Accepted Manuscript

A stacking model using URL and HTML features for phishing webpage detection

Yukun Li, Zhenguo Yang, Xu Chen, Huaping Yuan, Wenyin Liu

PII: S0167-739X(18)30503-X
DOI: https://doi.org/10.1016/j.future.2018.11.004
Reference: FUTURE 4568

To appear in: *Future Generation Computer Systems*

Received date : 17 March 2018
Revised date : 13 October 2018
Accepted date : 4 November 2018

A Stacking Model Using URL and HTML Features for Phishing Webpage Detection

Yukun Li[a], Zhenguo Yang[b,c,*], Xu Chen[a], Huaping Yuan[b], Wenyin Liu[a,b,**]

[a] *Department of Automation, Guangdong University of Technology, Guangzhou, China*
[b] *Department of Computer Science, Guangdong University of Technology, Guangzhou, China*
[c] *Department of Computer Science, City University of Hong Kong, Hong Kong, China*

Abstract

In this paper, we present a stacking model to detect phishing webpages using URL and HTML features. In terms of features, we design lightweight URL and HTML features and introduce HTML string embedding without using the third-party services, making it possible to develop real-time detection applications. Furthermore, we devise a stacking model by combining GBDT, XGBoost and LightGBM in multiple layers, which enables different models to be complementary, thus improving the performance on phishing webpage detection. In particular, we collect two real-world datasets for evaluations, named as 50K-PD and 50K-IPD, respectively. 50K-PD contains 49,947 webpages with URLs and HTML codes. 50K-IPD contains 53,103 webpages with screenshots in addition to URLs and HTML codes. The proposed approach outperforms quite a few machine learning models on multiple metrics, achieving 97.30% on accuracy, 4.46% on missing alarm rate, and 1.61% on false alarm rate on 50K-PD dataset. On 50K-IPD dataset, the proposed approach achieves 98.60% on accuracy, 1.28% on missing alarm rate, and 1.54% on false alarm rate.

Keywords: Anti-phishing, HTML string embedding, machine learning, stacking model

1. Introduction

Phishing is one type of the Internet fraud which refers to fake webpages impersonate legitimate webpages to trick users to send their sensitive information, such as username, password, bank account numbers or credit card numbers. Phishers usually imitate the identities of the well-known webpages to send emails, short message service (SMS) or instant messenger along with a phishing URL. However, victims believe that they are accessing credible webpages. Therefore, they may provide their debit or credit card numbers, PIN codes or other private information. According to the APWG global phishing

*Corresponding author
**Principal corresponding author
Email addresses: gdutkelvin@outlook.com (Yukun Li), zhengyang5-c@my.cityu.edu.hk (Zhenguo Yang), c[illegible]88@qq.com (Xu Chen), yuanhuaping@outlook.com (Huaping Yuan), liuwy@gdut.edu.cn (Wenyin Liu)

survey report 4Q2016 [1], the total number of phishing attacks in 2016 was 1,220,523, showing an increase of 65% over 2015. In the fourth quarter of 2004, the APWG saw 1,609 phishing attacks per month. However, in the fourth quarter of 2016, the APWG saw an average of 92,564 phishing attacks per month, showing an increase of 5,753% over the past 12 years. Phishing attacks are growing wildly, urging people to consider how to prevent it.

In order to detect emerging phishing webpages constantly, blacklist-based methods [2, 3] and heuristic-based methods are widely used. Blacklist-based methods checked the existing records in the blacklist to recognize the phishing ones, which could not deal with the newly emerging ones. There are many practical solutions based on heuristic methods. For example, quite a few works [4, 5, 6, 7] extracted text, image or URL features from webpages and used search engines to detect phishing webpages, which might be limited to the performance of search engines. Some works [8, 9, 10, 11] used visual similarity of phishing webpages and well-known webpages to recognize the phishing ones, which might be dependent on the accuracy of image similarity comparison algorithm. Some works [12, 13] used DNS anomaly information of webpages, which required the third-party service to provide DNS information, resulting in high development cost. Amounts of works [14, 15, 16, 17, 18, 19, 20, 21, 22] extracted text information from HTML, and images or special URL features to combine with heuristic or machine learning algorithms, while processing images suffers from high computing cost.

A report from APWG [23] states that the average uptime of phishing webpages is 32.5 hours and the median of the lifespan is 8.7 hours. Half of the phishing webpages are being shut down in less than a day, which shows that phishing attacks are fast and changeable. Since phishing attacks aim at exploiting weakness of humans, thus expecting users to understand phishing attacks is unrealistic. To this end, we develop a system by using machine learning technology for phishing webpage detection. Specifically, we extract two-fold features from URLs and HTML source codes, including artificially designed features, and HTML string embedding of HTML source codes extracted by the Word2Vec model [24]. In particular, the artificially designed features adopted in our work are lightweight, which only deal with the current page information and do not rely on third-party services. The embeddings of HTML strings are extracted automatically without increasing the workload of designing features. Furthermore, we devise a stacking model by combining GBDT, XGBoost and LightGBM to recognize the phishing webpages. Extensive experiments manifest that the proposed anti-phishing approach has achieved competitive performance on real-world datasets in terms of multiple performance metrics.

The main contributions in this paper are summarized as follows.

(1) In terms of feature extraction, we combine artificially designed features with HTML string embedding extracted by Word2Vec model.

(2) We propose a stacking model for detecting phishing webpages, combining the advantages of several machine learning models.

(3) We designed a real-time phishing webpage detection system that can be used to protect users from phishing attacks.

(4) We release a real-world phishing webpage detection dataset, which can hopefully be used to promote the research and applications on this topic.

This paper is structured as follows. Section 2 discusses related works, including different phishing webpage detection methods and the application scenarios of stacking methods. Section 3 introduces the designed features and the proposed stacking model. In Section 4, we conduct extensive experiments to evaluate the proposed methods. Finally, Section 5 offers some concluding remarks and expending works in the future.

2. Related Work

2.1. Phishing Webpage Detection

In the past few years, phishing webpage detection has received much research attention in both academia and the industry. However, the characteristics of phishing webpages, such as complexity, confusing, noising, etc., make them hard to be detected. There are quite a few research work on anti-phishing technology, which can essentially be classified into two categories: rule-based methods and machine learning-based methods.

2.1.1. Rule-based Phishing Webpage Detection

Rule-based approaches design rules according to the significant differences between phishing webpages and legitimate webpages. If a webpage meets one or more rules, it will be judged as a phishing one.

Cao et al. [25] presented a whitelisting-based approach named as Automated Individual White-List (AIWL). AIWL recorded all familiar Login User Interfaces (LUI) for a user as a white-list, where the familiar LUI refers to login webpages that are frequently used by users. The user is warned of a possible attack whenever he/she attempts to submit sensitive information to a webpage that is not included in the white-list. In order to obtain the white-list, they used the Naive Bayesian classifier to make a decision, which is confirmed further by users. In addition to the maintenance of the white-list, the approach relies on feedbacks from users and cannot proactively discover new phishing webpages.

Zhang et al. [26] presented a content-based approach named as CANTINA. They used the TF-IDF algorithm to extract the top-5 keywords from a webpage and then fed them into a search engine. Furthermore, they compared the domain name of the current webpage with the domain names of the top-N search results to determine whether it is a phishing webpage. To reduce the false positive rate, they developed several heuristics including age of domain, known images, suspicious URL, suspicious links, IP address, dots in URL. However, the accuracy of keyword extraction depends on the training corpus of the TF-IDF model. In addition, querying search engine needs much time, which has an influence on the performance.

Rami et al. [18] analyzed 17 different features that distinguished legitimate webpages and phishing webpages. Furthermore, they designed a rule for each feature. For example, if the age of domain is more than 6 months, the webpage is considered as legitimate. Otherwise, the webpage is considered as phishing. Moreover, several experiments conducted to select the most effective features on predicting phishing webpages. However, setting thresholds artificially for features requires tedious statistical work. In addition, the proposed features, such as DNS record, webpage traffic, and age of domain, rely on the third-party services.

2.1.2. Machine Learning-based Phishing Webpage Detection

Machine learning-based approaches extract various features from different sources to train phishing webpage classifiers.

Abu-Nimeh et al. [27] compared six machine learning algorithms for phishing e-mail detection, including Logistic Regression (LR), Classification and Regression Trees (CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random Forests (RF), and Neural Networks (NN). The results showed that there is a trade-off between false positive rate and false negative rate, i.e., if some algorithms have low false positive rate then they might have high false negative rate. Furthermore, they discussed that using straightforward accuracy or error rate as performance metrics is not considerate enough. Consequently, they suggested combining accuracy and false positive rate for evaluation.

Gowtham et al. [28] proposed an efficient anti-phishing system based on 15 heuristic features with a pre-filtering mechanism. Their proposed system contained three modules: Preapproved Site Identifier, Login Form Finder, and Webpage Feature Generator. Preapproved Site Identifier detected the legitimate webpages that users have visited to avoid redundant calculation. Login Form Finder filtered webpages which do not have a login form and prevented them from being processed. Webpage Feature Generator contained 15 heuristics to identify phishing webpages. Finally, they used SVM to train a phishing webpage classifier. However, the strict login page filtering mechanism makes it rely on the accuracy of the login window detection.

