On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifiers*

Rakesh Verma, Keith Dyer
Dept. of Computer Science, University of Houston
Houston, Texas, USA
rmverma@cs.uh.edu, kwdyer@uh.edu

ABSTRACT

Phishing attacks resulted in an estimated $3.2 billion dollars worth of stolen property in 2007, and the success rate for phishing attacks is increasing each year [17]. Phishing attacks are becoming harder to detect and more elusive by using short time windows to launch attacks. In order to combat the increasing effectiveness of phishing attacks, we propose that combining statistical analysis of website URLs with machine learning techniques will give a more accurate classification of phishing URLs. Using a two-sample Kolmogorov-Smirnov test along with other features we were able to accurately classify 99.3% of our dataset, with a false positive rate of less than 0.4%. Thus, accuracy of phishing URL classification can be greatly increased through the use of these statistical measures.

Keywords

Phishing URL Classification, Kolmogorov-Smirnov Distance, character distributions, Kullback-Leibler Divergence

1. INTRODUCTION

Phishing is the act of acquiring sensitive information by pretending to be a legitimate entity, through the use of electronic communications. Phishing attacks in 2006 cost victims up to $2.8 billion dollars worth of damages and affected nearly 2.3 million people, and unfortunately the incidence rate appears to be increasing, with 3.6 million people losing up to $3.2 billion dollars in 2007 [17]. Furthermore, the number of potential phishing targets is increasing as technology costs decrease allowing more areas to become fully integrated with online business. There are many methods that phishers use to distribute their attack, but all classical attacks [22] share one feature, they require a URL to direct their targets to the phishing website. Thus, we focus our efforts on URL classification, as a URL classifier will be useful across any classical method of phishing attack. Another reason for focusing on URL classification is to ensure more distance between the user and the phisher as visiting the site for visual cues, as some defensive methods entail, can install malware on the user's computer.

Traditional methods to combat phishing include blacklists, or lists of websites that once identified as phishing will be blocked by DNS servers and browsers. This method can be effective, but, more recently, phishers have been hosting their attacks and luring victims within short time frames while rapidly changing domains. This ensures that their damage is done by the time their attack gets blacklisted. A potentially more robust approach to combat phishing attacks is to identify automatically which websites are phishing or not before users access it. Some research has been done using machine learning techniques to develop such a URL classification model [16], but we believe our approach is novel in its use of statistical measures such as the Kolmogorov-Smirnov test and our testing using four different real data sets is much more comprehensive. In particular, researchers in [29] report a Logistic Regression algorithm to achieve an accurate model, but their approach requires *manual* extraction of the features, whereas our statistical measures perform at the same level of accuracy with completely automatic feature extraction.

1.1 Our Contributions and Results

Our contributions include the construction of multiple machine learning classifiers comprising a short list of features, and their evaluations on *four* unique real data sets. We present the results for each feature individually, as well as our most accurate multi-feature classifier. The results were obtained by analysis on four real data sets, the first was comprised of URLs from PhishTank (reported by humans) and Alexa organization. The second and third were provided to us by authors of [29], with the second consisting of phishing URLs from APWG (live feed for members), the third consisting of phishing URLs from Huawei Digital's phishing repositories and the fourth is the DMOZ Open Directory Project data set. The second and third data sets were combined with legit URLs collected by [29]. Our contributions therefore not only indicate the value of a feature, but also the robustness of a feature, due to the real data sets being collected at different times, and through completely unique organizations (APWG, Huawei digital, Phishtank, and DMOZ). In contrast to previous work such as [16], we do not use any host-based features. In contrast to previous works such as [4] and [25], who use more than 300,000 and millions of features respectively, we show that robust and

*Research supported in part by NSF grants CNS 1319212 and DUE 1241772

CODASPY'15 March 02 - 04, 2015, San Antonio, TX, USA

ACM 978-1-4503-3191-3/15/03 $15.00
http://dx.doi.org/10.1145/2699026.2699115.

effective classification is possible with no more than a few dozen features.

In addition to presenting the performance of each feature on multiple data sets, we present each feature evaluated using a different machine learning algorithm. We used the WEKA interface [12] to easily develop and compare multiple different machine learning algorithms for our feature sets, and we present the results for each algorithm, with each feature in isolation, and in best combinations. For all work in this paper we used WEKA stable version 3.6. This work can easily be recreated using WEKA or one of the many other popular machine learning packages.

The outline for the rest of the paper follows: Section 2 presents our hypothesis and gives a brief overview of our results. Section 3 presents our classifiers, the datasets they were trained on, the algorithms used, and covers our feature extraction process. Section 4 presents our analysis and results for our classifiers. Section 5 covers related research and comparisons. Section 6 concludes with implications of the results.

2. HYPOTHESIS

Many phishing website URLs can be identified by a trained person relatively easily, and we want to emulate this behavior automatically. Specifically, our primary question in this paper is:

1. Can statistical techniques applied to a URL be used to accurately differentiate between legitimate and phishing websites?

Specifically, we focus on the character distribution of URLs and a few other features, selected with the goal of robustness, in order to evaluate this question. To best answer this question in depth and breadth we present our results on multiple features and with multiple learning algorithms. Our results indicate that an accurate classification is possible using these methods. Additionally, we find that the use of statistical methods, together with the features selected, improves the robustness of the classifier.

3. CLASSIFIERS

3.1 Features

This section discusses our feature extraction algorithms. Prior to feature extraction "http://" and "https://" were removed from the URL, so as not to give extraneous information to the features such as number of punctuation, and to prevent URLs classified without such parts from being misclassified.

In contrast to previous work, e.g., [4, 25], we use a short list (few dozens) of URL-only features to study the effectiveness of each feature carefully and to investigate whether it is possible to design a robust and effective URL classifier with just a few features. Because we wish to design an *accurate and robust* automatic classifier, we have carefully selected a combination of new features supplemented with others that complement them or that test the claims made by previous researchers/antiphishing organizations. In the process some features will be similar to those used in previous work, since there are only so many dimensions to a URL, which is basically a string of symbols. Hence, at the outset, we wish to clarify and emphasize here the several novel aspects of our work. The first novel aspect of our work is our choice of aspects to focus on, specifically we focus on character distributions. The second, the way we extract the relevant information and construct the actual feature from the aspect of the URL is novel. The third is our use of efficient algorithms, e.g. Aho-Corasick pattern matching algorithm, for fast construction of features. Finally, our testing and validation is much more comprehensive than previous work wherein we use several machine learning algorithms and four different real data sets. Moreover, we show the generalization capability of our classifiers by training them on one data set and testing them on totally different data sets, which to our knowledge no previous work on phishing has done. This is important because it reveals how prone our models are to overfitting, and gives an idea of what kind of performance the system would have in a real scenario where phishing methodologies change rapidly.

