doi: 10.20532/cit.2020.1004899

A Detection Method for Phishing Web Page Using DOM-Based Doc2Vec Model

Jian Feng[1], Ying Zhang[2] and Yuqiang Qiao[1]

[1]Xi'an University of Science and Technology, Xi'an, China
[2]Information Technology Department for Head Office of SPD Bank, Xi'an, China

Detecting phishing web pages is a challenging task. The existing detection method for phishing web page based on DOM (Document Object Model) is mainly aiming at obtaining structural characteristics but ignores the overall representation of web pages and the semantic information that HTML tags may have. This paper regards DOMs as a natural language with Doc2Vec model and learns the structural semantics automatically to detect phishing web pages. Firstly, the DOM structure of the obtained web page is parsed to construct the DOM tree, then the Doc2Vec model is used to vectorize the DOM tree, and to measure the semantic similarity in web pages by the distance between different DOM vectors. Finally, the hierarchical clustering method is used to implement clustering of web pages. Experiments show that the method proposed in the paper achieves higher recall and precision for phishing classification, compared to DOM-based structural clustering method and TF-IDF-based semantic clustering method. The result shows that using Paragraph Vector is effective on DOM in a linguistic approach.

ACM CCS (2012) Classification: Security and privacy → Intrusion/anomaly detection and malware mitigation → Social engineering attacks → Phishing

Keywords: phishing detection, semantic similarity, Doc2Vec, DOM, clustering

1. Introduction

Phishing is a fraudulent attempt to obtain sensitive information such as usernames, passwords, and credit card details by disguising as a trustworthy entity in an electronic communication [1]. The PhishLabs reports that total phishing volume raised 40.9% in 2018 compared to 2017 [2]. The continued growth of phishing attacks has a huge negative impact on healthy development of the Internet and has become one of the most serious security threats to the Internet.

The approaches that tackle the task of phishing web page identification with the machine-learning perspective often views the task as a binary classification problem, so various classification algorithms are used, including Naive Bayes, SVM and deep learning [3-5]. At the same time, a small number of studies have regarded the detection of phishing web pages as clustering problems [6,7] and clustered according to the structural similarity in DOM (Document Object Model). The discriminative accuracy of the clustering method is lower than that of the classification method, but the unsupervised learning method is more in line with the human cognitive model, so it also has its research value.

In recent years, to better represent the textual content and internal semantic relationships of a document, different techniques for word embedding are proposed. Their potential for capturing the semantics of the plain text by dense vectors was demonstrated successfully by the past efforts like [8]. The work was extended to represent words, sentences, and documents efficiently in Doc2Vec [9]. These studies led to the author's thinking: can the idea of semantic analysis through word/document embedding be used with the clustering of DOM structures in order to achieve better results of phishing classification? The paper conducts research from this perspective.

DOMs of HTML (HyperText Markup Language) consist of HTML tags and their attributes, and can be seen as semi-structured documents. Unlike natural language, although HTML tags can be barely viewed as text, their original design goal was to express the page layout of HTML documents without giving semantics. So, whether DOM can be treated as text while HTML tags in DOM are treated as words and whether word/document embedding methods can be used to analyze their semantics, there is no previous research. However, based on the analysis of the existing literature, it is found that some studies have made other non-text content processing as text and achieved good results [10, 11]. Therefore, we try to consider DOM written in natural language, and attempt to learn the difference between the benign web page and phishing web page automatically through semantic analysis of DOM.

In this paper, a new approach for phishing web page detection is proposed. Firstly, we extract DOM structure of web page, and then use Doc2Vec model to learn its representation. After that, web pages are clustered by comparing the similarity in different web page vectors, and finally the obtained clustering categories are labeled as the basis for judging the web page to be tested.

The main contributions of this paper are, as follows.

- We proposed a novel method for detecting phishing web page based on semantic analysis of DOM, which is better than the existing ones.
- Utilized Doc2Vec models to generate vector representations to DOM.
- Verified that using Doc2Vec is effective on DOM in a linguistic approach.

Experiments show that our method achieves the accuracy higher than those achieved by the existing methods.

The rest of the paper is organized as follows. The next section briefly discusses related works and document representation methods. Section 3 proposes a phishing detection method based on the Doc2Vec. Section 4 describes the conducted experiments and presents the results and their evaluation. Section 5 draws the conclusions and points in the possible directions for future work.

2. Related Works

There are a lot of studies that typically regard phishing detection as a classification problem, including blacklist method, heuristic method, machine learning method, *etc.* However, some studies regard phishing web page detection as clustering problem, all of them which is based on the DOM structure of a web page. But, in general, insufficient research has been done. At the same time, semantic analysis technology based on document representation has become the mainstream of natural language processing (NLP) technology. In order to study the possibility of the combination of the two, the following two aspects are combed.

2.1. Detection Methods Based on DOM

The main idea of solving the problem of detecting phishing web pages from the perspective of clustering is to compare the structure of web pages. Firstly, extract the structure of web pages to form the DOM tree, and then measure the similarity between DOM trees by various distance metrics. Finally, clustering is performed according to the similarity, and the category of the web page to be tested is determined by labeling the clustered classes. The study of such methods began from the literature [12]. The researchers extracted the DOM tree from the HTML source code and compared the similarities in DOMs in two ways, including a simple comparison of tags and isomorphic subgraph recognition. Among them, the tag comparison method compared tags in different DOMs one-by-one, which led to inefficient comparison.

Therefore, later researchers tend to map DOM structural features into simplified vectors and then compare them. For example, Cui *et al.* [6] proposed a statistical method of tags, hereinafter referred to as PD (Proportional Distance), which generates fixed-length vectors for each web page to compare the differences and reduce computational complexity of tag comparison. The literature [7] focuses on shallow nodes in DOM, constructs vectors for hierarchical tags of DOM to characterize the structural features of web pages and calculate differences, the algorithm is HD (Hierarchical Distance).

