ABSTRACT Detecting phishing webpages is an essential task that protects legitimate websites and their users from various malicious activities. To classify the suspect webpage as phishing or legitimate, robust and effective features used for classification are in demand. However, recent phishing attacks usually make phishing webpages resemble the legitimate webpages in visual and functional aspects. This poses a greater difficulty for feature extraction. We herein propose SPWalk, an unsupervised feature learning algorithm for phishing detection. In SPWalk, similar property nodes refer to a collection of phishing webpages or legitimate webpages. We first construct a weblink network with nodes representing webpages. The edges between nodes represent the reference relationships that connect webpages through hyperlinks or similar textual content. Then, SPWalk applies the network embedding technique to mapping nodes into a low-dimensional vector space. A biased random walk procedure efficiently integrates both structural information between nodes and URL information of each node. The effectiveness and robustness of SPWalk come from three points. (1). Phishing attackers do not have full control over reference relationships. (2). The structural regularities generated by diverse reference relationships can be exploited to discriminate between phishing and legitimate webpages. (3). Node URL information makes the learned node representations more suited for phishing detection. Using node as numeric features, we conduct experiments to classify webpages as legitimate or phishing. We demonstrate the superiority of SPWalk over state-of-the-art techniques on phishing detection, especially in terms of precision (over 95%). Even in the case that phishing webpages are well camouflaged by attackers for evading detection, SPWalk exhibits better classification efficacy consistently.

INDEX TERMS Feature learning, network embedding, phishing detection, similar property.

I. INTRODUCTION Phishing is a concrete, widespread threat that combines social engineering with website spoofing. It leads to various malicious activities, including identity theft, financial gain, unauthorized account access, credit card fraud, etc. This threat causes not only tremendous financial losses to Internet users, but also long term reputation damage to the legitimate websites targeted by phishing scams.

A. BACKGROUND INFORMATION AND LITERATURE REVIEW FOR PHISHING DETECTION

One implementation means of phishing attacks is to leverage the advancement of web technology to create a new fraudulent website. Through manipulating textual or graphical form, the attackers make the newly-created phishing website look similar to the legitimate one. Another implementation form is to exploit the vulnerabilities in publicly-available websites to compromise legitimate websites. Through website compromise, a large scale deployment of phishing attacks is enabled. The automatically created phishing webpages share identical domain names and similar appearances with other (legitimate) webpages within the same compromised website.

Clearly, the attackers exploit the resemblance between phishing webpages and legitimate webpages. It is crucial and necessary to discriminate between phishing and legitimate webpages. The legitimate webpages imitated by phishing webpages are referred to as phishing targets. Existing
phishing detection mechanisms can be divided into target-independent and target-dependent methods. Target-independent methods aim to extract the common features shared by phishing webpages, which differ from the characteristics of legitimate webpages. Without an explicit comparison between the current webpage and its phishing target, such methods utilize the extracted features to build a binary classifier. The features can be computed from the webpage URL [1]–[6], HTML content [7], queries to remote servers [8], [9], and even publicly-available blacklisting services [10]. However, the attackers have the potential to infer relevant information about the classifier rules. In this case, the attackers can carefully manipulate classification features to evade detection. Target-independent methods exhibit the poor robustness against well-crafted phishing webpages that impose visual and functional deception.

On the other hand, target-dependent methods identify phishing webpages by highlighting the difference between the current webpage and its phishing target [11], [12]. The key idea is to discover the phishing target of a suspicious webpage. One way is to collect priori knowledge of the legitimate webpages potentially targeted by phishing scams (i.e., potential phishing targets). However, the offline collection of potential phishing targets leads to high computational complexity and space complexity. Another way is to retrieve phishing targets from search results through online search engines. Notably, phishing webpages may appear in search engine results. This increases the difficulty of locating phishing targets. The effectiveness of target-dependent methods is limited, especially when phishing targets are unpopular (legitimate) webpages.

B. INTRODUCTION OF WEBLINK NETWORK
To overcome the limitation of existing detection mechanisms, we construct a hierarchical network called a weblink network. A weblink network is constructed from a large quantity of suspicious webpages and reference relationships between different webpages. The reference relationships are directly implied by link relation, and indirectly implied by textual relation. Phishing attackers cannot clone a phishing webpage that resembles the legitimate one in terms of reference relationships, since they are unable to alter the reference relationships introduced by legitimate webpages. The structural regularity provides the potential to extract effective and robust classification features.

Specifically, link relation refers to the direct hyperlink from the current webpage to its destination webpage for page resource elements (i.e., images, CSS and script references). On the other hand, textual relation refers to the similar textual content between the current webpage and its destination webpages. Taking subject-matter keywords extracted from current webpage as query, textual relation can be derived from the associated webpages appearing in top search engine results. With legitimate webpages as destinations, phishing webpages frequently use such reference relationships to cover their real identities. Internet users may regard phishing webpages as legitimate webpages by clicking on hyperlinks or readily placing trust in similar keywords. In this way, phishing webpages can be camouflaged as legitimate ones. Within one fraudulent website, phishing webpages generate the hyperlinks connecting to each other, which enhances the camouflage effect to some extent. Different from phishing webpages, legitimate webpages call page resource elements from each other for the legitimate functionality. It is impossible for legitimate webpages to provide reference relationships back to phishing webpages. We construct a hierarchical network structure with webpages as nodes and reference relationships as edges. Such a network structure can be referred to as a weblink network.

C. APPLYING NETWORK EMBEDDING TO PHISHING DETECTION
To extract robust and effective features, we apply network embedding techniques to the constructed weblink network. Network embedding methods learn continuous feature representations for network nodes in an unsupervised way [47]. Each node can be represented with a low-dimensional vector, which captures meaningful semantic and structural information of the weblink network. The learned node representations can be used as numeric features for phishing detection with two important points. First, the attackers have no complete control over legitimate webpages. The reference relationships introduced by legitimate webpages cannot be altered. Second, the local structure centered on phishing webpages is different from the local structure centered on legitimate webpages, due to diverse reference relationships. The above two points provide effective structural information about identifying phishing webpages. The structural regularities conveyed by the weblink network can be preserved in node representations [16].

Nevertheless, existing network embedding methods cannot eliminate the camouflage effect produced by phishing webpages. We review the research activity on network embedding. The Skip-gram model [39] is originally designed for learning word representations from linear sequences consisting of natural language words. DeepWalk [13] extends word representation learning to network node representations. DeepWalk transforms a network structure into linear sequences consisting of nodes, using a uniform sampling strategy. A series of follow-up works have made a significant progress on network embedding [14]–[19]. The learned representations preserve co-occurrence patterns and relationships between context nodes and current node into linear sequences. When concerning a weblink network, the node sequences are sampled based on reference relationships between webpages. For the camouflage effect, phishing webpages introduce the reference relationships to their phishing targets. Consequently, there exist co-occurrence relationships between phishing and legitimate nodes. Phishing webpages are possible to have similar node representations with legitimate webpages.
D. BRIEF SUMMARY OF SPWalk

To overcome the camouflage effect of phishing webpages, we present a similar property oriented feature learning model called SPWalk. In a weblink network, there are two distinct collections of similar property nodes, wherein malicious nodes represent phishing webpages and legitimate nodes represent legitimate webpages. SPWalk concerns URL information of each network node which reflects to some extent the malicious degree of the current webpage. By integrating both network structural information discussed earlier and node URL information, SPWalk designs a biased sampling strategy. On this basis, SPWalk can provide effective, robust feature learning for phishing detection.

