Detecting Phishing Attacks: A Comprehensive Approach

by

Ram Bahadur Basnet

A dissertation submitted in partial fulfillment of the requirements for the Doctorate of Philosophy in Computer Science.

New Mexico Institute of Mining and Technology
Department of Computer Science and Engineering

Socorro, New Mexico

December, 2011

Acknowledgments

This research work was carried out under the supervision of Prof. Andrew H. Sung. I’m very grateful to Prof. Sung for his continuous guidance, support, and advices that immensely helped me in so many ways during my 6 years at New Mexico Tech. Moreover, I truly appreciate him for giving me ample freedom in conducting my research. I would like to thank my dissertation committee Prof. Subhasish Mazumdar, Prof. Dongwan Shin, and Prof. Quinzhong Liu for their continuous support throughout this research study. I am very thankful to Prof. Mazumdar for his time and effort during many discussions that helped shaping my research and academic works in so many ways. I would like to thank the rest of the faculty and the staff of the Department of Computer Science and Engineering for the help I received during my graduate studies at New Mexico Tech.

A lot of individuals have directly or indirectly helped me in so many ways. Because of their love, support and encouragement, I’ve been able to pursue and finish my Ph.D.

I’m very grateful to Dr. Urs Duersteler and Ms. Rosemarie Luzi-Keller for believing in me and providing me parental love and financial support throughout my school life without which my aspiration to pursue higher education would, probably, never have been possible.

I’m very thankful to Mr. Umesh Shrestha, Ms. Bidya Limbu and Mr. Rajesh Shrestha for believing in me and providing me scholarship to finish my higher secondary education in one of the best private schools in Nepal. Moreover, I’m forever indebted to them for helping me to pursue my higher studies in the United States.

I heartily thank Institute for Complex Additive Systems Analysis (ICASA) for providing me funding to conduct my Ph.D. research. Special thanks go to Darryl Ackley, Bill West, Max Planck, Michael Smith, and Tammy Roof for their help and support throughout my student and professional career at New Mexico Tech.

Thanks go to Dr. Mark Dredze for providing software library for online algorithms.

Thanks to Prof. Lori Payne for encouraging me to pursue my Ph.D. at New Mexico Tech.

I'm thankful to a lot of my wonderful friends and colleagues for their help and support. I would like to thank Rudra Kafle, Saurav Bhattarai, Tenzin Doleck, Dr. Srinivas Mukkamala, Krishna Yendrapalli, Madhu K. Shankarpani, Kesav Kancherla, Divya Suryakumar, Ashish Koirala, Anand Pature, and many others.

I've been fortunate to have the love and support of my family. I want to thank my parents and siblings for teaching me the importance of hard-work, sincerity, and dedication, and overall to be a good human being.

To my wife Basudha, I am certain I would not have been able to do this without your love and support. Thank you for your support and sacrifices during these five years. To our son Kushal, thank you for understanding me when I couldn't play with you; thank you for your brilliant questions and your continuous quest for knowledge that keep me motivated and focused to do my work and research. Your witty sense of humor and laughter teach me every day what is important in life, reminding me how lucky I am to have you in my life.

Dedication

To my family.
To Dr. Urs Duersteler and Ms. Rosemarie Luzi-Keller.
To Mr. Umesh Shrestha and Ms. Bidya Limbu.

Abstract

The convenience of Internet communications and e-commerce has been exploited by consumers and criminals alike, and in recent years phishing has become a serious problem for information security as cyber criminals prey on individual Internet users as well as government organizations and businesses. Though public awareness to phishing attacks is generally on the rise, new and novel schemes or “zero-day attacks” that circumvent phishing filters and blacklists often still succeed in fooling even savvy users. Therefore, machine-learning-based approaches have been implemented for phishing detection.

In this dissertation, we conduct a comprehensive study of phishing detection by investigating detecting phishing emails, URLs, and webpages, separately, using machine learning techniques. In addition, a simple rule-based system is developed for performance comparison against machine-learning-based systems.

The key contribution of this dissertation is that it covers major aspects of phishing attack cycle and that it performance analyzes various methods for phishing detection. It is demonstrated that our proposed method of including a number of discriminative features extracted locally or from Web, employing a suitable feature selection algorithm, and then applying a batch or online learning classifier or ensemble of classifiers, results in highly accurate phishing detectors with respect to real-world data sets, thereby improving the state-of-the-art for this important problem. Further, preliminary analysis of a simple rule-based phishing detection engine also shows very promising results, thus indicating the tremendous potential for developing highly effective phishing detection methods using merely the expert systems approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Phishing is a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials, according to Anti Phishing Working Group (APWG) (APWG, 2010). Phishing emails are a major attack vector to carry out the notorious phishing attacks. In a typical email-based phishing attack, scammers or "phishers" first setup forged websites targeting online popular brands such as financial institutions, social networking sites, e-commerce, payment services, etc. Attackers then send specially crafted deceptive emails masquerading as the reputable organization to a large number of random Internet users. Cleverly crafted phishing emails normally contain various deceptive tricks to fool users to reply with confidential account information, fill and submit a form embedded in the HTML-based email, or click on a link to a fraudulent site where user is asked to reveal private information, e.g., account password, bank account information, credit card number, social security number, etc.

Though there are several anti-phishing software and techniques for detecting potential phishing attempts in emails and on websites, phishers devise new and hybrid techniques to circumvent the available software and filtering techniques. Once phishing email receivers are lured into a fraudulent website, even the experienced, security-minded users are often fooled to fulfill the website's primary goal. Dhamija et al. (2006) have examined various aspects of bogus websites that make them credible. Successful phishers not only present a high credibility web presence to their victim, they also create a presence that is so impressive that it causes the victim to fail to recognize security measures installed in web browsers and/or corporate security systems. Data indicates that some phishing attacks have convinced up to 5% of their recipients to provide sensitive information to spoofed websites (Dhamija, Tygar, & Hearst, 2006).

1.1 Phishing Attack Cycle

In general, deceptive phishing attacks are performed with the following five steps:

- ✓ Phishers set up bogus websites which look exactly like the legitimate ones, including setting up the web server, applying the DNS server name, and creating the webpages similar to the destination website, etc.
- ✓ Phishers then send a large number of spoofed emails to potential victims in the name of the legitimate targeted companies and organizations trying to convince the receivers to perform the following actions – among others:
    - ✓ Reply with or enter their personal information such as username, password, social security number, bank account number, etc. on the HTML form embedded in message body or as an attachment.
    - ✓ Click a link to visit forged website and submit the necessary personal information.
- ✓ Receivers, if convinced, divulge their personal information either through email or by visiting to the website and submitting the required information.
- ✓ Phishers collect personal information and commit fraud of various kinds such as transfer money from victims' bank account, send spam and phishing emails using victims' email and social networking accounts, steal identity and apply for credit cards and various kinds of loans, etc.

The goal of this research work is to find ways to thwart phishing attacks by disrupting one or more steps required to carry out a successful phishing attack.

1.2 Motivation

Proliferation of phishing attacks in recent years has presented an important cybersecurity research area. Over the years, there has been an increase in the technology, diversity, and sophistication of these attacks in response to increased user awareness and countermeasures. In reaction to increasing response from service providers and law enforcement, criminals are using increasing technical sophistication to establish more survivable infrastructures that support phishing activities. The key building blocks for these infrastructures are the botnets that are used to send phishing emails and host phishing sites (Milletary).

According to Gartner study (McCall, 2007) released in December 2007, phishing attacks represent a staggering amount of fraud, costing organizations more than 3 billion dollars annually. Even more shocking than this cost is the fact that phishing is a steadily growing problem with no end in sight. According to a survey of more than 4,500 online U.S. adults in August 2007, 3.3% of those who received phishing emails have lost money in the attack. Roughly \$3.2 billion was lost to phishing in 2007 (McCall, 2007).

APWG report (APWG, 2009) states that more brands are under attack than ever before, hitting record high in the 4th quarter of 2009. The United States continues its position as the top country hosting phishing sites. In average, about 45 thousands unique phishing websites were detected every month in the 4th quarter of 2009. In average, about 30 thousands unique phishing email reports were received by APWG from consumers. Customers of both well-known brands and lesser-known companies alike have been victim to this pervasive form of online fraud. In fact, a recent report (APWG, 2011) indicated more sophisticated schemes seem to have been used in phishing attacks that also exploited an increased number of brands.

Phishing attacks can cost not only the individual consumers but also the organizations whose brands are constantly being hijacked in the attack. Depending on the type of organization, the type of sensitive data lost or severity of attacks, and the length of time taken to take down the fraudulent phishing site, it can cost an organization from thousands to millions of dollars per attack.

The design and implementation of effective phishing detection techniques to combat cyber crime and to ensure cyber security, therefore, is an important and timely issue that – as long as the cyber criminals are proceeding unabated in scamming Internet users – requires sustained efforts from the research community.

1.3 Detecting Phishing Attacks: A Comprehensive Approach

The focus of this dissertation is on technical approaches to detect and prevent deception-based phishing attacks normally carried out by emails. In this dissertation, we conduct a comprehensive study of phishing detection by investigating detecting phishing emails, URLs, and webpages, separately, using machine learning techniques.

Phishing detection techniques based on the machine learning methodology has proved highly effective, due to the large phishing data sets available and the advances in feature mining and learning algorithms. Since no single classifier is perfect, we evaluate several supervised batch and online learning classifiers and the ensemble of classifiers. As researchers, we have no vested interest in any particular classifier. These classifiers are chosen mostly because they have been applied to problems similar to ours – such as in detecting: spam and phishing emails, phishing and malicious URLs, phishing and malicious websites, etc. We simply want to empirically compare a number of classifiers based on their availability in implementation and determine the one that yields the best performance in terms of both training and testing time and accuracy to the problem of detecting phishing attacks (phishing emails, URLs, and webpages).

We treat the problem of detecting phishing attack as a binary classification problem with phishing instances belong to the positive class and benign instances belong to the negative class. For all the models, we use the following notation. Given training data set in the following format: $(x_1, y_1), \ldots, (x_n, y_n)$, we use $x_i \in R^d$ to denote feature vector i or i^{th} sample and $y_i \in \{1, -1\}$ to denote the label of i^{th} sample with $y = 1$ for phishing and $y = -1$ for non-phishing. Here, n denotes the number of samples in the training set and d denotes the size of feature set or the dimensionality of the feature space.

We present our contributions in phishing detection in Chapter 2 to 7, summarized as follows.

Detecting and filtering out phishing emails are a desired first line of defense to the end users. Chapter 2 introduces novel heuristic-based and keyword-based features that are prominent among phishing emails. Comparing five classifiers, we demonstrate that SVMs achieve an accuracy of 98% in classifying phishing and non-phishing emails.

In Chapter 3, we propose text classification-based approach to classify phishing and non-phishing emails utilizing Confidence-Weighted linear classifiers – a recently introduced online algorithm. Using only the text contents of the emails as features, the proposed approach achieves at best 99.8% accuracy with false positive and false negative rates of 0.1% and 0.7%, respectively.