The above studies show that it is feasible to detect phishing web pages according to DOM structure from the structural point of view, but it still needs to be improved in terms of structural feature extraction, vector representation, and similarity calculation method. The detection accuracy needs to be improved too.

2.2. Document representation methods

As document classification is one of the main text mining tasks, many related studies have exhibited significant progress to date.

Bag-of-Words (BoW) model, first proposed by Z. Harris in 1954 [13], is among the most common and most efficient approaches for document representation. The method involves the assumption that a document is simply a collection of words and represents the document vector using occurrence and frequency of features. TF-IDF (Term Frequency-Inverse Document Frequency) has been the most adopted BoW model [14]. It provides each word of a document with a weight according to the following two criteria:

1. frequency of its usage in the specified document (TF) and
2. rarity of its appearance in other documents in the corpus (IDF).

Another popular document representation approach is topic modeling LDA (Latent Dirichlet Allocation) [15]. The main purpose of LDA is to discover latent themes that permeate the corpus. Once the LDA is trained, two outputs are generated: word distribution per topic and topic distribution per document. The latter can be regarded as another document representation in which both the word frequencies and semantic information (topic constitution) are considered. BoW and LDA consider the text structure and semantic relevance, but regard the text as a collection of words and ignore context information such as the order, grammar, and syntax of terms in the text, so they still have limitations on capturing the semantics of a document. Doc2Vec is the newest among the three document representation schemes [9], it is a natural extension of Word2Vec [8], and the latter is a semantic representation of words using vectors. Doc2Vec extends Word2Vec from the word level to the document level. The vector representations are learned to predict the surrounding words in the contexts sampled from the paragraph.

Compare to LDA, Doc2Vec has low computational complexity and can learn embedding vectors of every word rather than only get the distribution of topic words. Compared to BoW, and other neural network-based classifiers, Doc2Vec results in very promising accuracy in many natural language processing (NLP) tasks, *e.g.*, in text classification, sentiment analysis, information retrieval, *etc.* [16]. In particular, [17] transformed a document using TF-IDF, LDA, and Doc2Vec separately for document classification and, among the three methods, Doc2Vec got the best predicted result. But to the best of our knowledge, there is no previous work has been done on DOM using document embeddings. However, [11] reads network packets as a natural language, and uses Doc2Vec to learn the feature automatically to detect malicious traffic. In the literature [10], segments of the speech are regarded as "words", and the similarities in different speeches are analyzed by Doc2Vec. These works inspire us to adopt Doc2Vec to express DOM in a linguistic way and explore its application effect on the phishing web page detection.

3. Methodology

We propose a novel method that can detect phishing web pages by Doc2Vec model over DOM, hereinafter referred to as DoD. The systematic design of DoD is illustrated in Figure 1.

The input comprises a set of benign web pages and phishing web pages. Firstly, DoD constructs DOM corpora for these web pages. Then, DOM corpus is vectorized by Doc2Vec model, and a vector is gotten for each web page. These vectors are used for clustering according to semantic similarity. Each class is labeled according to the third-party open-source blacklist library and manual intervention. After that, we convert the web pages to be tested into vectors. These unlabeled vectors are testing data onto the classifier. Finally, we input these unlabeled vectors to the trained classifier and obtain a predicted label. The predicted label is either phishing or benign.

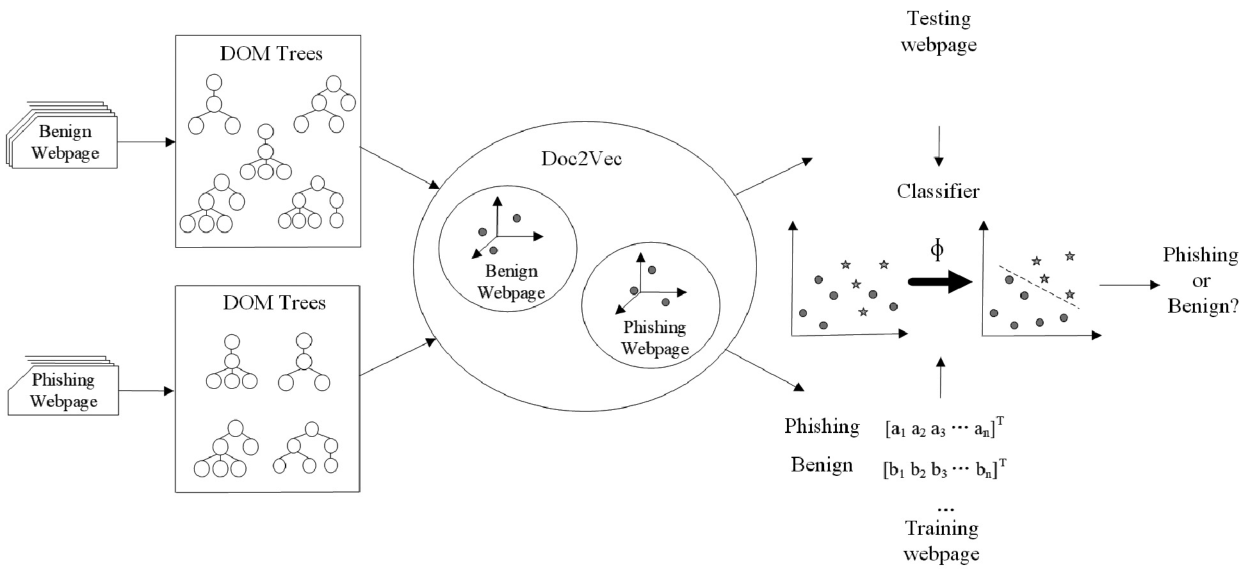


Figure 1. The architecture of DoD.

The key steps of DoD are, as follows.

