LGNet: Extreme Data Imbalance and Multi-Label Phishing URL Detection Network

Anonymous ACL submission

Abstract

Phishing is a form of cyberattack, that manipulates individuals into disclosing sensitive information. Typically, phishing URLs constitute a minor proportion of all searchable URLs, and are marked by multi-labeling and a significant imbalance in sample data. These characteristics considerably diminish the effectiveness and precision of existing detection methodologies. This paper introduces LGNet, an innovative network framework designed to identify phishing URLs in practical scenarios efficaciously. To tackle the multi-labeling issue, we have developed a label propagation algorithm that employs a confidence threshold limitation, ensuring high-confidence labeled URLs are acquired. We enhance the scalable tree system by using BILSTM and attention mechanism to address the substantial disparity between labeled and sample data in phishing URL prediction. Our experimental results demonstrate that LGNet markedly surpasses existing state-of-the-art algorithms in detecting phishing URLs.

1 Introduction

Phishing is a prevalent online attack aiming to fraudulently acquire sensitive information from network users, potentially leading to malware exposure, identity theft, or financial loss. The detection and blocking of phishing URLs present significant challenges, as attackers frequently employ sophisticated techniques to circumvent traditional detection methods [16, 5]. The growing incidence of phishing exacerbates these challenges.

Current research on phishing includes analyzing definitions [8], investigating human factors [19], and identifying phishing sites [20]. Traditional detection methods focus on training both end-users and automated software classifiers to recognize phishing URLs [2, 18]. However, these approaches often fall short in accuracy and effectiveness, especially as phishing URLs become increasingly indistinguishable from legitimate ones. Valentim et al. [23] suggest employing Generative Adversarial Networks (GAN) for enhancing phishing website detection in zero-day networks. Ozcan et al. [17] propose a hybrid model combining Long Short-Term Memory (LSTM) and Deep Neural Network (DNN) algorithms for this purpose. Notably, the datasets in these studies typically feature a balanced or nearly balanced ratio of phishing to normal URLs [15, 1]. In reality, however, phishing URLs often represent a small fraction of all URLs, and are marked by multi-labeling and extreme data imbalance, rendering existing detection methods less efficient and accurate.

This paper introduces LGNet, a neural network framework specifically designed to accurately identify phishing URLs amidst a vast array of largely unlabeled and highly unbalanced URL data. LGNet incorporates a label propagation algorithm with a confidence threshold limit for effective data augmentation, producing high-confidence labeled data. Moreover, we have enhanced the scalable tree boosting system with a Bidirectional Long Short-Term Memory (BILSTM) and an attention mechanism, optimizing our phishing URL prediction framework. Our experimental findings demonstrate that LGNet significantly outperforms existing state-of-the-art algorithms in multi-classification tasks involving unbalanced datasets.

2 Methodology

2.1 LGNet Framework

In this section, we propose a novel phishing URL detection network framework called LGNet for detecting phishing URLs that have an extreme imbalance in labeling and data volume in real-world situations.

To facilitate feature extraction by LGNet, URL sequences are encoded into word vectors through a two-step process. Initially, URL data are segmented at both character and lexical levels to gen-

erate respective corpora. These are then subjected to a dual encoding process, producing character and lexical encoding vectors for the URLs [13]. This results in the formation of word-embedded feature vectors, constituting both word and character matrices, crucial for subsequent extraction of semantic features.

Consider the ith URL u_i in the URL character-level corpus, and encode each character in it. If the jth character in u_i is denoted as g_j, $g_j = (g_{j1}, g_{j2}, ..., g_{jm})^T$ is an m-dimensional vector, where $m = 46$ is the size of the URL character dictionary. Each URL constitutes a one-hot matrix $G = G_{m\times \mathrm{n}} = (g_1, g_2, ..., g_n)$. Next, G is mapped by single-layer neural network embedding into its representation matrix S. where $\mathrm{W} \in \mathbb{R}^{\mathrm{p}\times\mathrm{m}}$ is the weight matrix of the embedding layer and $P = 128$ is the embedding dimension. The URL character embedding is computed as:

$$S^c = WG = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{p1} & w_{p2} & \cdots & w_{pm} \end{bmatrix} \times \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{mn} \end{bmatrix} \tag{1}$$

After word vector embedding, URLs are characterized as feature vectors $X_i = (S_i^c, S_i^w)$, where S_i^c is a character-level feature vector learned from u_i, and $S_i^w)$ is a vocabulary-level feature vector. Sequence vectorization makes feature-based mathematical computation possible. The feature vectors of URLs will then be learned and multicategorized.