Specifically, SPWalk includes three phases: preprocessing, sampling, and optimization. In the preprocessing phase, SPWalk mainly computes the URL quality score for each node to represent the malicious degree of its corresponding URL. The sampling phase designs a biased sampling strategy which fully exploits effective structural information and the pre-processed URL scores. Based on the two important points in terms of structural regularities, our sampling strategy maximizes the likelihood of embedding legitimate nodes closely together in feature space. Based on the URL score comparison of node-pairs, our sampling strategy minimizes the co-occurring probability of malicious and legitimate nodes. This minimizes the camouflage effect caused by the reference relationships from phishing webpages to legitimate webpages. The optimization phase uses stochastic gradient descent to learn feature representations of weblink network nodes. The learned node representations improve the robustness and effectiveness of phishing detection. The learned node representations can be classification features in phishing detection with higher effectiveness and robustness.

To summarize, we make the following contributions:

1). We put forward the notion of similar property nodes to denote the collection of phishing webpages or legitimate webpages. Moreover, we propose SPWalk, a similar property oriented feature learning model that applies the network embedding techniques to phishing detection. Through SPWalk, similar property nodes have similar node representations in a low-dimensional feature space.

2). We show how SPWalk is in accordance with phishing detection principles, learning node representations conforming to similar property. The core idea is to design a biased sampling strategy that integrates URL quality score of each webpage with the structural regularities conveyed by the weblink network.

3). We construct a weblink network based on real-world webpage data. On this network structure, we conduct experiments to classify each network node as malicious or legitimate. Experimental results validate the unique superiority of SPWalk on phishing detection. When used as classification features, the learned node representations are verified to be effective and robust.

To our knowledge, SPWalk is the first work that specifically leverages network embedding methods for phishing detection. The rest of this paper is organized as follows. Section II reviews the works related to phishing detection and network embedding. Section III formally defines the task of similar property oriented feature learning. Section IV introduces the calculation process of URL quality score for our sampling strategy. Section V presents our feature learning model in detail. Section VI gives a detailed experimental design as well as experimental results. Finally, we conclude the paper in Section VII.

II. RELATED WORK

In this section, we first review existing phishing detection mechanisms, including target-independent and target-dependent methods. Next, we make a comparative analysis of two mainstream methods for network embedding: neural embedding methods and matrix factorization based methods.

A. PHISHING DETECTION METHODS

We categorize most of today’s phishing webpage detection mechanisms into two main categories: target-independent and target-dependent, depending on whether they make a comparison between suspect webpages and potential phishing targets.

1) TARGET-INDEPENDENT DETECTION

Such methods rely on generic features computed from the webpage URL, from its HTML content and visual appearance, from remote servers, etc. By extracting unique features of a webpage, one can classify each webpage as phishing or legitimate.

From the URL string, many researchers can directly extract lexical features (the length of a URL/domain/directory/file name, the number of hyphens, whether a blacklisted word appears, etc.) and host-specific properties (host names, IP address, geolocation). N. Provos et al. [10] propose a detection framework that applies URL lexical features to train a binary classifier, using machine learning techniques. Host-specific properties can also be inferred from URL names to improve classification accuracy.

Relevant features and proposed features presented in New rule-based method [20] are extracted from webpage URL and HTML content separately. Relevant features are frequently used to create malicious URLs. Proposed features include the reference address and the access protocol of page resource elements. These two feature sets are used to identify the lack of similarity between the content and the URL of phishing webpages. The correlation between the feature sets and internet banking indicates its limitation in the detection range.

The notion of full features [9] introduces external features acquired from queries to remote servers. Such features include WHOIS properties, Team Cymru information, SSL certificate, etc. This adds accuracy to malicious URL detection. Nevertheless, almost all generic features are pre-defined,
which makes it possible to infer enough information on how a target-independent phishing detection mechanism works [21], [22]. Phishing attackers exploit this information to resemble the generic features of the legitimate webpages, by carefully manipulating phishing webpages without compromising their malicious functionality. URLPatternMining [23] improves the quality of feature selection without using any pre-defined features. This method parses URLs to dynamically extract lexical patterns that cannot be inferred by phishing attackers. This proposal has its limitation that its detection performance is completely dependent upon the quality of URL patterns.

2) TARGET-DEPENDENT DETECTION

Such methods typically measure how different the suspect phishing webpage is from potential phishing targets. Either offline information collection or online search engine provides enough information about potential phishing targets.

Earth Movers Distance [24] compares the visual appearance of the suspect phishing webpage against that of potential phishing targets, using offline data collection. H. Zhang et al. [25] utilize textual content and visual information to locate the domain names of phishing targets, which improves the accuracy of phishing detection. These methods require a priori knowledge of potential phishing targets. Accordingly, offline information collection comes with a significant overhead.

Instead of offline information collection, a more effective method that retrieves potential phishing targets during operation by querying search engines, is first put forward in Cantina [26]. With textual information of the suspect phishing webpage as query keywords, Cantina discovers the phishing targets from online search results. WebsiteLogo [27] exploits Google image search to retrieve potential phishing targets, using webpage images (i.e., logo) as query. Notably, these methods may take phishing webpages appearing in online search results as phishing targets. This makes the explicit comparison between webpages meaningless.

Semantic link network (SLN) [11] based method applies logical reasoning to locate phishing targets of the suspect webpages accurately. To be specific, this method semantically organizes associated web resources of the suspect webpage to construct a Semantic Link Network. Based on inferring rules of the Semantic Link Network, the implicit relationship between the suspect webpage and its phishing target is discovered. Such relationships can be effective and reliable to retrieve phishing targets from SLN nodes. Unfortunately, SLN-based method is unable to identify phishing webpages except for the suspect webpage itself.

All the above methods detect not only phishing webpages on newly-created phishing websites, but also phishing webpages on publicly-available (legitimate) websites compromised by phishing attacks. Note that the second group of phishing webpages remains active for a more extended period than the first group. To detect the second group of phishing webpages, DeltaPhish [28] takes the homepage in the same website of the suspect webpage as a potential phishing target. This method uses the dissimilarity measure (i.e., HTML code and visual differences) between the homepage and the suspect webpage, which makes its detection range limited to phishing webpages on compromised websites.

B. NETWORK EMBEDDING METHODS

The primary task of network embedding is to learn node representations by transforming a graph structure into a sample collection of node sequences. This is similar to word embeddings in models like Skip-gram [29], [30]. Two mainstream methods for learning word representations are applied equally to network embedding.

1) NEURAL EMBEDDING METHODS

The Skip-gram model [31], [32] employs a fixed slide window to preserve co-occurrence patterns and relationships between neighbor words of natural language corpus. Similarly, neural embedding methods sample linear sequences consisting of neighbor nodes, which generalizes supervised feature learning to incorporate information from network neighborhoods [13]. DeepWalk’s uniform exploration learns d-dimensional feature representations by simulating truncated random walks. LINE captures 1-step and 2-step local information to embed the nodes sharing similar neighbors closely together [19]. Node2vec develops a family of biased random walks to discover node representations conforming to different equivalences [33]. These methods learn network representations using separate local context information.

2) MATRIX FACTORIZATION BASED METHODS

These methods utilize global statistics because a network structure can be represented as an affinity matrix [34]. Through decomposing affinity matrices, matrix factorization based methods learn latent representations of network nodes. The Skip-gram model with negative sampling can be seen as a model that implicitly decomposes positive point-wise mutual information (PMI) matrix to generate word embeddings [35]. Redefining the loss function of Skip-gram, GraRep [16] decomposes the transition probability matrix to capture distinct relational information within different transitional steps. It reveals the useful global structural information associated with the network.