1. To create DOM corpora through analyzing the input HTML files and generating the corresponding DOM tree for the HTML pages.
2. To make DOMs vectorized by Doc2Vec.
3. To compute the similarity in DOM vectors and clusters.

The above steps are described in detail in the following subsections.

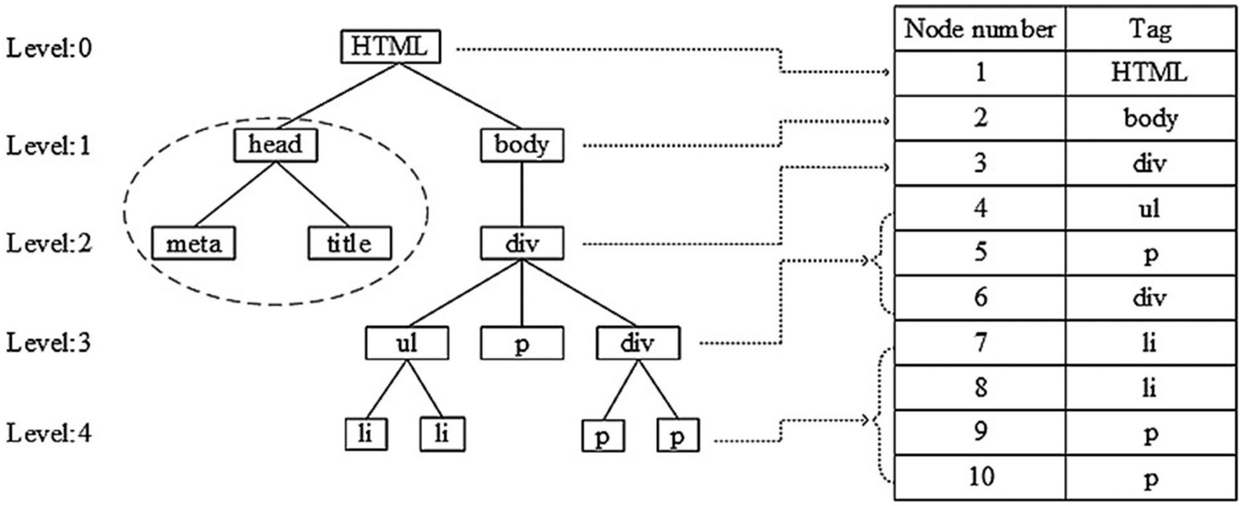
3.1. DOM Corpus Creation

An HTML document is a typical semi-structured document in which the tags have a nested relationship, which reflects the hierarchical structure of the web page, and can be characterized by the DOM tree. The DOM represents an HTML document as a tree structure with tags, attributes, and text nodes. For the sake of simplicity, we only use the DOM tag tree to represent an HTML document, ignoring attributes, texts, and comment nodes. The standard library of Python is used to iteratively obtain tags from the HTML page, and the DOM tag structure table is constructed. The table stores tag information about each layer of DOM tree. The construction steps are, as follows.

1. Acquire the content of the HTML document, parse the DOM tree of the document, obtain the root node of the tree and store it in the structure table as the current layer.
2. Starting from the current layer, traverse layer by layer with a breadth-first algorithm. The specific method is: traverse the child nodes of the nodes in the current layer from left to right, and save the tags to the structure table until all the child nodes are traversed.
3. Repeat 1. until all layers have been scanned, stop traversing, and return the structural table.

```
<!DOCTYPE HTML>
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <title>DOMtree</title>
  </head>
  <body>
    <div>
      <ul>
        <li>one</li>
        <li>two</li>
      </ul>
      <p>para</p>
      <div>
        <p>three</p>
        <p>four</p>
      </div>
    </div>
  </body>
</html>
```

Figure 2. HTML document.



Node number	Tag
1	HTML
2	body
3	div
4	ul
5	p
6	div
7	li
8	li
9	p
10	p

Figure 3. DOM tag tree and structure table of HTML document in Figure 2.

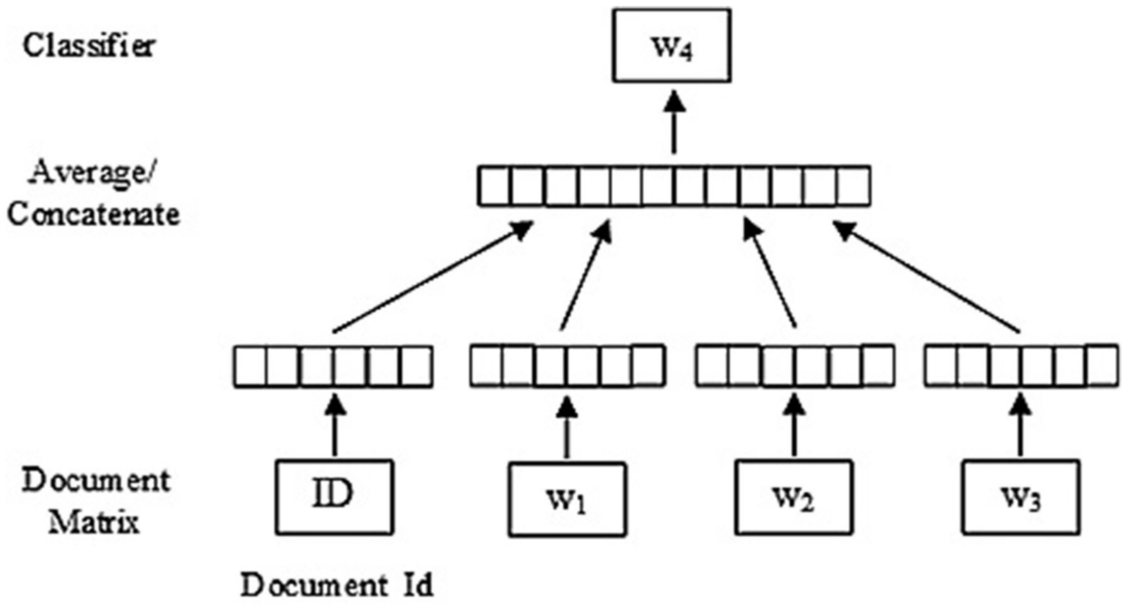


Figure 4. Framework for learning document vectors. Adapted from [9].