Kolmogorov-Smirnov Distance: In [18] the authors point out that character frequencies might be a good indicator for phishing detection, but they do not explain how it might be used nor did they build a classifier. We explore several different ways in which this distribution can be converted to a feature. First, we used the two sample Kolmogorov-Smirnov test to calculate how similar the distribution of English characters are in standard English to the distributions in legitimate and in phishing URLs, we then use this metric as a feature for our machine learning classifiers.

The Kolmogorov-Smirnov distance is calculated by calculating the two-sample Kolmogorov-Smirnov (K-S) test statistic for the normalized frequency distribution of English characters in the URL and comparing that to the normalized frequency distribution of English characters in standard English text. The accepted distribution for standard English text is readily available online [19]. In order to do the calculation, we first count the number of each character in the URL, next we normalize it to construct the normalized frequency distribution. Next we construct a cumulative distribution function (CDF) for the distribution of frequencies, and then we can compare the URL CDF and the standard English CDF using the K-S test. Through experiments, we found that we got better results by comparing the standard English CDF for only the characters that were present in the candidate URL's CDF. This preserved character importance.

Kullback-Leibler Divergence: As a second metric for comparing the distribution of character frequencies in standard URLs and phishing URLs we implemented a Kullback-Leibler (KL) Divergence test [14]. We first construct the two distributions we are comparing, the distribution of normalized character frequencies for the URL, and the distribution of normalized character frequencies in Standard English. Next we compute the following distance using this formula:

$$D = \sum_{i \in Q} ln\left(\frac{P(i)}{Q(i)}\right) P(i)$$

Here D is the KL-Divergence value, P represents the distribution of characters in standard English, and Q represents the distribution of characters in the URL. Characters not in the URL are absent from set Q and therefore they contribute no value to the divergence. The final KL-Divergence value was included as a feature in our classifier.

Euclidean Distance: The Euclidean Distance is a third metric for comparing character frequencies in URLs. For Euclidean Distance we calculated the sum of the squares of the difference in normalized character frequency in standard English and in the URL. Here, we compared all English characters, regardless of whether or not they were in the URL.

Character Frequencies: We used normalized character frequencies as a collection of features. For each character, we counted the number of times that character occurred in the URL, next we divided this number by the total length of the URL, finally we used this normalized frequency as a feature in our classifier. This greatly increased the size of the feature vector, but also led to a much higher accuracy.

The remaining features are carefully devised with the goal of increasing robustness and catch the phisher who tries to adapt to our character distribution approaches or uses techniques to avoid detection from the unwary user.

Edit Distance: Edit distance is defined as the number of substitutions, insertions and deletions required to change one string of characters into another. We pulled a set of 1000 random URLs from the DMOZ, specifically different URLs than were used in our DMOZ data set. Then when training we calculate the edit distance between each URL and our set of 1000 DMOZ legitimate URLs. We then take the minimal value from those 1000 distances, and record it as a integer based feature in our classifier.

The logic behind this was that our classifier's strongest classification features are based on the character frequencies of the phishing URLs vs standard english. We suppose that if a phisher has read this paper, and attempts to construct URLs that have a character distribution that is very much like legitimate URLs they might get through our classifier. The edit distance we believe will enhance the robustness of our classifier by finding URLs that are close to standard, but off by a small amount. For example, www.paypal.com and www.paypai.com. The paypai.com URL may not be caught by our character frequency methods, but the edit distance of 1 will make it look suspicious, and might catch it, thus increasing the robustness of our methods. Specifically, we calculate the edit distance for the URL as a whole to a set of whole legitimate URLs, and calculate the edit distance of the domain to a set of legitimate domains.

From our experiments we found that there was very little improvement in going from 100 URLs to 1000 URLs in our legitimate set, but the time to classify became much slower as the legitimate set grew large. Because of this reasoning we used a set of 100 URLs for all our edit distance tests instead of the initial 1000. This implies that phishing URLs that contain a target are typically aimed at a small set of legitimate companies. The literature has evidence to support this claim [18].

Length: In [18] the authors observed that the length of the domain is longer on average for legitimate URLs when compared to phishing URLs, while the length of the total URL was longer for phishing URLs than legitimate URLs. Other works such as [16, 4, 11] have used length as a feature, but we decided that the best way to capture the observation was to use the ratio of the length of the total URL over the length of the domain. For this, phishing URLs with a long URL length, divided by a small domain would yield a high ratio. Legitimate URLs with a shorter total length, and longer domain would then yield a smaller ratio.

@ and - Symbols: Another way that phishers might attempt to emulate a valid URL is through special characters such as the @ and - symbols. The @ symbol is special because everything to the left of the @ is ignored by the browser. Phishers could potentially exploit this by hosting a URL such as www.paypal.com@phishing.com. Many people might immediately recognize paypal.com, and determine it to be a legitimate website, whereas the browser really goes directly to phishing.com. This feature is simply the sum of the occurrences of @ and - symbols in the URL.

Number of Punctuation Symbols: We counted the number of punctuation symbols in the URL and used this value as a feature in our classifier. We found that phishing URLs tend to have more punctuation symbols than legitimate URLs. In contrast to the work in [30, 11], we include more types of punctuation in our summation: { . ! # $ % & * , ; : '}

Number of Top-Level Domains (TLDs) in URL: We found that many phishing URLs would attempt to imitate a legitimate website by placing TLDs in uncommon locations in the URL in order to trick the victim into believing the URL redirected to a website they were familiar with. For example, www.xyz.com/YourBank.com.php. Specifically, we counted the number of TLDs that were in the path of the URL, and not included in the domain. We found URLs that contained TLDs in the path were very likely to be phishing.

Number of Target Words: Since APWG reports claim that 40-50% of phishing attacks are based on common legal web sites, we decided to check this and so we compiled a list of target words which included many popular phishing targets, such as Ebay and paypal [1]. We then removed the domain from the URL in question, and then searched for any occurrences of our target words. For speed and efficiency, we build an Aho-Corasick automaton to very quickly find all occurrences of targets in the remaining URL. The Aho-Corasick automaton allowed us to create a very large target list, without significantly increasing the time required to classify a URL. Finally we sum the number of targets found, and use this value as a feature for the classifiers.

