Expert Systems With Applications 117 (2019) 345–357

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Machine learning based phishing detection from URLs

Ozgur Koray Sahingoz[a,*], Ebubekir Buber[b], Onder Demir[b], Banu Diri[c]

[a] Istanbul Kultur University, Computer Engineering Department, 34158 Istanbul, Turkey
[b] Marmara University, Technology Faculty, Computer Engineering Department, Istanbul, Turkey
[c] Yildiz Techical University, Computer Engineering Department, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 7 May 2018
Revised 25 July 2018
Accepted 12 September 2018
Available online 18 September 2018

Keywords:
Cyber security
Phishing attack
Machine learning
Classification algorithms
Cyber attack detection

ABSTRACT

Due to the rapid growth of the Internet, users change their preference from traditional shopping to the electronic commerce. Instead of bank/shop robbery, nowadays, criminals try to find their victims in the cyberspace with some specific tricks. By using the anonymous structure of the Internet, attackers set out new techniques, such as phishing, to deceive victims with the use of false websites to collect their sensitive information such as account IDs, usernames, passwords, etc. Understanding whether a web page is legitimate or phishing is a very challenging problem, due to its semantics-based attack structure, which mainly exploits the computer users' vulnerabilities. Although software companies launch new anti-phishing products, which use blacklists, heuristics, visual and machine learning-based approaches, these products cannot prevent all of the phishing attacks. In this paper, a real-time anti-phishing system, which uses seven different classification algorithms and natural language processing (NLP) based features, is proposed. The system has the following distinguishing properties from other studies in the literature: language independence, use of a huge size of phishing and legitimate data, real-time execution, detection of new websites, independence from third-party services and use of feature-rich classifiers. For measuring the performance of the system, a new dataset is constructed, and the experimental results are tested on it. According to the experimental and comparative results from the implemented classification algorithms, Random Forest algorithm with only NLP based features gives the best performance with the 97.98% accuracy rate for detection of phishing URLs.

1. Introduction

Due to the rapid developments of the global networking and communication technologies, lots of our daily life activities such as social networks, electronic banking, e-commerce, etc. are transferred to the cyberspace. The open, anonymous and uncontrolled infrastructure of the Internet enables an excellent platform for cyberattacks, which presents serious security vulnerabilities not only for networks but also for the standard computer users even for the experienced ones. Although carefulness and experience of the user are important, it is not possible to completely prevent users from falling to the phishing scam (Greene, Steves, & Theofanos, 2018). Because, to increase the success of the phishing attacks, attackers also get into consideration about the personality characteristics of the end user especially for deceiving the relatively experienced users (Curtis, Rajivan, Jones, & Gonzalez, 2018). End-user-targeted cyberattacks cause massive loss of sensitive/personal information and even money for individuals whose total amount can reach billions of dollars in a year (Shaikh, Shabut, & Hossain, 2016).

Phishing attacks' analogy is derived from "fishing" for victims, this type of attacks has attracted a great deal of attention from researchers in recent years. It is also a promising and attractive technique for attackers (also named as phishers) who open some fraudulent websites, which have exactly similar design of the popular and legal sites on the Internet. Although these pages have similar graphical user interfaces, they must have different Uniform Resource Locators (URLs) from the original page. Mainly, a careful and experienced user can easily detect these malicious web pages by looking at the URLs. However, due to the speed of life, most of the times, end users do not investigate the whole address of their active web page, which is generally forwarded by other web pages, social networking tools or by simply an email message as depicted in Fig. 1. By using this type of fraudulent URLs, a phisher tries to capture some sensitive and personal information of the victim like financial data, personal information, username, password, etc. (Gupta, Arachchilage, & Psannis, 2018). In the case of entering this type of fraudulent site, which is believed to be the original website, computer users can easily give their sensitive information

* Corresponding author.
E-mail addresses: o.sahingoz@iku.edu.tr (O.K. Sahingoz), ebubekirbbr@gmail.com (E. Buber), odemir@marmara.edu.tr (O. Demir), banu@ce.yildiz.edu.tr (B. Diri).

https://doi.org/10.1016/j.eswa.2018.09.029
0957-4174/

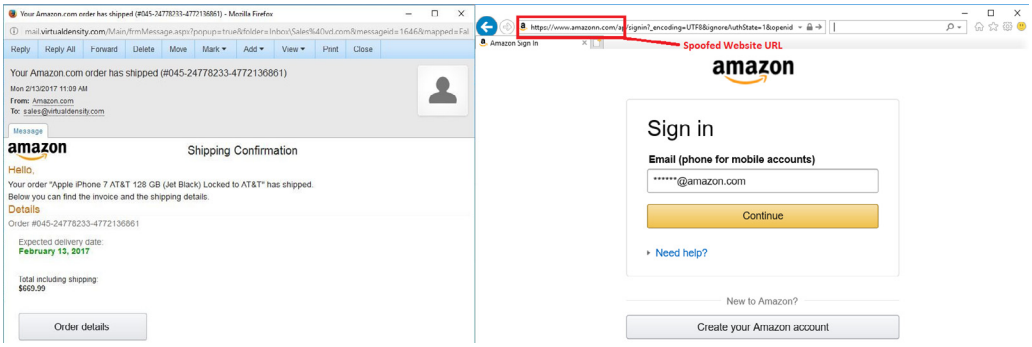

Fig. 1. Sample of a phishing mail and a web page.

without any doubt. Because the entered web page seems exactly same with the original web page.

In a related study about the user experiences of phishing attacks (Volkamer, Renaud, Reinheimer, & Kunz, 2017) computer users fall for phishing due to the five main reasons:

- Users don't have detailed knowledge about URLs,
- Users don't know, which web pages can be trusted,
- Users don't see the whole address of the web page, due to the redirection or hidden URLs,
- Users don't have much time for consulting the URL, or accidentally enter some web pages,
- Users cannot distinguish phishing web pages from the legitimate ones.

Anti-Phishing Working Group published a report about the position of the phishing attacks in the last quarter of 2016 (APWG, 2017). They emphasized that phishing attacks especially target the end users in developing countries, which are ordered as firstly China with the rate of 47.09% (infected computers) and then he is followed by Turkey and Taiwan with the rate of 42.88% and 38.98% respectively. Additionally, due to the increased use of smartphones, the end users are not so careful while checking their social networks in motion. Therefore, attackers target the mobile device users to increase the efficiency of their attacks (Goel & Jain, 2018).

In the literature, there are some studies, which are focused on detecting phishing attacks. In the recent surveys, authors discuss the general characteristics of the existing phishing techniques by categorizing the technical approaches used in these type of attacks, and some practical and effective combating techniques are highlighted (Chiew, Yong, & Tan, 2018; Qabajeh, Thabtah, & Chiclana, 2018).

Phishing attacks exploit the vulnerabilities of the human users, therefore, some additional support systems are needed for the protection of the systems/users. The protection mechanisms are classified into two main groups: by increasing the awareness of the users and by using some additional programs as depicted in Fig. 2. Due to the vulnerability of the end user, an attacker can even target some experienced users by using new techniques and before giving the sensitive information, he is believed that this page is legitimate. Therefore, software-based phishing detection systems are preferred as decision support systems for the user. Mostly preferred techniques are Black/White Lists (Cao, Han, & Le, 2008), Image Processing (Fu, Wenyin, & Deng, 2006), (Toolan & Carthy, 2009) of the web page, Natural Language Processing (Stone, 2007), Rules (Cook, Gurbani, & Daniluk, 2008), Machine Learning (Abu-Nimeh, Nappa, Wang, & Nair 2007), etc.

In one of the recent survey (Gupta et al., 2018) on phishing, authors emphasized that when some new solutions are proposed to overcome various phishing attacks, attackers came with the vulnerabilities of the solution and produced new attack types. Therefore, it is highly recommended to use hybrid models instead of a single approach by the security manager of the networks.

In this paper, we are focused on the real-time detection of phishing web pages by investigating the URL of the web page with different machine learning algorithms (seven of them implemented and compared in the paper) and different feature sets. In the execution of a learning algorithm, not only the dataset but also the extraction of the features from this dataset are crucial. Therefore, firstly we collect lots of legitimate and fraudulent web page URLs and construct our own dataset. After that, we defined three different types of feature sets as *Word Vectors, NLP based* and *Hybrid* features to measure the efficiency of the proposed system.