Marchal et al. [29] proposed an efficient phishing URL detection system named as PhishScore, which relied on URL lexical analysis. PhishScore analyzed the intra-URL relatedness, which was the quantification of the relatedness between words and words in different parts of URL, by leveraging search engine queries. Furthermore, 12 heuristics were used to increase the performance of the system, such as number of words in URL and Alexa ranking for domain name. Finally, a trained Random Forest model was used to detect phishing URLs. However, the dictionary of URL words is English, which can only extract English words from URLs. In addition, relying on search engine such as Google Trend and Yahoo Clues makes it cost much time while searching.

Xiang et al. [22] proposed CANTINA+ by extending CANTINA [26]. Firstly, a hash-based filter was used to recognize identical phishing webpages. Furthermore, the webpages without login forms were filtered to decrease false positive rate. Finally, a Bayesian network was trained by using the designed heuristic features to detect phishing webpages. However, the system relies on the PageRank, which service has stopped. Meanwhile, sending queries over the network and storing large amounts of data lead to high time and space costs.

2.2. Application Scenarios Using Stacking Models

Considering the insufficiency of single model, some researchers resort to the stacking strategy that combines multiple classifiers for predictions. Stacking strategy has achieved impressive performance results in quite a few application scenarios and data mining challenges, such as the Kaggle competitions, sentiment classification [30], workload prediction [31], and speech recognition [32]. To the best of our knowledge, there are no researchers apply stacking strategy on phishing webpage detection.

Georgios Sakkis et al. [33] evaluated a scheme of combined classifiers and further designed a stacking model for anti-spam filtering of E-mails. The strategy of stacking model is called cross-validation stacking, where each training set was prepared using a second-level 3-fold cross-validation. Their experiments showed that stacking outperformed the best methods such as NB and k-NN algorithm.

Anandita et al. [34] proposed an ensemble classifier for phishing E-mails filtering. The ensemble classifier contained five machine learning algorithms: Gaussian Naive Bayes, Bernoulli Naive Bayes, Random Forest Classifier, K-Nearest Neighbor and Support Vector Machines. Finally, the accuracy was improved from 94.09% (obtained by random forest) to 98.02%.

Mi et al. [35] were dedicated to applying data mining techniques to email spam detection. In their work, a hybrid model was proposed, which combined J48 and Naive Bayes machine learning algorithm. Research shows that hybrid models obtain performance improvement compared with single classifiers.

Stacking strategy has been verified to be effective on phishing e-mail and spam detection, while they have not been fully investigated in the context of phishing webpage detection.

3. Methodologies

The overview of our approach is shown in Fig.1. Firstly, we extract features from URLs and HTML codes of webpages, and concatenate them as feature vectors. Furthermore, we devise the stacking model to make predictions. The two components are introduced in the following subsections.

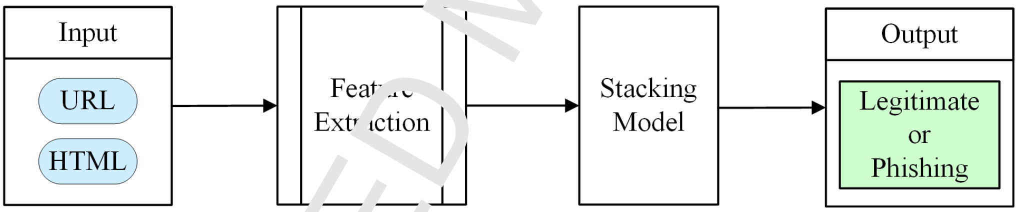


Fig.1. Overview of our approach

3.1. Feature Extraction

In the context of phishing webpage detection, we extract two-fold features, i.e., URL-related features, and HTML-related features. For the URL-related features, we extract 8 features in total, among which the first 6 features are designed by peer researchers. These features do not rely on the third-party services, which can be used to develop real-time systems. For the HTML-related features, 12 features are extracted, among which the last 6 features are designed by us. In particular, we propose to learn HTML string embedding by using the Word2Vec model. The two-fold features are introduced as follows.

3.1.1. URL-related Features

(1) **IP address** [26]. In practice, the domain names of phishing webpages usually are the IP address, e.g., http://62.141.45.54/portaleTitolaris8/. A binary value is used to indicate whether the domain name of a URL is an IP address or not.

(2) **Suspicious symbols** [36]. Some rarely used symbols usually emerge in phishing URLs, including '@', '−' and '∼', etc. If '@' appears in a URL, all strings at the right of the symbol will be ignored when a browser parses the URL. The number of each suspicious symbol in the URL is used as a feature. Specifically, we count the number of three symbols respectively, including '@', '−'and '∼'.

(3) **https** [37]. The transport protocols from webpage to webpage include "http" and "https", where "https" is a secure "http" data transfer method that provides authentication and encrypted communication. Applying "https" on personal webpage needs to apply to a professional agency, thus phishing webpages usually do not apply "https". A binary value is used to indicate whether the protocol of a URL is "https" or not.

(4) **URL length information** [38]. McGrath et al. observed that the lengths of the domain names of legitimate URLs usually were longer compared to phishing URLs, while the total lengths of the phishing URLs usually were longer than legitimate ones. Length information includes the number of the characters in a URL and its domain name, respectively.

(5) **Number of dots in domain name** [18]. Rami et al. observed that phishing webpages tended to use more dots in their URLs, while legitimate webpages usually used no more than three, e.g., "www.google.com". The number of '.' in domain name is used to represent this feature.

(6) **Sensitive vocabulary**. Some sensitive vocabularies are often used in phishing webpages. Garera et al. [39] summarized a set of eight sensitive words that frequently appeared in phishing URLs, i.e., ["secure", "account", "webscr", "login", "ebayisapi", "signin", "banking", "confirm"]. We use the statistics on the number of these words in URLs as a feature.

(7) **Top-level domain name related**. Top-level domain names can be divided into two categories, the ones for country code (e.g., 'cn' denotes China), and the generic ones (e.g., '.com' for the business enterprise, '.net' for the network provider, and '.org' for non-profit organization, etc.). Phishing URLs usually have multiple top-level domains, e.g., "http://www.ebay.com.urgd.com/path". Therefore, we design three features corresponding to top-level domain names, i.e., whether the top-level domain name is in the top-level domain name list of Stuffgate[1], the number of top-level domain names in the domain name, and the number of top-level domain names in the URL path (the part immediately following the domain name in a URL).

(8) **Similar target brands**. Phishing webpage designers usually imitate the URLs of target pages to confuse users. For example, quite a few phishing ones modify "paypal" as "paypa1" to confuse users. Therefore, we propose to discover the behavior that mimics target webpages in the URLs and take it as a feature. Specifically, we split a URL according '.' and '/' to extract strings. Furthermore, we calculate the Levenshtein distance [40] between each string and a given target brand from Phishtank[2] (the webpage has been attacked and reported on Phinshtank is seemed as a target brand). If there exists a distance value that is smaller than 2, we set this feature as 1, otherwise, we set this feature as zero.

[1] http://stuffgate.com/

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

3.1.2. HTML-related Features

In addition to the 6 existing HTML features (1-6) that do not need the third-party services, we propose 6 news features (7-12) considering the differences between phishing and legitimate webpage content.

(1) **Number of internal and external links** [18]. An internal or external link refers to the one has the same or different domain name as the URL. In order to deceive users that the page is legal, phishing webpages usually use external resources like their phishing targets to enrich their content, resulting in very few internal links and many external ones in a phishing webpage. We extract the domain names from all the links in the HTML codes, and compare them with the domain names of the webpage URL to recognize the internal and external links. We take the number of internal links as a feature and the number of external links as another feature.

(2) **Empty link** [36]. Phishing webpages usually use empty links to pretend the pages possessing a lot of hyperlinks. There are two kinds of empty links, i.e., " ", and " ". The number of the empty links in a webpage is used to represent this feature.

(3) **Login form**. Phishing webpages often have login windows for users to fill in sensitive information. Xiang et al. [22] proposed to recognize whether the page contains a login window in phishing webpage detection. Firstly, all the tags of <form> in the webpage have to be identified. Furthermore, for each sub-tag <input> in a tag <form>, the keywords 'password' and 'pass' or 'login' and 'signin' will be used for matching to decide whether there are login forms.

(4) **Length of HTML content**. The purpose of the phishing webpages is to cheat the user's login information, so their HTML contents usually are relatively simple. More directly, the length of phishing webpages HTML code is usually shorter than the length of legitimate ones. Specifically, we refine the length of HTML content to the length of the tag content. According to Alkhozae et al. [41], we choose five tags to calculate their length respectively, including "<style>", "<script>", "<link>", "<!−−>", and "<form>". Tags "<style>" and "<link>" mainly set the page style and CSS, while phishing webpages usually do not have such a style for quick development. Tag "<script>" is for changing the page content dynamically, while phishing webpages usually are static. Tag "<!−−>" is for comment, while phishing webpages usually are one-time development. The developers do not perform secondary maintenance, thus they usually do not write comments. Tag "<form>" is for setting forms inside a webpage. The length of each tags is taken as a feature, respectively. Furthermore, the length of the HTML code is also taken as a feature. Totally, we extract 6 features to represent the length of HTML content.