All the above network embedding methods find the low-dimensional embedding of a network structure without exploiting textual and label information attached with network nodes. It is rewarding to explicitly take full advantage of such information when capturing diverse semantic relationships between nodes. For example, CANE [36] integrates textual information into feature learning to show different aspects of one node when interacting with different neighbor nodes. CANE achieves significant improvement in node classification. Unfortunately, all these methods yield inferior classification results when applied to classify the suspect webpages as phishing or legitimate. The most probable cause
is that existing feature learning algorithms cannot integrate specific information about phishing webpages (i.e., malicious URLs) into feature learning.

In this paper, we fill this gap by proposing a similar property oriented feature learning model, which we call the SPWalk. SPWalk utilizes both lexical features of webpage URLs and reference relationships of webpage elements. The low-dimensional embedding obtained through SPWalk is effective and robust enough to identify phishing webpages.

III. PROBLEM DEFINITION

In this section, we formally define the task of similar property oriented feature learning. We first define a weblink network as follows:

Definition 1 (Weblink Network): A weblink network is a directed, weighted graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots , v_m\}$ is the set of nodes, each representing a webpage with a unique URL, and $E = \{e_{ij}\}$ is the set of edges, each representing a reference relationship from $v_i$ to $v_j$. The weight $w_{ij}$ associated with $e_{ij} \in E$ takes binary values. We set $w_{ij} = 1$ if $e_{ij}$ exists, and $w_{ij} = 0$ otherwise.

In practice, a weblink network is a specific part of the World Wide Web. The local structure in a weblink network is shown in Figure 1. To construct a weblink network, we start by collecting initially associated webpage sets of given webpages. Weblink network edges connect associated webpages through hyperlinks and similar textual content. Given a suspect webpage, we examine its HTML source to find many hyperlinks pointing to other particular webpages. Simultaneously, we collect its associated webpages from top search results, using its textual content as query. We repeat this process to find initial sets for a collection of suspect webpages. Intuitively, there are hardly any edges connecting two initial sets without content correlation. This forms multiple small-scale graphs disconnected from each other. To construct a large-scale network structure with strong connectivity, we extract all hyperlinks inside initial sets to further collect their associated webpages. Expanding the scale of associated webpage set adds the edges connecting different small-scale graphs.

Weblink network nodes represent either phishing webpages or legitimate webpages. A collection of nodes representing phishing webpages can be referred to as malicious nodes. Equally, a collection of nodes representing legitimate webpages can be referred to as legitimate nodes. Either malicious or legitimate can be regarded as a class label assigned to nodes.

As the current webpage call page resource elements from its destination webpages, the number of edges connecting to destination webpages is regarded as the out-degree of the current node. Similarly, the number of the edges in the opposite direction is regarded as the in-degree of the current node, which is derived from the webpages calling page resource elements from the current webpage. For the camouflage effect, malicious nodes introduce many reference relationships connecting themselves to legitimate nodes and associated malicious nodes. This leads to higher out-degrees of malicious nodes than legitimate nodes. On the contrast, legitimate nodes have higher in-degrees because of their legitimate resource elements. What is more, no single legitimate node provides reference relationships from themselves to malicious nodes. These structural regularities are preserved in a weblink network. In one sense, a weblink network is similar to a citation network in which higher academic value makes classic papers have higher in-degrees than general papers.

To divide weblink network nodes into malicious nodes and legitimate nodes, we define the notion of similar property nodes.

Definition 2 (Similar Property Nodes): Similar property nodes are the nodes with similarity property. As discussed, each weblink network node is assigned with a label: malicious or legitimate. Similarity property is an attribute attached to a collection of nodes, which indicates these nodes have the same label. Accordingly, phishing webpages and legitimate webpages form two collections of similar property nodes: malicious nodes and legitimate nodes.

As our goal is to predict weblink network nodes as malicious or legitimate, we embed a weblink network into a low-dimensional space to provide effective, robust, efficient features. The key to conducting the embedding is to preserve the similar property of weblink network nodes. Therefore, we proceed by extending the Skip-gram architecture to networks. As networks are not linear, we employ a family of random walks to sample linear sequences consisting of nodes. To investigate the similar property, we seek the optimal sampling strategy that biases random walks towards neighbor nodes with the same label. Unfortunately, such a sampling strategy is far beyond the range of possibility.

Therefore, we leverage the corresponding URL of each node to approximate the optimal sampling strategy. More concretely, we thoroughly analyze URL lexical features to calculate the possibility that the webpage URL is malicious. From now on, we refer to the calculated possibilities as URL quality scores assigned to weblink network nodes. This leads to an important inference that similar property nodes exhibit similar performance in terms of URL quality scores. Accordingly, URL quality scores of different neighbor nodes provide comparatively accurate information for sampling strategy design. To exploit the information, we define the notion of possible similar property neighborhood.

Definition 3 (Possible Similar Property Neighborhood): The notion of neighborhood in natural language corpora refers to a fixed slide window over consecutive word...
sequences that captures context words of current word. Given a weblink network node \( v \in V \), its possible similar property neighborhood refers to node neighbors which have similar URL quality scores with that of \( v \). Its abbreviation is “PSPN”.

Figure 2 presents an illustrative example. Exploring possible similar property neighborhood of each node uses URL quality scores of similar property nodes to full advantage. A desirable sampling strategy maximizes the probability to generate the sequences of similar property nodes by the exploration of PSPN. We achieve this by developing a random walk procedure defined as follows:

**Definition 4 (Similar Property Oriented Random Walk):** Similar property oriented random walk is a biased sampling procedure which maximizes the co-occurrence relationships between similar property nodes. For all \( v \in E \), similar property oriented random walk is biased towards PSPN of \( v \).

We employ similar property oriented random walk to learn node representations conforming to similar property, which is defined as follow:

**Definition 5 (Similar Property Oriented Feature Learning):** Given a weblink network \( G = (V, E) \), the task of similar property oriented feature learning is to learn a mapping function \( f : V \rightarrow \mathbb{R}^d \) from nodes to feature representations. The parameter \( d \) specifies the number of feature dimensions and \( d \ll |V| \). Equivalently, \( f \) is a matrix of size \( |V| \times d \) in which the row vectors representing similar property nodes are of high similarity.

Similar property oriented feature learning can be applicable to phishing detection, with the biased sampling strategy exploring PSPN.

**IV. URL QUALITY SCORE**

To provide the sampling bias for similar property oriented random walk, the calculation of URL quality scores must satisfy three requirements. 1). Capturing the major features of a webpage’s URL to predict whether it is malicious; 2). Being applied to the weblink network nodes to develop similar property oriented random walks; 3). Being pre-computed before the sampling phase. We adapt URLPatternMining [23] to calculate URL quality score, which satisfies all the three requirements.

As discussed earlier, URLPatternMining [23] dynamically extracts URL patterns from substantial quantities of URLs, without any pre-defined items. It provides relatively robust features that cannot be inferred by any existing frequent pattern mining methods. Hence, we follow this method and make minor changes for the calculation of URL quality score, which satisfies all the three requirements.

According to the URL specification[37], we divide a collection of URLs into three collections of URL segments: domain, directory, and file. For example, a given URL “https://www.airport-information.com/website/index.php/en/reference-list” is divided into three segments: domain segment “www.airport-information.com”, directory segment “website/index.php/en”, and file segment “reference-list”.