The rest of the paper is organized as follows: in the first following section, the related works about phishing detection are examined. Section 3 focuses on the factors that make the detection of phishing attack from URLs difficult. The details of the proposed system and acquisition of the dataset are detailed in Section 4. Some comparative experiments are conducted, and results are depicted in Section. 5. Advantages of the proposed system are discussed in Section 6. Finally, Conclusions and Future Works on this topic are presented.

2. Related works

The phishing detection systems are generally divided into two groups: *List Based Detection Systems and Machine Learning Based Detection Systems.*

2.1. List based detection systems

List-based phishing detection systems use two list, whitelists and blacklists, for classifying the legitimate and phishing web pages. Whitelist-based phishing detection systems make secure and legitimate websites to provide the necessary information. Each website that is not in the whitelist is considered as suspicious. Cao et al. (2008) developed a system that creates a whitelist by registering the IP address of each site, which has Login user interface that the user has visited. When the user visits a website, the system will warn if there is an incompatibility in the registered information of the website. However, this method is considered suspect in the legitimate sites visited by the user for the first time. Jain and Gupta (2016) developed a method that warns users on the web with a white-list of legitimate websites that is updated automatically. The method consists of two phases, the domain - IP address matching module and the extraction of the features of the links in the source code. According to the experimental results, 86.02% true positive rate and 1.48% false negative rate were achieved in this work.

Blacklists are created by URL records, which are known as phishing websites. These list entries are derived from a number of

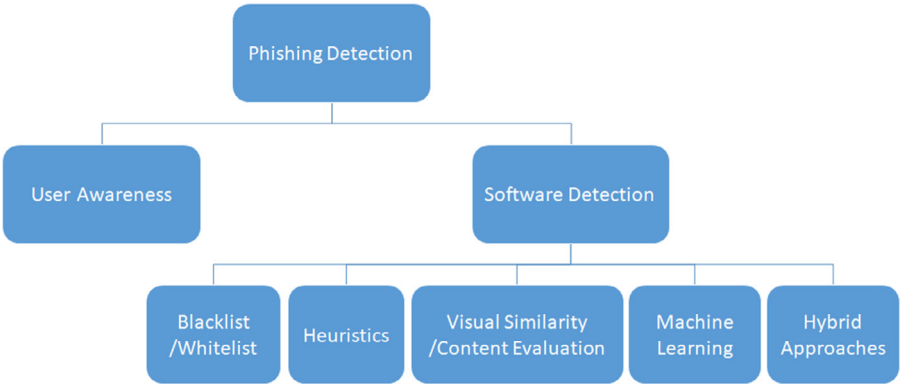

Fig. 2. Phishing detection models.

sources, such as spam detection systems, user notifications, third-party organizations, etc. The use of blacklists makes it impossible for attackers to attack again via same URL or IP address, which are previously used for attack. The security mechanism updates blacklists either by detecting malicious URLs / IPs or users can download these lists instantly from a server and protect their systems against the attacks listed in this list. The blacklist-based systems, however, do not have the ability to detect an actual attack or a first-time attack (zero-day attack). These attack detection mechanisms have a lower false positive rate than machine learning based systems. The success of the blacklist-based phishing attack detection system is about 20% (Khonji, Iraqi, & Jones, 2013; Sheng, Holbrook, Kumaraguru, Cranor, & Downs, 2010). Therefore, it seems that blacklist-based systems are not efficient as a reliable attack detection mechanism. Some companies service blacklist-based phishing attack detection systems such as Google Safe Browsing API (Google Safe Browsing, 2012), PhishNet (Prakash, Kumar, Kompella, & Gupta, 2010). These systems use an approximate matching algorithm to check whether the suspicious URL exists in the blacklist or not. The blacklist-based approaches require frequent updates. In addition, the rapid growth of the blacklist requires excessive system resources (Sharifi, & Siadati, 2008; Sheng et al., 2009).

Apart from the static techniques, dynamic techniques, which can learn from the previous data (especially big data) can produce a better solution with the help of machine learning approaches.

2.2. Machine learning based detection systems

One of the popular methods of malicious websites' detection is the use of machine learning methods. Mainly, detection of phishing attack is a simple classification problem. In order to develop a learning-based detection system, training data must contain lots of features, which are related to phishing and legitimate website classes. By the use of a learning algorithm, it can be easy to detect the unseen or not classified URLs with a dynamic mechanism.

A text-based phishing detection technique was developed by Zhang and named as CANTINA, which extracts keywords using the term frequency-inverse document frequency algorithm (Zhang, Hong, & Cranor, 2007). Then, the keywords were searched by Google search engine. If the website was included in search results, it was classified as legitimate. However, the success of the study is limited because it is sensitive only to the English language. The enhanced model is named as CANTINA+, and it includes 15 HTML-based attributes (Xiang, Hong, Rose, & Cranor, 2011). The system achieved a 92% accuracy rate, but it could produce a large number of false positives. Tan, Chiew, Wong, and Sze (2016) developed an anti-phishing system named PhishWHO, which has three steps to identify a website legitimate or not. At the first stage, keywords are identified from the suspicious website. In the second stage, these keywords are used for the detection of possible target domains by a search engine. The target domain is identified using the features extracted from these websites. Finally, the system decides whether the website queried in the third step is legitimate or not.

Le, Markopoulou, and Faloutsos (2011) identified phishing websites by classifying them with URL attributes such as length, number of special characters, directory, domain name, and file name. The system classifies websites offline using Support Vector Machines. Adaptive Regularization of Weights, Confidence Weighted, and Online Perceptron are used for online classification. According to the results of the experiments, using Adaptive Regularization of Weights algorithm increases the accuracy rate while decreasing system resource requirement.

The message title and priority ranking of the incoming message was taken into account by Islam and Abawajy's study (Islam & Abawajy, 2009). They designed a multi-layered classification to filter the message. Experimental results showed that the system reduces the number of false positives. Jeeva and Rajsingh (2016) extracted features related to transport layer security together with URL based features such as length, number of slashes, number and positions of dots in URL and subdomain names. Rule mining was used to establish detection rules using the apriori algorithm on the extracted features. Experimental results showed that 93% of phishing URLs were detected.

In a recent study, Babagoli, Aghababa, and Solouk (2018) use a nonlinear regression strategy for detecting whether a web site is phishing or not. They prefer the use of harmony search and support vector machine metaheuristic algorithms for training the system. According to them, harmony search produces a better accuracy rate of 94.13% and 92.80% for train and test processes, respectively by using about 11,000 web pages.

In Buber, Diri, and Sahingoz (2017a), which is the previous version of this study, we proposed a phishing detection system with 209 *word vector* features and 17 *NLP based* features. In the related study, the effect of the NLP based features was seen. However, there is a need to increase the number of NLP based features and word vectors. Therefore, in the ongoing study, we focused on this issue and reached better results with increasing the accuracy rate of 7%. In Buber, Diri, and Sahingoz (2017b) the number of NLP vectors are increased in the proposed system three different machine learning algorithms were compared according to their accuracy values.

Mohammad, Thabtah, and McCluskey (2014) implemented a phishing detection system, which uses adaptive self-structuring neural networks for classification. It uses 17 features, and some of them depend on the use of third-party services. Therefore, in the execution of real-time, it needs much more time; however, it can reach better accuracy rates. It uses a limited dataset with 1400 items but shows high acceptance for noisy data.

Jain and Gupta (2018) presents an anti-phishing approach, which uses machine learning by extracting 19 features in the client side to distinguish phishing websites from legitimate ones. They used the 2141 phishing pages from PhishTank (2018) and Openfish (2018), and 1918 legitimate web pages from Alexa popular websites, some online payment gateways, and some top banking websites. With the use of machine learning, their proposed approach reached 99.39% true positive rate.

Feng et al. (2018) propose a novel neural network based classification method for detection of phishing web pages by using Monte Carlo algorithm and risk minimization principle. They used 30 features, which are categorized into four main domains as *address bar-based features, abnormal-based features, HTML and JavaScript-based features and domain-based features*. The detection system reaches to 97.71% accuracy rate and 1.7% false positive rate in the experimental studies.