(5) **Alarm window** [42]. Some phishing webpages will pop up an alarm window for users to enter their personal information. We use a binary value to indicate whether a webpage has an alarm window.

(6) **Redirection** [42]. Some phishing webpages designers create legitimate webpages for users to access, which will redirect to phishing ones. The redirected phishing webpages use "redirect" string in the HTML codes. A binary value is used to indicate whether there exists such a string.

(7) **Hidden/Restricted information**. Some special codes in HTML codes can prevent the content from displaying or restricting the function of a tag, which may be

used by phishing webpages. These special codes work on specific tags, which are summarized below:

a. <div> : <div style = ″visibility: hidden″> or <div style = ″display: none″>. These tags hide the content inside the <div> and do not render the content on the webpage.

b. <button> : <button disabled = ″disabled″>. It disables the click function of the button.

c. <input> : <input type = ″hidden″> hides the input box, <input disabled = ″disabled″> prohibits input function of the input box, and <input value = ″hello″> fills in some irrelevant information in the input box.

For these three aspects, we can generate three features accordingly. For instance, the number of special codes inside all tag "<div>" in HTML codes is taken as a feature.

(8) **Consistency between title brand and URL brand**. In general, the title of a legitimate webpage contains its brand name. If the brand of a webpage is inconsistent with the URL brand, the webpage is likely a phishing one. We extract the domain name of URL and then split it by "." to get the URL brand. For instance, the URL brand of "www.google.com" is "google". Furthermore, a binary value is used to indicate whether there exists the same brand as the URL in the title.

(9) **Consistency between the most frequent link brand and URL brand**. Intuitively, due to extensive use of target webpage resources, the most frequent link brand should be inconsistent with the URL brand for phishing webpages. As shown in Fig.2, we extract all the links inside the HTML code, and calculate the number of times that each brand name appears in the links to obtain a brand name dictionary. Furthermore, the consistency is calculated based on whether the URL brand hits the most frequent link brand. Moreover, the number of times that the most frequent link brand appears in the links is used as a feature.

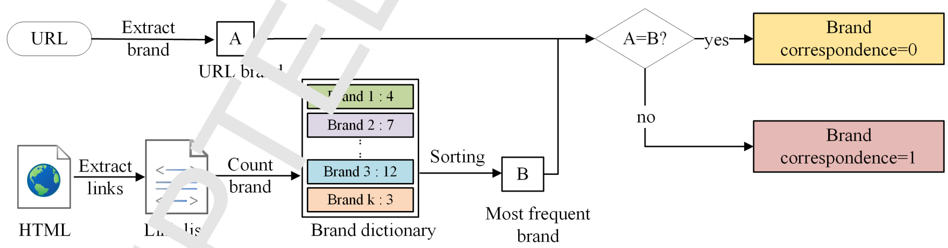


Fig.2. Evaluate the consistency between link brand and URL brand

(10) **Internal and external resources**. Based on the idea that phishing webpages tend to use external resources, we use four tags including"<link>", "<img>", "<script>" and "<noscript>", respectively, to calculate the number of internal resources and external resources as features.

(11) **Number of the URL brand name appears in HTML code**. The URL brand names of legitimate webpages usually appear a number of times in the HTML code due to the frequent use of internal resources. The opposite are usually the case for phishing webpages.

(12) **HTML string embedding**. Inspired by the Word2Vec model [43] that learns word embedding from textual documents, we propose to learn HTML string embedding in vector form from HTML documents in a similar manner. Word2Vec is a neural network language algorithm which focuses on learning distributed representations of words. In Word2Vec, there are an input layer, a projection layer, and an output layer. In the training phase of Word2Vec, given the context of a target word, we train the model to predict the target word as the output. After training the Word2Vec model, we can extract parameters of the projection layer to represent words. Specifically, in our task, HTML strings are extracted based on the space between them. Furthermore, we learn n-dimensional HTML string embedding by mapping the HTML string and HTML document to word and textual document, respectively. Finally, we obtain a n-dimensional vector to represent the HTML document by averaging each string embedding. The procedures are shown in Fig.3.

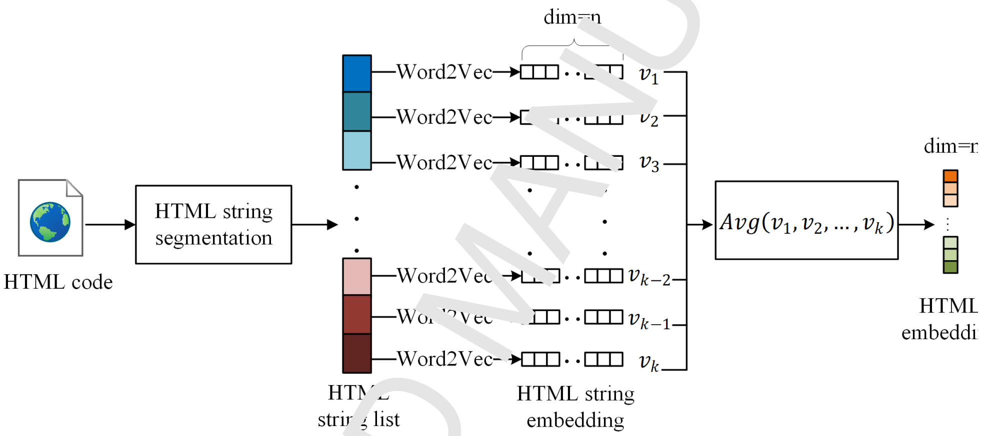


Fig.3. Learning HTML string embedding

3.2. Stacking Model

Given the aforementioned features, existing machine learning models can be used directly to identify the phishing and legitimate webpages. In the current work, we propose a stacking model by combining multiple machine learning models for this purpose. The proposed two-layer stacking model is shown in Fig.4, where each layer consists of three basic models, i.e., Gradient Boosting Decision Tree (GBDT) [44], XGBoost [45], and LightGBM [46]. More specifically, we use a strategy similar to K-fold cross validation to train the basic models. The training set is divided into Z copies, among of which Z-1 copies are used for training and 1 copy is used for testing. The training process will not be stopped until all the samples have been predicted by each basic model. Furthermore, we take the original features combining with the predicted results, which are obtained by majority voting in the first layer, as the input of the second layer of the stacking model. Finally, we combine the input features of the second layer and the output results of the second layer of the stacking models as the final features, which will be used to train a GBDT model to make predictions on the phishing webpages. For clarity, the procedures of the proposed stacking model using URL and HTML features for phishing webpage detection are summarized in Table 1.

Table.1. Phishing Webpage Detection Using the Proposed Stacking Model

Step 1. Feature extraction (refer to Section 3.1)

a. Extract the URL features
b. Extract the HTML features
c. Combine the URL features with the HTML features to obtain the features

Step 2. Dataset partition

a. Divide the dataset into a training set and a test set

Step 3. Construct the first layer of the stacking model

a. Select the basic models as GBDT, XGBoost and lightGBM
b. Divide the training set into Z copies
c. For (i=0; $i < Z$; i++)
    Choose Z-1 copies of the training set (except i) to train the basic models separately.
    Use the trained models to predict the i−th copy of training set
    Use the trained models to predict the test set
d. Combine the predictions of the training set with the original features as new features of training set
e. Vote the predictions of the test set, and combine the voting results with the original features as new features of test set

Step 4. Construct the second layer of the stacking model

a. Use the method of Step 3 to construct the second layer and obtain new features based on the output in Step 3.