IP Address: We implemented a simple algorithm that determined if the domain consisted of only numbers and punctuation, we used the value 1 for true, and 0 for false, and included it as a feature in our classifier. IP Address is a popular feature in the literature, being used by [16, 7, 4, 11, 13]. From the analysis of our data sets we find that phishing URLs encountered in the real world are many times more likely to be IP only URLs than legitimate websites. This provides the justification for us to use this feature.

Number of Suspicious Words: Examination of a small dataset of phishing URLs led to this feature, where we constructed a small list of action words, such as "login" and "account." We used many terms employed before, as other authors have already demonstrated their effectiveness. Our list included only nine words, so we did not implement any special data structures to increase speed, but if necessary the Aho-Corasick would suit this purpose if the list needed to be expanded. We summed the number of suspicious words found in each URL, and used this number as a feature. We searched for the following list of words in our implementation: {"confirm","account","secure","ebayisapi","webscr", "login","signin","submit","update"}.

3.2 Machine Learning Algorithms

We used the WEKA library to construct our models. Using the WEKA library allowed us freedom to quickly compare and contrast the different machine learning algorithms and their results without having to implement the feature extraction algorithms more than once. We used multiple different algorithmic types in our approach to ensure a large scope was covered. We examined the following tree based algorithms RandomForest and J48. We examined a rule based learner called PART which combines C4 trees and RIPPER learning. We looked at functional algorithms such as SMO, and Logistic Regression. Finally, we looked at NaiveBayes as a baseline for comparison.

PART: The PART algorithm constructs a decision list for classifying the URL. It was originally proposed in [8]. The algorithm uses separate-and-conquer methods. In each iteration it constructs a partial C4.5 decision tree and makes the "best" leaf into a rule. We used WEKA's default parameters for the classifier.

SMO: This is an implementation of Platt's sequential minimal optimization algorithm for training a support vector classifier [20]. The default SVM kernel for WEKA's implementation of Platt's SMO is the polynomial kernel:

$$K(x, y) =< x, y >^p$$

We selected the p value in the above formula by running a series of small experiments testing p values from 1 to 5. We concluded that a p value of 1 had the best combination of a low false positive rate and high accuracy rate of the set. For this experiment we constructed two data sets, set I - consisting of 1000 random phishing and legitimate URLs from our primary data set, and set II - containing 1000 random phishing and legitimate URLs from our DMOZ data set. We used the combination of all of our features for the following table.

P	Accuracy I	Accuracy II	FP-Rate I	FP-Rate II
1	99.20%	95.95%	0.1%	2.9%
2	99.25%	95.99%	0.6%	3.2%
3	99.25%	95.39%	0.6%	4.4%
4	99.25%	95.29%	0.7%	4.7%
5	97.45%	95.14%	0.8%	4.7%

Table 1: Accuracy and False-Positive Rate for small sets I and II.

All further references to the SMO algorithm correspond to p value of 1.

J48: Algorithm J48 generates a C4.5 decision tree for classification. The algorithm is discussed in [21]. We used the WEKA 3.6 default pruning threshold.

Logistic Regression: The logistic regression implementation in WEKA is a standard multinomial logistic regression model, with the addition of a ridge estimator. The original method is presented in [15]. There are a few modifications to leCessie's algorithm in the WEKA implementation, such as the addition of weighted instances [27].

NaiveBayes: These classifiers have been used quite successfully in many real-world problems, but have been shown to be less effective than more current approaches such as boosted trees [28]. The threshold values are calculated by default in the WEKA implementation, and we used these defaults for our experiments.

RandomForest: The random forest algorithm constructs a multitude of decision trees, and then during classification the algorithm returns the most often occurred class from the array of individual trees. The trees are all constructed using a random subset of features from the entire feature list. The method has been proven quite effective at classifying [28].

For values we used WEKA's default implementation of the RandomForest class. This involves ten unlimited depth trees, with a feature selection of $log\ M + 1$, where M is the total number of features.

3.3 Datasets

For our experiments we used four different real data sets. Data Set I consists of 24,545 URLs comprised of 11,271 phishing URLs from PhishTank.com, accessed on February 12, 2014. We then combined that with 13,274 legitimate URL's taken from Alexa.com on February 11, 2014. We acquired our second and third data sets thanks to authors of [29]. The second set consists of 14,920 phishing URLs provided from Huawei Digital's own phishing repository. This was paired with 14,999 of the original authors' legitimate URLs. The third data set consists of 18,395 phishing URLs acquired from APWG combined with 19,999 less popular legitimate websites gathered using the original authors' crawler. Finally, we extracted 11,275 random legitimate URLs from the DMOZ Open Directory Project [6], and paired them with the 11,271 phishing URLs we gathered from PhishTank. This is our fourth data set.

To summarize, our four data sets used legitimate URLs from Alexa.com, [29], and DMOZ. We acquired phishing URLs from PhishTank, Huawei's phishing repository, and the APWG. To demonstrate the diversity of our data we analyzed the breakdown of unique TLDs as well as unique domains in the four sets (Table 2). The main conclusion is that our datasets are diverse with one exception, the DMOZ set contains a less diverse selection of unique domains; the top 50 most popular domain names account for roughly 10% of the data for the other three groups, whereas the DMOZ-PhishTank group's top 50 domains account for nearly 25% of the URLs in the dataset.

4. ANALYSIS AND RESULTS

In this section, we present the results obtained using each of the classifiers, and each of the features. We begin by presenting each feature's results in isolation, and then expand to include our best classifiers using combinations of features. We present the comparison of features into two sections, character frequency based features, and other standard features. To save space, we omit results on different datasets that are similar (Tables 3-6 report results for Data set 1 and Tables in Sections 4.1-4.3 are for Data Sets 1 and 2 for this reason).