In the convolutional layer, multiple convolutional kernels perform convolutional operations on the input vectors to generate multiple feature mappings; in the pooling layer, the feature dimensions are dimensionality reduced by maximal pooling. Among them, LGNet removes the pooling layer of Web2Vec [9] in the vocabulary-level channel to prevent the pooling layer from losing the semantic information of the text and optimize the feature extraction of the original model.

The input X_i to the convolutional layer is the feature representation of the web page. For some convolution kernel W, the convolved matrix R_i is:

$$R_j = f\left(W \otimes V_{j:j+h-1} + b\right) \tag{2}$$

where b is the bias and $\otimes$ is the convolution operation is the activation function. In order to reduce the network parameters and extract the most important features, the feature maps after the convolution operation are used 1-Maxpooling.

The attention mechanism[11] can help the model to focus on the critical parts of the input sequence. In URL classification tasks, URLs are usually variable-length sequences [14], and some of their subsequences may contain critical information, such as specific domain names, paths, or parameters. By using the attention mechanism, the model can focus more on these key parts and improve the ability to capture key information. When dealing with sequence data, traditional recurrent neural networks [21] and BiLSTM [22] may face the problem of gradient vanishing or gradient explosion, and the attention mechanism can help the model to better capture long-distance dependencies and improve the model's ability of long-time memory. The calculation formula is as follows:

$$\begin{aligned} a &= \tanh\left(h\right) \\ h &= softmax(w^T a) \\ x &= \sum_t \alpha h \end{aligned} \tag{3}$$

where w is the weights, h is the output of BiLSTM, α is the attention, and x is the weighted summed feature vector.

In the process of classification prediction, scalable tree boosting system [7] is chosen as a classifier, which is able to integrate multiple weak classifiers to obtain better classification performance.In the given $D = \{(x_i, y_i)\}(|D| = n, x_i \in R^m, y_i \in R)$ training set, the prediction function is:

$$\hat{y}_i = \sum_{K=1}^{K} f_k(x_i), f_k \in F \tag{4}$$

where K denotes that there are K decision trees, $F = \{f(x) = w_{q(x)}\}(q{:}R^m \to T, w \in R^T)$ denotes the space of decision trees, q denotes the structure of each tree, which maps an input sample x to the corresponding leaf node, and T denotes the number of leaf nodes in each tree, each f_k corresponds to a separate tree structure and leaf weight w. Unlike decision trees, regression trees have a score on each leaf. Here i_w is used to represent the score of the i-th leaf. To learn the set of functions in the model, the regularized objective function can be written as:

$$L(\varphi) = \sum_i l(\hat{y}, y_i) + \sum_k \Omega(f_k) \tag{5}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2 \tag{6}$$

where l is the second-order derivable loss function used to measure the error between the true

value and the predicted value, and the canonical term represents the model complexity in order to prevent overfitting.