In Chapter 4, we present a novel scheme to automatically detect phishing URLs regardless of context and medium the phishing URL is distributed to Internet users. Unlike previous work in this area, we present a number of novel yet publicly available features on URLs and evaluate the viability of near real-time application of the proposed approach. Applying the scheme on real-world data sets, we demonstrate that the proposed approach is effective in detecting phishing URLs with an error rate of 0.3%, false positive rate of 0.2% and false negative rate of about 0.5%.

In Chapter 5, we propose several novel content-based features and apply batch and online learning algorithms. By augmenting the URL-based features with content-based features and employing the approach on real-world data sets, we demonstrate that the proposed approach can detect phishing webpages with error rates 0.04-0.44%, false positive and false negative rates of 0.0-0.3% and 0.06-0.73%, respectively using Random Forests classifier, thereby improving previous results on the important problem of phishing detection.

In Chapter 6, we present a novel rule-based method to detect phishing webpages. We first study a number of phishing websites to examine various tactics employed by phishers and generate a rule set based on observations. We propose a simple yet effective approach to detect phishing webpages by counting the number of rules satisfied by a webpage. This approach achieves false positive rate of 2.4% and false negative rate of 1.9%. We compare the approach with C4.5 decision tree and Logistic Regression learning algorithms. It is demonstrated that our rule-based method for phishing detection achieves performance comparable to learning machine-based methods, with the great advantage of understandable and flexible rules derived from experience.

Ensemble-based systems have shown to produce favorable results compared to those of single-expert systems for broad applications and under a variety of contexts. In Chapter 7, we investigate the efficacy of ensemble-based systems for phishing detection. Experiments results show that ensemble-based systems improve the error rates achieved by each base classifiers employed in the systems.

1.4 Classifier Performance Metrics

There are several metrics to measure the quality of binary classification models. We present the most widely used ones that are briefly described below.

Accuracy: Accuracy is the total number of correctly classified instances among all the instances available during the test.

Error Rate: Error rate is the total number of incorrectly classified instances among all the instances available during the test.

True Positive Rate (TPR): TPR (also called sensitivity, hit rate or recall) determines a classifier's test performance on classifying phishing instances correctly among all phishing instances available during the test.

$$TPR = \frac{TP}{\#\, positive\ instances} \tag{1-1}$$

False Positive Rate (FPR): FPR determines a classifier's test performance on classifying non-phishing instances incorrectly as phishing among all non-phishing instances available during the test.

$$FPR = \frac{FP}{\#\, negative\ instances} \tag{1-2}$$

True Negative Rate (TNR): TNR (also called specificity) is the proportion of instances that are predicted as non-phishing of all the instances that actually are non-phishing (true negative + false positive).

$$TNR = \frac{TN}{\#\, negative\ instances} \tag{1-3}$$

False Negative Rate (FNR): FNR determines a classifier's test performance on classifying phishing instances incorrectly as non-phishing.

$$FNR = \frac{FN}{\#\, positive\ instances} \tag{1-4}$$

F-Measure: F-measure is the harmonic mean of precision and recall. In F1-measure, precision and recall are equally weighted.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{1-5}$$

where,

$$Precision = \frac{TP}{TP + FP} \tag{1-6}$$

$$Recall = \frac{TP}{TP + FN} \tag{1-7}$$

Mathews Correlation Coefficient (MCC): MCC (Baldi, Brunak, Chauvin, Andersen, & Nielsen, 2000) takes into account true and false positive and negative and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes. MCC gives coefficient value between -1 and +1 where a coefficient of +1 represents a perfect prediction, 0 a random guess, and -1 an inverse prediction.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \tag{1-8}$$

Balanced Error Rate (BER): We also include another performance measure called balanced error rate (BER) which is one of the main judging criteria in feature selection. BER (Chen & Lin, 2003) is defined as:

$$BER = \frac{\mathrm{FPR} + \mathrm{FNR}}{2} \tag{1-9}$$

Receiver Operating Characteristic (ROC) Curves: The ROC is used to represent the plotting of the fraction of true positives (TP) versus the fraction of false positives (FP). In the ROC plot, the point (0, 1) is the perfect classifier, since it classifies all positive cases and negative cases correctly (Egan, 1975).

Chapter 2

Detecting Phishing Emails

2.1 Introduction

Fette et al. (2007) proposed a method for detecting phishing emails by incorporating features specific to phishing. Using the approach, they were able to accurately classify over 92% of phishing emails, while maintaining a false positive rate on the order of 0.1%. The results were obtained on a data set of approximately 860 phishing emails and 6,950 non-phishing emails. Further, they showed that the accuracy result on the data set was significantly better than that of SpamAssassin, a widely-used spam filter.

In this study, we use 7 features from Fette et al. and introduce 3 new heuristic-based and 6 keyword-based features to detect phishing emails. Applying the approach on a benchmark data set, we show that Support Vector Machines (SVMs) achieve an accuracy of more than 98% in detecting phishing and non-phishing emails.

2.2 Heuristic-Based Features

The heuristic-based features used in our approach are briefly described below.

HTML-based email: HTML tags are exploited by phishers to visually trick users. For instance, hyperlinks are active and clickable disguising the actual URL the users will be taken to only in HTML-based emails. Thus, an HTML-based email is flagged is used as a binary feature.

IP-based URL: Instead of buying and registering a domain name, phishers normally use botnets or compromised systems to host phishing websites. Using IP to obfuscate domain name is a common tactic as it provides easier and cheaper way to host phishing websites. A legitimate website usually has a domain name for its identification. Therefore, if an email contains a link whose host is an IP address (e.g., *http://81.215.214.238/bankofamerica/*), we flag it.

Age of domain name: Phishing websites usually have a short life of 3-4 days in average (APWG, 2006). We can, thus, use this feature to flag emails as phishing based on the fact that the linked-to domain is newly registered (less than 30 days old). This is a binary feature.

Number of domains: We extract and count the number of domains in the URL starting with http:// or https://. Two or more domain names are used in a URL to either trick users or to redirect them to a different domain. For instance, the URL

http://www.google.com/url?sa=t&ct=res&cd=3&url=http%3A%2F%2Fwww.antiphishing.org%2F&ei=-0qHRbWHK4z6oQLTm-BM&usg=uIZX_3aJvESkMveh4uItI5DDUzM=&sig2=AVrQFpFvihFnLjpnGHVsx has two domain names where google.com forwards the users to antihphishing.org domain. This is a continuous feature.

Number of sub-domains: Phishers obfuscate their links using two or more sub-domains with the target domain name to make links look legitimate. For instance, *https://login.paypal.somesite.com/verification.asp?d=1* has 2 sub-domains. This is a continuous feature.

Presence of JavaScript: JavaScript is usually employed in phishing emails, because it allows for deception on the client side using scripts to hide information or activate changes in the browser. Whenever an email contains the string "JavaScript", we flag it as a phishing email.

Presence of FORM tag: HTML forms are one of the techniques used to gather information from users. Instead of asking users to click on a link to go to a forged webpage controlled by phishers, users may be provided a form to fill in and submit with their personal information. We check whether an email has FORM tag and use it as a binary feature.

Number of links: In order to make phishing emails look more authentic and trick recipients, phishers may provide a number of links with anchor tags, mailto:, etc., pointing to the legitimate target addresses. We count number of links and use it as a continuous feature.

URL-based image source: To make the phishing emails look authentic, images and banner of targeted companies are used in the emails. Such images are usually linked to the target websites. Thus, if an email makes use of such URL-based images we flag it as a phishing email. This feature is binary.

Non-matching domains: Phishing emails may have different domains in the header and in the body part. For example, the 'From' field in the header part of the email may show *someone@paypal-site.com*, while the body can have company's legitimate domain *paypal.com* to provide an authentic look. This feature is binary.

Keywords: Phishing emails may contain a number of frequently used keywords such as *suspend*, *verify*, *username*, etc. We use *word frequency* (count of a keyword divided by total number of words in an email) of 6 groups of keywords.

2.3 Data Sets

In order to evaluate our methodology, we used two publicly available data sets: the ham corpora from the SpamAssassin project (SpamAssassin public corpus) as legitimate emails and the phishing emails from PhishingCorpus (Nazario). Our data set has a total of 4,000 emails, out of which 973 are phishing and 3,027 are legitimate (ham) emails.

2.4 Experiments and Results

We evaluated and compared the classification performance of the following classifiers: Support Vector Machines (SVM, BSVM & LOOMS), Neural Networks (NN) and Self Organizing Maps (SOMs). We used 5-fold cross-validation method to test the models.

2.4.1 Model Selection for SVMs

In any predictive learning task, such as classification, both a model and a parameter estimation method should be selected in order to achieve a high level of performance of the learning machine (Chapelle & Vapnik, 2000), (Cherkassy, 2002) and (Lee & Lin, 2000).

For SVM classifiers, usually the following parameters are chosen: (i) the penalty term *C* which determines the trade-off between the complexity of the decision function and the number of training examples misclassified, (ii) the mapping function Φ, and (iii) the kernel function such that:

$$K(x_i, x_j) = \Phi(\mathrm{x_i}) \cdot \Phi(\mathrm{x_j}) \tag{2-1}$$

In the case of RBF kernel, the width, which implicitly defines the high dimensional feature space, is the other parameter to be selected (Chapelle & Vapnik, 2000). We performed a grid search using 5-fold cross validation for each to automatically search *C* and γ values. Figure 2-1 is 3-D plot for different γ and *C* pairs. For BSVMs (Chan & King, 2004) we used LOOMs (Chang & Lin, 2003) technique to select models.

2.4.2 Results

Table 2-1 shows performance results of SVM-based classifiers with corresponding best *C* and γ values.

Using neural networks (NN), we achieve an overall accuracy of 97.8% with *trainscg* training function over all the 16 features. *trainrp* training function with all the 16 features achieved the best accuracy of 98%. Experiments with the various combinations of subsets of features achieved worse results compared to using all the 16 features emphasizing the importance of each feature.

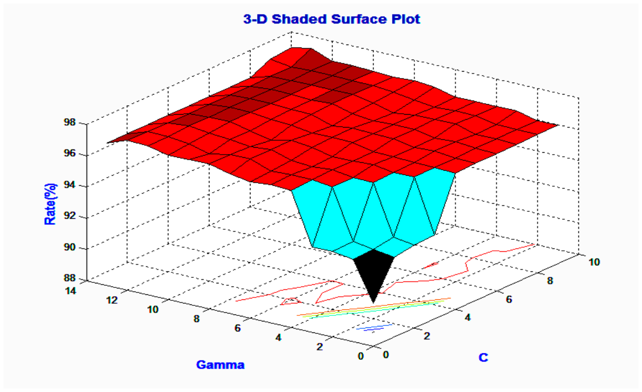

Figure 2-1: Model selection for BSVM using LOOMS. Accuracy rate for different Gamma (γ) and C pairs.

Classifier	Accuracy	*C*	γ
SVMs-rbf	98.04%	2097152.0	0.0625
BSVMs	97.99%	524288.0	0.125

Table 2-1: Performance of BSVM using 16 features with best C and γ pairs.