Character Frequency based Methods

First, we present the results for classifying using our different algorithms, based on a single feature in isolation. These values indicate the percent of accuracy achieved with classification following a 5-fold cross-validation test. We decided to use five fold cross validation as a way to evaluate the effectiveness of our models. Cross validation allows us to see how well our model generalizes to unseen data from the same domain. We originally decided on ten fold cross validation but eventually migrated our tests to five folds to reduce

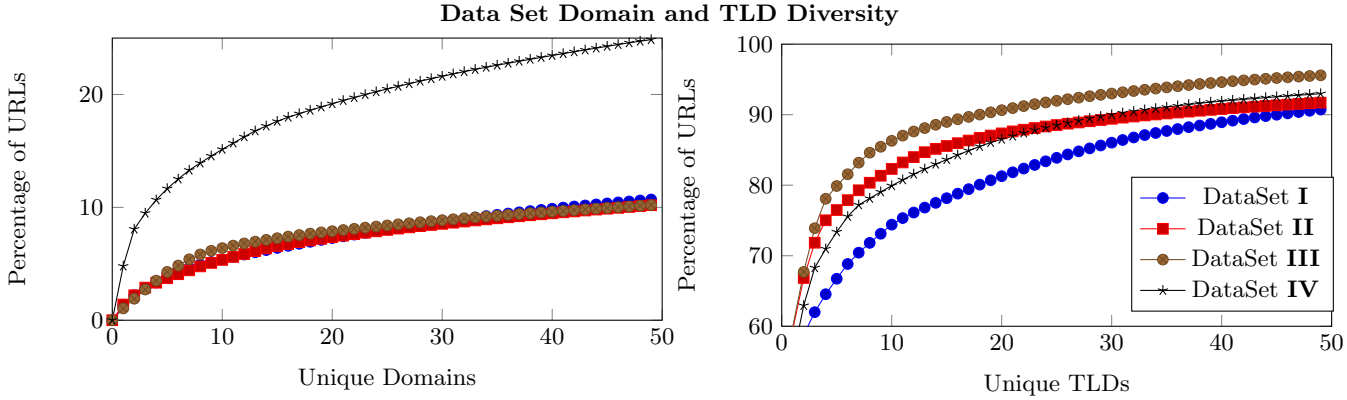

Table 2: Diversity of Data Sets I, II, III and IV. The charts represent how much of the data is contained in the top 50 domains and TLDs.

training time from ten trainings per test to five. Here we present the character frequency features, it is significant to note that KS-Distance, KL-Divergence, and Euclidean Distance are all single features, meaning the feature vector has a length of 1. The character frequencies feature result in a feature vector consisting of an individual feature for every English character found in the URL, resulting in a maximum size of 26.

Accuracy for character frequency features I

Features	Logistic	PART	SMO	J48
Euclidean	65.16%	68.08%	64.98%	68.42%
KS-Distance	92.51%	92.54%	92.49%	92.53%
KL-Divergence	94.78%	94.76%	94.78%	94.76%
Char. Frequencies	79.70%	95.65%	80.14%	95.01%
Edit Distance	87.54%	91.11%	75.88%	91.24%

Accuracy for character frequency features II

Features	RandomForest	NaiveBayes
Euclidean	76.49%	64.53%
KS-Distance	92.56%	92.48%
KL-Divergence	93.35%	94.79%
Char. Frequencies	96.30%	90.34%
Edit Distance	91.08%	87.44%

Table 3: Accuracies on Data Set I

The Euclidean distance, performs poorly across all algorithms, when compared to the other measures. The KS-Distance and the KL-Divergence are close, but the latter appears to give a slightly more accurate classification. The pure character frequencies perform well for tree and rule based learners, (PART, RandomForest, and J48) but underperformed on Logistic Regression, SMO, and NaiveBayes. From this data we can see that the statistical measures for KS-Distance and KL-Divergence provide similar classification performance as constructing a lengthy feature vector based on every unique character in the URL. Additionally, in certain learning algorithms the single feature statistical methods are better able to classify than constructing the lengthy feature vector for each character.

We define a positive occurrence to be a phishing URL. In the context of our tests then a false positive is a benign URL that was falsely classified as phishing. We chose to include false positive rate in our results because it has specific

False-Positives for character frequency features I

Features	Logistic	PART	SMO	J48
Euclidean	30.10%	21.03%	31.01%	19.31%
KS-Distance	4.37%	4.36%	4.37%	4.38%
KL-Divergence	3.18%	2.39%	2.64%	2.39%
Char. Frequencies	5.09%	3.79%	3.90%	3.49%
Edit Distance	1.13%	2.82%	1.00%	3.49%

False-Positives for character frequency features II

Features	RandomForest	NaiveBayes
Euclidean	23.79%	33.47%
KS-Distance	4.33%	4.37%
KL-Divergence	5.71%	3.22%
Char. Frequencies	1.60%	2.12%
Edit Distance	5.55%	1.16%

Table 4: False positive rates on Data Set I.

interest to the problem of phishing detection. When any software program is attempting to detect phishing attacks it is important that they have a very low false positive rate, because each false positive occurrence in a real system might mean an important message never reaches its recipient.

For character frequency based methods, the Euclidean distance feature not only has a low accuracy, but a very high false positive rate. The rule and tree based methods appeared to do a better job of keeping the false positive rate low, with PART, J48, and RandomForest all having an 8-12% lower false positive rate for the feature. The KL-Divergence algorithm had the lowest false-positive rate for four of the six experiments, and the highest accuracy for three of the six experiments. For the cases that KL-Divergence was not the best, only once was it not the second best either. This implies that on average the KL-Divergence algorithm is better than any of the three other methods for use as a feature in our classifier. In addition to these results, the KL-Divergence method has the advantage of a low dimensionality feature vector, resulting in faster training times for the chosen machine learning algorithm.

Other Methods

Here we present the features we used that were unrelated to character frequencies.

From the results in the following tables we conclude that

Accuracies for other features I

Features	Logistic	PART	SMO	J48
Length	71.11%	71.35%	71.49%	71.49%
@ and - symbols	68.11%	68.11%	68.11%	68.11%
Punctuation	84.66%	84.66%	66.65%	84.66%
TLD	83.50%	83.98%	83.50%	83.98%
Target Words	84.08%	84.08%	80.11%	84.08%
IP Address	55.60%	55.60%	55.60%	55.60%
Suspicious Words	64.81%	64.81%	64.81%	64.81%

Accuracies for other features II

Features	RandomForest	NaiveBayes
Length	71.39%	67.40%
@ and - symbols	68.11%	68.11%
Punctuation	84.66%	76.44%
TLD	83.98%	83.50%
Target Words	84.08%	80.91%
IP Address	55.60%	55.60%
Suspicious Words	64.81%	64.81%

Table 5: Accuracies for other methods. Data Set I.

punctuation, presence of target words, and number of TLDs following the domain of the URL were good features for phishing classification, with all achieving over 80% accuracy across multiple algorithms. Additionally, our results indicate that length is a decent feature, performing next best. The remaining in order of average accuracy were special symbols, suspicious words, and finally IP addresses.

Many of these features may not classify a large portion of the data set, for example the IP address feature accurately classifies only 55.6%, barely more than half, but it captures URLs that would not be correctly classified by the other features. Thus many of the low accuracy features add significant value to the final multi-feature models.