Figure 2 is an HTML document, which is converted into a corresponding DOM tag tree and stored in the structure table, shown in Figure 3.

Since the web page information is mainly within the body, we extract the part other than the head subtree (the dotted line in Figure 3). According to the construction steps of the structure table, the tags on the same level are stored in order from left to right.

After the above steps, a web page is translated into a DOM document, and HTML tags in the document can be seen as words.

3.2. DOM Vectorization

DOM vectorization is implemented through the Doc2Vec model. It learns features from the DOM corpus in an unsupervised manner. A framework for learning document vectors is shown in Figure 4.

Doc2Vec is, in fact, a generalization of Word2Vec achieved by extending the learning of embedding from words to word sequences. Word2Vec is a three-layer neural network of one input, one hidden and one output layer. There are two architectures including Continuous Bag of Words (CBOW) and Skip-gram implemented in Word2Vec. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. Take Skip-gram as an example: given a sequence of training words w_1, w_2, ..., w_T, the goal of Word2Vec is to maximize the predicted log probability as follows:

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p\left(W_t \mid W_{t-k}, ..., W_{t+k}\right), \quad (1)$$

where k is the window size for preserving the contextual information. The prediction is generally performed via multiclass classification using the softmax function as follows:

$$p\left(W_t \mid W_{t-k}, ..., W_{t+k}\right) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}, \quad (2)$$

where each y_i is the i-th output value of a feed-forward neural network computed using:

$$y = b + Uh\left(W_{t-k}, ..., W_{t+k}; W\right), \quad (3)$$

where b is the bias terms between the hidden and output layers, $\boldsymbol{U}$ is weight matrix between the hidden and output layers, h is the concatenation for context words, and $\boldsymbol{W}$ is word embedding matrix.

Doc2Vec extends Word2Vec framework by adding additional input nodes representing documents as additional context. Each additional node can be thought of just as an identifier for each input document. In Doc2Vec, document vector matrix $\boldsymbol{D}$ is a vector matrix for all documents in the training data, each document is mapped to a unique vector that is represented by a column in matrix $\boldsymbol{D}$, whereas each word is mapped to a unique vector that is represented by a column in matrix $\boldsymbol{W}$. Therefore, the only alteration in the network formulation is the addition of $\boldsymbol{D}$ in (4), as follows:

$$y = b + Uh\left(W_{t-k}, ..., W_{t+k}; W, D\right). \quad (4)$$

Doc2Vec utilizes two algorithms to produce a distributed representation of entire documents, namely Paragraph Vector. One is Distributed-Memory (DM), and the other is Distributed-Bag-of-Words (DBoW). DM is an extension of CBoW, and the only change in this model is adding a document ID as a window of the surrounding context words. DBoW is an extension of skip-gram, and the current word is replaced by the current document ID.

In this study, the word set is all HTML tags in each DOM document and document ID is the identifier for each HTML document. We used DM to train DOM corpus, and after the training process, we obtained W for HTML tags and D for all DOM documents.

3.3. Similarity Measurement and Clustering

The vectors generated by Doc2Vec can be used for finding semantic similarities between documents. We compute Manhattan distance between documents using the vectors produced by Doc2Vec to score the semantic distance between document pairs. Manhattan distance, also known as a city block, rectilinear or L1 distance, is mathematically defined as:

$$d_{man}\left(v_1, v_2\right) = \sum_{i=1}^{n}\left|v_{1,i} - v_{2,i}\right|. \quad (5)$$

Where v_1 and v_2 are vectors of two different web pages, n is the dimension of the vectors.

For the sake of comparison, DoD used an improved hierarchical clustering approach proposed in [7] to group vectors together. As usual with clustering algorithms, our method is based on a threshold H, which is also defined in [7] as quality of clustering.

The clusters are labeled after clustering. The web pages to be tested are classified according to the distance value between them and each class center, and they are labeled according to the label of the classes they belong to.

4. Experiments

To validate the method presented in this paper, we designed two sets of experiments to test its feasibility and validity.

4.1. Experimental Preparation

4.1.1. Experimental Environment

The experimental development environment is shown in Table 1. The program implements functions of web pages crawling, web pages preprocessing, DOM tree extraction, clustering, and web page classification. It uses several third-party libraries for python such as hashlib, BeautifulSoup, lxml, and so on.

4.1.2. Dataset

The dataset used in the experiments is obtained from the real network environment[1]. The benign web pages come from Alexa[2]. Alexa is a dedicated website managed by Amazon to publish an authoritative ranking of websites, so it has a large number of URLs and detailed ranking information. After filtering out some invalid, erroneous, and duplicate pages, we collected 10,922 benign web pages from Alexa.

The phishing web pages are from PhishTank.com[3]. PhishTank is an internationally renowned website that collects a timely and authoritative list of phishing web pages. We collected 10,944 phishing web pages listed on PhishTank from April 2016 to April 2017. In addition to pre-processing web pages that do not conform to grammar rules, normal web pages mixed in phishing data sets are also removed.

We collected and saved URL, HTML source file, and a screenshot of each collected page.

4.1.3. Evaluating Indicators

Summarizing the various evaluating indicators in literature, the most frequently used are True Positive Rate (TPR), False Positive Rate (FPR), Precision, and their calculation formula is as follows:

$$TPR = \frac{TP}{TP + FN}, \tag{6}$$

$$FPR = \frac{FP}{FP + TN}, \tag{7}$$

$$Precision = \frac{TP}{TP + FP}. \tag{8}$$

Whereas TP (True Positive) denotes the number of benign web pages correctly classified as benign web pages, FP (False Positive) denotes the number of phishing web pages classified as benign web pages, TN (True Negative) denotes the number of phishing web pages classified as phishing web pages, and FN (False Negative) denotes the number of benign web pages classified as phishing web pages.