Assuming that $\hat{y}_i^{(t)}$ is the predicted value of the ith sample at the tth iteration, the residuals are used to fit the loss function, denoted as:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{7}$$

The objective function becomes the following equation:

$$L^{(t)} = \sum_{i=1}^{n} l\big(y_i, \hat{y}^{(t-1)} + f_t(x_i)\big) + \Omega(f_t) \tag{8}$$

Performing a second-order Taylor expansion of the above equation, removing the constant term, and defining $I_j = \{i|q(x_i) = j\}$ as the set of instances of the jth leaf, the substitution collation yields the objective function to be solved as:

$$\begin{aligned}\widetilde{L}^{(t)} &= \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}\boldsymbol{h}_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}\boldsymbol{w}_j^2 \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)\boldsymbol{w}_j + \frac{1}{2}\left(\sum_{i\in IJ}\boldsymbol{h}_i + \lambda\right)\boldsymbol{w}_j^2\right] + \gamma T\end{aligned} \tag{9}$$

$$g_i = \partial_{\hat{y}(t-1)} l(y_i, \hat{y}^{(t-1)}) \tag{10}$$

$$h_i = \partial^2_{\hat{y}(t-1)} l(y_i, \hat{y}^{(t-1)}) \tag{11}$$

2.2 Loss Function

The above equation is used as a score function to measure the goodness of the tree structure, and in practice it is difficult to enumerate all tree structures q to select the one with the highest score. At this point, a greedy algorithm is used for node splitting. Starting from the root node as the only leaf node, branches are added by traversing the attributes, assuming that I_L and I_R denote the left and right subtree nodes after splitting, respectively, and letting $I_L \cup I_R$, the value of the loss that is reduced by splitting a node is:

$$\begin{aligned}L_{\text{loss}} = \frac{1}{2}\Bigg[&\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_{l\in L}} k_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} k_i + \lambda} \\ &- \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} k_i + \lambda}\Bigg] - \gamma\end{aligned} \tag{12}$$

Labels	Types	Numbers
0	Normal	8509522
1	Shopping	22
2	Dating	17780
3	Counterfeit Identity	173
4	Phishing	4658
5	Impersonation of Lawyers	2
6	Platform Fraud	2603
7	Recruitment	11
8	Killing Plate	883
9	Betting and Gambling	595
10	Credit management	998
11	Swipe fraud	549
12	Lottery Scam	7

Table 1: Data tagging and classification about URLs. Types represents the categorization of URLs, including normal URLs and different types of phishing URLs, and Numbers represents the number of different types of URLs in the dataset.

3 Experiments

In this paper, we examine data from three datasets provided by the Mobile Innovation Institute, identifying a significant imbalance among them. Normal websites account for the vast majority of the data set, the total number of data for other labels does not add up to more than 30,000, and the amount of data for different labels also varies greatly, with some exceeding 10,000 and others not even exceeding ten, and the data set suffers from a problem of extreme sample imbalance. To mitigate this imbalance, our approach initially employs a semi-supervised learning model to augment categories with fewer data points. Subsequently, we apply data augmentation techniques to further rectify the data imbalance, resulting in a more evenly distributed dataset. The final dataset division allocates 80% (45,416 URLs) for training and 20% (11,354 URLs) for testing. Following the labeling and classification of these datasets, we present a comprehensive data overview in Table 1:

In addressing the challenge of highly unbalanced datasets, this study employs label propagation based on threshold screening for data augmentation. Label propagation [10] occurs as category labels are transmitted from labeled to unlabeled data via these connecting edges. Typically, label propagation is more efficient between similar vertices, as their probability distributions tend to be closely aligned. Consequently, the resulting categorization forms a distribution that is not constrained

to a specific shape, thereby more accurately reflecting the true data distribution. The process of label propagation is executed on a weighted, undirected relational graph [12]. The weights on the graph's edges intuitively represent the similarity between samples, facilitating the analysis of label propagation intensity along these edges. Therefore, constructing a graph where labeled and unlabeled samples are interconnected through undirected edges is a preliminary step in label propagation.