Although most of the researchers focus on the phishing detection through URLs, some researchers tried to detect phishing emails by checking the data stored in the email packets. To detect phishing attacks, Smadi, Aslam, and Zhang (2018) combined the neural network approach with reinforcement learning for classification. The proposed system contains 50 features, which are grouped into four different categories as mail headers, URLs in the content, HTML content and main text. Although the focus of the system is on the emails, due to the URL extracted features, it has the similarity with our proposed model. It uses 9118 emails as the dataset in, which 50% of them are legitimate and the rest are phishing emails. In the experiments, it reached 98.6% of accuracy rate and 1.8% false positive rate.

In some researches, such as (Rao & Pais, 2018), the authors use a hybrid method by using not only machine learning approaches but also image checking. An important weakness of the image/visual based phishing detection is about the need of an initial image database or prior knowledge (web history) about the web page; however, the proposed approach is free from these dependencies. They used three categories of features: hyperlink-based features, third-party based features, and URL obfuscation features. Although the use of third-party services increases the detection time, it also increases the accuracy rate of the system up to 99.55%.

Use of natural language processing (NLP) has not been encountered in the literature much. In a recent study, Peng, Herris, and Sawa (2018), NLP is applied to detect phishing emails. It performs a semantic analysis of the content of emails (as simple text) to detect malicious intent. With the use of NLP, question and command sentences tried to catch. Specific blacklist of word pairs is used for detection of phishing attacks. They used 5009 phishing emails and 5000 legitimate emails for training and testing the system. They reached 95% precision rate in their experimental works.

The detailed comparison of the Machine Learning Based Phishing Detection Systems is depicted in Table 1.

3. URLs and Attackers' techniques

Attackers use different types of techniques for not to be detected either by security mechanisms or system admins. In this section, some of these techniques will be detailed. To understand the approach of attackers, firstly, the components of URLs should be known. The basic structure of a URL is depicted in Fig. 3.

In the standard form, a URL starts with its protocol name used to access the web page. After that, the subdomain and the Second Level Domain (SLD) name, which commonly refers to the organization name in the server hosting, is located and finally the Top-Level Domain (TLD) name, which shows the domains in the DNS root zone of the Internet takes place. The previous parts compose the domain name (host name) of the web page; however, the inner address is represented by the path of the page in the server and with the name of the page in the HTML form.

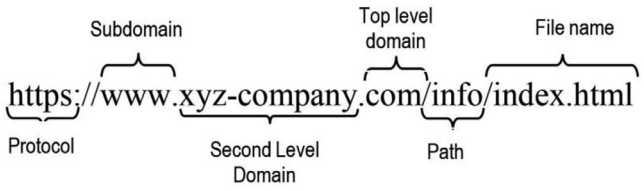

Fig. 3. URL components.

Although SLD name generally shows the type of activity or company name, an attacker can easily find or buy it for phishing. The name of SLD can only be set once, at the beginning. However, an unlimited number of URLs can be generated by an attacker with extending the SLD by path and file names, because the inner address design directly depends on attackers.

The unique (and critical) part of a URL is the composition of SLD and TLD, which is named as domain name. Therefore, cybersecurity companies make a great effort to identify the fraudulent domains by name, which are used for phishing attacks. If a domain name is identified as phishing, the IP address can be easily blocked to prevent from accessing the web pages located in it.

To increase the performance of the attack and steal more sensitive information, an attacker mainly uses some important methods to increase the vulnerability of victims such as the use of random characters, combined word usage, cybersquatting, typosquatting, etc. Therefore, detection mechanisms should take into consideration of these attack methods.

4. The proposed system and data processing

The dataset and its processing are very important parts of the machine learning based systems. The performance and efficiency of the system are directly related to them. Therefore, in this section, these topics are detailed.

4.1. Dataset

To compare the proposed system, we tried to find a worldwide accepted dataset; however, we cannot. Therefore, firstly, there was an urgent need to construct a good and big dataset. To construct an acceptable and balanced dataset, two classes of URLs are needed: legitimate and phishing. In this study, the phishing URLs were mostly provided by PhishTank (2018). However, PhishTank does not give a free dataset on their web page. Therefore, by writing a script, addresses of lots of malicious websites could easily be downloaded. At the same time, there was a need to collect legitimate websites. To collect these pages, we got help from Yandex Search API (YandexXML Yandex Technologies, 2013). Firstly, specific query_word_list was constructed, and then these words are sent to Yandex Search API to get the highest ranked web pages, which had a very low possibility to be phishing web pages. It stems from the fact that; search engines do not give high ranking to the malicious URLs due to their short lifetime.

As a result of these efforts, we collected a very great dataset and shared this on the website (Ebbu2017 Phishing Dataset, 2017) for the use of other researchers. We have performed our test on this dataset, which contains 73,575 URLs. This dataset totally contains 36,400 legitimate URLs and 37,175 phishing URLs.

4.2. Data Pre-processing

A URL consists of some meaningful or meaningless words and some special characters, which separate some important

Table 1
Comparison of machine learning based phishing detection systems.

Project	Description	Pros	Cons
Xiang et al. (2011)	Implements a content-based system to detect phishing web page by using a feature-rich machine learning approach.	* catch the constantly evolving novel phishing attacks * increase the number of features from their previous work (Zhang et al., 2007)	* limited dataset (8118 phishing and 4883 legitimate web pages) * use third-party services * use location-specific data (top 100 English sites) * 15 features (6 URL_based/ 4 HTML_based/ 5 web_paged features)
Le et al. (2011)	Identifies phishing websites by classifying them with URL attributes	* appropriate for client-side deployment * based on an online classification * resilient to noisy data (training)	* use third-party services * limited Dataset (6083 malicious URLs and 8155 legitimate URLs)
Jeeva and Rajsingh (2016)	Define some URL features, and with them, they generate some rules with apriori and predictive apriori rule generation algorithms.	Fast detection with rules (especially with apriori rules)	* use rules for classification. * depends on the quality of the rules. * limited dataset (1200 phishing URLs and 200 legitimate URLs) * 14 Heuristic features * 9 priori and 9 predictive apriori rules
Babagoli et al. (2018)	A meta-heuristic-based nonlinear regression algorithm by utilizing two feature selection methods: decision tree and wrapper	Original UCI dataset is decreased from 30 to 20, and this feature set will give a better result with decision trees.	* limited dataset (11055 phishing and legitimate web pages) * use third-party services * 20 features
Mohammad et al. (2014)	Based on artificial neural network particularly self-structuring neural networks.	* it uses an adaptive strategy in designing the network * language independence	* third-party services are used (such as the age of domain) * limited dataset (1400 data) * 17 features
Buber et al. (2017b)	Uses NLP for creating some features and with the use of these features classifies the URLs by using three different machine learning approach.	* NLP based features * use of 3 different ML algorithms * use of hybrid features * 7% increased performance with the (Buber et al., 2017a) * 278 features (238 word-features and 40 NLP features)	Limited dataset (3717 malicious URLs and 3640 legal URLs)
Jain and Gupta (2018)	Client-side detection of phishing web pages by use of ML techniques	* does not depend on third parties * real-time detection * high detection accuracy	* need to download the whole page (for accessing the source code) * limited dataset * 19 features (URLs/Source Code)
Feng et al. (2018)	Neural network based classification with a simple and stable Monte Carlo algorithm	* does not depend on third parties * real-time detection * improve the accuracy and stability of detection, * can detect new phishing websites (zero-day attack)	* need to download the whole page * use of third-party services * limited dataset (11055 data, 55.69% of them are phishing) * 30 features (address bar-based/ abnormal-based/ HTML and javascript-based/domain-based features)
Smadi et al. (2018)	Phishing email detection system, which combines the neural network approach with reinforcement learning for classification	* detection of phishing emails before end-user saw it. * does not depend on third parties * real-time detection	* limited dataset (9118 data, 50.0% of them are phishing) * use blacklist from PhishTank * 50 features (12 of them are URL based features)
Rao and Pais (2018)	Use principal component analysis random forest classifier by using new heuristic features and image analysis.	* language independence * high detection accuracy * check whether the web page is replaced with an image.	* limited dataset (1407 legitimate and 2119 phishing) * legitimate dataset is constructed only from Alexa's top websites * third-party service-based features * 16 features(hyperlink-based/ third-party based/ URL obfuscation features
Peng et al. (2018)	Detects phishing emails by using NLP techniques and machine learning (with Naïve Bayes classifier).	Use of natural language processing to detect the appropriateness of each sentence	* relies on analysis of the text of emails. * ML is used to construct the blacklist of malicious pairs * limited dataset (5009 phishing emails and 5000 legitimate emails)