Following the notion of segment pattern [23], a particular segment pattern \( \rho = \varsigma_1 \varsigma_2 \ldots \varsigma_t \) with wildcard symbols “*” covers the URL segment \( \rho' = \varsigma_1' \varsigma_2' \ldots \varsigma_t' \). For instance, “*informa*.com” is a segment pattern. It covers URL segment “www.airport-information.com”. There is a function \( \Omega: [1, \eta] \rightarrow [1, \tau] \) satisfying: for all \( \forall \iota \in [1, \tau] \), if \( \varsigma_i \neq * \), then there exists a unique \( \epsilon \in [1, \eta] \) such that \( \Omega(\epsilon) = \iota \) and \( \varsigma'_{\iota} = \varsigma_{\epsilon} \).

To dynamically extract meaningful URL features in less time cost, we follow the data structure of tri-gram model based inverted index to find the segment patterns sharing common lexical features. A URL segment (e.g., “www.airport-information.com”) can be split into successive tri-grams. As shown in Figure 3, the left is its corresponding linked list of tri-grams, in which each item is composed of current tri-gram, one pointer pointing to next tri-gram, another pointer pointing to a linked list of segments corresponding to current tri-gram (e.g., “www”). The right in Figure 3 is a set of linked lists of segments. Each linked list of segments organizes the URL segments sharing at least one common tri-gram. Each item in a linked list of segments is composed of the sequence number of current segment (i.e., DomainId/DirectoryId/FileId), the position of the common tri-gram in current segment, one pointer pointing to next segment. In Figure 3, we enumerate four different segments. The DomainId of the domain segment “www.airport-information.com” is 56. The DomainId of “www.aircomsystem.com”, “bb-informa-cadas.com”, and “infighter.co” is 33, 83, and 92 respectively. In a linked list of segments, each item is organized by the increment of DomainId.

Over the segment sets sharing at least one identical tri-gram, maximal segment patterns [23] can be computed to cover such pairs. For two segments \( \rho_1 = \varsigma_1 \varsigma_2 \ldots \varsigma_{\eta_1} \) and \( \rho_2 = \varsigma_1' \varsigma_2' \ldots \varsigma_{\eta_2}' \) appearing in the same linked list of segments, we extract only one maximal segment pattern \( \text{MAX}_{\rho_1, \rho_2} \), by traversing their linked lists of tri-grams to record their common tri-grams, and common characters in nonidentical tri-grams. \( \text{MAX}_{\rho_1, \rho_2} \) can be computed following dynamic programming with function \( \Omega \) called. For the specific details, please refer to [23]. In this way, the maximal segment pattern (i.e., “*informa*.com”) is computed to cover the

---

1. http://www.w3.org/Addressing/URL/url-spec.txt
two domain segments (i.e., “bb-informa-cadas.com” and “airport-information.com”). Meanwhile, the maximal segment pattern (i.e., “*infighter.co*”) covering the other two domain segments (i.e., “bb-informa-cadas.com” and “infighter.co”) can be computed.

The maximal segment patterns over three collections of URL segments (i.e., domain, directory, and file) can be computed respectively. We denote the maximal segment patterns corresponding to domain, directory, and file as $\rho_d$, $\rho_r$, and $\rho_f$, respectively. The three maximal segment patterns can be concentrated to form a tuple representing a URL pattern, i.e., $\phi = (\rho_d, \rho_r, \rho_f)$. Given two URLs, a URL pattern is said to cover them if and only if each maximal segment pattern covers the corresponding URL segment. For instance, a URL pattern “*infights.co/csss/alimi/index.html” is said to cover two URLs “bb-informa-cadas.com/cl/alibaba/login.html” and “infighter.co/csss/alimi/index.h-tml”.

Mining URL patterns has been proven to capture the major features of annotated URLs effectively in [23]. The malicious and legitimate URL patterns can be said to cover malicious URLs and legitimate URLs respectively. We denote the number of malicious URL patterns as $\Phi_m$, and legitimate URL patterns as $\Phi_w$. For each weblink network node, we analyze whether the computed URL patterns cover its URL. Denoting the number of malicious URL patterns covering the URL as $\Psi_m$, the malicious probability is $\frac{\Psi_m}{\Phi_m}$. Similarly, the legitimate probability is $\frac{\Psi_w}{\Phi_w}$, with $\Psi_w$ denoting the number of the legitimate URL patterns covering the URL. We calculate the malicious probability ratio to assign URL quality score $\phi$ to each node, i.e.,

$$\phi = \frac{\Psi_m}{\Phi_m} / \frac{\Psi_w}{\Phi_w}$$  \hspace{1cm} (1)

Notice that higher malicious probability and lower legitimate probability produce a higher value of URL quality score. The higher score indicates that the URL is more likely to be malicious. The underlying intuition is that a malicious URL should match much more malicious data and less legitimate data [23]. To determine a quality score threshold for measuring the malicious degrees, we review the experimental results of URLPatternMining. As the quality score threshold increases from 10 to 1000, more and more malicious URLs are mistaken as legitimate. Considering malicious nodes produce the camouflage effect to evade detection, we set the quality score threshold to be 10. This specifies the value interval of URL quality scores of malicious nodes (i.e., $\geq 10$).

With the URL quality scores and the determined threshold pre-computed, the application to the weblink network nodes provides the sampling bias for similar property oriented random walk.

V. SPWALK MODEL

Having provided URL quality score of each node in a weblink network, can our model learn node representations conforming to similar property to improve phishing detection performance? To answer this, we present a novel feature learning model called “SPWalk”, which implements similar property oriented feature learning.

A. MODEL OVERVIEW

The task of similar property oriented feature learning is to map the weblink network nodes into the low-dimensional embeddings. The learned node representations can be used as the numeric features for phishing detection. To this end, we design a novel sampling strategy that combines URL quality scores of weblink network nodes with reference relationships between nodes.

A desirable sampling strategy aims to maximize the probability of a legitimate/malicious node co-occurring with other legitimate/malicious nodes. In the weblink network, reference relationships between nodes provide both positive and negative effects. The positive effect comes from those reference relationships that connect legitimate node-pairs. Moreover, there exist no reference relationships from legitimate nodes to malicious nodes. The co-occurring probability of legitimate node-pairs can be accordingly increased. The negative effect comes from many reference relationships from malicious nodes to legitimate nodes. The probability of malicious nodes co-occurring with legitimate nodes can also be increased. There exists the potential to put malicious nodes into the category of legitimate nodes. Hence, we compute the sampling bias based on URL quality scores of network nodes. Under the sampling bias, our sampling strategy maximizes the co-occurring probability of similar property nodes, through similar property oriented random walks.

Remarkably, the accuracy of URL quality scores relies heavily on the quality of URL patterns. The resulting impact can be reduced in our sampling strategy for two reasons. First, URL quality scores of a few nodes are not in line with structural regularities generated by reference relationships of webpage elements. This may point out which nodes are with inaccurate URL quality scores. Second, we integrate URL quality scores into the biased random walks, instead of a straightforward comparison between URL quality scores and the quality score threshold. The learned feature representations are not directly dependent on specific information about webpage URLs.
In general, the SPWalk model fully integrates webpage URLs and reference relationships, and makes up for their shortcomings. In doing this, the learned node representations are effective and robust in identifying similar property nodes to improve phishing detection performance.