Step 5. Phishing webpage recognition

a. Use the output of Step 4 to train a GBDT model for classification.
b. Recognize the phishing webpages

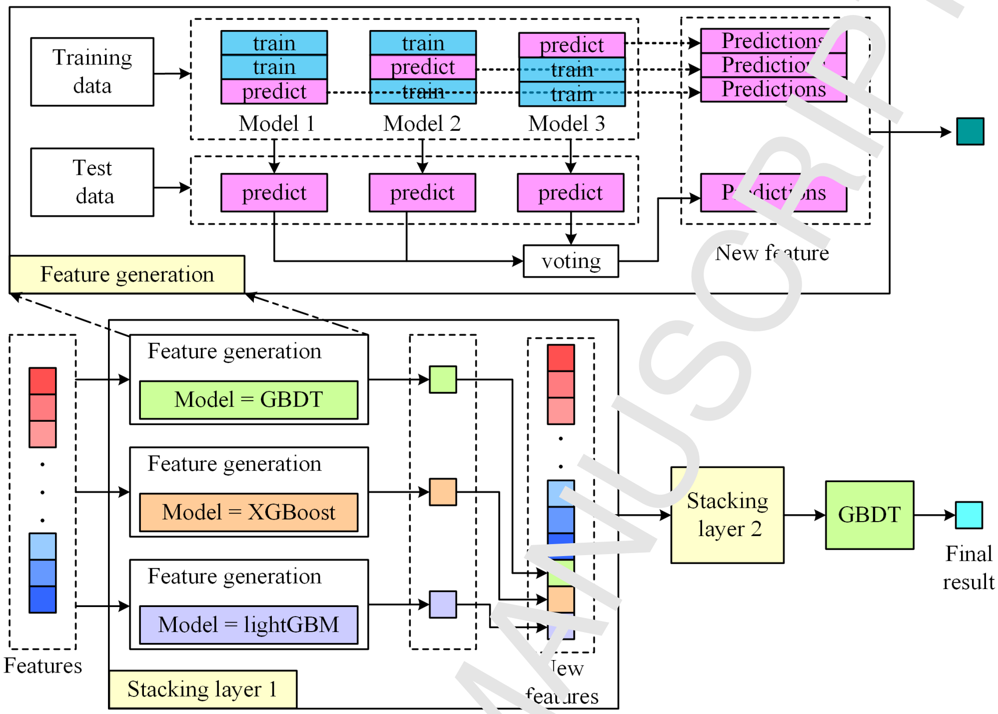


Fig.4. Overview of our stacking model (For the convenience of drawing, we divide training set into 3 copies)

4. Experiment

4.1. Dataset

For evaluations, we collect three datasets in different scales.

1) **2K Phishing Detection Dataset (2K-PD).** The small dataset contains 2,000 webpages and their HTML codes, where 1,000 ones are legitimate and 1,000 ones are phishing. The legitimate ones are collected from Alexa [47], which rankings are between 100,000 to 101,000. The phishing ones are from Phishtank [48], which have been validated from July 12, 2017 to July 15, 2017.
2) **50K Phishing Detection Dataset (50K-PD).** The large-scale dataset contains 49,947 webpages and their HTML codes, where 30,873 ones are legitimate and 19,074 ones are phishing. On one hand, we collect 2,000 legitimate webpages from Alexa which rankings are from 10,000 to 12,000. The remaining 28,873 legitimate ones are collected from the hyperlinks of these webpages. On the other hand, the phishing webpages are collected from Phishtank, which have been validated from June 2009 to June 2017. In particular, the distributions of the lengths of URLs in the 2K-PD and 50K-PD datasets are shown in Fig.5 and Fig.6 respectively. We can observe that the distributions of the lengths of URLs of two datasets are similar.

3) **50K Image Phishing Detection Dataset (50K-IPD).** The large-scale dataset contains 53,103 URLs, HTML codes and screenshots of webpages, where 28,320 ones are legitimate and 24,789 ones are phishing. On one hand, we collect 5,000 legitimate webpages from Alexa which rankings are from 10,000 to 15,000. The remaining 23,320 legitimate ones are collected from the hyperlinks of these webpages. On the other hand, the phishing webpages are collected from PhishTank which have been validated from June 2009 to February 2017. This dataset is prepared to evaluate the effectiveness of visual features. The distribution of the lengths of URLs in the 50K-IPD dataset is shown in Fig.7.

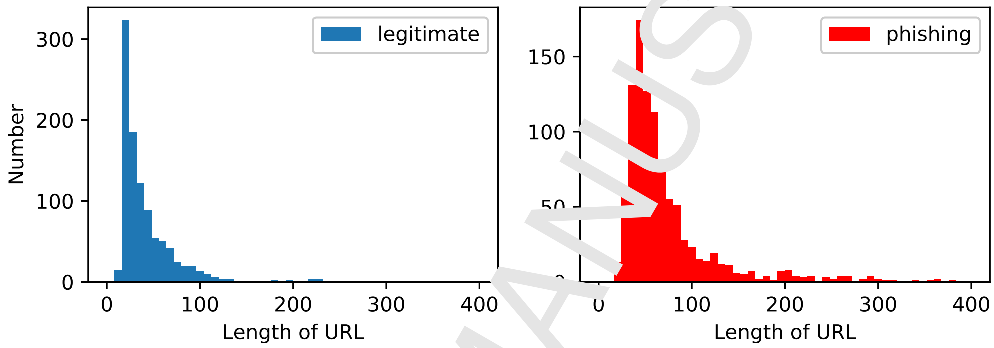


Fig.5. Distributions of the lengths of URLs in the 2K-PD dataset

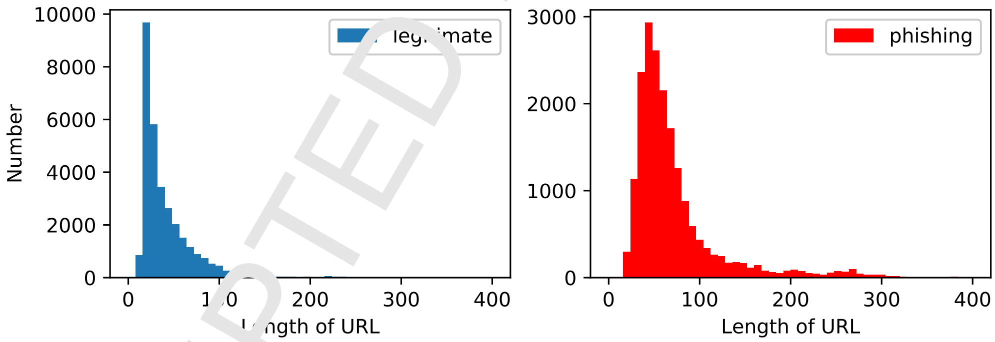


Fig.6. Distributions of the lengths of URLs in the 50K-PD dataset

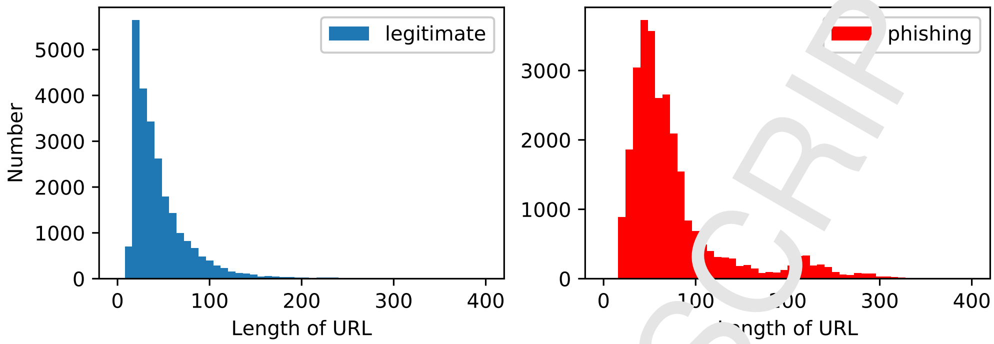


Fig.7. Distributions of the lengths of URLs in the 50K-IPD dataset

4.2. Performance Metrics

In the context of phishing webpage detection, we use accuracy rate, missing rate, and false alarm rate as performance metrics. Let P denote the truth number of phishing webpages in test set, L denote the truth number of legitimate ones, α denote the correct predicting number of phishing webpages, and β denote the correct predicting number of legitimate webpage. The three metrics are defined as follows:

$$\text{Accuracy} = \frac{\alpha + \beta}{P + L} * 100\% \tag{1}$$

$$\text{Missing rate} = (1 - \frac{\alpha}{P}) * 100\% \tag{2}$$

$$\text{False alarm rate} = (1 - \frac{\beta}{L}) * 100\% \tag{3}$$

It should be noted that missing rate is the percentage of misclassified phishing webpages in all phishing webpages, while false alarm rate is the percentage of misclassified legitimate webpages in all legitimate webpages. Obviously, the higher the accuracy and the lower the missing and false alarm are, the better the system performance is.

4.3. Baselines

In fact, there are quite a few existing phishing webpage detection approaches. However, some of them are hard to be achieved due to some features are difficult to be obtained. For instance, some approaches [22, 49] used PageRank service, while the service is not available now. Some approaches [12, 13] used DNS features, which rely on a third-party agency. In this work, the baselines include some popular and recently proposed phishing webpage detection approaches as specified as follows.

1) **CANTINA:**CANTINA [26] used the TF-IDF algorithm to extract keywords from HTML text content, and searches them by search engine to detect phishing webpages. The authors have also discussed some simple heuristics that can be applied to reduce false positive rate.

2) **Lightweight Phish Detector (LPD):**LPD [50] was a search engine based phishing webpage detection system. The authors extracted the domain names of URLs and the titles of webpages, and searched them by search engine. Furthermore, they designed a decision-making algorithm to detect the phishing webpages according to the search results.
3) **URL Character Statistical Learning Classifiers (UCSLC)** Verma et al. [51] focused on character distributions by using a two-sample Kolmogorov-Smirnov test, and supplement them with a selected set of heuristic features to enhance robustness.

4.4. Comparing with Baseline Models

CANTINA and LPD rely on search engine, which are difficult to be applied on large-scale data. Therefore, we apply the baseline methods on 2K-PD dataset to compare with our method. Note that CANTINA and LPD do not need training data unlike our approach. Therefore, 70% samples in 2K-PD dataset are selected as the training set, while the remaining ones are for testing. The results are shown in Table 2, from which some observations can be concluded.