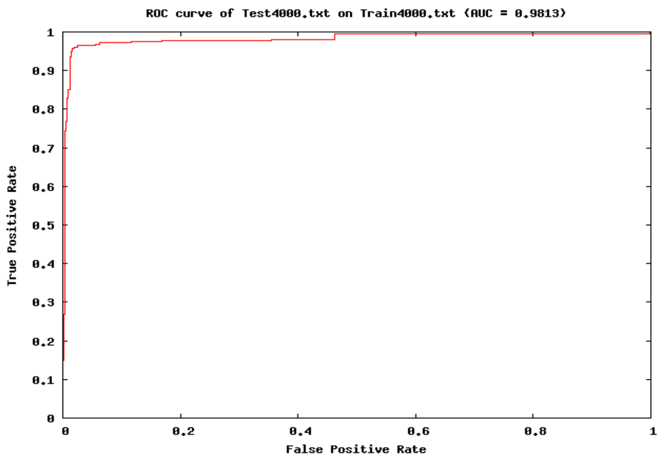

Figure 2-2: ROC curve on data set using SVMs.

When applying K-Means (Anderberg, 1973) on the data set, 635 (9.2%) instances were incorrectly clustered.

Accuracy can be measured by the area under the ROC curve (AUC). An area of 1 represents a perfect test and an area of .5 represents a worthless test. In our experiment, we achieved an AUC of 0.9813 as shown in Figure 2-2.

2.5 Conclusions and Future Work

We have proposed novel technical-based and keyword-based heuristics and shown that the proposed features are effective in automatically detecting phishing emails. Although the classification accuracies given by SVMs, BSVMs and NN are statistically similar (~98%), SVM-rbf achieved the best result in this context.

An interesting future direction would be to compare machine learning classifiers from other paradigms such as Bayesian and tree-based. As the tactics used by phishers evolve over time, new discriminative features (perhaps features from URLs and webpages) should be extracted from newer phishing emails to improve classification results on detecting phishing attacks.

Highly targeted and sophisticated phishing attacks called spear phishing (FBI, 2009) is on the rise (Goldman, 2011), (Kurtz, 2010) and detecting it – perhaps using semantics of email contents such as sentiment polarity, writing style and quality, etc. – is an open and interesting research problem.

Chapter 3

Detecting Phishing Emails: A Text Classification-Based Approach

3.1 Introduction

Detecting phishing emails using heuristic-based features is a common approach used by many research studies (Basnet, Mukkamala, & Sung, 2008), (Bergholz, Beer, Glahn, Moens, Paab, & Strobel, 2010), (Fette, Sadeh, & Tomasic, 2007), etc. Can we classify phishing emails from legitimate ones using only the text contents of the emails without using any heuristics? This chapter tries to answer this question. Fette et al. (2007) stated that phishing email classification appears to be simple text classification problem but, the classification is confounded by the fact that the class of "phishing" emails is nearly identical to the class of legitimate emails. From a learning perspective, this is a challenging problem. However, they didn't experimentally verify or show it otherwise.

Confidence-Weighted (CW) linear classifiers have recently attracted much attention and demonstrated their effectiveness in detecting malicious websites (Ma, Saul, Safage, & Voelker, 2009b). In this Chapter, we formulate the problem of detecting phishing emails as text-classification problem and experiment with CW classifiers and, based on a large phishing data set and in comparison with other learning machines, present impressive and highly accurate results as compared to those previously published.

3.2 Related Work

The idea of using contents of the text documents to automatically classify them into various pre-defined categories has a long history in text mining. However, we do not know of any previous work that used only the text contents as features to classify phishing emails against their ham counterparts.

The popular open source SpamAssassin project (SpamAssassin for Win32) uses a variety of mechanism including header text analysis, Bayesian filtering, DNS blacklists, and collaborative filtering databases to combat Spam. Bergholz et al. (2010) have proposed advanced email features generated by adaptively trained Dynamic Markov Chains and by latent Class-Topic Models. They show that classifiers trained using features extracted with these two techniques together with heuristic-based features outperforms the previous benchmark.

3.3 Background

In this section, we briefly discuss the underlying techniques we employ to achieve our goal.

3.3.1 Confidence-Weighted (CW) Linear Classifiers

CW (Dredze, Crammer, & Pereira, 2008) algorithm has been applied on a range of NLP tasks. They show that the algorithm improves over other state-of-the-art online and batch methods, learns faster in the online setting, and lends itself to better classifier combination after parallel training.

Ma et al. (2009b) have applied CW algorithm on a large-scale URL data sets from real-time source, large Web mail provider. They showed that recently-developed online algorithms such as CW can be highly accurate classifiers, capable of achieving classification accuracies up to 99% on experiments over a live URL feed. The study shows that CW clearly outperforms other online (Passive

Aggressive and Logistic Regression with Stochastic Gradient Descent) and batch algorithm such as LIBLINEAR (Fan, Chang, Hsieh, Wang, & Lin, 2008).

3.3.2 Text Categorization

The goal of text categorization is the classification of documents into a number of predefined categories. The first step in text categorization is to transform documents which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task (Joachims, 1998). Information retrieval research suggests that word stems work well as representation units and that their ordering in a document is of minor importance for many tasks. This leads to an attribute value representation of text. Each email document is an instance represented as a vector of stemmed words which is commonly called “bag of words” representation. Each distinct term w_i corresponds to a feature with the number of times term w_i occurs in the document as its value. More on text categorization can be found in (Lewis, Yand, Rose, & Li, 2004).

In texts classification tasks, millions of features derived from words and word combinations, most of which are binary and are infrequently on, can be weakly indicative of a particular class. These properties make the data very sparse which in turn demands large training sets, and very high dimensional parameter vectors. Therefore, the size and complexity of individual instances in the classification problem make it difficult to keep more than a small number of instances in main memory. These particularities make online algorithms, which process a single instance at a time, a good match for natural-language tasks (Dredze, Crammer, & Pereira, 2008).

3.4 Experiments and Results

In this section, we present the results of running CW linear classifiers and LIBLINEAR on data sets of emails described in subsection 3.4.1. The results in classifying the data sets are shown in subsection 3.4.3.

3.4.1 Data Sets

We used publicly available data sets from two different sources. For phishing emails, we used the phishing data sets available from (Nazario). This data set covers many phishing schemes and contents that evolved over the years. For non-phishing emails, we used public data set published by SpamAssassin Project (SpamAssassin public corpus). The ham corpus contains more than 20,000 emails.

We generated 5 sets of data containing varying number of phishing and ham emails out of those public data sources. The data sets provided in (Nazario) cover a variety of common phishing schemes. The phishing emails are collected at different times making them the most comprehensive public data sets. Corpus5 contains all the phishing emails collected between November 27, 2004 and August 7, 2007, making it the comprehensive data set. The details on each data set are summarized in Table 3-1.

Dataset	**# Samples**		**Feature Size**
	Phishing	**Ham**	
Corpus1	412	969	18,953
Corpus2	421	969	18,647
Corpus3	4,516	969	31,617
Corpus4	4,516	4,880	66,658
Corpus5	5,349	14,677	139,742

Table 3-1: Summary of data sets.

3.4.2 Experimental Setup

We parsed emails in mbox formats and discarded the attachment, HTML tags, JavaScript, etc. Using a standard list of stop words (e.g., articles, prepositions, etc.)

in information retrieval, we removed these words that do not aid in the text categorization process. We used Natural Language Toolkit (NLTK), an open source Python library, for preprocessing the email texts. Detail on texts preparation and feature extraction can be found in (Basnet, Torres, Sung, & Ribeiro, 2009), (Lewis, Yand, Rose, & Li, 2004).

Figure 3-1 shows the flowchart of the methodology we followed to carry out our experiments.

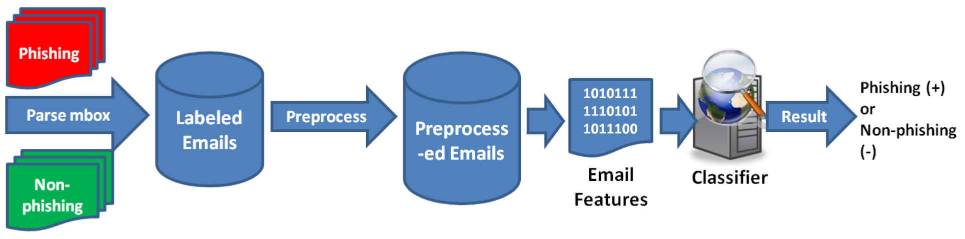

Figure 3-1: Graphical overview of experimental setup.

We used the holdout method to train and evaluate the classifiers. Each data set was divided into two groups, training and testing set, using 2/3rd-1/3rd split, respectively. The training set was used to train the classifier and the test set to estimate the error rate of the trained classifier. We show the average classification results of those 10 random splits on each data set.

We used the LIBLINEAR implementation of SVMs as our batch algorithm. LIBLINEAR is a linear classifier for millions of instances and features (Fan, Chang, Hsieh, Wang, & Lin, 2008).

3.4.3 Results

Table 3-2 shows the classification performance by CW and LIBLINEAR classifiers. Both CW and LIBLINEAR gave competitive results. Interestingly, CW algorithm gave better FPR, while LIBLINEAR classifier gave better FNR on these benchmark data sets.

Confidence-Weighted classifier achieved accuracy of more than 99% with FPR of less than 1% across all data sets. CW achieved slightly higher false negative rates of 2.1% and 3.4% in Corpus2 and Corpus1, respectively. LIBLINEAR, on the other hand, gave the best accuracy of 99.6% with FPR 0.6% and FNR of 0.2% on Corpus4. Though test accuracy, i.e., the fraction of correctly classified emails, is of limited interest in phishing classification, we report them for comparisons with related works.

Data Set	**Accuracy**		**FPR**		**FNR**	
	CW	**LIBLINEAR**	**CW**	**LIBLINEAR**	**CW**	**LIBLINEAR**
Corpus1	98.96%	**99.28%**	**0.06%**	0.74%	3.41%	**0.68%**
Corpus2	99.31%	**99.33%**	**0.06%**	0.49%	2.13%	**1.08%**
Corpus3	**99.72%**	99.45%	**0.84%**	2.30%	**0.17%**	**0.17%**
Corpus4	**99.77%**	99.58%	**0.15%**	0.63%	0.32%	**0.20%**
Corpus5	**99.76%**	99.46%	**0.08%**	0.48%	**0.68%**	**0.68%**

Table 3-2: Average accuracy, FPR and FNR for CWLC and LIBNEAR.

We also applied Support Vector Machines (SVMs) with RBF kernel. However, the results were not that impressive as the best test accuracy result it achieved was 82% on Corpus3 data set. The rest of the results were in the range of 70-75%.

Other performance metrics in evaluating binary classifier are also calculated and shown in Table 3-3.

Data Set	**Precision**		**Recall**		**F-Measure**	
	CWLC	**LIBLINEAR**	**CWLC**	**LIBLINEAR**	**CWLC**	**LIBLINEAR**
Corpus1	**99.84%**	98.26%	**96.59%**	99.32%	98.18%	**98.79%**
Corpus2	**99.86%**	98.85%	97.87%	**98.92%**	98.85%	**98.89%**
Corpus3	**99.82%**	99.51%	**99.83%**	**99.83%**	**99.83%**	99.67%
Corpus4	**99.84%**	99.33%	99.68%	**99.80%**	**99.76%**	99.56%
Corpus5	**99.79%**	98.68%	**99.32%**	**99.32%**	**99.55%**	99.00%

Table 3-3: Average precision, recall and F-measure for CWLC and LIBLINEAR.