Consider constructing a labeling matrix to represent the labeling changes during the propagation process. Assuming that the sample set has a total of c category labels and l labeled samples, therefore, this paper defines a lxc labeling matrix Y_L , with the ith row denoting the labeling probability composition of the ith sample. Given an unlabeled sample u define a labeling matrix Y_U of $u \times c$. So the total training sample set can be represented by a labeling matrix $Y^T = (Y_L, Y_U)$ of $(l+u) \times c$. where the conceptual composition of the sample x_i $a_i = (a_{i1}, a_{i2}, \ldots, a_{ij})$, a_{ij} denotes the jth labeled conceptual part of the x_i sample l_j. Assume that $a_{ij} > 0$ and that $a_i^T = 1$ for each sample x_i. Accordingly, the labeling matrix Y is constructed as shown in the following equation:

$$\begin{array}{c} \begin{array}{ccccc} \alpha_{(1)} & & \alpha_{(j)} & & \alpha_{(c)} \\ \downarrow & & \downarrow & & \downarrow \end{array} \\ \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1j} & \cdots & \alpha_{1c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_{i1} & \cdots & \alpha_{ij} & \cdots & \alpha_{ic} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nj} & \cdots & \alpha_{nc} \end{bmatrix} \end{array} = \begin{array}{cl} \begin{bmatrix} \alpha_1^T \\ \vdots \\ \alpha_i^T \\ \vdots \\ \alpha_n^T \end{bmatrix} & \begin{array}{l} \\ \\ \leftarrow x_1 \\ \\ \leftarrow x_i \\ \\ \leftarrow x_n \end{array} \end{array} \tag{13}$$

In the initialization of matrix Y, the initial value of the unlabeled data labels in this paper is taken as 0. The formula for generating the matrix Y is shown in the following equation:

$$a_{ij} = \begin{cases} \dfrac{1}{|Y_i|}, l_j \in Y_i \\ 0, otherwise \end{cases} \tag{14}$$

In the process of label propagation, the label matrix needs to maintain the label probability distribution of the samples, and when determining the labels of the sample categories to be tested, the ones with larger label probability values are taken as the predicted labels. With one label propagation, the category labels will be propagated from labeled data to unlabeled data, and if the distribution of the training data satisfies the real data distribution, then the unlabeled data will play a role in helping to improve the learning.

Model	Accuracy	F1	AUC	Score
Phishnet	0.453	0.456	0.502	0.513
MPURNN	0.836	0.825	0.848	0.857
URLNet	0.896	0.899	0.945	0.954
DNN-LSTM	0.908	0.914	0.953	0.964
Aljofey et. al	0.911	0.913	0.953	0.967
Web2Vec	0.925	0.925	0.959	0.975
LGNet	**0.974**	**0.965**	**0.984**	**0.990**

Table 2: Benchmark comparison of different datasets, deeper color is the best result

To validate the effectiveness of our method, we compare it with several state-of-the-art methods for detecting phishing URLs, including a traditional software URL classifier Phishnet [18], and five machine learning methods, including MPURNN [6], URLNet [13], DNN-LSTM [17], Aljofey et. al [4], and Web2Vec [9]. We used our dataset to retrain these sota models. There are four metrics to compare the strengths and weaknesses of the fishing URL prediction models, including accuracy, F1, AUC and overall score. Higher values of accuracy, F1, AUC and total score prove that the model predicts phishing URLs with better quality.

Table 2 compares the numerical results of our LGNet with the state-of-the-art sota method. As can be seen from Table 2, our LGNet searches through a large number of URLs to predict phishing URLs with multi-labeling and extremely unbalanced sample data significantly outperforms other methods in all four metrics. The comparison results show that our LGNet has more accurate retrieval ability in phishing URL detection on real networks.

4 Conclusion

In this paper, we propose a neural network framework, LGNet improving the accurate identification of phishing URLs in a large amount of actual unlabeled URL data with a highly unbalanced sample size. Our LGNet employs a label propagation algorithm, impling it can apply semi-supervised learning to obtain new data with high-confidence labels. We improve the scalable tree enhancement system to obtain a better prediction framework for phishing URLs in the prediction. Experimental results show that our LGNet significantly outperforms other SOTA method in prediction results.