Additionally from this we conclude that there is little variability between learning algorithms for these specific features. There are two notable exceptions, firstly NaiveBayes appears to perform slightly worse for length and punctuation, and secondly, punctuation performs significantly worse for SMO, at 66.65% accuracy compared to the 84.66% that was achieved by all the other algorithms but Naive Bayes.

False-Positive rates for other features I

Features	Logistic	PART	SMO	J48
Length	17.83%	14.60%	12.67%	12.67%
@ and - symbols	4.14%	4.14%	4.14%	4.14%
Punctuation	9.50%	9.51%	0.08%	9.51%
TLD	21.45%	22.83%	21.45%	22.83%
Target Words	5.96%	5.96%	2.13%	5.96%
IP Address	0.05%	0.05%	0.05%	0.05%
Suspicious Words	0.19%	0.19%	0.19%	0.19%

Our test results indicate that IP Address, and suspicious words have a very low false positive rate. While the IP Address feature and the suspicious word feature may not classify a large number of URLs as phishing, the ones that do get classified are almost always a phishing URL. This is very good at increasing the total accuracy of the multi-feature classifier without substantially increasing the false-positive rate.

Following IP and suspicious words, we found target words and special symbols to be the next best in regards to a low false-positive rate. Target words was one of the highest accuracy features, and one of the lowest false-positive rate features in our experiments. This demonstrates that using target words to help catch phishers is especially effective.

False-Positive rates for other features II

Features	RandomForest	NaiveBayes
Length	14.67%	4.14%
@ and - symbols	4.14%	4.14%
Punctuation	9.50%	2.13%
TLD	22.83%	21.45%
Target Words	5.96%	18.06%
IP Address	0.05%	0.05%
Suspicious Words	0.19%	0.19%

Table 6: False-positive rates for Data Set I.

The last three features in order of increasing false-positive rate are punctuation, length, and number of TLDs after the domain. It appears logical that these features would have a higher false-positive rate than the preceding ones. For example, the number of legitimate URLs that contains suspicious words such as "Account" or "Login" would be very few, compared to the number of legitimate URLs that might have an abnormally high number of punctuation marks, or an odd domain to url length ratio.

The TLD feature had the highest false-positive rate at over 20%. We considered removing it because of this, but when we tested it in our final multi-feature models we found that the presence of the TLD feature increased accuracy without having any significant impact on the classifiers false-positive rate.

The same trends above are present here as well. Specifically, the punctuation values for SMO are considerably different than the other algorithms. While the punctuation classification is 20% less accurate from SMO, it has an extremely low false-positive rate comparatively.

Aho-Corasick

We utilized the Aho-Corasick automaton for finding occurrences of the targets in our URLs. The Aho-Corasick data structure is a fast tree based search structure that allows searching for any of a large set of pattern strings in a large amount of text quickly. The Aho-Corasick algorithm was developed in 1975 at Bell Labs and first presented in [2]. Adopting the Aho-Corasick algorithm allowed us to expand the number of targets we were able to search for without negatively impacting classification time, or training time. We ran some experiments to demonstrate the effectiveness of using this data structure in the URL classification problem. For our targets we took the 100,000 topmost popular websites from Alexa.com, and then extracted target names from that list.

In the following table, N is the number of targets we looked for in our URLs. The columns represent how long it took to construct our feature vector for targets on Data Set I, a set of 24,545 URLs. Test A is using the Aho-Corasick, and Test B is using Java's built in function String.contains(String), which returns true or false. We indexed the numbers for readability with 100 corresponding to 2.32 seconds.

From our tests we find that by implementing the Aho-Corasick data structure for finding targets within the URLs we can look for up to 2,000 targets before we begin to see any effect on the performance. This allows us to cover nearly all possible targets without negatively impacting the perfor-

Feature Construction Time

N	Test A	Test B
20	100	100
200	100	107.32
2,000	100	202.59
20,000	123.71	782.33
100,000	166.38	4021.55

Table 7: Time to construct features with and without Aho-Corasick.

mance of our classifier in the training stage, or in real-time use of the classifier for detection of phishing URLs. Additionally, we found that increasing the target count above 2,000 had no statistical change in the accuracy or false-positive rate of the classifier.

4.1 Multi-Feature Models

In this section we combine the features into a single multi-feature classification model, and present the results for running a 5-fold cross-validation test on each of our data sets. For this data we used no feature selection, meaning we include every feature in the model. There has been some research to indicate that machine learning performance can be further improved through feature selection [3].

Features	Accuracy	False-Positive Rate
PART	98.98%	0.889%
Logistic	97.70%	2.682%
J48	99.01%	0.8286%
RandomForest	98.88%	0.512%
SMO	98.51%	0.798%
NaiveBayes	79.85%	2.192%

Table 8: Accuracy and false-positive rate for 5-fold cross-validation on Data Set I.

Features	Accuracy	False-Positive Rate
PART	95.35%	4.805%
Logistic	94.71%	4.660%
J48	94.97%	4.447%
RandomForest	95.65%	3.247%
SMO	94.79%	4.547%
NaiveBayes	83.88%	8.087%

Table 9: Accuracy and false-positive rate for 5-fold cross-validation on Data Set II.

From these results we demonstrate that these classifiers can achieve very high classification accuracy, with low false-positive rates. We found that for the most part all the learning algorithms were similarly effective at classifying the URLs, with the exception of the NaiveBayes algorithm. The NaiveBayes algorithm was significantly less effective at classification using our feature extraction techniques.

Our results indicate that the tree and rule based learners performed slightly better than SMO and Logistic Regression algorithms, but the difference is very small, and may not be statistically significant.

For DataSet II we get on average 4% lower classification results. From examining this data we attribute this difference due to the fact that the second data set consists of many normal URLs along with a small amount of randomly selected URLs from yahoo's directory. Some of these randomly selected URLs are not logical URLs but perhaps redirected links. Here is an example of such a legitimate URL from this data set.

"http://www.youtube.com/results?
search_query= stephenie+meyer&search_type=&aq=f"

This URL is legitimate, but isn't a typical URL one might receive in an e-mail, or type into an address bar. It is a search query for youtube, and because of that it contains many uncommon symbols and characters for regular legitimate URLs. We believe the addition of URLs similar to this one into the data set might be the reason behind a slightly lower classification percent.

Information Gain

A popular method for evaluating how effective a feature is in a multi-feature classifier is to calculate the information gain. The general definition of information gain for decision trees is:

$$IG(T, a) = H(T) - H(T|a)$$

Where IG is the information gain, T is the class, and a is the feature. The function H is entropy.