We achieved the best F-measure of 99.8% compared to 97.6% achieved by Fette et al. (2007) and 99.5% by Bergholz et al. (2010).

The results given by CW are as good or better than the results in (Abu-Nimeh, Nappa, Wang, & Nair, 2007), (Basnet, Mukkamala, & Sung, 2008), (Bergholz, Beer, Glahn, Moens, Paab, & Strobel, 2010), (Fette, Sadeh, & Tomasic, 2007) though the experimental conditions and approaches are different.

3.5 Conclusions and Future Work

In this chapter, we have shown that it is possible to detect phishing emails with high accuracy by using Confidence-Weighted linear classifiers over features that are readily available from the email contents without applying extra effort to retrieve heuristic-based phishing specific features.

As the text of phishing emails are often similar to the text of legitimate emails, learning rules like Naïve Bayes might not actually help the classifier (Fette, Sadeh, & Tomasic, 2007). We didn’t closely examine our data sets to see if there were any highly similar ham and phishing emails, however. We would like to further investigate this matter to see how effectively CWLC can classify highly similar phishing email from its ham counterpart.

The method we’ve proposed obviously will not work on Image content. Phishers create images that contain the text of the message only in graphical form to bypass the content-based phishing filter. This is an interesting research area we would like to look into.

The results motivate future work to explore feature selection techniques and inclusion of those selected variables with the most common heuristic-based features as described in (Fette, Sadeh, & Tomasic, 2007) and apply CWLC to see if the predictive accuracy of the classifier can be further improved. An interesting research work would be to apply CWLC on the feature sets proposed in (Basnet, Mukkamala, & Sung, 2008) and (Bergholz, Beer, Glahn, Moens, Paab, & Strobel, 2010) for accurate comparisons of different machine learning techniques using different feature sets.

Chapter 4

Detecting Phishing URLs

4.1 Introduction

In this Chapter, we study the anatomy of phishing URLs that are created with the specific intent of impersonating a trusted third party to trick users into clicking the link to load a forged website. Unlike previous work in this area, we only use a number of publicly-available features on URLs alone; in addition, we compare performance of different machine learning techniques and evaluate the efficacy of real-time application of our method.

The means of distribution of phishing URLs include, among others, spam or phishing messages with links to the phishing site, Blackhat search engine optimization (SEO) techniques, social networking sites, Internet downloads, peer-to-peer (P2P) file sharing networks, visiting vulnerable websites such as blogs, forums, comment accepting news portals, instant messaging (IM), Internet Relay Chat (IRC), etc.

Blacklisting, perhaps, is the most common anti-phishing technique used by modern web browsers. However, study shows that centralized blacklist-based protection alone is not adequate enough to protect end users from new and emerging zero-day phishing webpages that appear in thousands and quickly disappear every day (Ludl, McAllister, Kirda, & Kruegel, 2007). Another study shows that heuristics based phishing techniques outperform centralized blacklisting techniques used by most web browsers (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). To address these shortcomings, what are needed are methods that are discovery-oriented, dynamic, and semi-automated. This approach should not replace but compliment the blacklist to provide defense-in-depth mechanism to effectively combat the phishing attacks.

To that end, we present a heuristic-based methodology for automatically classifying URLs as being potentially phishing in nature. This methodology could then be used to thwart a phishing attack by either masking the potentially phishing URL, or by alerting the user about the potential threat. Because of the focus on the URL itself, this approach can be applied anywhere that a URL can be embedded, such as in emails, webpages, chat sessions, etc. We evaluate our approach on real-world data sets with more than 16,000 phishing and 31,000 non-phishing URLs. We experimentally demonstrate that our approach can obtain an error rate of less than 0.5% while maintaining low false positive and negative rates. Featured with high accuracy rate, we believe that our light-weight approach can be used by individual users in their system for near real-time phishing URL detection.

4.2 Background

In this section, we provide our definition of phishing URL and review some related works.

4.2.1 Definition of Phishing URL

Whittaker et al. (2010) define a phishing webpage as “any webpage that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewers would only trust a true agent of a the third party.” This definition, which is similar to the definition of “web forgery”, covers a wide range of phishing pages from typical ones – displaying graphics relating to a financial company and requesting a viewer’s personal credentials – to sites which claim to be able to perform actions through a third party once provided with the viewer’s login credentials. We use the same definition of phishing and thus define phishing URL as a URL that leads user to a phishing webpage. Our study, by this definition, is therefore independent of the attack vector by which a phishing URL is distributed.

4.2.2 Anatomy of a URL

An HTTP URL, a specific type of URI (Universal Resource Identifier), locates an existing resource on the Internet using HTTP (Berners-Lee, Masinter, & McCahill, 1994), (HyperText Transfer Protocol), (URL Syntactic Components). Phishers will try to construct their "bait" – phishing URL – in every possible way in order to deceive unsuspecting users. Thus, it is important to understand the anatomy of URL to effectively combat against phishing URLs.

A complete HTTP URL takes the form:

<scheme>://<netloc>:<port>/<path>?<query>. Each component is briefly explained below using the following URL example:

http://username:password@www.site.com:8080/dir1/dir2/file1.php?id=100&query2=2.

<scheme>: The scheme identifies the protocol to be used to access the resource on the Internet. In an HTTP URL, it can be *http* or *https* (using SSL/TLS). Obviously, the above example has *http* scheme.

<netloc>: Network location component, that holds the resource, is commonly called host name. It is possible to specify a username or combination of username and password in a URL along with port number. Port number can be omitted if web server uses the default port number for the given scheme (port 80 for HTTP). Though no username or password is allowed in the actual syntax of HTTP URL, most modern browsers allow it and thus, phishers may use these tokens to trick users. In the example URL, username and password are separated by colon ':', the host name or network location is *www.site.com* and the web server is configured to serve the domain through port 8080.

<path>: The path component identifies the specific resource within the host that the web client wants to access. */dir1/dir2/file1.php* in the given example is the path in the URL.

<query>: The query component provides query information that the resource can use for some purpose for example as data to be processed. Query string is usually a name value pairs separated by '='. Multiple query parameters can be supplied with '&' special character. For example, *id=100&query2=2* is query component in the given URL.

Within the <path> and <searchpart> components, "/", ";", "?" are reserved. The "/" character may be used within HTTP to designate a hierarchical structure. The <path> is optional, as is the <query> and its preceding "?". If neither <path> nor <query> is present, the "/" may also be omitted.

4.2.3 Major URL Obfuscation Techniques

URL obfuscation is technically simple yet very popular and effective technique employed by phishers to lure their victims to forged websites. URL obfuscation misleads the victims into thinking that a link and/or website, they have come across, is that of a trusted legitimate site. One of the common tricks used is to obfuscate the host with an IP address. Instead of using domain name in URL such as *http://www.phishingsite.com/...*, phishers use the public IP of the system hosting the phishing website in place of the hostname part of the URL such as *http://245.222.111.123/**bankofamerica.com**/...*. Some other notable techniques are: obfuscating URL with unknown or misspelled domain to take advantage of "typo squatting" or to visually deceive the users (e.g., *paypa1.com* to forge the target *paypa1.com,* e.g., *faecbook.com* to forge *facebook.com*), obfuscating domain with a large number of sub-domains (e.g., *www2.**bankofamerica.com**.jk09.boa.ru/...*), obfuscating domain with special characters such as "-" soft hyphen, Unicode and visually similar looking characters (e.g., *www.**google**-accounts.c-o.in/*), obfuscating the host with another domain where the URL path contains the web domain of the target being phished and exploiting URL redirection technique to redirect users to phishing page (e.g., *www.**paypal**.com@www.paypa1.com/signIn.html*).

Phishers can craft a link, with some social engineering, pertaining to a targeted user on a legitimate looking URL. If the user's actual username from the target site is embedded in the URL, unsuspecting user may think that the URL belongs to the legitimate site and thus she may be duped into revealing her password. In our data sets, two phishing and zero non-phishing URLs had username in them.

We try to identify these tactics and use them as features in our classifiers. These features are grouped under lexical-based features in section 4.3.3.1.

4.2.4 Related Work

4.2.4.1 Machine Learning Approaches

The work by Garera et al. (2007) is the most closely related to our work. They use logistic regression over 18 hand-selected features to classify phishing URLs. The features include the presence of certain red flag key words in the URL and some proprietary features based on Google's PageRank and webpage quality guidelines. Though their approach doesn't analyze the page contents to use as features, they use the pre-computed page-based features from Google's proprietary infrastructure which they call *Crawl Database*. They achieve a classification accuracy of 97.3% over a set of 2,500 URLs. Direct comparison with our approach, however, is difficult without access to the same data sets or features. Though similar in goal, our approach differs significantly in both methodology (considering new publicly available features based on URLs alone and comparing several machine learning algorithms) and scale (considering more features and an order-of-magnitude more samples).

Ma et al. (2009a, 2009b) propose a method to classify malicious URLs using variable number of lexical and host-based properties of the URLs. Using these features, they compare the accuracy of four batch and four online learning algorithms. Though we use some similar features and classification models, our approach is different in a number of ways. First, the scope of our work is limited to detecting phishing URLs as opposed to detecting wide range of malicious URLs. Our techniques can certainly be extended to detecting and classifying wider range of malicious URLs. Secondly, we have a fixed set of smaller number of features. Thirdly, we do not use host-based properties of webpages such as WHOIS entries, connection speed, etc. Though WHOIS information can be very useful in determining the reputation of hosts and registrars and the reputation of the domains overall, it makes the repetition of the experiments difficult. Last but not least, we evaluate the effectiveness of our approach with an expanded set of machine learning algorithms.

4.2.4.2 Non-machine Learning Approaches

Besides machine learning (ML) based techniques, there exist a number of other approaches in phishing detection. Perhaps, the most widely used anti-phishing technology is the URL blacklist technique that most modern browsers are equipped with (Google Safe Browsing API), (SmartScreen Filter). Other popular methods are browser-based plug-in or add-in toolbars such as SpoofGuard (Chou, Ledesma, Teraguchu, Boneh, & Mitchell, 2004). SpoofGuard uses domain name, URL, link, and images to evaluate the spoof probability on a webpage. There has been an attempt to detect phishing attack using user generated rules (Basnet, Sung, & Liu, 2011). Other anti-phishing tools include SpoofStick (SpoofStick Home), SiteAdvisor (McAfee SiteAdvisor Software), Netcraft anti-phishing toolbar (Netcraft Toolbar), AVG Security Toolbar (AVG Security Toolbar), etc.

4.3 Our Method

In this section, we describe in detail our approach to detecting phishing URLs.

4.3.1 Method Overview

We propose a heuristic-based approach to classifying phishing URLs by using the information available only from URLs. We first run a number of scripts to collect our phishing and benign URLs and create our data sets. Our next batch of scripts then extracts a number of features by employing various publicly available resources. We then apply various machine learning algorithms to build models from training data which is comprised of pairs of feature values and class labels. Separate set of test data are then supplied to the models, and the predicted class of the data instance is compared to the actual class of the instance to compute the accuracy of the classification models. Figure 4-1 provides the overview of graphical representation of phishing URL detection framework.