Limitations

We propose a neural network framework, LGNet, for phishing URL detection in real-world scenarios with multiple labels and extremely unbalanced sample data.While LGNet can improve the detection accuracy of phishing URLs in real-world scenarios, phishing attacks are constantly evolving, and attackers frequently change their tactics and techniques.The LGNet framework may face challenges in keeping up with these rapid changes, particularly when relying on historical data for training. In addition, detecting new or unknown phishing URLs that have not been previously encountered or labeled (like zero-day attacks [3]) may still be a challenge for LGNet. These challenges need to be addressed and optimized by subsequent research in order to better protect against phishing URLs and improve network security.

References

[1] 1987. Domain names - implementation and specification. RFC 1035.

[2] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. 2007. A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 60–69.

[3] Rasheed Ahmad, Izzat Alsmadi, Wasim Alhamdani, and Lo'ai Tawalbeh. 2023. Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, pages 1–79.

[4] Ali Aljofey, Qingshan Jiang, Abdur Rasool, Hui Chen, Wenyin Liu, Qiang Qu, and Yang Wang. 2022. An effective detection approach for phishing websites using url and html features. *Scientific Reports*, 12(1):8842.

[5] Luca Allodi, Tzouliano Chotza, Ekaterina Panina, and Nicola Zannone. 2019. The need for new antiphishing measures against spear-phishing attacks. *IEEE Security & Privacy*, 18(2):23–34.

[6] Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, and Fabio A González. 2017. Classifying phishing urls using recurrent neural networks. In *2017 APWG symposium on electronic crime research (eCrime)*, pages 1–8. IEEE.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

[8] Zuochao Dou, Issa Khalil, Abdallah Khreishah, Ala Al-Fuqaha, and Mohsen Guizani. 2017. Systematization of knowledge (sok): A systematic review of software-based web phishing detection. *IEEE Communications Surveys & Tutorials*, 19(4):2797–2819.

[9] Jian Feng, Lianyang Zou, Ou Ye, and Jingzhou Han. 2020. Web2vec: Phishing webpage detection method based on multidimensional features driven by deep learning. *IEEE Access*, 8:221214–221224.

[10] Steve Gregory. 2010. Finding overlapping communities in networks by label propagation. *New journal of Physics*, 12(10):103018.

[11] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. 2022. Attention mechanisms in computer vision: A survey. *Computational visual media*, 8(3):331–368.

[12] Sen Jia, Xianglong Deng, Meng Xu, Jun Zhou, and Xiuping Jia. 2020. Superpixel-level weighted label propagation for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(7):5077–5091.

[13] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. 2018. Urlnet: Learning a url representation with deep learning for malicious url detection. *arXiv preprint arXiv:1802.03162*.

[14] Yunji Liang, Qiushi Wang, Kang Xiong, Xiaolong Zheng, Zhiwen Yu, and Daniel Zeng. 2021. Robust detection of malicious urls with self-paced wide & deep learning. *IEEE Transactions on Dependable and Secure Computing*, 19(2):717–730.

[15] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. 2014. Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471.

[16] Miranda Mowbray and Josiah Hagen. 2014. Finding domain-generation algorithms by looking at length distribution. In *2014 IEEE international symposium on software reliability engineering workshops*, pages 395–400. IEEE.

[17] Alper Ozcan, Cagatay Catal, Emrah Donmez, and Behcet Senturk. 2021. A hybrid dnn–lstm model for detecting phishing urls. *Neural Computing and Applications*, pages 1–17.

[18] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. 2010. Phishnet: predictive blacklisting to detect phishing attacks. In *2010 Proceedings IEEE INFOCOM*, pages 1–5. IEEE.

[19] Asadullah Safi and Satwinder Singh. 2023. A systematic literature review on phishing website detection techniques. *Journal of King Saud University-Computer and Information Sciences*.

[20] Said Salloum, Tarek Gaber, Sunil Vadera, and Khaled Shaalan. 2022. A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access*, 10:65703–65727.

[21] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.

[22] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2019. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International conference on big data (Big Data)*, pages 3285–3292. IEEE.

[23] Rodolfo Valentim, Idilio Drago, Martino Trevisan, Federico Cerutti, and Marco Mellia. 2021. Augmenting phishing squatting detection with gans. In *Proceedings of the CoNEXT Student Workshop*, pages 3–4.