components of the address. For instance, a dot mark (".") is used for separating the SLD and TLD. Similarly, the domain names and the subdomain names are also separated by the same character. However, in the path address folders are separated by the "/" sign. Additionally, each component of the URL may also contain some separation marks such as ".", "_" etc. as can be seen in the following example "xyz_company.com". Similar characters can also be used with "=", "?", "&", characters in the file path area. Therefore, in the data preprocessing part, firstly, each word is extracted from the URL and then they are added to the "word_list" to be analyzed in the ongoing execution. Additionally, the similarity or these words with the most targeted websites and random created words are also detected in this module. The execution diagram of the data preprocessing is depicted in Fig. 4.

The main aims of data pre-processing part are as follows:

1. Detecting the words, which are similar to known brand names,
2. Detecting the keywords in the URL,

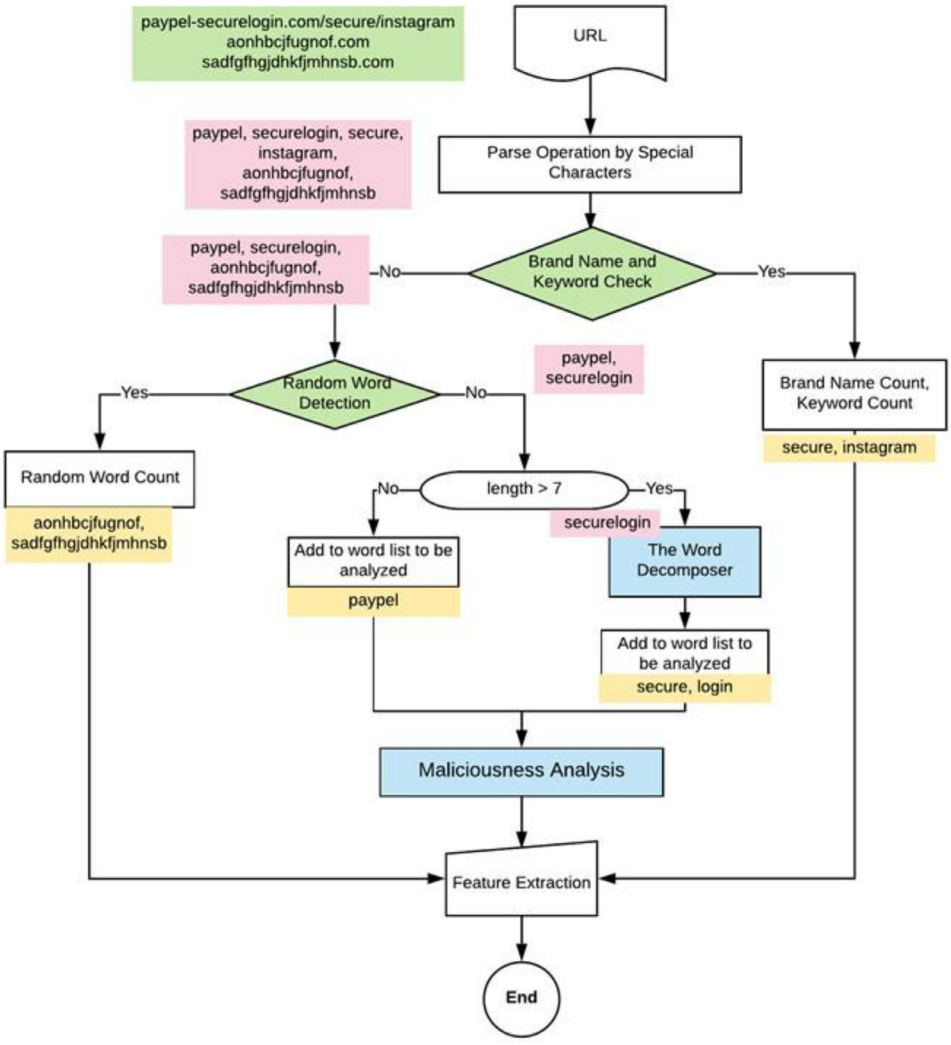

Fig. 4. Execution of the data preprocessing module.

3. Detecting the words, which are created with random characters.

Finding these words are crucial for the classification of the URL. Therefore, a data preprocessing module is very critical in the proposed system. Firstly, the words are compared with the brand names and keywords lists with an acceptable similarity. Then the Random Word Detection Module is executed. This module firstly checks whether a word contains some random characters or not. After that, the detected random words are added to the random_word_list, and they are deleted from the word_list.

When the words used in the address of phishing web pages were collected and analyzed, it was explicitly seen that the there are lots of compound words, which are a combination of more than two words that have single meanings (or without meaning in the address). Therefore, a heuristic based threshold value is determined to detect adjacent words as if they are compound. Words, which contain more than seven characters, are checked in Word Decomposer Module (WDM) to separate the subwords that they contain. If the word is not a compound word, then WDM returns only the original word. The words obtained from the WDM and words with less than seven characters are analyzed in the Maliciousness Analysis Module. Finally, some additional features, which are related to these words have been extracted.

4.3. Word decomposer module

Word Decomposer Module (WDM) analyzes the incoming URL and divide it into the separate words/objects if there is more than one word. Firstly, it removes the digits in the original words. Because an attacker can add some numeric values for making the address more complex. After that, the remaining string is checked whether it exists in the dictionary or not. If it is a dictionary word, then this word is added to word_list, else the word is divided to substrings to reach the adjacent words. After that these words are also added to the word_list. The execution flow of the word decomposer is shown in Fig. 5.

Before starting the extraction process of a word, firstly, it must be checked whether it exists in the dictionary or not.

If it exists in the dictionary, then there is no need to parse it. The word is directly added to the word_list. The decomposer module is implemented to understand the difference between the dictionary words that are written in contiguous form. To check this, we use a publicly available package: Enchant (Pyenchant, 2017). By using this, all possible substrings are extracted from the original word by dividing it into substrings according to consecutive characters, which have the length as greater than 2. A sample of the extraction process is depicted in Fig. 6.

After extracting to substrings, they are sorted according to their lengths. Firstly the ones with the longest length are checked from

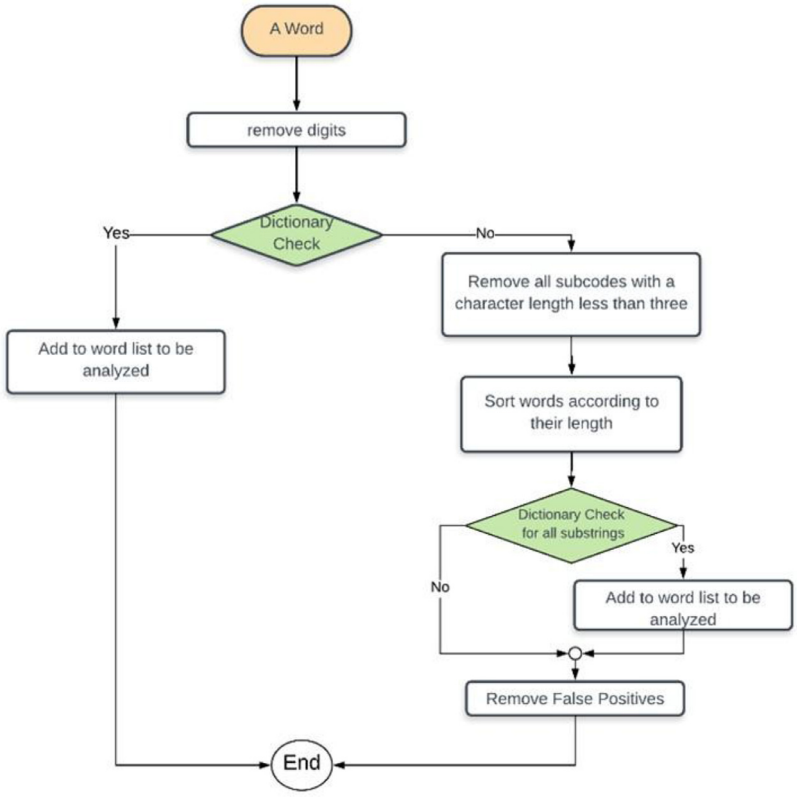

Fig. 5. Execution flow of word decomposer module.

