7}$$

where $\mathbf{W}_b$ contains the learnable parameters.

For each node in the graph, TAGCN collects the features from its neighbors and aggregates them with different weights, then updates the feature representation of itself.

C. Graph classification and model training

The output of the above graph convolutional network is the final feature representation, which is still a group of vectors corresponding to every vertex in the graph. Since the number of nodes of each sample is variable, the number of vectors is also different. Therefore, we use a reduction function to

aggregate the vector representation for each node to obtain a graph-level vector representation:

$$\mathbf{g} = \text{Maxpooling}(\mathbf{H}) \tag{8}$$

This vector is then passed through a fully connected layer and an activation function, outputting the probability that the sample is a phishing web page:

$$y = \text{Sigmoid}(\mathbf{W}\mathbf{g} + \mathbf{b}) \tag{9}$$

The goal of training is to minimize the cross-entropy loss between ground truth labels and predicted labels.

In this way, we have achieved end-to-end training. Starting from the loss function, the gradient is first passed back to the graph convolutional layer, and then to the vector representation of the nodes. For the Tag part of the vector, the loss will be passed back to the embedding matrix. For the Attribute part of the vector, the loss will be passed back to the RNN to guide its parameter learning.

IV. Experiments

In this section, we conduct extensive experiments to evaluate the proposed method. First, we introduce the settings of the experiments, including the datasets used for evaluation, the metrics, the baseline methods to compare, and the implementation details of our method. Then we compare the phishing web detection performance of our proposed method with other state-of-the-art methods. Finally, we do ablation studies to make sure that every part of our model has some contribution to the improvement of detection accuracy .

A. Dataset

To reliably test the effectiveness of the proposed method, we collected a large number of real-world samples, including phishing web pages and benign web pages.

Phishing web pages are obtained from PhishTank and OpenPhish. These two web pages regularly publish community-reported phishing web pages, and they are widely used as data sources in previous phishing web detection researches [6]. Since these two web pages only provide URLs, we build a crawler to visit the reported URLs to obtain their HTML content.

For normal web pages, we rely on the TrancoTop1M [39]. TrancoTop1M is the top one million domain names ranked by traffic. We use it as the start point and crawl recursively following the links in the pages. Many phishing pages rely on input forms to acquire sensitive information, which is typically similar to login pages on normal web pages. Adding these benign login pages will pose more difficulties for the phishing detection method. Thus the links with keywords such as login, sign in, signin, account, are crawled with high priority.

We adopt the group split method to divide the dataset into the training set and the test set. Specifically, we use the domain of each web page as the group id and divide the dataset by this id to ensure that samples of the same domain will not appear in the training set and test set at the same time. The rationale behind this is that most machine learning methods assume the i.i.d property of the samples, but the web pages within the same domain are similar, which violates this assumption.

TABLE I
THE STATISTICS OF THE DATASET.

Class	Train	Validation	Test	All
Phishing	12713	3224	10641	26578
Benign	58163	14495	49325	121983
All	70876	17719	59966	148561

Table I lists the statistics of the data set used in our experiments. Note that the number of benign samples in the data set is far larger than phishing samples. This is more realistic and can better evaluate the performance of the phishing detection methods.

B. Performance Metric

Since our data set is unbalanced, we selected multiple indicators to reliably evaluate the performance of the methods. F1 and accuracy are common indicators that need to specify the classification threshold in advance, while AUC-ROC and AUC-PR are thresholds-invariant metrics that aggregate the performance of the model across all possible thresholds. Note that accuracy is inappropriate when the data set is highly unbalanced, but we still report it for completeness and as a basic sanity check. We consider F1 and AUC-PR as the most important metrics in our experiments.

C. Compared methods

We compare our method with several state-of-the-art methods. The first group of methods only use the URL as input.

- **Manual features+SVM**: Das et al. [6] summarizes the manual features used by the state-of-the-art phishing detection methods. We extract these features and then use SVM as the classifier. The reason for choosing SVM is that the character n-gram will lead to a very high feature dimension, and SVM has the best classification accuracy for high dimension data.
- **PDRCNN**: PDRCNN [10] is a deep learning-based phishing detection method that takes URL as input. It proposed improved neural network architecture and achieved decent detection accuracy improvement.

The second group of methods uses HTML as input.