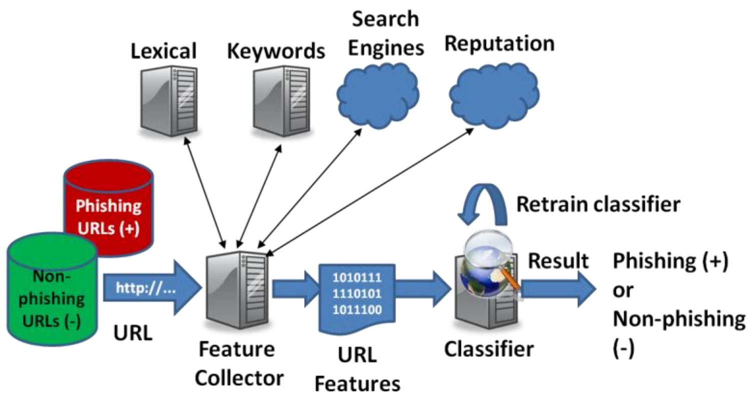

Figure 4-1: Overview of phishing URL detection framework.

4.3.2 Data Sets

For experiments, we collected our data from various credible sources that are also used by Ma et al. (2009a), Zhang et al. (2007), and many others. For phishing URLs, we wrote scripts to automatically download confirmed phishing websites' URLs from PhishTank. PhishTank is a collaborative clearing house for data and information about phishing on the Internet (PhishTank). After signing up, developers and researchers can download confirmed phishing URL lists in various file formats with an API key provided for free. A potential phishing URL once submitted is verified by a number of registered users to confirm it. We collected first set of 11,361 phishing URLs from June 1 to October 31 of 2010 and call it OldPhishTank data set.

Phishing is an arms race where tactics used by scammers evolve over time. In order to follow these evolving URL features and to closely mimic the real-world scenario, we collected second batch of 5,456 confirmed phishing URLs that were submitted to PhishTank for verification from January 1 to May 3, 2011. We call it NewPhishTank data set.

In order to address URLs that were produced using shortening services such as bit.ly, goo.gl, etc., we developed a Python library (Basnet, 2010b) to utilize the web service API provided by *longurl.org* to automatically detect and expand shortened URLs. The service currently supports about 333 popular shortening services. However, some short URLs – either from some new and unsupported shortening services or because the shortening services do not expand because the target URL has been reported as phishing or malicious – do exist in our data sets.

We collected our non-phishing URLs mainly from two public data sources: Yahoo! directory[1] and DMOZ Open Directory Project[2]. We used Yahoo's server redirection service, *http://random.yahoo.com/bin/ryl*, which randomly selects a web link from Yahoo directory and redirects browser to that page. In order to cover wider URL structures, we also made a list of URLs of most commonly phished targets (using statistics of top targets from PhishTank). We crawled webpages from these URLs, parsed the retrieved HTML contents, and harvested the hyperlinks therein to also use as non-phishing URLs. We made the assumption,

[1] http://dir.yahoo.com
[2] http://www.dmoz.org

which we think is reasonable, to treat those additional hyperlinks as benign since they were extracted from a legitimate source. We use 22,213 legitimate URLs using these two sources and call it Yahoo data set. These URLs were collected between September 15, 2010 and October 31, 2010. The other source of legitimate URLs, DMOZ, is a directory whose entries are vetted manually by editors. We use 9,636 randomly chosen non-phishing URLs from this source and call it DMOZ data set.

We then paired OldPhishTank and NewPhishTank data sets with non-phishing URLs from a benign source (either Yahoo or DMOZ). We refer to these data sets as the OldPhishTank-Yahoo (OY), OldPhishTank-DMOZ (OD), NewPhishTank-Yahoo (NY), and NewPhishTank-DMOZ (ND).

4.3.3 Feature Analysis

We develop our set of 138 features based on related works, drawing primarily from (Basnet, Mukkamala, & Sung, 2008), (Garera, Provos, Chew, & Rubin, 2007), (Ma, Saul, Safage, & Voelker, 2009a, 2009b), (Whittaker, Ryner, & Nazif, 2010), (Zhang, Hong, & Cranor, 2007). Most of these features are newly proposed; some are different in the way they are extracted and selected. The use of relatively small number of fixed set of features makes the decision boundaries less complex, and therefore less prone to over-fitting as well as faster to evaluate for most batch algorithms.

We group features that we gather into 4 broad categories described as follows.

4.3.3.1 Lexical-Based Features

Lexical features, the textual properties of the URL itself, have been widely used in literature (Basnet, Mukkamala, & Sung, 2008), (Fette, Sadeh, & Tomasic, 2007), (Garera, Provos, Chew, & Rubin, 2007), (Ma, Saul, Safage, & Voelker, 2009a, 2009b), (Whittaker, Ryner, & Nazif, 2010), (Zhang, Hong, & Cranor, 2007) in detecting phishing attacks.

Feature Description	URL Type	Max	Min	Mean	Median
Length of Host	Phishing	240	4	21.38	19
	Non-phishing	70	5	18.77	18
Number of ‘.’ in Host	Phishing	30	0	2.13	2
	Non-phishing	5	1	2.14	2
Number of ‘.’ in Path	Phishing	18	0	0.86	1
	Non-phishing	13	0	0.25	1
Number of ‘.’ in URL	Phishing	30	0	3.00	3
	Non-phishing	15	1	2.38	2
Length of Path	Phishing	380	0	24.55	15
	Non-phishing	360	0	10.74	1
Length of URL	Phishing	999	13	66.09	1
	Non-phishing	383	15	41.22	33

Table 4-1: Lexical-based real valued features and their statistics.

Feature Description	% Phishing URLs	% Non-phishing URLs
‘-’ in Host	2.02%	9.03%
Digit [0-9] in Host	30.06%	3.11%
IP Based Host	4.15%	0.00%
Hex Based Host	0.18%	0.00%
‘-’ in Path	15.82%	6.64%
‘/’ in Path	98.39%	96.18%
‘=’ in Path	4.58%	0.16%
‘;’ in Path	0.07%	0.00%
‘,’ in Path	0.15%	0.28%
Has Parameter Part	0.18%	0.77%
Has Query Part	0.07%	0.01%
‘=’ in Query Part	13.45%	10.43%
Has Fragment Part	0.18%	0.77%
‘@’ in URL	0.33%	0.08%
‘Username’ in URL	0.33%	0.08%
‘Password’ in URL	0.02%	0.00%
Has Non-Standard Port	0.01%	0.00%
‘_’ in Path	11.16%	8.41%

Table 4-2: Summary of lexical-based binary valued features and their statistics.

We examine various obfuscation techniques (see Section 4.2.3) phishers may employ and derive a number of phishing like features to use in our classifiers.

There are 24 features in this category. We summarize the real-valued and binary features separately in Table 4-1 and Table 4-2, respectively. The decimal numbers are rounded to 2 digits after the decimal point.

4.3.3.2 Keyword-Based Features

Phishing URLs are found to contain eye-catching word tokens (e.g., *login, signin, confirm, verify,* etc.) to visually trick users. Garera et al. (2007) select 8 red flag keywords to use as features. Whittaker et al. (2010) use every string token separated by non-alphanumeric characters out of a URL to use as features. However, they rely on the feature selection methods built into their machine learning framework to incorporate only the most useful of these features into their classification models. Though our word based feature extraction technique is somewhat similar to theirs, the selection technique differs significantly.

Using the *Random Set* (see Section 4.3.3), we tokenize each phishing URL by splitting it using non-alphanumeric characters. After applying Porter stemmer (NLTK), we obtain 12,012 unique root tokens and their frequencies. While we could use every word token appearing on phishing URLs as a feature – an approach taken by Ma et al. (2009a, 2009b) in detecting malicious URLs – this many keyword based features plus other features per URL can burden our batch learning algorithms without yielding any performance benefit. Instead, we discard all tokens with length < 3 such as *d*, *c*, *e*, *br*, *fr*, *it*, etc. Some single-character tokens have high frequencies, but offer no meaning. Two-character tokens are mostly the country code top-level domains (ccTLD). We also discard several common URL parts such as *http*, *www*, *com*, etc. and webpage file extensions such as *htm, html, asp, php,* etc. Since top target organizations such as *paypal*, *ebay*, *bankofamerica*, *wamu,* etc. are covered by top target list under reputation based features, we discard them as well. A large number of tokens have frequency one, suggesting that they are not frequently used in phishing URLs. Most of these words either do not make sense as a whole or they have random characters such as *ykokejox*, *riversid*, *sxkretyvwufatnrmomgpqjdw*, *njghlfi*, etc. We discard these tokens as well. With this preliminary selection, we are left with 1,127 tokens.

We then apply feature selection technique commonly used in text classification. Feature selection serves two main purposes. First, it makes training and applying a classifier more efficient by decreasing the size of the discriminative features. This is of particular importance for classifiers that, unlike Naïve Bayes, are expensive to train. Second, feature selection often increases classification accuracy by eliminating noise features. We compute mutual information (MI) of each term in phishing class. MI measures how much information the presence or absence of a term contributes to making the correct classification decision on a class (Manning, Raghavan, & Schutze, 2008). MI is computed using the following equation:

$$MI(U;C) = \frac{N_{11}}{N} log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} log_2 \frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N} log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} log_2 \frac{NN_{00}}{N_{0.}N_{.0}} \quad (4\text{-}1)$$

where U is a random variable that takes values e_t = 1 (the URL contains term t) and e_t = 0 (the URL does not contain term t), C is a random variable that takes values e_c = 1 (the URL is in class c) and e_c = 0 (the URL is not in class c), N s are counts of URLs that have the values of e_t and e_c that are indicated by the two subscripts. For example, N_{10} is the number of URLs that contain t (e_t = 1) and are not in c (e_c = 0). $N_{1.} = N_{10} + N_{11}$ is the number of URLs that contain t (e_t = 1) and we count URLs independent of class membership ($e_c \in \{0, 1\}$). $N = N_{00} + N_{01} + N_{10} + N_{11}$ is the total number of URLs in the training set.

Terms with high MI values indicate that they are more relevant to the class and are good discriminative features, whereas terms with lower MI values indicate that they are less relevant. For brevity, Table 4 shows only the top 10 terms based on

MI along with the percentage of each term appearing in phishing and non-phishing URLs in the selected training data set.

Root Term	MI	% Phishing URLs	% Non-phishing URLs
log	0.1740	21.77%	1.71%
pay	0.1027	13.26%	0.50%
web	0.0778	14.90%	1.62%
cmd	0.6840	10.08%	0.37%
account	0.0559	7.86%	0.34%
dispatch	0.0390	5.69%	0.01%
free	0.0362	7.20%	0.48%
run	0.0331	4.89%	0.16%
net	0.0320	13.05%	5.05%
confirm	0.0292	3.42%	0.00%

Table 4-3: Top 10 root terms (based on mutual information) and their statistics.

Note that statistical values of each feature among phishing and non-phishing URLs are rounded to 2 and MI values are rounded to 4 decimal digits. Because some keywords are so sparsely present among non-phishing URLs in our training data set, their values are rounded to 0.00%. The only feature term that is entirely absent among non-phishing URLs from all data sets is *config*.

By ordering the terms based on MI values from high to low, we then use these terms as binary features on OY data set. Using forward selection method, we train and test Naïve Bayes 1,127 times for each feature set size from 1 to 1,127 and record its error rate for each run. We choose Naïve Bayes because of its speed and its effectiveness in spam filtering application (SpamAssassin for Win32) – a problem similar to ours – that uses "spammy" word tokens. Figure 4-2 shows the error rates on feature size from 1 to 1,127.