B. THE EXPLORATION OF PSPN

The key to implementing similar property oriented feature learning is our sampling strategy. The core idea of our sampling strategy is to explore possible similar property neighborhoods.

We first provide an understanding of DeepWalk’s uniform sampling strategy from a probabilistic perspective. Formally, given a starting node \( v \), we simulate a truncated random walk of fixed length \( l \). The generated node sequence is denoted as \( c_1 \ c_2 \ldots \ c_l \). With starting node \( v = c_0 \), let \( c_i \) denote the \( i \)-th node in the walk. The probability distribution of the transition from \( c_{i-1} \) to \( c_i \) is defined as:

\[
P (c_i = x|c_{i-1} = v) = \frac{T_{vx}}{D} \tag{2}
\]

Here \( T_{vx} \) is the unnormalized transition probability from \( v \) to \( x \), and \( D \) is the normalizing constant. In DeepWalk, we set \( T_{vx} \) to be the static edge weight (i.e., \( T_{vx} = w_{vx} \)), and set \( D \) to the out-degree of node \( v \).

To explore possible similar property neighborhood of the current node \( v \), we re-define the transition probability to bias our random walks. The key to transition probability calculation is to comprise of URL quality score of each node. As described in Section IV, the value interval of URL quality scores is specified for similar property nodes. Accordingly, we argue that the transition probability on edge \((v, x)\) increases, if two nodes \( v \) and \( x \) are within the same interval of URL quality scores. With the quality score threshold (i.e., 10), we define the similarity measurement over a specific \((v, x)\):

\[
sm_{v,x} = \begin{cases} 
1 - \frac{|\max(0, \phi_v) - \max(0, \phi_x)|}{\phi_{max} - 10} & \text{if}\ max(0, \phi_v) = 10 \\
\frac{\phi_v}{\max(0, \phi_x)} & \text{otherwise}
\end{cases} \tag{3}
\]

Here, \( \phi_{max} \) is the maximum value from URL quality scores of all weblink network nodes. We consider two cases that the node \( x \) is a legitimate or malicious node. In the former one, the value of \( sm_{v,x} \) is close to 1 if URL quality scores of \( v \) and \( x \) are both below the quality score threshold. At the same time, the value of \( sm_{v,x} \) is reduced, if URL quality score of \( v \) indicates a malicious node. The latter case makes the value of \( sm_{v,x} \) approximate to 1 if two URL quality scores are both within the value interval of malicious nodes (i.e., \( \geq 10 \)). As there are no edges from legitimate nodes to malicious nodes, we discard the value of \( sm_{v,x} \) when URL quality score of \( v \) indicates a legitimate node. We can observe that the value of \( sm_{v,x} \) is maximized when \( x \) is within \( v \)'s PSPN.

The similarity measurement of node-pairs can be regarded as the sampling bias used in our random walks. What is more, a higher value of \( sm_{v,x} \) indicates the higher probability of the transition from node \( v \) to node \( x \). Thus, we introduce a new definition of the normalized transition probability from \( v \) to \( x \):

\[
P (c_i = x|c_{i-1} = v) = \frac{T_{vx}}{D} = sm_{v,x} \cdot \frac{1}{D} \tag{4}
\]

As the edge weight takes binary values, the transition probability from \( v \) to \( x \) is proportional to the value of \( sm_{v,x} \).

C. LOSS FUNCTION ON WEBLINK NETWORK

Similar property oriented feature learning can be regarded as an optimization problem which maximizes the log-probability of observing a PSPN \((v)\) conditioned on \( v \)'s feature representation \( f(v) \). Accordingly, our loss function is defined as follows:

\[
L = -\sum_{v\in V} \log Pr(PSPN(v)|f(v)) \tag{5}
\]

where

\[
Pr(PSPN(v)|f(v)) = \prod_{x\in PSPN(v)} Pr(x|f(v)) \tag{6}
\]

With neighborhood size \( k \), each member \( x \in PSPN(v) \) is equivalent to a context node of the current node \( v \) in SkipGram [32]. We use the softmax function to approximate the conditional probability of every current-PSPN node pair \((v, x)\):

\[
Pr(x|f(v)) = \frac{\exp(f(v) \cdot f(x))}{\sum_{\tilde{x} \in V} \exp(f(v) \cdot f(\tilde{x}))} \tag{7}
\]

Experimentally, the per-node partition function \( Z_v = \sum_{\tilde{x} \in V} \exp(f(v) \cdot f(\tilde{x})) \) results in expensive computational overhead. To optimize computational efficiency, we employ negative sampling [38] to define our loss function. This leads to a local loss defined over each current-PSPN node pair \((v, x)\):

\[
L(v, x) = -\log \sigma (f(v) \cdot f(x)) - \lambda E_{x' \sim P_d(V)} \left[ \log \sigma (-f(v) \cdot f(x')) \right] \tag{8}
\]

Here, \( f(x) \) is the feature representation of a positive sample, \( \sigma(\cdot) \) is sigmoid function, and \( \lambda \) is the number of negative samples. The term \( E_{x' \sim P_d(V)} [\cdot] \) is the expectation with a negative sample \( x' \) following the noise distribution \( P_d(V) \) proposed by Mikolov [39]. We optimize Equation 8 using stochastic gradient descent (SGD).

D. ALGORITHM

The pseudocode for similar property oriented feature learning is given in Algorithm 1. Furthermore, we detail the biased random walk procedure in Algorithm 2.

The preprocessing phase contains the procedures of URL quality score calculation, similarity measurement calculation, and the transition probability \( T \) calculation.
Algorithm 1 Feature Learning

Require: graph $G(V, E, W)$
- neighborhood size $k$
- embedding size $d$
- walks per node $r$
- walk length $l$

Ensure: matrix of node representations $f \in \mathbb{R}^{|V| \times d}$

1: $T = \text{Preprocess}(G, \text{sm}_{v_i})$
2: $G' = (V, E, T)$
3: Initialize walks to empty
4: for iter = 1 to $r$
5: \hspace{1em} $O = \text{Shuffle}(V)$
6: \hspace{1em} for each $v_i \in O$
7: \hspace{2em} $\text{walk} = \text{SPWalk}(G', v_i, l)$
8: \hspace{1em} Append walk to walks
9: end for
10: end for
11: $f = \text{StochasticGradientDescent}(k, d, \text{walks})$

Algorithm 2 SPWalk

Require: graph $G'(V, E, T)$
- start node $v_i$
- walk length $l$

Ensure: walk

1: Initialize walk to $[v_i]$
2: for step = 1 to $l$
3: \hspace{1em} $V_{curr} = \text{walk}[\text{step-1}]$
4: \hspace{1em} $N_{curr} = \text{GetNeighbors}(curr, G')$
5: \hspace{1em} $x = \text{AliasSample}(N_{curr}, T)$
6: \hspace{1em} Append $x$ to walk
7: end for

(Line 1-2, Algorithm 1). Then, the sampling phase simulates $r$-iterations of random walks, each starting from every node (Line 3-10, Algorithm 1). At each step within fixed length $l$, our sampling strategy comprises of the computed transition probabilities $T$ to sample next node from node neighbors $N_{curr}$, using alias sampling (Line 5, Algorithm 2). With node sequences generated by similar property oriented random walk, the optimization phase is subsequently executed to learn $d$-dimensional feature representations. Stochastic gradient descent (SGD) is used to minimize the local loss in Equation 8 (Line 11, Algorithm 1). Transition probability pre-computing, biased sampling and optimization are parallelizable and executed asynchronously, which contributes to the scalability of SPWalk.