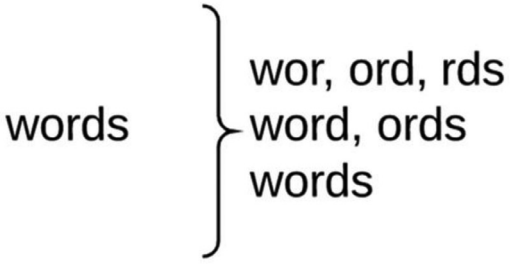

Fig. 6. Extraction of a word to substrings.

the dictionary. If they exist in the dictionary, then they are added to the word_list. If not, then the smaller substrings will be checked in the same order. However, we have to be careful in this process because some false positive words can be encountered while word_list is constructed. For example, “secure” is one of the most preferred words in this type of attacks. In the case of extracting process, we can reach a substring of “cure”, which is a valid word in English dictionary. Therefore, in the similar cases, we need to eliminate some words. We prefer the longer words if there exist, the smaller substring ones to eliminate the false positive words.

In this process/module, brand names and keywords, which have more than seven characters are not processed in the parser, because they are processed and controlled in the previous stage. On the other side, dictionary words, which have a length greater than seven are not controlled in the WDM. Therefore, this type of words has behaved like a single word, and they are not attempted for parsing.

4.4. Random word detection module

In the phishing URLs, it is seen that some words are formed from random characters. Therefore, it is an efficient way to detect the number of the random words (or possible random words) with their lengths. To detect these words, we implemented the Random Word Detection Module (RWDM) by getting help from an open source project in GitHub (Gibberish detector, 2015). In the referenced study, the Markov Chain Model was used for the detection of random words. Firstly, the system is trained with texts written in the controlling language. After that, the probability of two successive consecutive characters is calculated in the training stage. The calculated value will be used as a new feature in the proposed system. In this study, the consecutive characters can only be either alphabetical characters or empty character such as space. Therefore, there is no need to calculate the probabilities of the other characters.

To understand whether a given the word is random or not, consecutive letters in the related word must be examined in the test stage. In this process, firstly the probabilities of the letter pairs are calculated in the training stage. In the test stage, these probabilities are multiplied to reach the fitness value. By looking at this value, it is aimed to understand whether the word is random or not. If the fitness value is a high value then it is classified as a real word; otherwise, it is classified as a random word. To decide this, a threshold value needs to be determined.

4.5. Maliciousness analysis module

To detect whether the words in the given/tested URLs are used for fraudulent purposes or not, we have implemented a Maliciousness Analysis Module (MAM), which mainly focuses on detection of Typosquatting. Typosquatting is also known as URL hijacking, which targets computer users who incorrectly type a website

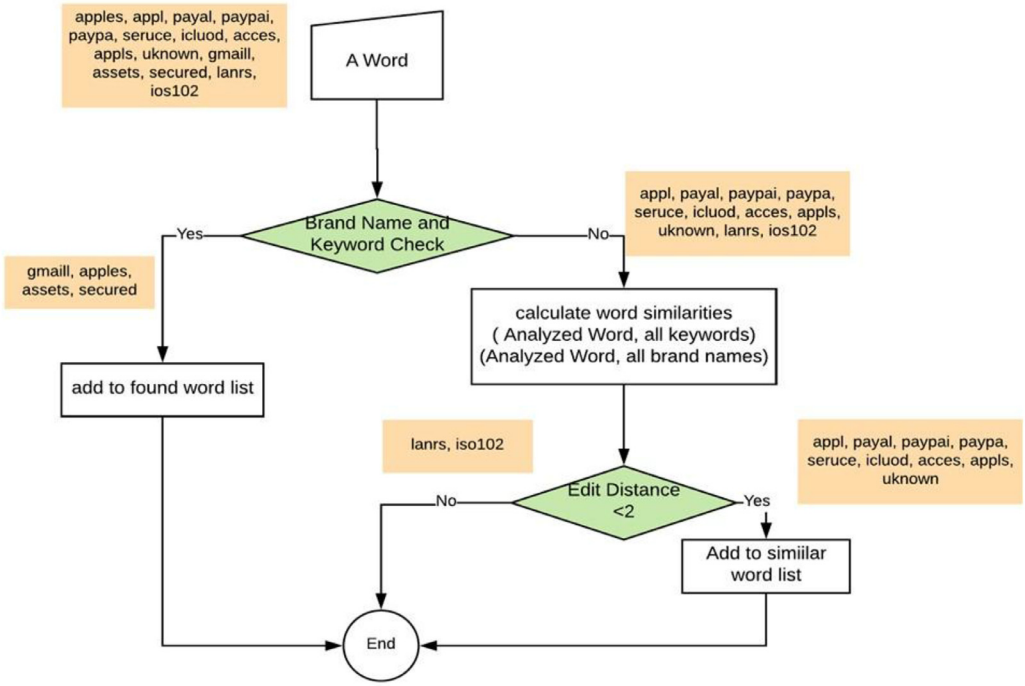

Fig. 7. Execution of maliciousness analysis module.

address or link from an email or web page by a web browser such as "Goggle.com" instead of "Google.com". This module gets a word as input and then analyzes it according to the flow chart depicted in Fig. 7.

In the previous stages, we have gathered some words in WDM, and they may include some brand names or some keywords. Therefore, in this module, we rechecked the words whether they are in the brand name and keyword lists. At the same time, the similarity of each word is also calculated with the use of Levenshtein Distance, which is also named as Edit Distance algorithm. This algorithm measures the similarity between two strings by taking into account the deletions, insertions, or substitutions operations in the strings. This algorithm mainly calculates the number of moves, which is needed to convert a source word to the target one. The calculated value gives the distance between the source and target words, and this value is used as the similarity value.

5. Experimental results

This section gives the experimental details of the proposed model's classification algorithms and used feature extraction types (NLP based features, Word Vectors, and Hybrid) are detailed. Then, the comparative test results between these algorithms with related features are depicted.

5.1. Used classification algorithms

In this paper, we have used seven different classification algorithms (*Naive Bayes, Random Forest, kNN* ($n = 3$)*, Adaboost, K-star, SMO and Decision Tree)* as machine learning mechanism of the proposed system and then compared their performances.

The Naïve Bayes classification is a probabilistic machine learning method, which is not only straightforward but also powerful. Due to its simplicity, efficiency and good performance, it is preferred in lots of application areas such as classification of texts, detection of spam emails/intrusions, etc. It is based on the Bayes theorem, which describes the relationship of conditional probabilities of statistical quantities. It is based on the assumption of independence between the attribute values. For calculating the conditional probability, Eq. (1) is used.

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)} \tag{1}$$

where P(H) is the prior probability, which is the probability of H being true. P(E) is named as the probability of the evidence. P(E|H) is the probability of the evidence given that H is true. P(H|E) is the posterior probability in, which is the probability of the H being true according to given evidence.

Random forests or random decision forests are ensemble machine learning methods, which can be used for regression and/or classification. This type of classifiers firstly constructs a number of decision trees on a randomly selected subset of training sets. After that, they aggregate the decisions of these trees and get the average of them not only for improving the predictive accuracy but also for controlling over-fitting. Additionally, with its forest structure, the instabilities of the individual decision trees can disappear.