4.1.4. Baselines

In order to measure the performance and efficiency of document-based representation methods, DoD is compared with other methods based on DOM clustering, namely PD and HD, which are both based on structural clustering. In order to compare with the semantic-based algorithm, the method of adopting the TF-IDF model (referred to as TF-IDF) on the DOM tree is explicitly compared. Table 2 illustrates the differences between the above methods.

The training of DoD adopts DM mode, and the specific hyperparameter setting is shown in Table 3.

Table 1. Development environment.

Operating system	CPU	RAM	Development environment	Development language
Windows 10	Intel Core i5-3337U CPU@1.8GHz	4 GB	Eclipse	Python 2.7

[1] https://pan.baidu.com/s/1zdxN9brTh1e4R_3jZgidCQ Access code: 0rnm

[2] https://www.alexa.com/

[3] https://www.phishtank.com/

Table 2. The comparison of methods.

Method	Tags from DOM	Clustering Algorithm	Starting Point
PD	Tag frequency	Hierarchical clustering	Structure-based clustering
HD	Hierarchical tag vector	Improved K-mediods	Structure-based clustering
TF-IDF	Tag list	Hierarchical clustering	Semantic clustering
DoD	Tag list	Hierarchical clustering	Semantic clustering

Table 3. Hyper-parameter used for Doc2Vec.

Method	Vector Size	Window Size	Min Count
DM	120	4	2
Sub-Sampling	**Negative Sample**	**Epoch**	**Workers**
0.1	10	200	6

4.2. Experiment 1

Experiment 1 compares the results of DoD with the baselines. On DOM corpus of the dataset, four different methods are applied to be vectorized and clustered. The clustering results by different methods are shown in Table 4. Using a histogram, Figure 5 compares TPR and FPR after the execution of each method.

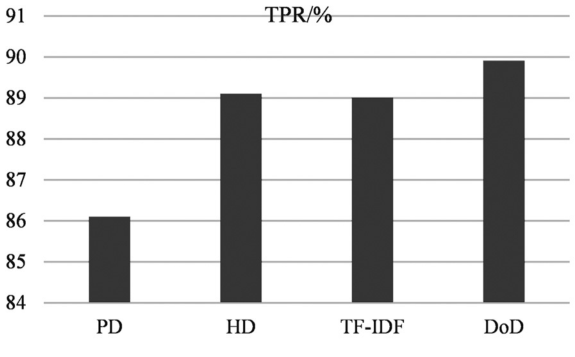


Figure 5. Comparison of TPR by different methods.

4.2.1. Efficiency Comparison of Different Methods

As can be seen from Table 4, Figure 5 and Figure 6, DoD has the best classification effect in general, while the semantic-based methods (TF-IDF and DoD) are better than the structure-based methods (PD and HD). In principle, PD counts the frequency of the occurrence of 107 types of tags on web pages. HD considers the hierarchicality of tags when comparing tag sequences. Both algorithms are based on the structural features of DOM. When similar web pages have large differences in shallow structures, the two algorithms have poor results. Both TF-IDF and DoD are based on the semantic features of the DOM tree. TF-IDF focuses on the calculation of weights for feature words, while the Doc2Vec model adopted by DoD is not only good at capturing potential relationships between feature words, but can also express the relationship between these words and specific documents. This means DoD can neglect the structure changes of DOMs and capture the similarity in DOMs, so the best classification effect is achieved.

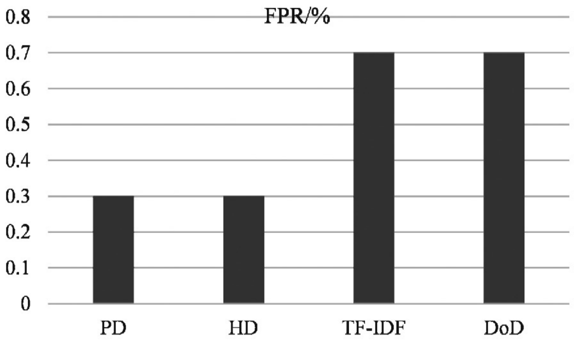


Figure 6. Comparison of FPR by different methods.