Initially, the error rate decreases significantly from ~29% to ~24% as the number of features increases. But after 100 features, the change in error rate is statistically insignificant (< 0.1%). Thus, we decide to use the top 101 terms based on their MI as keyword based features.

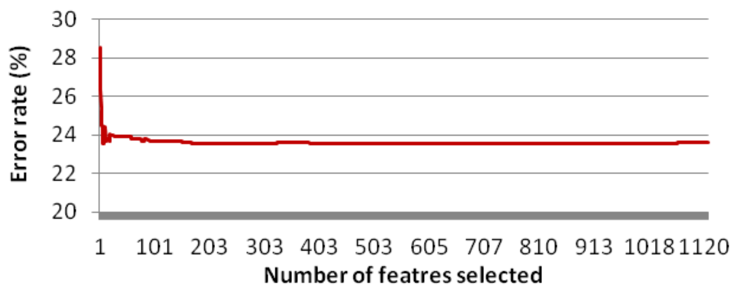

Figure 4-2: Effects of keyword feature set size on error rate using Naïve Bayes on OY data set.

4.3.3.3 Reputation-Based Features

PhishTank produces various top 10 statistical reports on phishing websites every month. We downloaded 3 types of statistics: Top 10 Domains, Top 10 IPs, and Top 10 Popular Targets from the first batch of statistics published in October 2006 to October 2010. The idea behind this is to make use of the historical data on top IPs and domains that host phishing websites. If a URL has many other phishing related heuristics and also its host belongs to top IP and/or top domain that has historic reputation of hosting phishing webpages, then we can increase our confidence level to classify the URL at hand as phishing. There are 311 unique domains, 354 unique IPs, and 43 unique targets in the top 10 statistics during 4 years of period.

Features from PhishTank statistics may appear little biased to use against detecting phishing URLs from the same source. The idea, however, is to use features based on many statistical reports similar to these on phishing webpages. For instance, we include statistics from StopBadware.org which we explain next. We plan to find more public statistics to use in our future work, and we hypothesize that these features help in detecting phishing URLs if included with many other discriminative features.

StopBadware.org works with its network of partner organizations such as Google, Sunbelt Software, etc. and individuals to fight back against viruses, spyware, etc. (StopBadware). It produces top 50 IP address report from number of reported URLs. We check if the IP address of a URL belongs to this top 50 report and flag it as potentially phishing if it does.

Table 4-4 summarizes the distribution of reputation based features in phishing and non-phishing URLs.

Feature Description	% Phishing URLs	% Non-phishing URLs
PhishTank Top 10 Domain in URL	20.98%	4.87%
PhishTank Top 10 Target in URL	32.65%	14.21%
IP in PhishTank Top 10 IPs	17.30%	0.87%
IP in StopBadware Top 50 IPs	2.31%	1.37%
URL in Phishing Blacklist	42.41%	0.00%
URL in Malware Blacklist	0.45%	0.05%
URL in RegTest Blacklist	0.16%	0.00%

Table 4-4: Reputation based features and their statistics.

Several blacklists have been used in Ma et al. (2009a, 2009b). We use Safe Browsing API (Google Safe Browsing API) to check URLs against Google's constantly updated blacklists of suspected phishing and malware pages and use 3 binary features for membership in those blacklists. Essentially, these blacklists are also used by Google Chrome and Mozilla Firefox to warn users of potentially malicious websites.

4.3.3.4 Search Engine-Based Features

Google search engine has been used in Garera et al. (2007), Whittaker et al. (2010), and Zhang et al. (2007). Whittaker et al. use PageRank from Google proprietary infrastructure. Garera et al. use Google's proprietary technologies such as PageRank, page index, and page quality scores. These are pre-computed during Google's crawl phase and are stored in a table, which they call *Crawl Database*. Though the features generated from the Google proprietary infrastructure look plausible, it makes the repetition and validation of experiments extremely difficult if not impossible. On the other hand, our search engine based feature gathering technique uses either publicly available APIs or mimics users using search engines to gather information on a URL.

Zhang et al. select top 5 words with highest TF-IDF value to generate lexical signature of a page. They feed each lexical signature to Google search engine and check if the domain name of the current webpage matches the domain name of the top 30 results. If yes, they consider it to be a legitimate website. Though the goal is similar, we utilize search engines in different ways. Instead of using query terms, we use URL or its domain part.

We check if a URL exists in the search engines' index in the following manner. First, we search for the whole URL and retrieve the top 30 results. Our preliminary experiments showed that retrieving top 10 results was enough to check if a URL has been indexed. We use top 30 results, even so, to be on the safe side as the work by Zhang et al. show that retrieving more than 30 results doesn't yield any performance improvements. If the results contain the URL, we consider it as a potentially benign URL, phishing otherwise. We also check if the domain part of a URL matches the domain part of any links in the results. Similarly, if there is a match, we flag the URL as a potentially legitimate URL. Otherwise, we query the search engine again with just the domain part of a URL. If none of the returned links matches the query URL, we flag the URL as potentially phishing. If both the URL and the domain do not exist in search engines index, it is a high indication that the domain is a newly created one and the URL in question is more likely to be phishing. Hence, we believe that these features also compliment the 'age of domain' feature based on WHOIS used by most of the related works.

Search engine based features and their statistics are summarized in Table 4-5.

Our heuristic, however, makes certain assumptions that the leading top 3 search engines index the vast majority of legitimate websites and that legitimate sites usually live longer and hence, the search engines will index them sooner or later. On the other hand, the average time a phishing site stays online is 4.5 days or even less (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009), (Zhang, Hong, & Cranor, 2007). Moreover, there won't be that many links pointing to the phishing website. Because of the low life span and lack of links pointing to the phishing site, we assume that search engines' crawlers may not get to the site before they are taken down. We employ 3 major search engines with the strong reason that at least one of them may have indexed legitimate website if not all. Furthermore, search engines may try to filter out known malicious links from the search results using their proprietary technologies. Thus, the heuristic effectively leverages on top search engines crawling resources and proprietary filtering techniques.

Feature Description	% Phishing URLs	% Non-phishing URLs
URL NOT in Google Top Results	98.71%	4.85%
Domain NOT in Google Top Results	98.27%	2.64%
URL NOT in Bing Top Results	96.95%	34.63%
Domain NOT in Bing Top Results	96.34%	12.77%
URL NOT in Yahoo Top Results	98.93%	17.74%
Domain NOT in Yahoo Top Results	98.71%	13.95%

Table 4-5: Search-engine-based features and their statistics.

To the best of our knowledge, this is the first work that utilizes search engines in this manner to detect phishing URLs.

4.3.3.5 Classification Models

We evaluate the following 9 classifiers implemented in WEKA (Waikato Environment for Knowledge Analysis), a popular data mining library (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) with their default parameter values:

1) Support Vector Machines (SVMs with rbf kernel) (Vapnik, 2000)
2) SVMs with linear kernel (SVM-linear)
3) AdaBoost (AB)
4) Multilayer Perceptron (MLP)

5) Bagging (Breiman L. , 1996)

6) Random Forests (RF) (Breiman L. , 2001)

7) Naïve Bayes (NB) (Rish, 2001)

8) Logistic Regression (LR) (Brannick)

9) C4.5 (Quinlan, 1993)

Note that C4.5 algorithm is implemented as J48 in WEKA.

4.4 Empirical Evaluation

Using the features described in Section 4.3.3, we encode each individual URL into a feature vector with 138 dimensions. We scale the real-valued features, available mostly in lexical-based features, to lie between 0 and 1. Scaling equalizes the range of the features in real-valued and binary features further emphasizing that we are treating each feature as equally informative and important.

We use 10 times 10-fold cross-validation (unless otherwise stated) to evaluate the classifiers. The experiments are run on a machine with 2 dual-core 2 GHz Intel processors and 4 GB memory.

4.4.1 Classifier Evaluation

In this experiment, we evaluate classification performance of 9 classifiers on all data sets using the whole feature set. Table 4-6, Table 4-7, Table 4-8, and Table 4-9 compare classifiers' classification performances on OldPhishTank-Yahoo, OldPhishTank-DMOZ, NewPhishTank-Yahoo, and NewPhishTank-DMOZ data sets, respectively.

The differences in overall error rates on the top 3 classifiers are not significant on each data set. Random Forests (RF) performs the best in all performance metrics followed by J48 and Bagging on each of four data sets. Naïve Bayes (NB) consistently performs the worst followed by AdaBoost (AB) and SVM-rbf on all data sets. Classifiers yield worst performance on NewPhishTank-Yahoo data set mostly due to high false negatives that range between 2.8–8.9%. RF's error rate ranges between 0.16–0.95%, whereas the NB's error rate ranges between 0.86–3.74%. AdaBoost (AB) yields 0% false positive rates on NewPhishTank-Yahoo and NewPhishTank-DMOZ data sets. J48 and Bagging yield 0% false positives on NewPhishTank-DMOZ data set. False positives rates are normally better than false negative rates for all classifiers on each of four data sets.

Classifier	**Error**	**FPR**	**FNR**	**Precision**	**Recall**	**F-Measure**	**ROC**	**MCC**	**BER**
RF	**0.31%**	**0.20%**	**0.53%**	**99.60%**	**99.47%**	**99.54%**	**0.999**	**0.993**	**0.37%**
J48	0.37%	0.23%	0.66%	99.56%	99.34%	99.45%	0.997	0.992	0.44%
Bagging	0.38%	0.23%	0.67%	99.56%	99.33%	99.44%	0.999	0.992	0.45%
MLP	0.40%	0.20%	0.79%	99.61%	99.21%	99.41%	0.998	0.991	0.50%
LR	0.43%	0.31%	0.69%	99.40%	99.31%	99.36%	0.998	0.990	0.50%
SVM-lin	0.44%	0.23%	0.85%	99.55%	99.15%	99.35%	0.995	0.990	0.54%
SVM-rbf	1.44%	1.07%	2.17%	97.91%	97.83%	97.87%	0.984	0.968	1.62%
AB	1.48%	1.35%	1.73%	97.38%	98.27%	97.83%	0.997	0.967	1.54%
NB	2.24%	2.24%	2.25%	95.72%	97.75%	96.72%	0.989	0.950	2.25%

Table 4-6: Comparison of classifiers' performance on OY data set.

Classifier	**Error**	**FPR**	**FNR**	**Precision**	**Recall**	**F-Measure**	**ROC**	**MCC**	**BER**
RF	**0.16%**	0.05%	**0.25%**	99.96%	**99.75%**	**99.85%**	**0.999**	**0.997**	**0.15%**
J48	0.19%	0.02%	0.33%	99.98%	99.67%	99.83%	0.998	0.996	0.17%
Bagging	0.18%	0.06%	0.28%	99.95%	99.72%	99.83%	0.999	0.996	0.17%
MLP	0.21%	0.04%	0.35%	99.96%	99.65%	99.81%	0.999	0.996	0.20%
LR	0.23%	0.09%	0.34%	99.92%	99.66%	99.79%	0.999	0.995	0.22%
SVM-lin	0.20%	0.06%	0.31%	99.95%	99.69%	99.82%	0.998	0.996	0.19%
SVM-rbf	0.60%	0.27%	0.89%	99.77%	99.11%	99.44%	0.994	0.988	0.58%
AB	0.48%	**0.00%**	0.89%	**100.00%**	99.11%	99.55%	0.998	0.990	0.45%
NB	1.01%	0.02%	1.86%	99.98%	98.14%	99.05%	0.998	0.980	0.94%

Table 4-7: Comparison of classifiers' performance on OD data set.