(1) The missing rate of CANTINA is relatively high. The reason might be that it highly depends on the result of TF-IDF, which may extract different keywords between training set and test set.
(2) The false alarm rate of LPD is relatively high because that it totally relies on retrieval results of the search engine. However, a large number of low-profile webpages may not be returned under the strategies of the most search engines.
(3) CANTINA and LPD take relatively long time due to using search engines. UCSLC outperforms the previous baseline methods in terms of missing rate and accuracy. However, the false alarm rate is relatively high because the information inside URL is relatively barren. Different from the baselines, our method takes advantage of two-fold information about URL and HTML, outperforming the baselines in terms of the three metrics. Meanwhile, we do not use any third-party service.

Table.2. Performance of the approaches on 2K-PD Dataset

Method	Missing rate(%)	False alarm rate(%)	Accuracy(%)
CANTINA	70	7.5	61.25
LPD	7.6	48	72.2
UCSLC	7.8	9.5	91.35
Our Method	**3.4**	**3.7**	**96.45**

4.5. Comparing with Single Machine Learning Models

In order to show the effectiveness of stacking, we compare the proposed stacking model with the single machine learning models, including Support Vector Machine (SVM), Nearest Neighbor Classifier (NN), Decision Tree (DT), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), XGBoost (XGB) and LightGBM (LGB). In particular, GBDT, XGB, and LGB are the basic models adopted by the proposed stacking model. The evaluations are conducted on the 50K-PD dataset, which is divided into a training set consisting of 70% data samples, and a test set containing the rest ones. The performances of the approaches are shown in Table 3. We can observe that the proposed stacking approach achieves better performance than other single models on the three metrics.

Table.3. Performance of the approaches on 50K-PD Dataset

Model	Missing rate(%)	False alarm rate(%)	Accuracy(%)
SVM	10.13	2.71	94.45
NN	10.87	2.80	94.12
DT	7.48	4.92	94.10
RF	7.51	1.08	96.46
GBDT	5.3	1.94	96.77
XGB	5.70	1.92	96.64
LGB	4.82	1.68	97.12
Our Method	**4.46**	**1.61**	**97.30**

4.6. Impact of Dimensionality of HTML String Embedding

As we mentioned in Section 3.2, we use Word2Vec to learn HTML string embedding in vector forms. Therefore, we evaluate the impact of the dimensionality of the HTML string embedding. As shown in Fig.8, with the increase of the dimensionality of the HTML string embedding, the accuracy rate increases first and then decreases. The proposed approach achieves best performance using the HTML string embedding with 200 dimensions. Overall, the influence is quite small with a fluctuating rate no more than 0.5% on the performance.

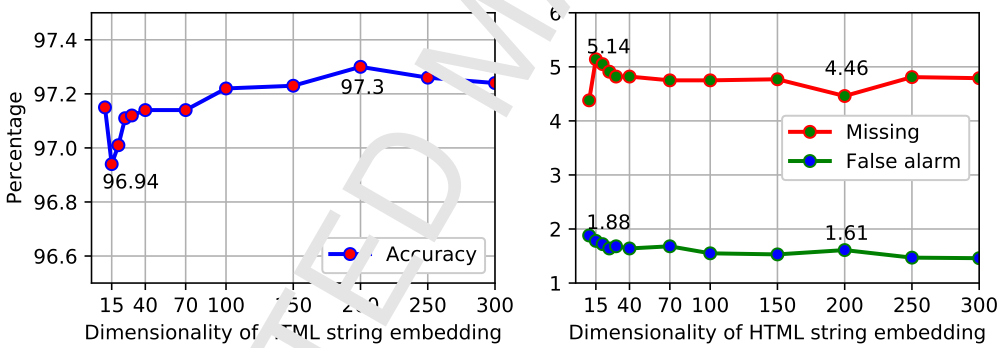


Fig.8. Impact of the dimensionality of the HTML string embedding

4.7. Comparing HTML String Embedding with Artificial Features

In Fig.9, we report the performance of GBDT classifier using the proposed HTML string embedding and artificially designed features (which are features mentioned in Section 3.1 except HTML string embedding) on 50K-PD dataset. The dimensionality of the HTML string embedding is set as 200. As shown in Fig. 9, the HTML string embedding performs quite close to the artificially designed features in terms of accuracy rate, missing rate, and false alarm rate. It is worth noting that HTML string embedding does not take any domain knowledge of phishing, and the embedding learning process does not rely on any supervised or external information. The experimental results indicate the significance of the HTML string embedding for phishing webpage detection.

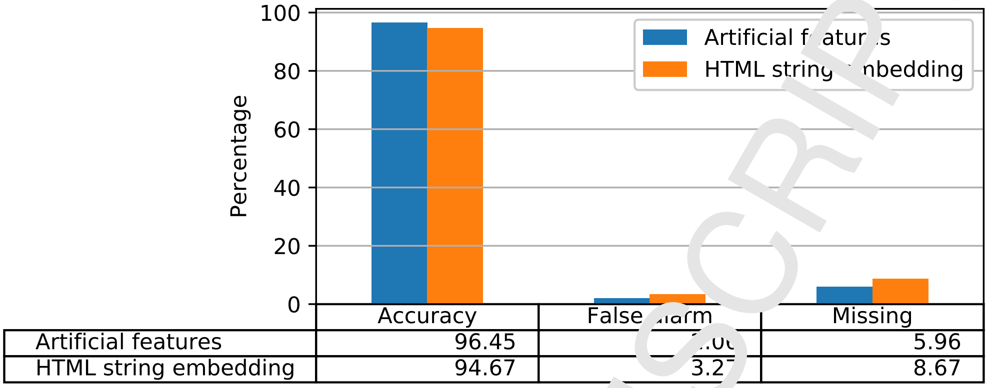


Fig.9. HTML string embedding versus artificial features

4.8. Importance of the Features

As specified in Section 3.1, two-fold features related to URL and HTML are introduced, including 238 features in total. We use the importance coefficient of the features in GBDT to visualize the importance of the features as shown in Fig.10, from which we can observe that the top-3 important features are "HTML string embedding", "URL length information" and "Internal and external resources". It is interesting that the importance of "HTML string embedding" is very high, even 10 times beyond the second-ranked "URL length information", although the specific meanings of them are not explainable like artificial ones. In addition, "Length of HTML content", "Number of the URL brand name appears in HTML code" and "Top-level domain name related" are also quite important for phishing webpage detection. Besides, "IP address", "Similar target brands" and "Login form" are relatively unimportant. For "IP address" and "Login form", all of these features have been used for phishing webpage detection for a long time, phishers already have strategies to avoid this type of detection. For "Similar target brands", there are 20 brands in the brand set, which is too small.

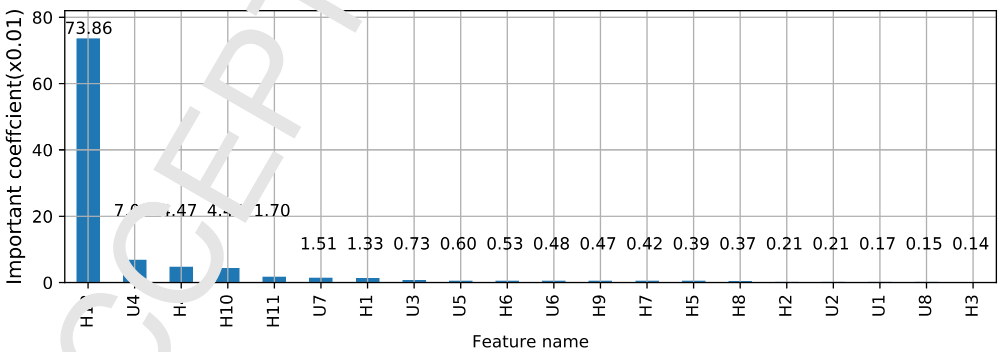


Fig.10. The importance of the features. In order to facilitate the display, we use the short name to represent the feature name, i.e. 'U2' represents the second URL-based

feature and 'H1' represents the first HTML-based feature (refer to Section 3.1), and so on. Furthermore, the numbers on the figure are the result of a 100x magnification.

In addition, we evaluate the performance of using a single feature. The result is shown in Fig.11, where the features are sorted by accuracy. Among these features, "Alarm window", "Redirection", "Https", "Number of dots in domain name" and "IP address" have a high missing rate when being used alone (nearly 100%), indicating that they are ineffective.