4.2.2. Time Comparison of Different Methods

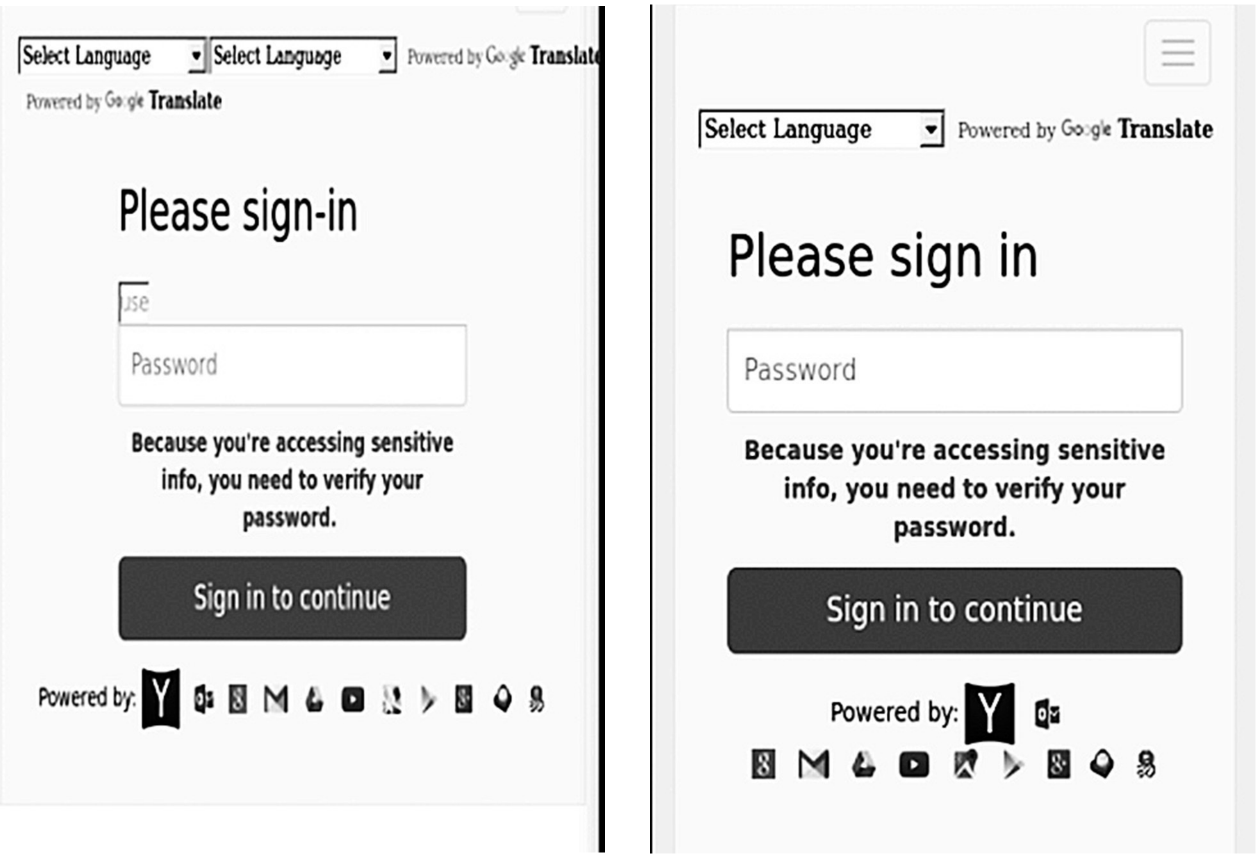
As can be seen from Table 4, training the DoD model based on Doc2Vec takes the longest time. The time complexity is closely related to the training dataset and dimension. The larger the dataset, the higher the dimension, and the longer the training model takes. This is because the number of parameters in the model that needs to be adjusted is N*P+M*Q, which is positively related to the size of the corpus and the size of the dimension, where *N* is the number of web pages in the dataset, *P* is the dimension of the paragraph vector, *M* is the number of tags, and *Q* is the dimension of the word vector. In future experiments, in order to reduce duration of the experiment, consider building a cluster to train the model with the large-scale corpus.

4.3. Experiment 2

Experiment 2 tries to explain the clustering results of Experiment 1 intuitively. In Experiment 1, two structure-based clustering methods, namely PD and HD, can group similarly structured web pages into one class, and the results are relatively intuitive. As shown in Figure 7, Web page 1 and Web page 2 are clustered into one class because they have similar structures.

In contrast to DoD, after finishing the training process, we obtain word embeddings *W* of HTML tags and document embedding *D* of DOM tree from the training corpus. But can embedded HTML tags express some semantic information? In order to observe the learned representation more intuitively, representation of HTML tags after DOD training is drawn in Figure 8. To shed light on what is being learned, we plot the word embeddings induced by Doc2Vec using t-SNE [18].

In order to analyze the results of Figure 8, let us first check the categories of tags in HTML5.0, shown in Table 5. In HTML 5.0, a web page can be created from 102 tags of 12 categories.



Because you're accessing sensitive info, you need to verify your password.

Because you're accessing sensitive info, you need to verify your password.

(a) Web page 1 *(b)* Web page 2

Figure 7. Two login pages in a class.

Table 4. Clustering results.

Method	Number of clusters	TPR/%	FPR/%	Clustering time/s
PD	1759	86.1	0.3	3631
HD	1655	89.1	1.3	1917
TF-IDF	1700	89.0	0.7	4503
DoD	1688	89.9	0.7	26,644

Table 5. HTML tag by category.

Category	Tag			
Basic	<!DOCTYPE>	<html>	<head>	<title>
	<h1>to<h6>	<br>	<hr>	<body>
	<!--…-->	<p>		
Formatting	<address>	<abbr>	<b>	<bdi>
	<bdo>	<mark>	<meter>	<pre>
	<s>	<samp>	<em>	<font>
	<cite>	<code>	<del>	<strong>
	<blockquote>	<big>	<var>	<dfn>
	<time>	<q>		
Forms and Input	<textarea>	<keygen>	<input>	<button>
	<optgroup>	<datalist>	<form>	<output>
	<label>	<fieldset>	<select>	<option>
Frames	<noframes>	<img>	<frame>	<iframe>
Images	<frameset>	<map>	<area>	<picture>
	<figcaption>	<figure>		
Audio/Video	<audio>	<source>	<track>	<video>
Links	<a>	<link>	<nav>	
Lists	<ul>	<ol>	<li>	<dir>
	<dl>	<dt>	<dd>	<menu>
	<menuitem>			
Tables	<table>	<caption>	<th>	<tr>
	<tbody>	<tfoot>	<col>	<td>
	<colgroup>	<thead>		
Styles and Semantics	<style>	<div>	<span>	<header>
	<summary>	<dialog>	<data>	<section>
	<footer>	<main>	<article>	<aside>
Meta Info	<basefont>	<meta>	<base>	<head>
Programming	<noscript>	<script>	<applet>	<embed>
	<object>	<param>		