- **Manual features+SVM**: We also implement the HTML-based features summarized by Das et al. [6] to build a traditional phishing detection method.
- **RCNN**: Since there are rather limited deep learning based-methods using HTML as input, we build a classification model based on the RCNN [40] model as a baseline. RCNN is currently one of the best deep learning-based text classification model. We treat the HTML source code as the sequences of characters.
- **HTMLPhish**: HTMLPhish [3] is the state-of-the-art deep learning-based phishing HTML detection approach that

TABLE II
PHISHING WEB PAGE DETECTION PERFORMANCE COMPARISON.

Input	Method	AUC-ROC	AUC-PR	F1	Precision	Recall	Accuracy	Training time (s)	Test time (s)
URL	Manual features+SVM	0.9373	0.8583	0.7683	0.8712	0.6872	0.9265	46	1.3
	PDRCNN	0.9503	0.8927	0.8208	0.8427	0.8001	0.9380	1050	24
HTML	Manual features+SVM	0.9512	0.8964	0.8331	0.8753	0.7949	0.9435	241	11
	Char-RCNN	0.9541	0.8996	0.8499	0.9148	0.7937	0.9502	16141	317
	HTMLPhish	0.9611	0.9177	0.8550	0.9191	0.7992	0.9519	6351	135
	Our method	**0.9682**	**0.9260**	**0.8634**	**0.9345**	**0.8025**	**0.9550**	7153	124

TABLE III
ABLATION STUDIES RESULTS.

Input	Model	AUC-ROC	AUC-PR	F1	Accuracy
tag+attribute	**graph**	**0.9682**	**0.9260**	**0.8634**	**0.9550**
tag	graph	0.9341	0.8244	0.7178	0.9119
attribute	graph	0.9602	0.9073	0.8421	0.9476
tag+attribute	bag	0.9549	0.8945	0.8384	0.9471

combines a character-level CNN and a word-level CNN into an end-to-end classifier.

D. Implementation details

We implement the proposed method with PyTorch and run the experiments on a GPU with 11GB memory. The hyperparameters of our model include the number of graph convolutional layers, embedding dimensions, and hidden layer dimensions. We adjust these hyperparameters based on the performance on the validation set. Finally, a two-layer graph convolution layer is used. Stacking more graph convolutional layers has no significant improvement. Both the embedded dimension and the hidden layer dimension are set to 64. After each layer of the graph convolution layer, there is a Dropout layer with a drop probability of 0.2 to improve the stability of the model and reduce the over-fitting of the model. We use Adam as the optimizer with a learning rate of 0.01. When the loss stops decreasing for 3 consecutive rounds, reduce the learning rate to 0.001, and then learn until the loss stops falling for another 3 consecutive rounds.

E. Performance comparison

In this section, we compare our proposed method with other state-of-the-art phishing web detection methods. There are three major questions we aim to address:

- Is HTML more useful than URL for phishing web detection?
- Is it necessary to utilize deep learning methods?
- Is our method more accurate than vanilla deep learning based method?

The phishing detection performance of our method and other competing methods are listed in Table II. From the table, we can see that the methods using HTML as input are more accurate than the methods using URL as input. This demonstrates that using HTML information can indeed improve detection performance. Also, it is evident that deep learning-based methods RCNN and our method outperform traditional detection methods, which indicates that deep learning-based methods can beat sophisticated manually designed features. Comparing our method with RCNN and HTMLPhish, we can obtain the **main result** that our graph convolutional network-based phishing detection method achieves significant performance improvement, which indicates that our model can effectively leverage the inherent structural information of HTML to improve the phishing detection accuracy.

F. Ablation Study

The purpose of this section is to verify the effectiveness of each part of our proposed model. We respectively remove a part of our proposed model and compare the detection accuracy.

In the first group of experiments, we build two models that respectively using Tag or Attribute information only. From the experimental results listed in Table III we can find that these two models perform significantly worse than the complete model, which indicates that these two parts both have contributions.

In the second set of experiments, we keep the tag and attribute information, but use a bag-of-words model without the knowledge of the graph structure to verify that the information in the HTML structure really helps our model to improve the recognition accuracy. It can be seen from the Table III that the detection accuracy is pretty low. This shows that our proposed method has indeed learned the structural information in HTML, and this information can effectively help improve recognition accuracy.

V. CONCLUSIONS

We have presented our improved graph convolutional network for phishing web detection. We first construct the graphs from the DOM trees of HTML source codes. Then we utilize RNN and embedding method to extract local features in the nodes and model the long-range semantics among the nodes with graph neural network. The experimental results demonstrate that our approach of combining the advantage of RNN and graph neural network can effectively improve the phishing detection accuracy.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China with Grant No. Y950131112. The corresponding author is Yongzheng Zhang.