Classifier	Error	FPR	FNR	Precision	Recall	F-Measure	ROC	MCC	BER
RF	**0.95%**	**0.48%**	**2.86%**	98.04%	**97.14%**	**97.59%**	**0.995**	**0.970**	**1.67%**
J48	1.11%	0.45%	3.79%	**98.11%**	96.21%	97.15%	0.987	0.965	2.12%
Bagging	1.04%	0.50%	3.24%	97.96%	96.76%	97.35%	0.997	0.967	1.87%
MLP	1.44%	0.69%	4.49%	97.13%	95.51%	96.31%	0.993	0.954	2.59%
LR	1.42%	0.56%	4.89%	97.65%	95.11%	96.36%	0.991	0.955	2.73%
SVM-lin	1.32%	0.47%	4.78%	98.02%	95.22%	96.60%	0.974	0.958	2.63%
SVM-rbf	2.15%	0.73%	7.92%	96.86%	92.08%	94.41%	0.957	0.931	4.33%
AB	2.86%	1.36%	8.98%	94.29%	91.02%	92.62%	0.993	0.909	5.17%
NB	3.74%	3.62%	4.22%	86.67%	95.78%	91.00%	0.987	0.888	3.92%

Table 4-8: Comparison of classifiers' performance on NY data set.

Classifier	Error	FPR	FNR	Precision	Recall	F-Measure	ROC	MCC	BER
RF	0.23%	0.03%	0.59%	99.94%	99.41%	99.68%	**0.998**	0.995	0.31%
J48	0.25%	**0.00%**	0.68%	**100.00%**	99.32%	99.66%	0.996	0.995	0.34%
Bagging	**0.19%**	**0.00%**	0.53%	**100.00%**	**99.47%**	**99.73%**	**0.998**	**0.996**	**0.27%**
MLP	0.20%	**0.00%**	0.55%	**100.00%**	99.45%	99.72%	0.997	**0.996**	**0.27%**
LR	0.27%	0.06%	0.64%	99.89%	99.36%	99.62%	0.997	0.994	0.35%
SVM-lin	0.21%	0.01%	0.55%	99.98%	99.45%	99.72%	0.997	**0.996**	0.28%
SVM-rbf	0.66%	0.26%	1.36%	99.54%	98.64%	99.09%	0.992	0.986	0.81%
AB	0.85%	0.00%	2.36%	100.00%	97.64%	98.80%	0.997	0.982	1.18%
NB	0.85%	0.47%	1.54%	99.17%	98.46%	98.81%	0.997	0.981	1.00%

Table 4-9: Comparison of classifiers' performance on ND data set.

We choose RF classifier in the rest of the experiments primarily because its training and testing times are reasonably fast (see Section 4.4.3) with best overall classification results.

4.4.2 Feature Evaluation

In this experiment, we compare various combinations of feature sets to evaluate how effective each feature category is in detecting phishing URLs. Specifically, we compare individual feature category and combine it with the lexical-based feature category – the most commonly used feature category in phishing detection.

We use RF classifier on OldPhishTank-Yahoo data set as it has sufficiently good number of phishing and non-phishing URLs with varieties in URL structures covering most feature categories. Results on these experiments are displayed in Table 4-10.

When using lexical-based feature category alone, RF classifier achieves an error rate of 14.6%. Similarly, keyword based feature type, which has 101 features, yields a 15.4% error rate. When combined lexical with keyword-based features, the error rate improves to 9.3%. Reputation based feature set with 7 features provide the worst error rate of 15.7%. What is interesting about these feature sets is that each set provides better true negative rate, but worse true positive rate. This indicates that the absence of these features can detect non-phishing URLs with good accuracy, but their presence doesn't necessarily result in higher accuracy in detecting phishing URLs. When the first three feature categories are combined, the error rate significantly improves to 5.9%.

Feature Category	Feature Count	Error Rate	FPR	FNR
Lexical based	24	14.62%	8.22%	27.12%
Keyword based	101	15.39%	5.36%	35.01%
Lexical + Keyword based	125	9.25%	5.73%	16.14%
Reputation based	7	15.71%	2.69%	41.16%
Lexical + Reputation based	31	8.37%	4.74%	15.46%
Lexical + Keyword + Reputation	132	5.88%	3.37%	10.77%
Search Engine based	6	**0.47%**	**0.15%**	1.09%
Lexical + Search Engine based	30	0.37%	0.20%	0.69%
All Features	138	**0.31%**	0.20%	**0.53%**

Table 4-10: Comparison of various feature sets using RF classifier on OY data set.

Individually, search engine-based feature set alone provides the lowest error rate of 0.5%. It also provides the highest true positive and true negative rates. The experiment on the data set with all feature categories combined gives the best performance results across all the metrics. The combined features provide at best a 0.3% error rate. Search engine based features provide the lowest false positive rate of 0.2% indicating that very few non-phishing URLs are misclassified as phishing using this feature set alone.

These results demonstrate that search engine-based features are the most discriminative features in detecting phishing URLs. Combining all the features, however, provide the best accuracy confirming the importance of each feature category.

4.4.2.1 Feature Ranking and Selection

In order to find the importance of each individual feature, we calculate the F-score (Fisher score) value of all features. F-score is a simple but effective criterion to measure the discrimination between a feature and the label. Based on statistic characteristics, it is independent of the classifiers (Chen & Lin, 2003). Using the same notations explained in Chapter 1.3, we denote n_+ as the total number of positive instances and n_- as number of negative instances. The F-score of the *j*th feature is given by:

$$F(j) = \frac{(\bar{x}_j^{(+)} - \bar{x}_j)^2 + (\bar{x}_j^{(-)} - \bar{x}_j)^2}{\frac{1}{n_+ - 1}\sum_{i=1}^{n_+}(x_{i,j}^{(+)} - \bar{x}_j^{(+)})^2 + \frac{1}{n_- - 1}\sum_{i=1}^{n_-}(x_{i,j}^{(-)} - \bar{x}_j^{(-)})^2} \quad (4\text{-}2)$$

where $\bar{x}_j, \bar{x}_j^{(+1)}, \bar{x}_j^{(-1)}$ are the average of the *j*th feature of the whole, positive, and negative classes, respectively; $x_{i,j}^{(+)}$ is the *j*th feature of the *i*th positive instance, and $x_{i,j}^{(-)}$ is the *j*th feature of the *i*th negative instance. The numerator indicates the discrimination between the positive and negative classes, and the denominator is the sum of the variance within each class. A larger F-score value indicates that the feature is more discriminative. One known disadvantage of F-score is that it considers each feature separately and does not say anything about the mutual co-relation among features.

After ranking the features in descending order of F-score value, we ran LIBSVM (Chang & Lin, 2001) with RBF kernel using the backward elimination process. We used grid search to find the best parameter C and γ value and trained and tested the classifier using 10-fold cross-validation method. We achieved the least error rate using all features with parameters $C = 256$ and $\gamma = 0.00048828125$. Performance metrics of the best result are shown in Table 4-11.

Error	**FPR**	**FNR**	**Precision**	**Recall**	**F-Measure**	**ROC**	**MCC**	**BER**
0.44%	0.22%	0.88%	99.57%	99.12%	99.34%	0.994	0.99	0.55%

Table 4-11: Best performance metrics of SVM-rbf after parameters and feature selection.

For brevity, we list top 20 features based on F-score value in Table 4-12. We found that the new features we proposed in our methodology are ranked higher and in fact 80% of the top 20 features are the new features. Search engine based features are ranked the highest followed by reputation-based, and keyword-based features. There are 4 lexical based features as well in the top 20.

Not surprisingly, our hunch is experimentally confirmed that the search engine-based features are an accurate indicator of phishing URLs and search engines' indexes do act as a rudimentary white-lists. Looking at the statistics shown in Table 4-5, the large percentage of phishing URLs and domains not being in search engines results and a large percentage of legitimate URLs and domains being in search engine results as well as the highest F-scores of these features do confirm that these features are highly discriminative.

Reputation-based features are also discriminative as a large number of phishing URLs have those properties compared to a few legitimate URLs. Presence of digit (0-9) in URLs seems to be more relevant to the phishing URLs and the presence of URL in blacklists does provide a higher indication that the URL in fact is phishing. Presence of root terms '*Username* in URL', '*Password* in URL', 'Has Non-Standard Port', 'Number of Dots in Host', and '*lottery* in URL' are in the bottom 5 of the list.

Feature Description	F-score	Feature Description	F-score	Feature Description	F-score
Domain NOT in Google	14.453	URL NOT in Google	10.215	Domain NOT in Yahoo!	3.901
Domain NOT in Bing	3.450	URL NOT in Yahoo!	3.110	URL NOT in Bing	1.321
URL in Phishing Blacklist	0.487	Digit [0-9] in Host	0.179	Number of Dots in Path	0.138
IP in PhishTank Top 10	0.124	Top Target in URL	0.121	'log' in URL	0.113
PhishTank Top Domain in URL	0.106	Length of URL	0.087	Length of Path	0.083
'pay' in URL	0.079	'web' in URL	0.074	Number of Dots in URL	0.069
'cmd' in URL	0.065	'account' in URL	0.045		

Table 4-12: Top 20 ranked features based on F-score.

SVM-rbf with default parameters yields 1.6% BER on OY data set. However, SVM-rbf with parameter selection yields at best a balanced error rate of 0.6% while including all the features on the same data set. This experiment not only emphasizes the importance of parameter selection in SVM-rbf but also emphasizes the importance of all the features. Further, it shows that SVM-based feature selection method doesn't improve the performance of a classifier in our approach.

4.4.3 Time Analysis

Next we investigate the feasibility of real-time application of the proposed anti-phishing approach. In order to prevent Internet users from clicking on phishing URLs in real-time, such a proposed system needs to be highly accurate (very low false positive and negative rates) and needs to have tolerably low lag time. In this experiment, we analyze time taken by our prototype system in classifying whether a given URL feed is phishing by considering time taken from generating a feature vector to testing and getting the final verdict.

4.4.3.1 Feature Collection Time

We expect the system to take the most time in accessing the web and collecting the proposed search engine and some reputation based features. In our prototype experimental system, the single-threaded feature collector takes about 3.8 seconds in average to generate feature vector from a phishing URL and 3.2 seconds in average from a non-phishing URL. Thus, in general, it takes about 3.5 seconds to collect all the features and generate the feature vector in the format readable by the classifiers. The lower time taken for non-phishing URLs is because most non-phishing URLs are usually present in search engines results and as such the feature collector doesn't have to query search engines again for just the domain part of the URLs. We believe that we can significantly lower the time to collect features by employing parallel processing and local caching techniques that can access the web and gather various features simultaneously. We leave this as our future work when we build a robust, scalable system to test in the real world.