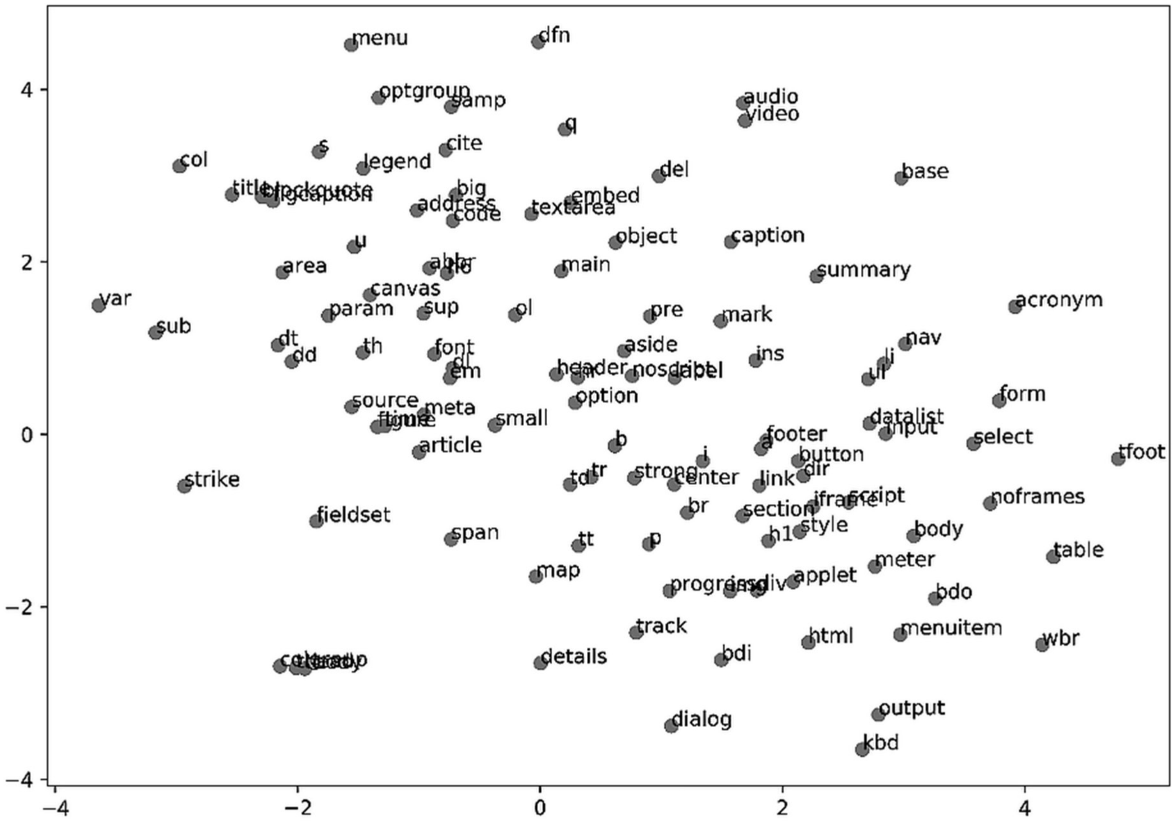


Figure 8. Two-dimensional t-SNE projection of Doc2Vec embeddings.

Websites are characterized by the order for these tags, the number of appearances, and the constructions of text documents.

From Figure 8, we can see that <dt> is near to <dd>, and that there is some distance between these two tags and <dl>. In Table 5, it is obvious that all three tags belong to the Lists. In fact, <dl>, <dt> and <dd> is a combination tag. Its standard usage is as follows:

```
<dl>
        <dt>title1</dt>
        <dd>list1</dd>
        <dd>list2</dd>
</dl>.
```

In web page design, <dl> is the outermost tag, while <dt> and <dd> are often used in pairs, so they can be seen as having similar semantics. Doc2Vec is based on the distributional hypothesis that words occurring in similar context tend to have similar meanings, so <dt> and <dd> have small distance. The same situation can be seen in <td> and <tr>. <td> is used to define the cell of a table, and <tr> is a row in the table, which is often used together. In Figure 8, the two tags are very close to the <strong> and <b>. This can correspond to the actual situation, as these three tags in HTML are often used to describe the format of the content of the table.

Through the above analysis, it can be considered that DoD has learned the semantic association with HTML tags, regards HTML tags as words of natural language, regards DOM tree as documents composed of HTML tags, and it is feasible to learn semantic representation and detect phishing web pages through DoD.

5. Conclusion

In this paper, we presume that DOMs of web pages are composed of natural language and use Doc2Vec model to capture semantic similarity in DOMs. This is different from the phishing web page detection methods from the perspective of structural analysis of DOM trees. Our method in certain settings outperforms the state-of-the-art approaches for the phishing detection task based on DOM.

But still, the method has not been implemented in large datasets and actual scenario. In the future, we will implement DoD in larger datasets collected from public websites. As word/document embedding is considered as a promising approach to text representation, we expect that this line of exploration can target the tasks which can be modeled as a word sequence.

Acknowledgement

This work is supported by Shaanxi Provincial Natural Science Foundation Project (No. 2020JM-533 and No. 2020JM-526).

References

[1] Z. Ramzan, "Phishing Attacks and Countermeasures", *Handb. Inf. Commun. Secur.*, Springer Berlin Heidelberg, pp. 433–448, 2010. http://dx.doi.org/10.1007/978-3-642-04117-4_23

[2] L. C. John, "2019 Phishing Trends and Intelligence Report: The Growing Social Engineering Threat", Phishlabs, [Online]. Available: https://info.phishlabs.com/hubfs/2019%20PTI%20Report/2019%20Phishing%20Trends%20and%20Intelligence%20Report.pdf

[3] A. Aleroud and L. Zhou, "Phishing Environments, Techniques, and Countermeasures: A Survey", *Comput. Secur.*, vol. 68, pp. 160–196, 2017. http://dx.doi.org/10.1016/j.cose.2017.04.006