VI. EXPERIMENTS

To investigate the effectiveness of the node representations learned by SPWalk in phishing detection, we conduct experiments of node classification on the weblink network.

A. DATASET

We collect the annotated URLs used in URL quality score calculation from December 2016 to June 2017, including 0.5 million malicious URLs obtained from PhishTank [40] and OpenPhish [41], and 1 million legitimate URLs obtained from Alexa [42] and Dmoz [43].

To construct a weblink network, we collect real-world data through the inspection of online public information. Notice that SPWalk can be conducted over any hierarchical structure with the following steps:

Step 1. Collect 5000 phishing webpages as the zero layer $L_0$ of the weblink network, using the GNU Wget [44] tool. The widely available tool Wget is driven by an automated Python script. By downloading files from the specified URLs, Wget makes local copies of the webpages, including the HTML documents and page resource elements. To extract reference relationships between webpages, we extract plain text and hyperlinks from the downloaded HTML-files.

Step 2. Locate the initially associated webpage set as the first layer $L_1$. The $L_1$ data include: 1). 5,932 directly associated webpages pointed by the hyperlinks inside the HTML code of $L_0$; 2). 27,826 indirectly associated webpages sharing similar textual content with the webpages in $L_0$. For each webpage in $L_0$, we employ the TF-IDF algorithm [48] on its plain text to extract keywords as query data. From the top 10 Google search results, indirectly associated webpages containing all query words can be retrieved. The reference relationships consist of direct, indirect associations from 5000 $L_0$ webpages to 33,758 $L_1$.

Step 3. Build an initial weblink network $G = (V_0, E_0)$ with $L_0, L_1$ as $V_0$ and the reference relationships as $E_0$. There are multiple dense subgraphs in $G$ with few relationships connecting each other, each consisting of the webpages with content correlation. Then, we expand the initial set to collect the $L_2$ webpages directly linked to (or by) $L_1$. Intuitively, the newly-collected webpages in $L_2$ introduce the reference relationships from $L_1$ to $L_2$. Remarkably, there exists an intersection between the newly-collected webpages and $V_0$. The overlapping webpages introduce the reference relationships from $L_1$ to $L_0$, and the reference relationships between different webpages in $L_1$. We gather 140,542 $L_2$ webpages, and expand $V_0$ to include $L_2$. $E_0$ can also be expanded to include the introduced reference relationships. Next, we repeat the expansion process one more time to add $L_3$ to form the complete node set $V$ and edge set $E$. To increase the density of the resulted network, we expand $E$ to include additional edges. Additional edges connect those node-pairs that point to the same node. The weblink network $G$ is a large-scale network without structural sparsity.

Based on the above process, a specific part of our weblink network is shown in Figure 4. Table 1 summarizes the statistics of our weblink network. The ground truth can be obtained from four well-known sources (PhishTank, OpenPhish, Alexa, and Dmoz). There are 78,368 malicious nodes and 761,688 legitimate nodes in our dataset. Malicious nodes
camouflage their identities, mainly through direct associations with legitimate nodes.

### B. BASELINE METHODS

For a node binary classification task, we compare the performance of SPWalk against two network embedding models (i.e., DeepWalk, Node2vec) and two traditional phishing detection methods.

- **Network embedding models.**
  - DeepWalk [13] first learns $d$-dimensional node representations, using uniform random walks. For each node-pair $(v, x)$, its sampling strategy is a special case of SPWalk with $T_{vx} = w_{vx}$ and $sm_{v,x} = 1$.
  - Node2vec [33] employs a 2nd order random walk procedure to learn $d$-dimensional node representations. With two parameters $p$ and $q$, it discovers diverse neighborhoods to implement a mixture of breadth-first search and depth-first search. In the case of $p = 1$ and $q = 1$, Node2vec is equivalent to DeepWalk.

- **Traditional phishing detection methods.**
  - URLPatternMining [23] is a typical target-independent method for phishing detection, capturing malicious URLs. It dynamically extracts URL patterns without any pre-defined features. As discussed in Section IV, this method can be adjusted to calculate the URL quality score of each node in the weblink network.
  - WebsiteLogo [27] is a typical target-dependent method. With the extracted webpage logo as input of Google image search, it retrieves phishing targets from online search results, and conducts a comparison between the target and real identity of the suspect webpage. The consistency indicates a normal webpage and inconsistency indicates a phishing webpage.

We exclude DeltaPhish [28] and New rule-based method [20]. The former requires the homepage hosted in the same domain of the suspect webpage. Many malicious nodes represent newly-created phishing webpages without normal homepages as potential phishing targets. The latter requires the correlation between its classification feature and internet banking, which limits its detection range.

### C. EXPERIMENTAL SETTINGS

To compare SPWalk’s performance in node binary classification with the other two network embedding models, we consider the parameter settings as follows. First, in the sampling phase, DeepWalk, Node2vec and SPWalk generate an equal number of samples $K$ for a fair comparison between the three models. The parameter settings satisfy $K = r \cdot l \cdot |V|$. With the fixed number of network nodes $|V|$, we follow the setting parameters in DeepWalk and Node2vec, i.e., $r = 10$, $l = 80$. The other two
parameters (neighborhood size $k$ and dimension $d$) used for SPWalk are in line with typical values used for DeepWalk, Node2vec, i.e., $d = 128$, $k = 10$. Worth remarking, there are two key parameters ($p$, $q$) used for the sampling strategy in Node2vec. The following analysis helps us identify the optimal parameter settings that maximize the likelihood to locate similar property nodes. In a weblink network, those webpages with similar subject-matter form the node cluster with dense intra-connections. Within such node clusters, nodes are highly intra-connected, which is in line with the notion of homophily [33]. However, the nodes with the similar property do not exhibit homophily, since the phishing webpages without content correlation have few reference relationships with each other. Thus, we follow the notion of structural equivalence [33] which is based on similar structural roles of the nodes with the similar property. Specifically, there exists a phenomenon that many malicious nodes have higher out-degrees and lower in-degrees with comparison to legitimate nodes. Such malicious nodes frequently point to dense subgraphs mainly consisting of legitimate nodes. In contrast, there are only few edges in the opposition direction. The difference in structural connectivity makes such malicious nodes similar to peripheral nodes. Meanwhile, many legitimate nodes play the roles of core nodes. Accordingly, we use $p = 1, q = 4$ to discover different structural roles played by two collections of similar property nodes.

Then, the three network embedding models can be optimized using SGD with one key difference. DeepWalk uses hierarchical softmax to update output vector representations, while SPWalk and Node2vec use negative sampling to approximate the softmax probabilities [45]. For fair comparisons, we switch to negative sampling to speed up the feature learning of DeepWalk. For each model, we set the number of negative samples as 5, learning rate of starting value as 0.025, the mini-batch size of stochastic gradient descent as 1. The node representations obtained through unsupervised feature learning are input to a logistic regression classifier with L2 regularization implemented by sklearn [46]. For phishing detection, node binary classification task can be solved in a semi-supervised way. With the varying amount of network nodes used as training, the classification task can be solved in a semi-supervised way. With the varying amount of network nodes used as training, the remaining nodes are used as test to evaluate the classification performance.

To compare SPWalk’s phishing detection effect with traditional phishing detection methods, we consider 10-fold cross-validation to evaluate the detection performance. We randomly partition the weblink network nodes into 10 equally-sized subsets, each subset having the same percentage of malicious nodes as the complete dataset. In each fold, we train a binary classifier using 9 subsets, and test it using the remaining subset. We repeat this process 10 times, using each subset as test data. Finally, we combine the results of 10 folds, providing the mean of false positives and false negatives.