For our multi-feature model we present the information gain values for each feature as a demonstration of which of the features was most effective when used together. The information gain values are for a model trained on data set I using the RandomForest algorithm.

Information Gain by Feature

Feature	Information Gain
Punctuation	0.557
Length	0.485
Target Words	0.456
TLD	0.392
Suspicious Words	0.126
@ and - symbols	0.126
IP Address	0.016

For the set of features that are not based on the character distribution, it appears that punctuation and length gave us the most value in our classifier, followed closely by Target Words and out of place TLDs.

Information Gain by Feature

Feature	Information Gain
KL-Divergence	0.811
KS-Distance	0.725
Edit-Distance	0.534
Euclidean	0.151

For features based on character distribution, we see that KL-Divergence adds the most, followed some what closely by KS-Distance. These values coincide with our single feature experiments, KL-Divergence performed better than KS-Distance, but only by a little bit, and euclidean distance performed much worse than either other method.

Rather than list all information gain values for character frequency, we present the top three.

It is probably not a coincidence that the letters in ".com" are the three most valuable by information gain in data set I. Perhaps phishing URLs are more or less likely to use a ".com" TLD in that data set. It is worth noting that in

Information Gain for Character Frequencies

Feature	Information Gain
c	0.603
o	0.596
m	0.595

Table 10: Data Set I

Feature	Information Gain
w	0.605
h	0.450
e	0.435

Table 11: Data Set II

Data Set II, the letters c, o, and m are also ranked between 0.39 and 0.41. Therefore, there may be a positive or negative correlation between phishing URLs and use of a ".com" TLD.

4.2 AdaBoost

A popular method for constructing strong learners is the Adaptive Boosting or AdaBoost method first presented in [9]. Using WEKA's built in AdaBoost implementation we were able to apply adaptive boosting to our top models to build more accurate classifiers. We used the AdaBoostM1 booster to build the following models.

Base Classifier	Accuracy	False-Positive Rate
PART	99.30%	0.475%
Logistic	98.73%	1.153%
J48	99.28%	0.535%
RandomForest	99.27%	0.324%
SMO	98.53%	0.761%
NaiveBayes	79.85%	2.192%

Table 12: AdaBoosted results on Data Set I.

Base Classifier	Accuracy	False-Positive Rate
PART	96.23%	3.160%
Logistic	94.71%	4.660%
J48	96.11%	3.554%
RandomForest	96.59%	2.554%
SMO	94.80%	4.574%
NaiveBayes	91.54%	6.907%

Table 13: AdaBoosted on Data Set II.

From our experiments we found that AdaBoosting provided a significant advantage for all algorithms. All of our algorithms reported higher accuracy and lower false-positive rate on the 5-fold cross validation tests once AdaBoosting was used in the training process. Additionally, to compare the improvement of AdaBoosting with other popular enhancement methods we ran a test on Stacking classifiers.

4.3 Stacking

Another popular method of constructing a strong learning is to use stacking. In stacking multiple learners are used to provide input to a meta learner that then builds the final model. The method takes advantage of the strengths of multiple different learning algorithms. Our implementation used WEKA's stacking classifier, using the ZeroR default implementation for the meta-classifier. We stacked the following classifiers, PART, J48, SMO, RandomForest, and Logistic Regression to build our final stacked learner.

Data Set	Accuracy	False-Positive
DataSet I	99.22%	0.580%
DataSet II	96.32%	3.167%

Table 14: Stacked classifier results.

Our stacked algorithm performed comparably with our best AdaBoosted algorithms, but not statistically better. The two methods both seem to be comparable for increasing the effectiveness of our classifier.

4.4 Cross Data Set Validation

We were interested in how our models generalized from one data set to another, so we did an experiment where we trained on one data set, and then evaluated on the other three data sets. This reveals some insight into how effective the dataset was for the phishing problem, and how well our methods generalize. For this we constructed a model using the RandomForest algorithm because from our earlier results it appeared to perform fairly accurately. We might have received better results with a stacked or boosted algorithm, but we chose to use random forest for this experiment due to the quicker training and evaluation. In the follow tables the top row indicates the training set and the first column indicates the testing set.

Cross Data Set Accuracy

	Set I	Set II	Set III	Set IV
Set I	99.98%	88.64%	86.09%	81.37%
Set II	55.99%	99.64%	87.10%	85.90%
Set III	58.14%	94.87%	99.72%	91.40%
Set IV	50.11%	81.27%	79.827%	99.58%

Table 15: Accuracy for classifier tests.

Cross Data Set False-Positives

	Set I	Set II	Set III	Set IV
Set I	0.0%	0.0%	7.1%	67.8%
Set II	83.1%	0.0%	7.7%	16.4%
Set III	83.4%	3.5%	0.0%	15.9%
Set IV	100.0%	9.2%	23.2%	0.0%

Table 16: False positives for classifier tests.

From our results we conclude that training on the Alexa data set (Set I) poorly generalized to the other data sets. Alexa consists of the top most popular websites on the web, and perhaps there are some malicious websites present in this list. Additionally, many popular websites are simple domain names, such as "www.facebook.com" with no path, while most phishing URLs have some sort of path, therefore if too many of the Alexa URLs lack a path, then it would cause the model to poorly generalize to the phishing URLs. Our classifier when trained on data from the DMOZ set, Zhang's sets, and the PhishTank URLs appear to generalize better on the other datasets.

Combined Data Set. Next we report in Table 16 our results of several classifiers with and without Adaboosting on a dataset that *combined all four datasets together*, a total of 115,405 urls with 59,548 legitimate and 55,857 phishing. The second number in each box is the Adaboosted result when the experiment finished in less than 8 hours or so.

4.5 Digging Deeper

We hypothesized that the normalized character distributions of phishing URLs are different from those of legitimate URLs because of the presence of digits, special symbols and the use of words like 'login', '.php', etc., in the URLs. Next we did a count of the URLs containing special symbols, numbers, strings such as ".php," ".js" and "login." The results are

Classifier	Accuracy	False-Positive Rate
PART	93.17%	7.08%
Logistic	90.45%/90.45%	9.69%/9.69%
J48	94.00%	6.13%
RandomForest	95.24%/96.22%	4.69%/3.61%
SMO	90.79%	8.48%
NaiveBayes	77.34%/83.45%	7.76%/4.43%

Table 17: Accuracy and false-positive rate for 5-fold cross-validation on combination of all datasets.

interesting. We report a feature only if it was present in at least 5% of the URLs in each set. The word login appears in a total of 9836 phishing URLs out of a total of 9839 URLs containing the word across all 4 sets. Results for the underscore and semicolon were also interesting, but we omit them here to save space. One might wonder, why phishers persist in their use of special symbols and strings in the URL. We suspect that there are several reasons for this, including: URL obfuscation using various encoding schemes (such as escape encoding, inappropriate UTF-8 encoding, etc.), cross-site scripting attacks through URL formatting, and preset session attacks. We leave testing of this hypothesis for the future. Note that Data sets I and IV shared phishing URLs, hence the first number is the same.