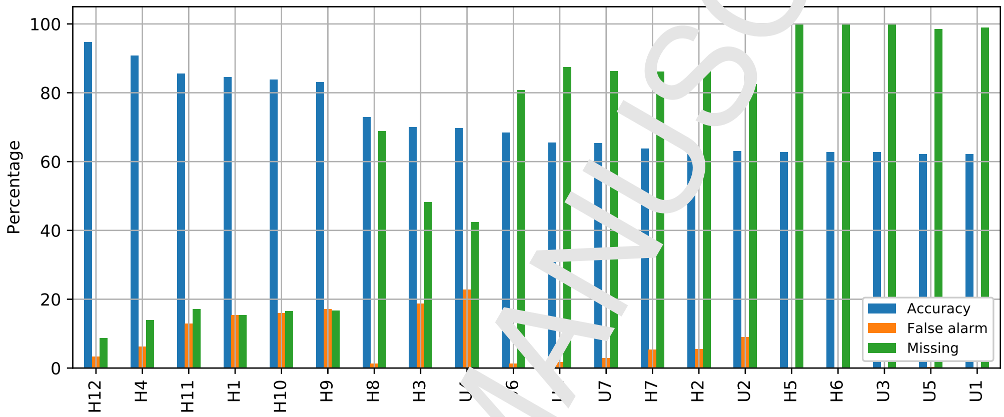


Fig.11. The performance of each single feature. In order to facilitate the display, we use the short name to represent the feature name, i.e. 'U2' represents the second URL-based feature and 'H1' represents the first HTML-based feature (refer to Section 3.1) and so on.

To illustrate the effectiveness of the new features designed by us, we divide the features into an existing feature set E including 6 URL-related features and 6 HTML-related features (refer to Section 3.1); a new URL-related feature set U designed by us, including 2 features (refer to Section 3.1); and a new HTML-related feature set H designed by us including 6 features (refer to Section 3.1). Furthermore, we compare the performance of the proposed stacking model on the different combinations of the feature sets. As shown in Fig.12, when we combining E with U, our approach can get some improvement. When we combine E with H, the improvement on the performance is significant. The experimental results show the effectiveness of the new URL and HTML features designed by us.

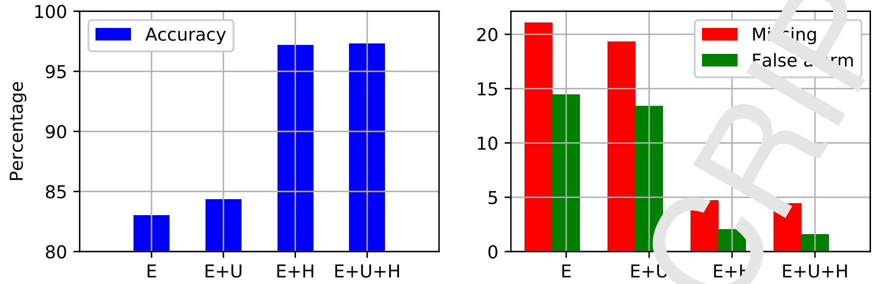


Fig.12. Comparison of the combinations of the features

4.9. Impact of Number of Layers in Stacking Model

In this section, we study the impact of the number of layers of the stacking model. As shown in Fig.13, the proposed stacking model performs the best when the number of stacking layer is set as 2. The performance of the stacking model will not be improved significantly with the increase of the layers after a layer number of 2. It is determined by the amount of information contained in the features. As differences between the outputs of the high-layers tend to become smaller and smaller as the increase of the layers, thus benefiting the predicting model less and less.

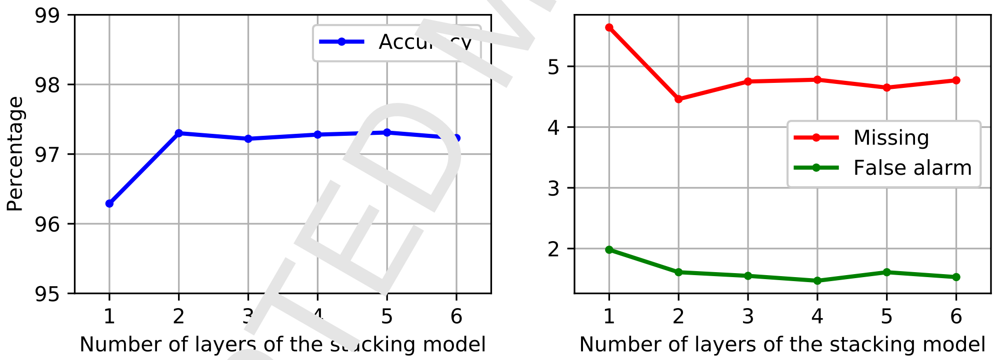


Fig.13. The influence of the number of layers

4.10. Explanation of the Selection of Basic Models

Generally speaking, ensemble methods work better when using diverse and effective basic models [52]. On one hand, to measure the diversity of the models, we use Kappa-Statistic [53] to measure the predicted results of pairs of classifiers. More specifically, given two classifiers h_i and h_j, let a denote the number of samples that are predicted as phishing by both h_i and h_j, b denote the number of samples which are predicted as phishing by h_i and are predicted as legitimate by h_j, c denote the number of samples are predicted as legitimate by h_i and are predicted as phishing by h_j, and d denote the number of samples are predicted as legitimate by both h_i and h_j. Then we can estimate the Kappa-measure between h_i and h_j according to:

$$k_p = \frac{p_1 - p_2}{1 - p_2} \tag{4}$$

where p_1 and p_2 can be calculated according to

$$p_1 = \frac{a + d}{a + b + c + d} \tag{5}$$

$$p_2 = \frac{(a + b)(a + c) + (c + d)(b + d)}{(a + b + c + d)^2} \tag{6}$$

The closer k_p is to 1, the less diversity between h_i and h_j is.

On the other hand, to measure the effectiveness of the models, we adopt average error as a performance metric. We select Support Vector Machine (SVM), Nearest Neighbor classifier (NN), Decision Tree (DT), Gradient Boosting Decision Tree (GBDT), XGBoost (XGB) and LightGBM (LGB) as candidates of basic models for stacking, and conduct the experiments on the training set of 50K-PD dataset by applying 5-fold cross validation. As shown in Table 4, some pairs of classifiers have low value of k_p, but the average error of them is high, i.e., SVM and DT, NN and LGB.

Table.4. Kappa-Statistic and average error of pairs of classifiers

Classifiers	k_p	Average Error	Classifiers	k_p	Average Error
SVM,NN	0.913	0.063	NN,LGB	0.888	0.055
SVM,DT	0.852	0.060	DT,GBDT	0.890	0.057
SVM,GBDT	0.908	0.056	DT,XGB	0.900	0.056
SVM,XGB	0.904	0.053	DT,LGB	0.886	0.056
SVM,LGB	0.912	0.051	GBDT,XGB	0.950	0.037
NN,DT	0.952	0.061	GBDT,LGB	0.901	0.033
NN,GBDT	0.886	0.058	XGB,LGB	0.907	0.032
NN,XGB	0.891	0.055			

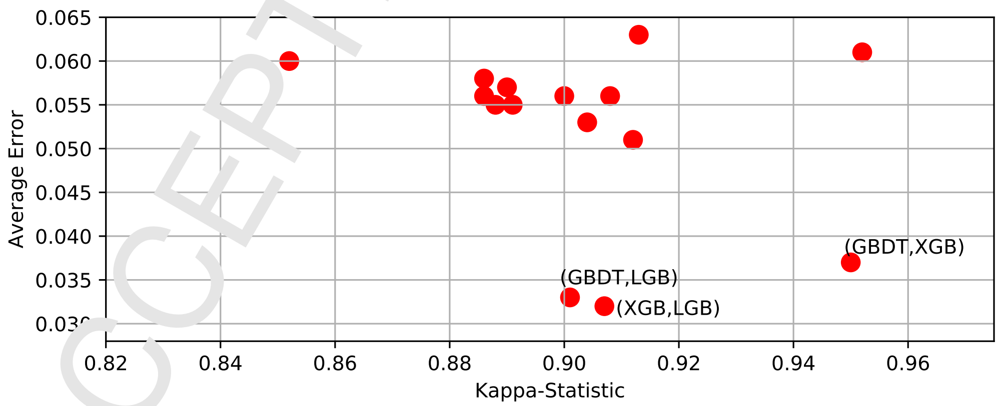


Fig.14. k_p and average error distribution of the individual classifiers

The average error and Kappa-Statistic of each pair of candidates are shown in Fig.14. We select the basic models for stacking by the principals of low average error and high diversity. Therefore, we choose GBDT, XGB and LGB as basic models.

4.11. Implementation Details

The parameters of our stacking model include two aspects, i.e., the parameters of each basic model, and the parameters in the mechanism of stacking (e.g., the number of the basic model in each stacking layer, and the number of layers). In the training stage, we conduct 5-fold cross validation on the training set. Whenever tuning one parameter, we fix the other parameters to train the model, and determine the values of the parameters by observing the performance of the model on the validation set. For simplicity, we set the parameters of the basic models to be default values in practice. As mentioned in Section 4.10, our model has only three basic models in each layer and stacks for two layers finally, considering that complex models may have the risk of overfitting.

We train the stacking model layer by layer. In the training stage of each layer, we train the basic model parallelized. While training each basic model, we conduct hold out validation to measure the training error and evaluating error. In terms of the time cost for training, our method takes 64 minutes (CPU: i7-6700HQ, RAM: 16G) on the training set of 50K-PD dataset (around 35K training samples).