**D. EXPERIMENTAL RESULTS**

We analyze the experimental results through 1) applying network embedding models to a node binary classification task; 2) reproducing the detection effect of phishing detection methods.

1) EXPERIMENTAL RESULTS OF NETWORK EMBEDDING MODELS

We use the precision, recall, and F1-scores for the performance evaluation of network embedding models. To be specific, we randomly sample 10%-90% of weblink network nodes as training data, with the remaining nodes as test data. The varying proportion of training data (i.e., 10%-90%) covers a wide range of labeled data. Under different proportions of labeled data, the classification results can effectively reflect the efficacy of network embedding models. Extremely low (i.e., $<10\%$) and extremely high (i.e., $>90\%$) proportions are not conducive to reflect the performance disparity of network embedding models. We repeat this process 10 times to report the average performance.

From the results shown in Figure 5, we can see that SPWalk performs consistently better than DeepWalk and Node2vec, even with only 10% of network nodes labeled. To explain the superiority of SPWalk, we first analyze the uniform random walk of DeepWalk. Applying DeepWalk to a weblink network is not a complete failure. The reason is that the weblink network preserves co-occurrence relationships between legitimate nodes. Specifically, about 90% of the network nodes are legitimate nodes. Each legitimate node only provides directed edges connecting to other legitimate nodes, instead of malicious nodes. There are a large amount of directed edges that connect legitimate node-pairs to preserve legitimate node co-occurrences. It is conducive to embed legitimate nodes close together. Unfortunately, DeepWalk is unable to counteract the camouflage effect produced by phishing webpages, which makes it easy to mistake malicious nodes as legitimate.

Neighborhood exploration conforming to structural equivalence makes Node2vec relatively robust against the camouflage effect produced by a small number of malicious nodes. Malicious nodes with higher out-degrees point to legitimate nodes to produce the camouflage effect. Instead of providing directed edges back to malicious nodes, many legitimate nodes with higher out-degrees and in-degrees are connected with each other. By using metaphor, these malicious nodes may represent novel characters that are at the periphery and have limited interactions with the main characters. Conversely, legitimate nodes may play the roles of main characters that frequently interact with each other. Although this improves the classification performance, the problem brought by the camouflage effect remains unresolved. Not all similar property nodes play the similar roles in terms of structural connectivity. Many malicious nodes with similar subject-matter share webpage elements with each other to increase their in-degrees, which makes them less similar to
peripheral nodes. Hence, DeepWalk and Node2vec are not suitable for phishing detection.

SPWalk minimizes the camouflage effect produced by malicious nodes, through the biased random walks towards possible similar property neighborhood of the current node. The minimized camouflage effect explains why SPWalk achieves higher precision (over 95%), recall (around 84%) and F1-score. Compared to other feature learning models, the sampling bias derived from URL quality scores brings the unique advantage of phishing detection to SPWalk. Note that all three models have higher precisions than their recalls. Their precisions are enhanced by classifying legitimate nodes correctly. Their recalls are reduced by the camouflage effect that produces a tendency to misclassify malicious nodes as legitimate. The biased random walks of SPWalk add the robustness against the camouflage effect produced by the majority of malicious nodes.

2) EXPERIMENTAL RESULTS OF PHISHING DETECTION METHODS

Here, we thoroughly evaluate the detection effect of SPWalk against URLPatternMining and WebsiteLogo, with the weblink network nodes as test data.

Table 2 summarizes the phishing detection results of the three phishing detection methods, which demonstrates the relatively significant performance gain of SPWalk. The lowest false positive error of SPWalk (4.75%) is mainly due to structural regularities generated by reference relationships. In terms of false negative, SPWalk has a slight advantage (0.22%) over URLPatternMining.

Comparative results of SPWalk and URLPattern-Mining indicate the contribution of reference relationships to the high-quality feature learning. SPWalk has its unique advantage to classify legitimate nodes correctly, through preserving legitimate node co-occurrences. On the other hand, URLPatternMining is adapted to compute URL quality scores, based on which SPWalk simulates the biased random walks. The sampling bias is conducive to classify malicious nodes correctly, which is verified in the performance comparison. Remarkably, a small fraction (around 16%) of malicious nodes are assigned with excessively low URL quality scores. This leads to a slight decline in the effectiveness of the sampling bias.

Compared with the results of WebsiteLogo, SPWalk demonstrates its unique advantage in terms of false positive. The reason behind this is the same as the above analysis. The inferiority of WebsiteLogo comes from the incredibility of third-party services (i.e., Google search engine). Compared to other target-dependent methods, website logo can be a more effective query to increase the accuracy of retrieving phishing targets. SPWalk exhibits a slight advantage over WebsiteLogo in false negatives (0.72%).

3) SCALABILITY

To test for scalability, we learn node representations using network embedding models for the weblink network with increasing sizes from $10^5$ to $10^6$ nodes. In Figure 6, we empirically observe that DeepWalk, Node2vec and SPWalk scale linearly with the increasing number of nodes. These three models generate representations for one million nodes in around three hours. Note that the y axis denotes the running time comprising of the sampling and optimization phrases in the log scale. The time cost for the preprocessing phase is not counted.
The time complexity of SPWalk and Node2vec is slightly expensive than that of DeepWalk. This is due to the time overhead introduced by calculating transition probabilities. For SPWalk, the sampling phase simulates biased random walks efficiently in \(O(1)\) time, using alias sampling. The optimization phase generates representations for all weblink network nodes in an efficient way, using negative sampling and asynchronous SGD. Hence, SPWalk can scale to large networks with millions of nodes.

4) THRESHOLD ANALYSIS FOR URL QUALITY SCORE
In preprocessing phase (i.e., Section IV), we implement URL quality score of each network node as malicious probability ratio. We also specify the value interval of URL quality scores of malicious nodes, by setting different quality score threshold in the experiments. Taking the setting threshold as the classifying criterion, we evaluate the classification performance obtained from the preprocessing phase in Figure 7.

From Figure 7, we can observe that setting the threshold as 10 achieves the optimal performance, when jointly considering precisions and recalls. As URL quality score threshold increases, the precision increases slightly, and the recall is decreasing sharply. The higher threshold is conductive to classify legitimate webpages correctly, reducing the false positive error. However, many malicious webpages may be classified as legitimate ones under higher thresholds, which increases false negative error. Overall, setting the quality score threshold as 10 is conductive for SPWalk to improve the classification results.

E. EFFECTIVENESS ANALYSIS OF SAMPLING BIAS
Our encouraging experimental results demonstrate the efficacy of SPWalk in phishing detection. Meanwhile, the effectiveness of the sampling bias is still a matter of discussion. The effectiveness relies much on the accuracy of URL quality scores. A small percentage of malicious nodes may be assigned excessively low URL quality scores. These malicious nodes introduce many directed edges pointing to legitimate nodes, which produces higher values of similarity measurement over such edges. The reduced effectiveness of the sampling bias increases the transition probabilities from malicious to legitimate nodes. The co-occurrence relationships between malicious and legitimate nodes in the same walk produce false negatives shown in Table 2.