[4] A. C. Bahnsen *et al.*, "Classifying Phishing URLs Using Recurrent Neural Networks", in *Proc. of the IEEE 2017 APWG Symp. Electron. Crime Res. (eCrime)*, 2017, pp. 1–8. http://dx.doi.org/10.1109/ECRIME.2017.7945048

[5] R. Vinayakumar *et al.*, "Evaluating Deep Learning Approaches to Characterize and Classify Malicious URL's", *J. of Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1333–1343, 2018. https://doi.org/10.3233/JIFS-169429

[6] Q. Cui *et al.*, "Tracking Phishing Attacks over Time", in *Proc. of the 26th Int. Conf. World Wide Web*, New York, USA, 2017, pp. 667–676. https://doi.org/10.1145/3038912.3052654

[7] J. Feng and Y. Zhang, "A Detection Method for Phishing Web Page Based on Document Object Model Structure Clustering", *Sci. Technol. Eng*, vol. 18, no. 23, pp. 81–89, 2018. (in Chinese) https://doi.org/10.3969/j.issn.1671-1815.2018.23.012

[8] T. Mikolov *et al.*, "Distributed Representations of Words and Phrases and Their Compositionality", in *Advances in Neural Information Processing Systems (NIPS 2013)*, vol. 26, pp. 3111–3119, 2013. https://dl.acm.org/doi/10.5555/2999792.2999959

[9] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents", in *Proc. of the Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1188–1196. https://dl.acm.org/doi/10.5555/3044805.3045025

[10] J. Tao *et al.*, "DNN Online with iVectors Acoustic Modeling and Doc2Vec Distributed Representations for Improving Automated Speech Scoring", in *Interspeech 2016*, San Francisco, USA, 2016, pp. 3117–3121. http://dx.doi.org/10.21437/Interspeech.2016-1457

[11] M. Mimura and H. Tanaka, "Reading Network Packets as a Natural Language for Intrusion Detection", in *Proc. of the Int. Conf. on Inform. Secur. Cryptology*, Springer, Cham, 2017, pp. 339–350. https://doi.org/10.1007/978-3-319-78556-1_19

[12] A. P. E. Rosiello *et al.*, "A Layout-Similarity-Based Approach for Detecting Phishing Pages", in *Proc. of the Int. Conf. Secur.*, New York, USA, 2017, pp. 454–463. http://dx.doi.org/10.1109/SECCOM.2007.4550367

[13] Z. S. Harris, "Distributional Structure", *Word*, vol. 10, no. 2-3, pp. 146–162, 1954. https://doi.org/10.1080/00437956.1954.11659520

[14] B. Baharudin *et al.*, "A Review of Machine Learning Algorithms for Text-Documents Classification", *J. Adv. Inf. Technol.*, vol. 1, no. 1, pp. 4–20, 2010. http://dx.doi.org/10.4304/jait.1.1.4-20

[15] D. M. Blei *et al.*, "Latent Dirichlet Allocation", *J. Mach. Learn. Res.*, vol. 3, no. 1, pp. 993–1022, 2003. http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993

[16] A. M. Dai *et al.*, "Document Embedding with Paragraph Vectors", *J. Comput. Sci.*, 2015. https://arxiv.org/pdf/1507.07998.pdf

[17] D. Kim *et al.*, "Multi-Cotraining for Document Classification Using Various Document Representations: TF-IDF, LDA, and Doc2Vec", *Inf. Sci.*, vol. 477, pp. 15–29, 2019. https://doi.org/10.1016/j.ins.2018.10.006

[18] L. V. D. Maaten and G. Hinton, "Visualizing Data using t-SNE", *J. Mach. Learn. Res.*, vol. 9, no.11, pp. 2579–2605, 2008. https://doi.org/10.1007/s10846-008-9235-4

Received: November 2019
Revised: June 2020
Accepted: June 2020

Contact addresses:

Jian Feng
Xi'an University of Science and Technology
Xi'an
China
e-mail: fengjian@xust.edu.cn

Ying Zhang
Information Technology Department for
Head Office of SPD Bank
Xi'an
China
e-mail: zhangy292@spdb.com.cn

Yuqiang Qiao
Xi'an University of Science and Technology
Xi'an
China
e-mail: 825113305@qq.com

JIAN FENG was born in Xi'an, Shaanxi, China, in 1973. She received her BSc degree in printing technology from Beijing Institute Of Graphic Communication, Beijing, China, in 1994, and the MSc degree in printing engineering from Xi'an University of Technology, Xi'an, China, in 2001, and the doctoral degree in computer software and theory from Northwest University, Xi'an, China, in 2008. In 2008, she joined the Department of Network Engineering in College of Computer Science & Technology, Xi'an University of Science and Technology, as a Lecturer, and in 2010, she became an Assistant Professor. She is the author of two books, and of more than 30 articles. Her research interests include computer networks and communications, network security, and distributed computing. She holds three patents.

YING ZHANG was born in Shaanxi, China. She received her BSc in computer science and technology from Xi'an University of Science and Technology, Xi'an, China, in 2016 and the MSc degree in computer application technology from Xi'an University of Science and Technology, Xi'an, in 2019. She is currently working in the Head Office Information Technology Department of SPD Bank, Application Development Service Sub-center (Xi'an). Her research areas include network security and Internet data mining. Ms. Zhang's awards and honors include the National Scholarship for Masters, the Second Prize for the China Graduate Mobile Application Innovation Competition, and the Second Prize for the China Graduate Electronic Design Competition (Business Plan Special Competition).

YUQIANG QIAO was born in Yulin, Shaanxi, China, in 1996. He received the BSc degree in software engineering from the City College of Xi'an Jiaotong University in 2019. He is currently pursuing the MSc degree in software engineering, Xi'an University of Science and Technology, China. Beginning in 2019, he was engaged in the research of phishing web pages detection. His research areas include network security and graph machine learning. Mr. Qiao's awards and honors include the Academy Scholarship and the National Inspirational Scholarship.