In terms of updating the model, we use incremental learning strategy to train XGB and LGB if there are new phishing data, which avoids training from scratch. For the basic model GBDT, we may add the new phishing data into training set to train an updated model.

4.12. Exploring on Visual Features

The aforementioned features do not include visual features. Therefore, we further investigate the effectiveness of visual features. As both 2K-PD and 50K-PD datasets do not contain images, we collect a new dataset named as 50K-IPD, which contains URL, HTML code and the screenshot of final rendered webpage for each sample. The 50K-IPD dataset has been introduced in Section 4.1.

We design two methods to extract visual features. On one hand, we use a pre-trained ResNet18 [54] model for feature extraction. We revise the output of the ResNet18 into two dimensions and fix all the parameters of the CONV layers. Furthermore, we train the model on 50K-IPD dataset to classify the images of webpages. Finally, we take the second-to-last layers output of the model as the image features. On the other hand, we design a small-scale convolutional neural network (CNN), which has three convolutional layers and three fully connected layers. We train the CNN model on 50K-IPD dataset and take the second-to-last layers output of CNN as the webpages image features. Table 5 shows a comparison of the performance of our model with and without visual features.

Table.5. The performance of our stacking model using visual features

Visual Features Extractor	Number of Visual Features	Missing rate(%)	False alarm rate(%)	Accuracy(%)
RseNet18	64	2.67	2.97	97.19
CNN	64	1.28	1.54	98.60
Without visual features	0	3.02	2.75	97.11

From the table, we can see that the performance of our model using visual features has been improved significantly. It shows that visual features are quite helpful to the current task. To our surprise, when using the visual features extracted by CNN, the performance is improved more obviously. However, visual features extracted from ResNet18 contributes a little. The reason might be that most of the parameters of ResNet18 are trained on ImageNet. There is a gap between the pictures in ImageNet and the pictures of the webpages.

5. Conclusions and Future Work

5.1. Conclusions

In this work, we design two-fold features considering the characteristics of URLs and HTML documents of phishing webpages. The features do not rely on the third-party services so that they can be used for developing real time applications. In particular, we use Word2Vec model to learn HTML string embedding from HTML codes. The newly-designed features are effective for phishing webpage detection. Given the two-fold features, we design a stacking model by combining GBDT, XGBoost, and LightGBM to detect phishing webpages. Experiments show that the proposed approach reaches 97.3% in terms of accuracy rate.

5.2. Future Work

There are several works we plan to investigate in the future as follows.

1. **Dataset enlargement.** The current phishing URLs are collected from Phishtank. We plan to collect more data samples released by some phishing webpage detection competitions or finding some business partners focusing on this problem, in order to increase the diversity of the dataset.
2. **Increase the robustness of HTML strings embedding model.** To learn the HTML strings embedding, we have to obtain the HTML strings in advance. The disadvantage is that it cannot learn embedding for the new HTML strings that never appear in the training set or training corpus. Therefore, we have to consider how to generate the HTML strings corpus, including extracting the tags, variables, and parameters as corpus or splitting each character inside the HTML code to create the corpus.
3. **Target Recognition.** Researchers have developed quite a few methods to find out the target of a phishing webpage. However, most of them usually depend on multi-page information or search engines, which are time-consuming. We plan to design an intelligent system to recognize the phishing target based on single webpage information, especially the visual features, in a more efficient manner.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61703109, No.917[illegible]7), and the Guangdong Innovative Research Team Program (No. 2014ZT05G157).

References

[1] APWG, Phishing activity trends report, http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf, accessed Dec 10, 2017.

[2] G. Developers, Safe browsing api-developer guide v3, https://developers.google.com/safebrowsing/developers_guide_v3, accessed May 13, 2014.

[3] L. Li, E. Berki, M. Helenius, S. Ovaska, Towards a contingency approach with whitelist- and blacklist-based anti-phishing applications: What do usability tests indicate? Behaviour and Information Technology 33 (11) (2014) 1136–1147. doi:10.1080/0144929X.2013.875221.

[4] E. H. Chang, K. L. Chiew, S. N. Sze, W. K. Tiong, Phishing detection via identification of website identity, in: International Conference on IT Convergence and Security - ICITCS'13, 2013, pp. 1–4. doi:10.1109/ICITCS.2013.6717870.

[5] M. Dunlop, S. Groat, D. Shelly, GoldPhish: Using images for content-based phishing analysis, in: Proceedings of the 5th International Conference on Internet Monitoring and Protection - ICIMP'10, 2010, pp. 123–128. doi:10.1109/ICIMP.2010.24.

[6] J. H. Huh, H. Kim, Phishing detection with popular search engines: Simple and effective, in: Foundations and Practice of Security, 2012, pp. 194–207. doi:10.1007/978-3-642-27901-0_15.

[7] G. Ramesh, I. Krishnamurthi, K. S. S. Kumar, An efficacious method for detecting phishing webpages through target domain identification, Decision Support Systems 61 (1) (2014) 12–22. doi:10.1016/j.dss.2014.01.002.

[8] T.-C. Chen, T. Stepan, S. Dick, J. Miller, An Anti-Phishing System Employing Diffused Information, ACM Transactions on Information and System Security 16 (4) (2014) 1–31. doi:10.1145/2584680.

[9] P. D. Ryck, N. Nikiforakis, TabShots: client-side detection of tabnabbing attacks, in: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security - ASIA CCS'13, 2013, pp. 447–455. doi:10.1145/2484313.2484371.

[10] J. Mao, P. Li, K. Li, T. Wei, Z. Liang, BaitAlarm: Detecting phishing sites using similarity in fundamental visual features, in: Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems - INCoS'13, 2013, pp. 790–795. doi:10.1109/INCoS.2013.151.

[11] I. F. Lam, W. C. Xiao, S. C. Wang, K. T. Chen, Counteracting phishing page polymorphism: An image layout analysis approach, in: Advances in Information Security and Assurance, 2009, pp. 270–279. doi:10.1007/978-3-642-02617-1_28.

[12] B. Sun, Q. Wen, X. Liang, A DNS based anti-phishing approach, in: Proceedings of the 2nd International Conference on Networks Security, Wireless Communications and Trusted Computing - NSWCTC'10, Vol. 2, 2010, pp. 262–265. doi:10.1109/NSWCTC.2010.196.

[13] H. Kim, J. Huh, Detecting DNS-poisoning-based phishing attacks from their network performance characteristics, Electronics Letters 47 (11) (2011) 656. doi:10.1049/el.2011.0399.

[14] M. Aburrous, M. A. Hossain, K. Dahal, F. Thabtah, Intelligent phishing detection system for e-banking using fuzzy data mining, Expert Systems with Applications 37 (12) (2010) 7913–7921. doi:10.1016/j.eswa.2010.04.044.

[15] P. A. Barraclough, M. A. Hossain, M. A. Tahir, G. Sexton, N. Aslam, Intelligent phishing detection and protection scheme for online transactions, Expert Systems with Applications 40 (11) (2013) 4697–4706. doi:10.1016/j.eswa.2013.02.009.

[16] B. Eshete, Effective analysis, characterization, and detection of malicious web pages, in: Proceedings of the 22nd International Conference on World Wide Web - WWW'13, 2013, pp. 355–360. doi:10.1145/2487788.2487942.

[17] M. He, S. J. Horng, P. Fan, M. K. Khan, R. S. Run, J. L. Lai, R. J. Chen, A. Sutanto, An efficient phishing webpage detector, Expert Systems with Applications 38 (10) (2011) 12018–12027. doi:10.1016/j.eswa.2011.01.046.

[18] M. M. Rami, F. Thabtah, L. McCluskey, Intelligent Rule based Phishing Websites Classification, IET Information Security 8 (3) (2014) 153–160. doi:10.1049/iet-ifs.2013.0202.

[19] L. A. T. Nguyen, B. L. To, H. K. Nguyen, M. H. Nguyen, A novel approach for phishing detection using URL-based heuristic, in: International Conference on Computing, Management and Telecommunications - ComManTel 2014, 2014, pp. 298–303. doi:10.1109/ComManTel.2014.6825621.

[20] L. Xu, Z. Zhan, S. Xu, K. Ye, Cross-Layer Detection of Malicious Websites, in: Proceedings of the 3rd ACM conference on Data and application security and privacy - CODASPY'13, 2013, pp. 141–152. doi:10.1145/2435349.2435366.

[21] W. Zhuang, Q. Jiang, T. Xiong, An intelligent anti-phishing strategy model for phishing website

detection, in: Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops - ICDCSW'12, 2012, pp. 51–56. doi:10.1109/ICDCSW.2012.66.

[22] G. Xiang, J. Hong, C. P. Rose, L. Cranor, CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites, ACM Transactions on Information and System Security 14 (2) (2011) 1–28. doi:10.1145/2019599.2019606.

[23] APWG, Phishing activity trends paper, http://www.antiphishing.org/resources/apwg-papers/, accessed June 15, 2017.

[24] G. Code, word2vec: Tool for computing continuous distributed representations of words., https://code.google.com/archive/p/word2vec/, accessed Feb 26, 2018.