Apart from other classification algorithms, the k-nearest neighbor algorithm (k-NN) is a non-parametric algorithm, which can be used in both regression and classification. It classifies unknown objects based on k closest training examples in the feature space. The parameter k plays a very important role in the performance of the kNN for finding the nearest neighbors. Closest neighbors are selected according to some distance functions such as Euclidian, Hamming, Manhattan, Minkowski, etc. Mainly, no learning model is required for kNN classification. Therefore, it is also identified as a lazy learning algorithm, which needs to store the whole training dataset. The consistency of the training data is very important. Although it is a simple classifier, which can be used in many recognition applications, it is seen as a slow classification for real-time prediction especially if the size of the data is huge.

Adaboost classification is a method used to find targets by training different weak classifiers for the same training set to form a strong classifier where multiple models vote on the best prediction. Then the boosting algorithm can be explored. For example,

classification of a person as either male or female can be done according to his/her height. If the person is over 170 cm, then he is a male. Although there are lots of misclassifications, accuracy will be greater than 50%. This classification can be combined with other classifiers with the selection of training set at every iteration and setting the correct weights for final voting. Due to its dependence on many weak classifiers for the final decision, this algorithm, like the random forest classifier, gives more accurate predictions, especially in image recognition systems.

KStar - K* is a simple instance-based classifier whose execution is very similar to k-nearest neighbor algorithm. It was implemented as part of the WEKA project, at the University of Waikato. This algorithm is originally an unsupervised algorithm, which is suitable for clustering purposes. However, in WEKA, it provides a supervised learning approach with an appropriate distance calculation. As a distance metric, KStar uses an entropy-based distance function, however, for classifying the instance, it calculates the average of the closest attributes. The use of entropic distance has many benefits especially handling of missing values and use of real-valued attributes.

The Decision Tree classification is one of the popular supervised learning processes that is used not only for classification but also for regression tasks. This classifier repetitively divides the training dataset into the subparts to identify the separation lines in a tree-like structure. Then these lines are used to detect the appropriate class for the target item. Each decision node splits the data into two or more categories according to a single attribute value. Each leaf node is assigned to a class (especially by calculating a probability) in the classification algorithm.

Support Vector Machine (SVM) is an important classifier in the machine learning concept, which searches non-linear decision boundaries using kernel trick in the trained data. Sequential Minimal Optimization (SMO) is a fast training method for SVMs. Due to its simplicity; SMO is one of the highly preferred algorithms for classification problems. At the same time, it is also used to solve the optimization problems in training time.

For testing the proposed system, we have used different types of features to measure the efficiency. Related tests are grouped into three main classes as NLP based Features, Word Vectors, and Hybrid Features.

5.2. Natural language processing based features

By the use of data preprocessing as detailed in previous sections, it can be easy to extract some distinctive features. These features are extracted by using the Natural Language Processing (NLP) operations. Therefore, these features depend on the used language. For the efficiency of the system, features are extracted according to the English language; however, according to aim it can be easily adapted to any language. Selection and design of these features are very trivial issues to accomplish, and most of the works focus on phishing detection used different feature list according to their algorithms.

The selected features mainly need to parametrize the URL of the web page. Therefore, the text form of web address must be decomposed to the words that it contains. However, this is not an easy task. Because a web address can contain some combined texts in, which finding each word is a trivial task. In this decomposition operation, firstly the URL was parsed by taking into account some special characters such as (“?”, “/”, “.”, “=“, “&”). Then, a raw word list is reached in, which each word can have meaning alone or can be combined with the use of two or more distinct words in a meaningful order. The latter one is especially preferred for the attackers to convince the victim as if it is a legitimate web page. To deceive the users, attackers can use different techniques. According to attack type or targeted site, malicious URLs may contain some publicly known brand names such as *Apple, Google, Gmail, Oracle, etc.* or some important keywords such as *login, secure, account, server, etc.* Therefore, in addition to some features, which are proposed in the previous literature, we defined some additional features for detection of phishing websites. There are totally 40 different NLP based features, which are listed in Table 2. This number is not too much, therefore, there is no need to apply a feature reduction mechanism while using NLP features alone.

5.3. Word vectors

In the text processing or text mining approaches, converting words into vectors is mostly preferred for reaching some crucial features. In our system, we are related to the URL of the web page, which is mainly constructed as a text that contains lots of words in it. Instead of converting these words manually, an automatic vectorization process is preferred. In this module, each URL is converted into word vectors with the help of a specific function of Weka named as “StringtoWordVector”. After obtaining the related vectors, they can be easily used in the selected machine learning algorithm.

In the proposed system, 73,575 URLs are used for the testing. In the vectorization process, 1701 word-features are extracted. Then a feature reduction mechanism is applied to decrease the number of features in the list by using a feature selection algorithm named as “CfsSubsetEval” algorithm, which runs with the best first search method. With this reduction mechanism, the sufficient number of features has dropped from 1701 to 102. Some examples of these reduced words are listed in Table 3.

5.4. Hybrid features

To increase the efficiency of the proposed system we wanted to combine both features (NLP features and word vectors) in a hybrid model. After the implementation of the word vectorization step, we have totally 1701 word-features, and then we joined them with the 40 NLP features and there were 1741 total features before making a hybrid test. Then a similar feature reduction mechanism is executed, and the total number is decreased to 104 features.

5.5. Test results

One of the important problems for testing the proposed system is the use of a worldwide accepted dataset. We cannot reach this the dataset, therefore, produced our own dataset as detailed in Section 4. The dataset is also published in (Ebbu2017 Phishing Dataset, 2017). Due to its huge size and lack of test device capacity, we have performed our test on this dataset, which contains 73,575 URLs. This dataset contains 36,400 legitimate URL and 37,175 phishing URLs.

Experiments are executed on a MacBook Pro device with 2.7 GHz Intel Core i5 processor and 8 GB of 1867 MHz DDR3 RAM. For testing the proposed system Weka was used with some predeveloped libraries. 10-fold Cross Validation and the default parameter values of all algorithms were used during the tests.

Each test set is executed with seven different machine learning algorithms. Firstly, the confusion matrix for the tested learning algorithms is constructed as depicted in Table 4 (Only the best test type (NLP based features, Word Vectors or Hybrid) is listed).

By using the values in confusion matrix, 4 different statistics as precision, sensitivity, f-measure, and accuracy are calculated to measure the usefulness and efficiency of the algorithms. These statistics, whose formulation is depicted in Eqs. (2–5), are also important for making a comparison between the tested machine

Table 2
NLP based features.

Feature	Explanation
Raw Word Count	The number of words obtained after parsing the URL by special characters.
Brand Check for Domain	Is domain of the analyzed URL in the brand name list?
Average Word Length	The average length of the words in the raw word list.
Longest Word Length	The length of the longest word in the raw word list.
Shortest Word Length	The length of the shortest word in the raw word list.
Standard Deviation	Standard deviation of word lengths in the raw word list.
Adjacent Word Count	Number of adjacent words processed in the WDM module
Average Adjacent Word Length	The average length of the detected adjacent words.
Separated Word Count	The number of words obtained as a result of decomposing adjacent words.
Keyword Count	The number of keywords in the URL.
Brand Name Count	The number of the brand name in the URL.
Similar Keyword Count	The number of words in the URL that is similar to a keyword.
Similar Brand Name Count	The number of words in the URL that is similar to a brand name.
Random Word Count	The number of words in the URL, which is created with random characters.
Target Brand Name Count	The number of target brand name in the URL.
Target Keyword Count	The number of target keyword in the URL.
Other Words Count	The number of words that are not in the brand name and keyword lists but are in the English dictionary (e.g. computer, pencil, notebook etc …).
Digit Count (3)	The number of digits in the URL. Calculation of numbers is calculated separately for domain, subdomain and file path.
Subdomain Count	The Number of subdomains in URL
Random Domain	Is the registered domain created with random characters?
Length (3)	Length is calculated separately for the domain, subdomain and path.
Known TLD	[*"com", "org", "net", "de", "edu", "gov", etc.*] are the most widely used TLDs worldwide. Is the registered TLD known one?
www, com (2)	The expression of "www" and "com" in domain or subdomain is a common occurrence for malicious URLs.
Puny Code	Puny Code is a standard that allows the browser to decode certain special characters in the address field. Attackers may use Puny Code to avoid detecting malicious URLs.
Special Character (8)	Within the URL, the components are separated from each other by dots. However, an attacker could create a malicious URL using some special characters {'-', '.', '/', '@', '?', '&', '=', '_'}.
Consecutive Character Repeat	Attackers can make small changes in brand names or keywords to deceive users. These slight changes can be in the form of using the same character more than once.
Alexa Check (2)	Alexa is the name of a service that places frequently used websites in a certain order according to their popularity. Is the domain in Alexa Top one million list?