Feature	Set I	Set II	Set III	Set IV
Digits	4239/4369	4358/5032	5561/6447	4239/4918
&	1457/1457	3906/3996	2967/3070	1457/1612
?	2009/2009	6204/6401	4933/5211	2009/2665
=	2030/2034	6397/6579	5213/5472	2030/2653
-	3982/4532	7395/9077	6854/9048	3982/5685
.php	2150/2162	5082/5152	6339/6425	2150/2540

Table 18: Characters in URLs. The first number is count of phishing URLs and the second is the total number of URLs containing the feature.

4.6 Security Analysis of Our Method

An adaptive phisher may try to develop techniques that get past our classification model. They could have some success, given that our most successful techniques involved comparing the normalized character frequency of phishing vs legitimate URLs, either through statistical tests such as Kolmogorov-Smirnov, the Euclidean Distance between the two character distributions, or other methods. If phishers are to construct URLs that more closely resemble legitimate URLs then they might get past this method of classification. However, the phisher will face two problems in beating our overall classifier. The first problem with constructing more legitimate seeming URLs is that domain space is limited, when phishers have valid websites for only a few days before switching domains, they quickly run out of phishing URLs that remain somewhat close to their target. For example, if the phisher is attempting to emulate PayPal, there are so many previous attempts made to phish PayPal that anything that is extremely close to PayPal's official website URL is either already a registered domain, or is blacklisted. Therefore, in order for a phisher to emulate a Paypal URL, they have to start introducing misspellings, such as www.páypal.com, or they must construct another domain name to look like PayPal, such as www.paypal.com.anotherurl.com. Alternatively, if the phisher constructs valid URLs that emulate no target company, few people will click the URL such that the phishers work won't be profitable. The second problem is that the phisher who tries to mimic a legal URL closely in character-frequency distribution may find himself ensnared by one of our other features used in our classifiers such as edit-distance from a legal URL.

All of these phishing techniques enhance our classification methods, and help promote the robustness of our classifier. Ultimately, the end target must be fooled into clicking the URL, and therein lies the robustness of our methods.

4.7 URL Shortening Services

One of the contributions of [18] is that phishers are more frequently employing URL shortening services. Other authors have also demonstrated that this is an ever increasing problem regarding twitter, which places a maximum character count on tweets. In order to preserve characters, nearly all links that are tweeted employ a shortening service, and recently phishers have taken advantage of this.

Our work is specifically designed to analyze the lexical features of a URL, and therefore shortening services may result in mis-classifications. An improvement for our work might be to retrieve the final URL from the service, and then apply our classifier to the final URL. This is a potential tactic that might be employed in a future work.

5. RELATED RESEARCH

Many researchers have presented their work on phishing detection, so there are some similarities between our work and theirs, but also some key differences. As the astute reader will note from the presentation below, there are also several similarities among the features used by previous researchers, since the URL is basically a string of symbols.

In [18] the authors do a good job at outlining phisher modi operandi. However, they do not create a classifier nor do they indicate how one could be created. In contrast, we propose features based on statistical methods and then use machine learning to build classifiers significantly extending their observations. Specifically, we believe we are novel in our analysis of the character frequency information that [18] provides in their work.

In [10], Garera et al. divide phishing URLs into four different types, and select features based on each type. The authors use a set of suspicious words that have some overlap with the words we have used. However, unlike their work, we chose not to include features outside of lexical analysis of the URL, as our goal was to analyze the effectiveness of using only URL analysis and keep a margin of safety between users and the attackers.

After this work was completed, we discovered the work in [16], which we missed since they focus on classifying "malicious" URLs rather than phishing URLs. Ma et al. used a strict subset of the classifiers we have used (SVM, logistic regression, and a naive bayes classifier) and also their data sets are not as rich and diverse. We have seen in the results section that naive bayes and SVM are not among the strongest classifiers with our features. One of our data sets that we use is similar to theirs in the use of DMOZ data. Unfortunately, after learning of this work, we tried to find their second source for legitimate URLs, but it is no longer available. The authors did use some lexical features such as IP address and lengths of domain and URL. Unlike their

work we have not used a bag-of-words feature implementation because our goal was to demonstrate the effectiveness of a carefully chosen small feature vector. It is logical to assume we might achieve better results by including a bag-of-words, but this was specifically outside the goal of our work. Moreover, we do not use features beyond URL analysis for above mentioned reasons, whereas Ma et al. do.

The work by [30, 26] has been influential in the detection of phishing websites. There is some overlap between their work and ours, specifically in their URL analysis section of their methods. We share a set of three features: presence of @ and - symbols, number of punctuation symbols, and whether or not the URL is an IP address (also used by Ma et al. and others). Although, our punctuation filter includes more than just the dots that they use. However, bigger differences between our work and theirs are that they also use content-based features and Internet search and tested their approach on smaller datasets. The filters for IP and number of dots were also employed in the related work by [7] in the author's e-mail detection work.

Recently in [4], the authors have created a system based only on URLs to detect phishing websites. The authors achieve good accuracy, but lower than ours, with a confidence weighted classifier, using a bag-of-words implementation that generates a huge set of 369,000 features. Additionally, the authors have some hand selected features that correspond to ours, such as IP detection, TLD analysis, and Length analysis. Similar in URL features is the work of [25].

In another similar work, [11], the URL problem is analyzed further. Specifically the authors attempt to evaluate how much accuracy can be acquired by semi-supervised learning, and additionally address the problem of imbalanced data. Phishing URLs are far less common in day to day activities than legitimate URLs, this creates a natural imbalanced set of data. The authors alleviate the problem with undersampling, and feature selection methods. Their work shares some similar features with ours: length of the URL, the Dot Count, IP Address, and TLD. Additionally the authors use a target based feature that looks for popular phishing targets such as PayPal and Chase Bank. However, they do not consider character frequencies, and they tried a limited set of classifiers.