[25] Y. Cao, W. Han, Y. Le, Anti-phishing based on automated individual white-list, in: Proceedings of the 4th ACM workshop on Digital identity management - DIM'08, 2008, p. 51. doi:10.1145/1456424.1456434.

[26] Y. Zhang, J. Hong, L. Cranor, Cantina: a content-based approach to detecting phishing web sites, in: Proceedings of the 16th international conference on World Wide Web - WWW'07, 2007, pp. 639–648. doi:10.1145/1242572.1242659.

[27] S. Abu-Nimeh, D. Nappa, X. Wang, S. Nair, A comparison of machine learning techniques for phishing detection, in: Proceedings of the the anti-phishing working groups 2nd annual eCrime researchers summit, 2007, pp. 60–69. doi:10.1145/1299015.1299021.

[28] R. Gowtham, I. Krishnamurthi, A comprehensive and efficacious architecture for detecting phishing webpages, Computers and Security 40 (2014) 23–37. doi:10.1016/j.cose.2013.10.004.

[29] S. Marchal, J. Francois, R. State, T. Engel, PhishScore: Hacking phishers' minds, in: Proceedings of the 10th International Conference on Network and Service Management - CNSM'15, 2015, pp. 46–54. doi:10.1109/CNSM.2014.7014140.

[30] D. Vishwanath, S. Gupta, Adding CNNs to the Mix: Stacking models for sentiment classification, in: 2016 IEEE Annual India Conference - INDICON'16, 2017, pp. 1–4. doi:10.1109/INDICON.2016.7839062.

[31] H. M. Nguyen, S. Woo, J. Im, T. Jun, D. Kim, A workload prediction approach using models stacking based on recurrent neural network and autoencoder, in: Proceedings of the 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, 2017, pp. 929–936. doi:10.1109/HPCC-SmartCity-DSS.2016.0133.

[32] Z. Q. Wang, D. Wang, Recurrent deep stacking networks for supervised speech separation, ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (2017) 71–75arXiv:1612.04675, doi:10.1109/ICASSP.2017.7952120.

[33] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, P. Stamatopoulos, Stacking Classifiers for Anti-spam Filtering of E-mail, Empirical methods in Natural Language Processing (2001) 44–50arXiv:0106040.

[34] Anandita, D. P. Yadav, P. Paliwal, D. Kumar, R. Tripathi, A Novel Ensemble Based Identification of Phishing E-Mails, in: Proceedings of the 9th International Conference on Machine Learning and Computing - ICMLC'17, 2017, pp. 447–451. doi:10.1145/3055635.3056583.

[35] M. ZhiWei, M. M. Singh, Z. F. Zaaba, Email Spam Detection:A Method of Metaclassifiers Stacking, in: Proceedings of the 6th International Conference on Computing and Informatics - ICOCI'17, 2017, pp. 750–757.

[36] X. Gu, H. Wang, T. Ni, An efficient approach to detecting phishing web, Journal of Computational Information Systems 9 (14) (2013) 5553–5560. doi:10.12733/jcis6350.

[37] S. Marchal, K. Saari, N. Singh, N. Asokan, Know Your Phish: Novel Techniques for Detecting Phishing Sites and Their Targets, in: Proceedings of the 36th International Conference on Distributed Computing Systems - ICDCS'16, 2016, pp. 323–333. arXiv:1510.06501, doi:10.1109/ICDCS.2016.10.

[38] D. McGrath, M. Gupta, Behind phishing: an examination of phisher modi operandi, in: Proceedings of the Usenix Workshop on Large-Scale Exploits and Emergent Threats - LEET'08, 2008, p. 4.

[39] S. Garera, N. Provos, M. Chew, A. D. Rubin, A framework for detection and measurement of phishing attacks, in: Proceedings of the 2007 ACM workshop on Recurring malcode - WORM '07, 2007, pp. 1–8. doi:10.1145/1314389.1314391.

[40] L. Yujian, L. Bo, A normalized Levenshtein distance metric, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6) (2007) 1091–1095. doi:10.1109/TPAMI.2007.1078.

[41] M. G. Alkhozae, O. A. Batarfi, Phishing Websites Detection based on Phishing Characteristics in the Webpage Source Code, International Journal of Information and Communication Technology Research (2011) 1 (6) (2011) 283–291.

[42] R. Basnet, A. Sung, Q. Liu, Rule-Based Phishing Attack Detection, in: Proceedings of the International Conference on Security and Management-SAM'11, 2011, pp. 1–8.

[43] T. Mikolov, K. Chen, G. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality, in: Proceedings of the 26th Neural Information Processing Systems - NIPS'13, 2013, pp. 1–9. arXiv:1310.4546, doi:10.1162/jmlr.2003.3.4-5.951.

[44] J. H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics 29 (5) (2011) 1189–1232. doi:doi:10.1214/aos/1013203451.

[45] T. Chen, C. Gusetrin, Xgboost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD'16, 2016, pp. 785–794. arXiv:1603.02754, doi:10.1145/2939672.2939785.

[46] G. Ke, Q. Meng, T. Wang, W. Chen, W. Ma, T.-Y. Liu, LightGBM:A Highly Efficient Gradient Boosting Decision Tree, 2017, pp. 3148–3156.

[47] Alexa, Keyword research, competitive analysis, & webpage ranking, https://www.alexa.com/, accessed Feb 26, 2018.

[48] Phishtank, Phishtank > developer information., http://www.phishtank.com/developer_info.php, accessed Feb 26, 2018.

[49] N. Abdelhamid, A. Ayesh, F. Thabtah, Phishing detection based Associative Classification data mining, Expert Systems with Applications 41 (13) (2014) 5948–5959. doi:10.1016/j.eswa.2014.03.019.

[50] G. Varshney, M. Misra, P. K. Atrey, A phish detector using lightweight search features, Computers and Security 62 (2016) 213–228. doi:10.1016/j.cose.2016.08.003.

[51] R. Verma, K. Dyer, On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifier, in: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy - CODASPY'15, 2015, pp. 111–122. doi:10.1145/2699026.2699115.

[52] Z. H. Zhou, Ensemble Methods Foundations and Algorithms, 2012.

[53] T. G. D. Dragos D Margineantu, Pruning Adaptive Boosting, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 112–122.

[54] S. R. Kaiming He, Xiangyu Zhang, J. Sun, Deep Residual Learning for Image Recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

Yukun Li is currently a graduate student in the School of Automation, Guangdong University of Technology, China. His research interests include anti-phishing, data mining, machine learning and natural language processing.

Zhenguo Yang is a post-doctoral fellow at Guangdong University of Technology. He received his Ph.D. degree in Computer Science from City University of Hong Kong, in 2017, and the M.E. degree in Computer Science from Zhejiang Normal University, China, in 2013, and the B.E. degree in Computer Science from Shandong Normal University, China, in 2010. His research interests include phishing detection, social media analysis, event detection, and transfer learning, etc.

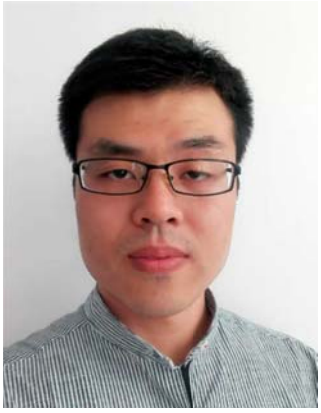
Xu Chen is currently a graduate student in the School of Automation, Guangdong University of Technology, China. His research interests include anti-phishing, data mining, and machine learning.

Huaping Yuan is currently a graduate student in the School of Computer Science and Technology, Guangdong University of Technology, China. His research interests include anti-phishing, data mining, machine learning and natural language processing.

Wenyin Liu is currently a Professor in School of Computer Science and Technology, Guangdong University of Technology. He was Deputy Director of Multimedia software Engineering Research Centre at the City University of Hong Kong from 2013 to 2016, an assistant professor in the Department of Computer Science at the City University of Hong Kong between from 2002 to 2012, and a full time researcher at Microsoft Research China/Asia from 1999 to 2001. His current research interests include blockchain, anti-phishing, Web identity authentication and management. He has a BEng and MEng in computer science from Tsinghua University, Beijing and a DSc from the Technion, Israel Institute of Technology, Haifa. In 2003, he was awarded the International Conference on Document Analysis and Recognition Outstanding Young Researcher Award by the International Association for Pattern Recognition (IAPR). He had been TC10 chair of IAPR for 2006-2010. He had been on the editorial boards of the International Journal of Document Analysis and Recognition (IJDAR) from 2006 to 2011 and the IET Computer Vision journal from 2011-2012. He is a Fellow of IAPR and a senior member of IEEE.

Yukun Li

Zhenguo Yang



Xu Chen

Huaping Yuan

ACCEPTED MANUSCRIPT

Wenyin Liu

A real-time phishing webpage detection system that can be used to protect users from phishing attacks is proposed.

HTML string embedding is proposed for extracting features from HTML code automatically.

Stacking model combines multiple machine learning models for better performance.