Table 3
Some word features.

Word features				
Verification	Store	Remove	Customer	Update
Configuration	Recover	Support	Com	Activity
Billing	Online	Provider	Protect	Services
Service	Resolved	Secure	Home	Setup
Center	Summary	Contact	Server	Solution

learning approaches.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3}$$

$$F - Measure = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{4}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{5}$$

where TP means the true positive, TN means true negative, FP means false positive, and FN means the false negative rate of classification algorithms. According to these equations, calculated results of the implemented machine learning algorithms are depicted in Table 5 in a comparative format.

As can be seen from this table Random Forest algorithm with NLP based features has the best classification performance (97.98% accuracy). Additionally, the effect of the features also can be seen from this table.

Table 4
Confusion matrix.

	Confusion matrix		Predicted	
			P	N
Decision Tree	Actual	P	36,328	847
(NLP based)		N	1348	35,052
Adaboost	Actual	P	35,813	1362
(NLP based)		N	3609	32,791
Kstar[1]	Actual	P	3596	121
(Hybrid)		N	227	3413
kNN ($n = 3$)	Actual	P	36,214	961
(Hybrid)		N	2082	34,318
Random Forest	Actual	P	36,806	369
(NLP based)		N	1120	35,280
SMO	Actual	P	36,256	919
(NLP based)		N	2817	33,583
Naïve Bayes	Actual	P	27,663	9512
(Hybrid)		N	1247	35,153

[1]Due to the high computation time, only 10% of dataset is used.

This accuracy rate is interpreted as an acceptable and good result for phishing detection. A 100% accuracy rate cannot be possible. Because while the security managers of the systems try to use some new techniques, the attackers try to enhance their attack techniques to bypass the existing/developed antiphishing systems. At the same time, our approach depends on the URL of the phishing web page. After we examined the undetected phishing web pages, we saw that some of these pages have short domain and subdomains without any paths. If the URL contains only a single domain name such as "www.testbank.com", due to the NLP based features of the proposed solution, these pages cannot be detected mostly. In a standard phishing attack, a web page is designed as

Table 5
Test results of the classification algorithms.

Algorithm	Features	Precision	Sensitivity	F-Measure	Accuracy
Decision Tree	NLP Features	0.964	0.977	0.971	**97.02%**
	Word Vector	0.944	0.695	0.800	82.48%
	Hybrid	0.933	0.973	0.953	95.14%
Adaboost	NLP Features	0.908	0.963	0.935	**93.24%**
	Word Vector	0.936	0.536	0.682	74.74%
	Hybrid	0.915	0.940	0.927	92.53%
Kstar	NLP Features	0.936	0.936	0.936	93.56%
	Word Vector	0.845	0.811	0.806	81.05%
	Hybrid	0.953	0.953	0.953	**95.27%**
kNN ($k=3$)	NLP Features	0.940	0.977	0.958	95.67%
	Word Vector	0.955	0.697	0.806	83.01%
	Hybrid	0.946	0.974	0.960	**95.86%**
Random Forest	NLP Features	0.970	0.990	0.980	**97.98%**
	Word Vector	0.958	0.697	0.807	83.14%
	Hybrid	0.953	0.976	0.964	96.36%
SMO	NLP Features	0.928	0.975	0.951	**94.92%**
	Word Vector	0.947	0.697	0.803	82.71%
	Hybrid	0.923	0.972	0.947	94.48%
Naive Bayes	NLP Features	0.940	0.977	0.958	95.67%
	Word Vector	0.955	0.697	0.806	83.01%
	Hybrid	0.946	0.974	0.960	**95.86%**

Table 6
Performance enhancements.

	Performance Difference Between (%)		
	NLP vs WV	Hybrid vs NLP	Hybrid vs WV
Decision Tree	14.54	−1.88	12.66
Adaboost	18.51	−0.71	17.79
K-Star	15.43	1.83	17.54
kNN ($n=3$)	12.66	0.20	12.86
Random Forest	14.83	−1.61	13.22
SMO	12.21	−0.45	11.76
Naive Bayes	−12.13	18.30	6.16
Average	10.86	2.24	13.14

if a legal web page, therefore, attackers tried to hide with the use of a long URL by using some special words to deceive the users. Because the shorter URLs can be caught by the users, who have an introductory knowledge about phishing attacks.

To see the overall performance enhancements in a comparative way, Table 6 is constructed. Almost in all machine learning algorithms, except Naive Bayes, NLP Features produce better performance than Word Vectors for classification of URLs, with the average rate of 10.86%. Additionally, the use of Hybrid Features also increases the performance of the system about 2.24% according to NLP Features and 13.14% according to Word Vectors.

6. Advantages of the proposed approach

As can be seen from the design of the proposed system, comparison table of the machine learning based phishing detection systems in Table 1 and the experimental results, our model have six main advantages as listed below.

Language independence: In most of the phishing detection system, language is very critical for the execution of the system. However, in the proposed system, we are using only URLs whose texts are constructed with random and long strings, which contain some specific keywords in a vector structure. Therefore, the success of our system depends on this word vector, and this can be constructed in a language independent way.

Huge Size of Phishing and Legitimate Data: Constructing a dataset for the anti-phishing system is a trivial issue. There are some web-based services, which give URLs of the phishing web pages. However, they share a limited amount of data in their web pages. Therefore, we wrote a script to collect all these data with different criteria. This is relatively easy than collecting the legitimate web page addresses. These addresses can be reached from the search engines and some services, which share the most attacked sites. We have obtained these URLs by writing some scripts and construct our dataset, which wholly contains 36,400 legitimate URLs and 37,175 phishing URLs. This dataset is publicly available in Ebbu2017 Phishing Dataset (2017).

Real-time Execution: Since creating a web page is a cheap and easy task, for phishing the users, an attacker can quickly construct a fraudulent web page, which is active in a short lifespan (maybe for a few hours). Therefore, detection of the phishing web page in real-time is essential for the prevention from this type of attacks. In the proposed system, features depend on the address of the web page, and some features are constructed with the help of NLP algorithms. Therefore, it is executed faster and classifies the web page in a negligible amount of time.

Detection of new Websites: Due to NLP based and word vector features, the proposed system can detect new phishing websites, which are not labeled as phishing previously. With this property, our system is robust for the zero-day attack, which is one of the most dangerous attack types in phishing.

Independence from Third-Party Services: In the literature, there are many works, which use third-party services such as who is records, web-based blacklist/whitelist, ranking pages, network traffic measures, the age of domain detection, etc. Mainly, use of these services increases the efficiency of the detection/prevention system. However, if the aim is about the execution in the real-time, then these services increase the detection time; therefore, they cannot be useful.

Use of Feature-Rich Classifiers: In the general form of feature-based machine learning algorithms, the first aim is to find a set of discriminative features, which can help to differentiate the phishing and legitimate web pages. After that, an appropriate machine learning model is executed. Most of the proposed detection/prevention systems use the limited number of features between 8 and 50 as depicted in Table 1. Obtaining a high-quality feature set is very crucial for the effectiveness of the system. There are lots of features in the literature, and some of them are not distinguishing enough. Therefore, we try to increase the number of features that can be used in our system and then eliminate the ones, which are not discriminative enough. We used 40 NLP

based features and 1701 Word Features, and then this number is decreased to the 102 features with a feature reduction mechanism.

7. Conclusion and future works

In this paper, we have implemented a phishing detection system by using seven different machine learning algorithms, as Decision Tree, Adaboost, K-star, kNN ($n = 3$), Random Forest, SMO and Naive Bayes, and different number/types of features as NLP based features, word vectors, and hybrid features. To increase the accuracy of the detection system, construction of an efficient feature list is a crucial task. Therefore, we have grouped our feature list in two different classes as NLP based features, which are mainly human-determined features and word vectors, which focus on the usage of the words in the URL without performing any other operations.