The authors in [13] have constructed a system for classifying e-mails as phishing, in their system they use a series of lexical URL features, some of which we also use. They share the following features with our work: special symbols, a collection of suspicious words, IP address, and number of periods. They also use a Random Forest algorithm to classify their e-mails. Their work demonstrates that phishing e-mail classification can be significantly improved by analysis of the URL. This result coincides with our results for phishing detection. This work differs from theirs in several respects: we consider many other classifiers than they do and we analyze more lexical features, including the character distributions of the URLs. More on phishing email detection can be found in [24, 23].

The authors in [5] use an SVM with lexical features to classify for phishing URLs. The author was specifically interested in thwarting "spear-phishing", phishing attacks focus on high-valued victims. The authors share some features with our work, and the previously mentioned related works. Uniquely shared between our two works is the feature Edit-Distance. The authors use two features, one for brand-name distance in the domain, and another from brand-name distance in the URL. Based on their performance for each individual feature, domain brand-name distance was their most effective feature. Our edit-distance implementation differs from theirs in that their method finds the minimal edit distance from all substrings in their URL to a set of a brands, or target words. Our implementation involved finding the minimal distance between the URLs as a whole and a small subset of legitimate URLs. We achieve similar results for this feature with our method. Our single feature classifiers on edit-distance have accuracies in the upper 80%, to lower 90%, while [5]'s domain edit distance achieves an accuracy of 88.44%.

The authors of [29] used a logistic regression classifier for their anti-phishing system in place. Their classifier achieves good accuracy, but lower accuracy than us, using both host and lexical based features. We get better performance on their data sets with purely lexical based classifiers. Moreover, some of their features are manually extracted whereas ours are completely automatic.

6. CONCLUSIONS

In this paper we addressed the problem of detecting phishing websites by analyzing their URLs. Phishing is a serious crime which involves stealing of information and/or funds from victims by tricking them into disclosing their information. Phishers must convince the victim that their website is a legitimate organization. Because the victim must be convinced, certain qualities are present in the phisher's URLs.

We have demonstrated that machine learning classifiers are able to classify URLs as phishing or non-phishing with a high degree of accuracy based entirely on lexical URL features. As features for our machine learning classifier we focused on character distributions and supplemented them with a carefully chosen set of features that enhance their robustness. We found that statistical tests for distributions of character frequencies yielded highly accurate classifiers. We believe this approach is novel, and as such we present the results of four different methods used to analyze character frequencies. We found that three of the four methods we proposed were able to classify URLs with above a 90% accuracy across multiple, independent, phishing data sets.

We found that, by combining features, we can reach extremely high accuracy with a very low false-positive rate for a variety of real data sets. Additionally, we found that our best models were robust enough to classify completely independent URLs without any training and still achieve a decent level of accuracy. The implication for this work is that phishing tools can be developed to be more accurate, and more time-efficient, using the methods we have presented. Thereby we provide evidence for our purpose, to demonstrate that a robust and accurate classifier for phishing detection can be created using a small number of intelligent features, based entirely on the URL.

7. REFERENCES

[1] Greg Aaron. Phishing activity trends report. http://docs.apwg.org/reports/apwg_trends_report_q1_2014.pdf, 2014.

[2] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975.

[3] Ram B. Basnet, Andrew H. Sung, and Quingzhong Liu. Feature selection for improved phishing detection. In *Proceedings of the 25th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems: Advanced Research in Applied Artificial Intelligence*, IEA/AIE'12, pages 252–261, Berlin, Heidelberg, 2012. Springer-Verlag.

[4] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *AISec*, pages 54–60, 2010.

[5] Weibo Chu, Bin B. Zhu, Feng Xue, Xiaohong Guan, and Zhongmin Cai. Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing urls. In *ICC*, pages 1990–1994. IEEE, 2013.

[6] Netscape Communications Corporation. Open directory rdf dump. http://rdf.dmoz.org/, 2004.

[7] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *Proc. 16th int'l conf. on World Wide Web*, pages 649–656. ACM, 2007.

[8] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Fifteenth International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[9] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[10] S. Garera, N. Provos, M. Chew, and A.D. Rubin. A framework for detection and measurement of phishing attacks. In *Proc. 2007 ACM workshop on Recurring malcode*, pages 1–8, 2007.

[11] Binod Gyawali, Thamar Solorio, Manuel Montes y Gómez, Brad Wardman, and Gary Warner. Evaluating a semisupervised approach to phishing url identification in a realistic scenario. In *CEAS*, pages 176–183, 2011.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[13] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Lexical url analysis for discriminating phishing and legitimate websites. In *CEAS*, pages 109–115, 2011.

[14] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 03 1951.

[15] S. le Cessie and J.C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[16] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Learning to detect malicious urls. *ACM TIST*, 2(3):30, 2011.

[17] Thomas McCall. Gartner survey shows phishing attacks escalated in 2007. http://www.gartner.com/newsroom/id/565125, 2007.

[18] D. Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. In *LEET*, 2008.

[19] Pavel Micka. Letter frequency (english). http://en.algoritmy.net/article/40379/Letter-frequency-English, 2008.

[20] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[21] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[22] Tanmay Thakur and Rakesh Verma. Catching classical and hijack-based phishing attacks. In Atul Prakash and R.K. Shyamasundar, editors, *Proceedings 10th International Conference on Information Systems Security*, pages 318–337. December 2014.

[23] Rakesh Verma and Nabil Hossain. Semantic feature selection for text with application to phishing email detection. In Hyang-Sook Lee and Dong-Guk Han, editors, *Proc. 16th International Conference on Information Security and Cryptology*, pages 455–468, November 2013.

[24] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. Detecting phishing emails the natural language way. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Proc. 17th European Symposium on Research in Computer Security*, pages 824–841, September 2012.

[25] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. *Proc. of 17th NDSS*, 2010.

[26] Guang Xiang, Jason I. Hong, Carolyn Penstein Rosé, and Lorrie Faith Cranor. CANTINA+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Inf. Syst. Secur.*, 14(2):21, 2011.

[27] Xin Xu. Logistic classifier. http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/Logistic.html, 2009.

[28] Harry Zhang. Exploring conditions for the optimality of naïve bayes. *IJPRAI*, 19(2):183–198, 2005.

[29] Jianyi Zhang and Yonghao Wang. A real-time automatic detection of phishing urls. In *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*, pages 1212–1216, Dec 2012.

[30] Y. Zhang, J.I. Hong, and L.F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proc. 16th int'l conf. on World Wide Web*, pages 639–648. ACM, 2007.