REFERENCES

[1] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2010*. The Internet Society, 2010.

[2] G. Ho, A. Sharma, M. Javed, V. Paxson, and D. A. Wagner, "Detecting credential spearphishing in enterprise settings," in *26th USENIX Security Symposium, USENIX Security 2017*. USENIX Association, 2017, pp. 469–485.

[3] C. Opara, B. Wei, and Y. Chen, "Htmlphish: Enabling phishing web page detection by applying deep learning techniques on HTML analysis," in *2020 International Joint Conference on Neural Networks*. IEEE, 2020, pp. 1–8.

[4] A. AlEroud and L. Zhou, "Phishing environments, techniques, and countermeasures: A survey," *Computers & Security*, vol. 68, pp. 160–196, 2017.

[5] Z. Dou, I. Khalil, A. Khreishah, A. I. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (sok): A systematic review of software-based web phishing detection," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2797–2819, 2017.

[6] A. Das, S. Baki, A. E. Aassal, R. M. Verma, and A. Dunbar, "Sok: A comprehensive reexamination of phishing research from the security perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 671–708, 2020.

[7] P. Zhao and S. C. H. Hoi, "Cost-sensitive online active learning with application to malicious URL detection," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 919–927.

[8] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran, "A lexical approach for classifying malicious urls," in *2015 International Conference on High Performance Computing & Simulation, HPCS*. IEEE, 2015, pp. 195–202.

[9] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know your phish: Novel techniques for detecting phishing sites and their targets," in *36th IEEE International Conference on Distributed Computing Systems*. IEEE Computer Society, 2016, pp. 323–333.

[10] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: precise phishing detection with recurrent convolutional neural networks," *Security and Communication Networks*, vol. 2019, pp. 2 595 794:1–2 595 794:15, 2019.

[11] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing URL detection via CNN and attention-based hierarchical RNN," in *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 13th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE*. IEEE, 2019, pp. 112–119.

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2009.

[13] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in *Proceedings of the 4th Workshop on Digital Identity Management*. ACM, 2008, pp. 51–60.

[14] H. Shahriar and M. Zulkernine, "Trustworthiness testing of phishing websites: A behavior model-based approach," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1258–1271, 2012.

[15] G. Ramesh, I. Krishnamurthi, and K. S. S. Kumar, "An efficacious method for detecting phishing webpages through target domain identification," *Decision Support Systems*, vol. 61, pp. 12–22, 2014.

[16] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *CEAS 2009*, 2009.

[17] J. Mao, P. Li, K. Li, T. Wei, and Z. Liang, "Baitalarm: Detecting phishing sites using similarity in fundamental visual features," in *2013 5th International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 2013, pp. 790–795.

[18] K. Chiew, J. C. S. Fatt, S. Sze, and K. S. C. Yong, "Leverage website favicon to detect phishing websites," *Security and Communication Networks*, vol. 2018, pp. 7 251 750:1–7 251 750:11, 2018.

[19] G. Xiang, J. I. Hong, C. P. Rosé, and L. F. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Transactions on Information and System Security*, vol. 14, no. 2, pp. 21:1–21:28, 2011.

[20] J. Stobbs, B. Issac, and S. M. Jacob, "Phishing web page detection using optimised machine learning," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 483–490.

[21] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[22] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL, 2014, pp. 1746–1751.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[26] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *CoRR*, vol. abs/1812.04202, 2018.

[27] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *CoRR*, vol. abs/1901.00596, 2019.

[28] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web - 15th International Conference, ESWC*, vol. 10843. Springer, 2018, pp. 593–607.

[29] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, 2015, pp. 2224–2232.

[30] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *CoRR*, vol. abs/1506.05163, 2015.

[31] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *CoRR*, vol. abs/1710.10370, 2017.

[32] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR*, 2018.

[33] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML*, vol. 70. PMLR, 2017, pp. 1263–1272.

[34] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR*, 2017.

[35] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, 2016, pp. 3837–3845.

[36] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *The Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 7370–7377.

[37] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, "Text level graph neural network for text classification," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*. Association for Computational Linguistics, 2019, pp. 3442–3448.

[38] WWW. (2020) Html tag set. https://www.w3.org/TR/2012/WD-html-markup-20121025/elements.html.

[39] V. L. Pochat, T. van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *26th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2019.

[40] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI, 2015, pp. 2267–2273.