Due to the absence of a worldwide acceptable test set for phishing systems, we needed to construct our own dataset with 73,575 URLs. This set contains 36,400 legitimate URLs and 37,175 phishing URLs. This set is also shared in a web page (Ebbu2017 Phishing Dataset, 2017) to be used for other phishing detection system developers. According to experimental results, it is clearly seen that the NLP based features have better performance than word vectors with the average rate of 10.86%. Additionally, the use of NLP based features and word vectors together also increases the performance of the phishing detection system with the rate of 2.24% according to NLP based features and 13.14% according to word vectors.

Although the obtained results are acceptable in the content of the detection rate, to increase the efficiency of the system; some up to date learning technologies, such as deep learning, can be used. Additionally, we have a relatively huge dataset. With the use of this deep learning technology, all dataset can be used for constructing the knowledge base. However, this needs additional time for training. Therefore, some parallel processing techniques can be adapted to the system.

Additionally, a new subsystem can be conducted for shorter URLs, which contain only a domain/subdomain. To detect this type of web pages, some active data about the web page is needed such as the number of visitors in the last week, when the domain name is received, etc. Checking these features need some extra time, therefore they may not be preferred for real-time detection. However, if a web page is detected as phishing this page can be added to the local blacklist of the network and can be forbidden for the further requests.

Acknowledgment

Thanks to Roksit for their support in the implementation of this work.

References

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual ecrime researchers summit, eCrime '07, ACM, New York, NY, USA* (pp. 60–69).

APWG. Accessed 24 July 2018.. http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf.

Babagoli, M., Aghababa, M. P., & Solouk, V. (2018). Heuristic nonlinear regression strategy for detecting phishing websites. *Soft Computing*, 1–13.

Buber, E., Diri, B., & Sahingoz, O. K. (2017a). Detecting phishing attacks from URL by using NLP techniques. In *2017 International conference on computer science and Engineering (UBMK)* (pp. 337–342). 28.

Buber, E., Diri, B., & Sahingoz, O. K. (2017b). NLP based phishing attack detection from URLs. In A. Abraham, P. K. Muhuri, A. K. Muda, & N. Gandhi (Eds.), *Intelligent systems design and Applications, springer international Publishing, cham* (pp. 608–618).

Cao, Y., Han, W., & Le, Y. (2008). Anti-phishing based on automated individual white-list. In *Proceedings of the 4th ACM workshop on digital identity*.

Cook, D. L., Gurbani, V. K., & Daniluk, M. (2008). Phishwish: A stateless phishing filter using minimal rules. In *Financial cryptography and data security, Berlin, Heidelberg: Springer* (pp. 182–186).

Chiew, K. L., Yong, K. S. C., & Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications, 106*, 1–20.

Curtis, S. R., Rajivan, P., Jones, D. N., & Gonzalez, C. (2018). Phishing attempts among the dark triad: Patterns of attack and vulnerability. *Computers in Human Behavior, 87*, 174–182.

Ebbu2017 Phishing Dataset. Accessed 24 July 2018. https://github.com/ebubekirbbr/pdd/tree/master/input.

Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., & Wang, J. (2018). The application of a novel neural network in the detection of phishing websites. *Journal of Ambient Intelligence and Humanized Computing*.

Fu, A. Y., Wenyin, L., & Deng, X. (2006). Detecting phishing web pages with visual similarity assessment based on earth mover's distance. *IEEE Transactions on Dependable and Secure Computing, 3*(4), 301–311.

Gibberish detector. *A small program to detect gibberish using a markov chain* Accessed 24 July 2018. https://github.com/rrenaud/Gibberish-Detector.

Goel, D., & Jain, A. K. (2018). Mobile phishing attacks and defence mechanisms: State of art and open research challenges. *Computers & Security, 73*, 519–544.

Google Safe Browsing.. *Google Developers* Accessed 24 July 2018. https://developers.google.com/safe-browsing/?csw=1.

Greene, K., Steves, M., & Theofanos, M. (2018). No phishing beyond this point. *Computer, 51*(6), 86–89.

Gupta, B. B., Arachchilage, N. A. G., & Psannis, K. E. (2018). Defending against phishing attacks: Taxonomy of methods, current issues and future directions. *Telecommunication Systems, 67*(2), 247–267.

Islam, R., & Abawajy, J. (2009). A multi-tier phishing detection and filtering approach. Journal of Network Computer. Applications , *36*(1), 324–335.

Jain, A. K., & Gupta, B. B. (2016). A novel approach to protect against phishing attacks at client side using autoupdated white-list. *EURASIP Journal on Information Security*.

Jain, A. K., & Gupta, B. B. (2018). Towards detection of phishing websites on client–side using machine learning based approach. *Telecommunication Systems, 68*(4), 687—700.

Jeeva, S. C., & Rajsingh, E. B. (2016). Intelligent phishing URL detection using association rule mining. *Humancentric Computing and Information Sciences, 6*(1).

Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing detection: A literature survey. *IEEE Communications Surveys Tutorials, 15*(4), 2091–2121.

Le, A., Markopoulou, A., & Faloutsos, M. (2011). Phishdef: URL names say it all. In *2011 Proceedings IEEE INFOCOM, 2011* (pp. 191–195).

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications, 25*(2), 443–458.

Openfish (2018). *Phishing dataset* Accessed 24 July 2018, available at: https://www.openphish.com/.

Peng, T., Harris, I., & Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. In *IEEE 12th international conference on semantic computing (ICSC)* (pp. 300–301).

PhishTank. *Verified phishing URL* Accessed 24 July 2018. https://www.phishtank.com/.

Prakash, P., Kumar, M., Kompella, R. R., & Gupta, M. (2010). Phishnet: Predictive blacklisting to detect phishing attacks. In *2010 Proceedings IEEE INFOCOM* (pp. 1–5).

Pyenchant.. *Pyenchant v1.6.6 documentation* Accessed 24 July 2018. https://github.com/rfk/pyenchant.

Qabajeh, I., Thabtah, F., & Chiclana, F. (2018). A recent review of conventional vs. automated cybersecurity anti-phishing techniques. *Computer Science Review, 29*, 44–55.

Rao, R. S. &, & Pais, A. R. (2018). Detection of phishing websites using an efficient feature-based machine learning framework. *Neural Computing and Applications*.

Shaikh, A. N., Shabut, A. M., & Hossain, M. A. (2016). A literature review on phishing crime, prevention review and investigation of gaps. In *10th international conference on software, knowledge, information management & applications (SKIMA)* (pp. 9–15). *2016*.

Sharifi, M., & Siadati, S. H. (2008). A phishing sites blacklist generator. In *2008 IEEE/ACS international conference on computer systems and applications* (pp. 840–843).

Sheng, S., Wardman, B., Warner, G., Cranor, L. F., Hong, J., & Zhang, C. (2009). An empirical analysis of phishing blacklists. In *Proceedings of the 6th conference in email and anti-spam, CEAS09*.

Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L. F., & Downs, J. (2010). Who falls for phish?: A demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI conference on human factors in computing systems, CHI '10, ACM, New York, NY, USA* (pp. 373–382).

Smadi, S., Aslam, N., & Zhang, L. (2018). Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decision Support Systems, 107*, 88–102.

Stone, A. (2007). Natural-language processing for intrusion detection. *Computer, 40*(12), 103–105.

Tan, C. L., Chiew, K. L., Wong, K., & Sze, S. N. (2016). Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems, 88*, 18–27.

Toolan, F., & Carthy, J. (2009). Phishing detection using classifier ensembles. In *2009 eCrime researchers summit* (pp. 1–9).

Volkamer, M., Renaud, K., Reinheimer, B., & Kunz (2017). A. User experiences of torpedo: Tooltip-powered phishing email detection. *Computers and Security, 71*, 100–113.

Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011). Cantina+: A feature-rich machine learning framework for detecting phishing web sites. ACM Transactions on Information and System Security, *14*(2), 1–28 20.

Yandex XML Yandex Technologies (2013). Accessed 24 July 2018, https://tech.yandex.com.tr/xml/.

Zhang, Y., Hong, J. I., & Cranor, L. F. (2007). Cantina: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07, ACM, New York, NY, USA* (pp. 639–648).