We discuss how the effectiveness of the sampling bias can effect the performance of SPWalk. The sampling bias can be derived from URL quality scores. In real applications, phishing attackers can carefully construct malicious URLs with excessively low URL quality scores. Here, we simulate one worst-case scenario in which a certain proportion of malicious nodes are assigned with inaccurate URL quality scores. A fraction of malicious nodes can be randomly selected with URL quality scores randomly sampled from a uniform distribution in \([0, 10]\). Based on inaccurate URL quality scores, the effectiveness of sampling bias is reduced. We measure the resulting performance as a function of the fraction of randomly selected malicious nodes (relative to all malicious nodes). Here, we abbreviate SPWalk, URLPatternMining in a worst-case scenario as SPWalk-W, URLPatternMining-W, respectively. The recalls are summarized graphically in Figure 8.

The dynamically extracted URL patterns are difficult to be inferred using existing frequent pattern mining methods. Consequently, phishing attackers cannot guarantee that well-crafted URLs obtain URL quality scores below the threshold. To simulate such cases, we properly increase the proportion of undetected malicious nodes (around 16%) in SPWalk. The fraction can be within the range of 26%-36%. As we can see in Figure 8, the decrease in the recall of URLPatternMining-W as the increment of the fraction is roughly linear. Compared to the poor robustness of the classification rules in URLPatternMining-W, SPWalk-W is more robust against inaccurate URL quality scores. The advantages of robustness are more and more obvious as the fraction increases from 31% to 36%. At the same time, the recall rate remains approximately 70%. The recalls of SPWalk-W largely reflects the changes in the effectiveness of the sampling bias.

To delve into the better robustness of SPWalk-W, let us review the detection results shown in Table 2. When the fraction of malicious nodes with low URL quality scores is below 20%, the reduced effectiveness of the sampling bias lowers the performance of SPWalk. Fortunately, the performance decline of SPWalk becomes more and more flat, with the increasing fraction from 26% to 36%. This is because of the following case. Given two malicious nodes linked by an edge, they may both have excessively low URL quality scores.
Consequently, the transition probability on the corresponding edge is increased, which helps SPWalk classify malicious nodes correctly. Moreover, higher fractions make the logistic regression classifier aware of inaccurate URL quality scores of partial malicious nodes. By adjusting the weights assigned to different numeric features, the decision function depends less on the edges connecting malicious and legitimate nodes.

Even with the decline in the effectiveness of the sampling bias, SPWalk exhibits better performance than URLPatternMiner. This comes from the robustness against inaccurate URL quality scores.

**VII. CONCLUSION**

In this paper, SPWalk, a novel feature learning model for phishing detection is proposed. Our model, with similar property oriented random walks integrating specific information about cyber security into a weblink network structure, captures robust and effective feature representations for phishing detection. The notion of similar property nodes specifies the collection of either phishing webpages or normal webpages. To locate similar property nodes, the structural regularities generated by reference relationships and webpage URL information can be leveraged to provide new flexibility and capability on neighborhood exploration. The learned node representations with phishing oriented characteristics are of high applicability, effectiveness in phishing detection.

We empirically demonstrate the unique advantages of SPWalk over relevant baselines. In order to delve into the impact of the sampling bias, we specifically perform a perturbation analysis demonstrating the robustness of our model against well-crafted malicious URLs. In conclusion, we believe this study provides an interesting direction to efficiently learn feature representations suited for phishing detection.

**REFERENCES**

[1] D. Liu, S. Hao, and H. Wang, “All your DNS records point to us,” in Proc. Conf. Secur. Symp., Oct. 2016, pp. 1–15.

[2] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, “PhishNet: Predictive blacklisting to detect phishing attacks,” in Proc. IEEE INFOCOM, Mar. 2010, pp. 1–5.

[3] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy, “A crawler-based study of malware in the Web,” in Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS), San Diego, CA, USA, 2006, pp. 18–32.

[4] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs,” in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Paris, France, Jul. 2009, pp. 1245–1254.

[5] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Identifying suspicious URLs: An application of large-scale online learning,” in Proc. Int. Conf. Mach. Learn., 2009, pp. 681–688.

[6] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, “Design and evaluation of a real-time URL spam filtering service,” in Proc. IEEE Symp. Secur. Privacy, May 2011, pp. 447–462.

[7] M. Aburrous, M. A. Hassain, K. Dahal, and F. Thabtah, “Intelligent phishing detection system for e-banking using fuzzy data mining,” Expert Syst. Appl. Int., vol. 37, no. 12, pp. 7913–7921, Dec. 2010.

[8] C. Whittaker, B. Ryner, and M. Nazif, “Large-scale automatic classification of phishing pages,” in Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS), San Diego, CA, USA, Feb./Mar. 2010, pp. 1–14.

[9] A. Le, A. Markopoulou, and M. Faloutsos, “PhishDef: URL names say it all,” in Proc. IEEE INFOCOM, Apr. 2011, pp. 191–195.
[35] O. Levy and Y. Goldberg, “Neural word embedding as implicit matrix factorization,” Advances in Neural Information Processing Systems, vol. 3, pp. 2177–2185, 2014.

[36] C. Tu, H. Liu, Z. Liu, and M. Sun, “CANE: Context-aware network embedding for relation modeling,” in Proc. 55th Annu. Meeting Assoc. Comput. Linguistic, 2017, pp. 1722–1731.

[37] T. Berners-Lee. Uniform Resource Locators (URL). Accessed: May 5, 2017. [Online]. Available: http://www.w3.org/Addressing/URL/url-spec.txt

[38] Y. Goldberg and O. Levy, “word2vec explained: Deriving Mikolov et al.’s negative-sampling word-embedding method,” 2014, arXiv:1402.3722. [Online]. Available: https://arxiv.org/abs/1402.3722

[39] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[40] PhishTank. Phishtank Developer Information. Accessed: Jun. 23, 2017. [Online]. Available: https://www.phishtank.com/

[41] Openphish. Phishing Intelligence Feeds. Accessed: Aug. 9, 2017. [Online]. Available: https://openphish.com/

[42] Alexa. Alexa the Web Information Company. Accessed: Sep. 7, 2017. [Online]. Available: http://www.alexa.cn/

[43] Dmoz. Open Directory Project. Accessed: Oct. 10, 2017. [Online]. Available: http://www.dmoz.org/

[44] R. Stallman. GNU Wget. Accessed: Oct. 31, 2017. [Online]. Available: http://www.gnu.org/software/wget/

[45] A. Mnih and G. Hinton, “A scalable hierarchical distributed language model,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2008, pp. 1081–1088.

[46] F. Pedregosa, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Pret-tenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, “Scikit-learn: Machine learning in pythonGnu Wget,” J. Mach. Learn. Res., vol. 12, no. 10, pp. 2825–2830, 2016.

[47] L. Qiao, H. Zhao, X. Huang, K. Li, and E. Chen, “A structure-enriched neural network for network embedding,” Expert Syst. Appl., vol. 117, pp. 300–311, Mar. 2019.

[48] Z. Zhu, J. Liang, D. Li, H. Yu, and G. Liu, “Hot topic detection based on a refined TF-IDF algorithm,” IEEE Access, vol. 7, pp. 26996–27007, 2019.

XIUWEN LIU received the bachelor’s degree in computer science from Yunnan University, in 2013. She is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Wuhan University. Her research interests include cyber security and web data mining.

JIANMING FU received the bachelor’s and master’s degrees from the Huazhong University of Science and Technology, in 1994 and 1991, respectively, and the Ph.D. degree from Wuhan University, China, in 2010. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. His research interests include software security, cyber security, and web data mining.

* * *

VOLUME 8, 2020