4.4.3.2 Training and Testing Times

To provide better comparisons of time taken by the classifiers to train and test the models and to find out the time taken to classify an instance of URL, we use 80/20 percentage split on OldPhishTank-Yahoo data set with all features for training and testing purposes respectively. We summarize the average results in Table 4-13 after running the experiments for 5 times. The cross-validation time is the time taken by classifiers in Experiment 1 on NewPhishTank-Yahoo data set.

Naïve Bayes is widely used in spam filters partly because the training and testing times are fast. We see similar result with NB taking the fastest 0.15 minutes to build the model. However, it also performs the worst in terms of error rate. MLP has the worst training time (in hours) and also worst testing time. We can observe that Random Forests (RF) has one of the best tradeoff between train and test time and overall accuracy. RF's training time is second best and comparatively very close to that of NB's, and its overall performance results are the best among all the classifiers (see Experiment 1).

Average Time	**Total Samples**	**RF**	**J48**	**Bagging**	**MLP**	**LR**	**SMV-lin**	**SVM-rbf**	**AB**	**NB**
Train	26860	1.05 m	2.28 m	4.66 m	1.69 h	0.56 m	1.65 m	1.56 m	1.63 m	**0.15** m
Test	6714	0.03 s	**0.003 s**	0.006 s	88.08 s	0.03 s	2.0 s	2.24 s	0.004 s	0.20 s
Test	1	0.021 ms	**0.003 ms**	0.01 ms	13.12 ms	0.03 ms	1.33 ms	2.01 ms	**0.003 ms**	0.19 ms
Cross-validation	27669	14.6 m	36.2 m	45.6 m	20.9 h	6.3 m	14.1 m	12.7 m	19.2 m	**1.7 m**

Table 4-13: Training and Testing time taken by all classifiers on OY data set using all features.

Though training time is generally higher compared to testing time, training may be done offline and less frequently. It's the test time that is crucial in providing the real-time service of detecting phishing URLs. Time taken to test a URL, once the model is trained, by most of the classifiers is in very low milliseconds and negligible compared to time taken to collect features. Overall, our single-threaded experimental system can classify a URL in less than 3.5 seconds in average. We understand that this time may not be acceptable in real-time classification system. Nonetheless, we believe that there's a lot of room for improvements during the design and implementation of the system for large-scale deployment and provide near real-time service. Furthermore, users may likely tradeoff some speed for its high accuracy when it comes to detecting potentially dangerous phishing URLs.

In these experiments, we demonstrate how a classifier's performance varies when using mismatched data sources and temporal-based data sets.

4.4.4 Training on One Data Set and Testing on Another

Features extracted by observing a particular data set can yield impressive low classification error rates when trained and tested on disjoint sets of the same data source using the right classifier. However, experiment results from study by Ma et al. (2009b) show that when training and testing sources are completely mismatched, a classifier's performance decreases significantly. To investigate if this phenomenon holds in our case, we experiment with training and testing on various combinations of phishing and non-phishing sources of URLs. There's one caveat, however, in our phishing URLs data sources. As we couldn't find the second credible source for phishing URLs, we generate two phishing data sets (separated by two whole months of collection time) from the same source PhishTank as described in Section 4.3.2. Even though it may not make a strong case for different sources, we argue that it does make a very strong case for investigating data drift (evolving features in phishing URLs, see Section 4.4.5). We use the abbreviations defined in Section 4.3.2 to refer to each combination of data sets, e.g., OY for Old PhishTank-Yahoo.

Table 4-14 shows classification results of training and testing on mismatched data sets using Random Forests (RF) classifier.

As expected, when trained and tested RF classifier with same data set, the overall error rates are normally better compared to when trained and tested with mismatched – possibly different sources – data sets (see the diagonal values in Table 4-14). When new phishing URLs are tested against the model trained from old phishing URLs, the error rate is 2.7% contributed mostly by false negatives. When only the non-phishing URL source is mismatched (e.g., OD and OY), error rate increases due to higher false positives. This observation is similar to the one observed by Ma et al. (2009a). However, when only the phishing URL data sets are mismatched (e.g., OD, ND), we do not see a big increase in error rates. The worst error rate in this category is 2.7% (OY and NY). This small increase in error rate may be due to the fact that the source of phishing URLs are not technically different but they differ by the time when phishing URLs were submitted for verification. We look into this in detail in the next section. When classifier is trained with combination of any phishing data sets plus DMOZ (OD, ND), and tested with combinations of phishing data sets and Yahoo data set (OY, NY), the error ranges between 10–13%, contributed mostly by high false positives. When trained with URLs from all the data sets, the model generalizes well and yields good performance results across all test data sets (last row). This experiment concludes that the training data set needs to be selected properly that represent the actual test environment in the real world in order to achieve the best performance from a classifier.

Though instances in data sets may be different, we can compare these results with those in Ma et al. (2009a) in cases where the data sources are similar. Their data sets include 5,500 phishing URLs from PhishTank and all of their benign URLs come from Yahoo and DMOZ. When trained and tested with the various combination of Yahoo, DMOZ, and PhishTank data sources, our method achieves errors in the range 0.06–13.3%, while their approach report errors in the range 1.2–33.5% on the same data sources. They achieve at best 0.9% on the split of the Yahoo-Spamscatter data source showing that the approach is better at detecting spam URLs than at detecting phishing URLs. We argue that our approach achieves qualitatively superior performance in detecting phishing URLs.

Training	**Testing (Error rate)**			
	OY	**OD**	**NY**	**ND**
OY	0.31%	0.08%	2.69%	4.96%
OD	10.92%	0.09%	13.39%	0.33%
NY	0.47%	0.79%	0.87%	0.21%
ND	10.45%	0.27%	12.54%	0.33%
All data sets (OYND)	0.33%	0.06%	0.10%	0.16%

Table 4-14: Overall error rates when training on one data set and testing on another (possibly different source) using RF classifier.

4.4.5 Concept Drift

Phishing tactics and URL structures keep changing continuously over time as attackers come up with novel ways to circumvent the existing filters. As phishing URLs evolve over time, so must the classifier’s trained model to improve its performance. Ma et al. (2009b) conclude that retraining algorithms continuously with new features is crucial for adapting successfully to the ever evolving malicious URLs and their features. An interesting research work would be to find the effect of variable number of features using online algorithms in detecting phishing URLs. We leave this for our future work. By comparing all the classifiers, we want to find out which classifier is most robust to data drift.

We use OldPhisTank data set and 22,722 (twice the number of phishing URLs) randomly selected non-phishing URLs from Yahoo and DMOZ data sets as our "base" training set. Figure 4-3 shows the classification error rates for classifiers after training them only once using the "base" training set. The *x*-axis shows number of weeks in the experiment with the phishing URLs collected from January 1st week to May 1st week of 2011, and the *y*-axis shows the error rates on testing the classifier with phishing URLs collected each week. For non-phishing URLs, to generate each week's test data, twice the number of phishing URLs is randomly selected from Yahoo and DMOZ data sets.

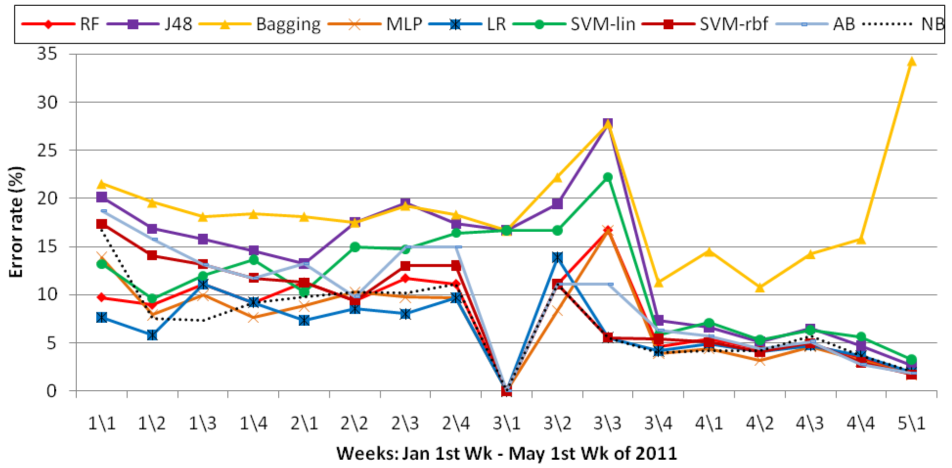

Figure 4-3: Error rates for all classifiers after training them once and testing them on weekly data.

The error rates reach as high as 28% for J48 and Bagging when testing with data from 3rd week of March (3\3). While most classifiers perform poorly in this experiment, Bagging performs the worst across all the weeks with error rate reaching as high as 34% for the last week. Random Forest (RF) and Logistic Regression (LR), in general, perform better with error ranging from 0–13%. For all the classifiers, the high error rate is due to high false negatives which get as high as 60–80% for some weeks. For week 3\1 data, most classifiers yield a 0% error rate as there are only 2 phishing URLs and, therefore, 4 non-phishing URLs for the week's test data. These high error rates due to high false negative rates suggest that, to achieve better accuracy, the model must be retrained on fresh data to account for new combinations of features on phishing and non-phishing URLs over time.

To address this issue, we retrain classifiers every week with training data collected up to that week and test on the data from the following week. We show these results in Figure 4-4. To test the models for data collected on January 1st week (1\1), for example, we train classifiers using "base" training set. To test data collected on week 1\2, we retrain all the classifiers with "base" training set plus all data collected up to the previous week (1\1 in this case).

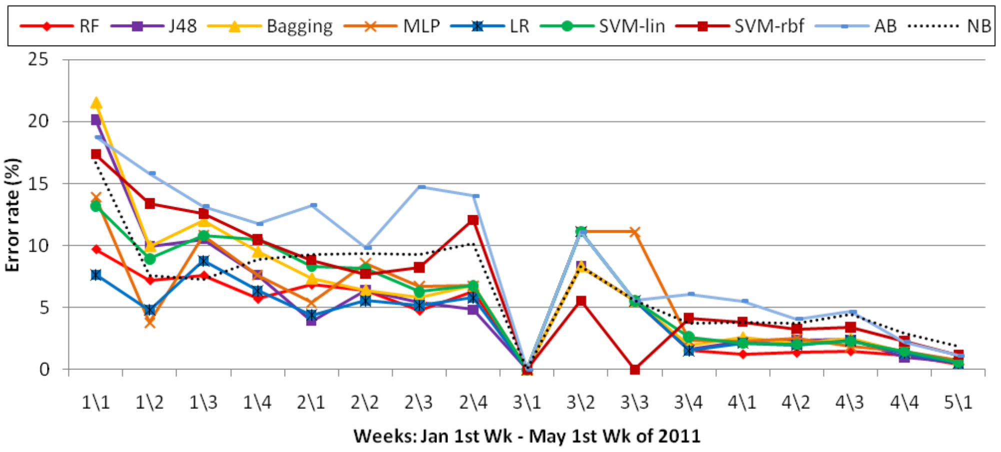

Figure 4-4: Error rates for all classifiers after retraining them every week.

As a result, the error rates decrease over time due to significant improve in false negative rates week after week. For RF classifier, error rate decreases from 9% on the first week to 0.4% for the last week. Interestingly, we see the most improvement in performance of Bagging. Its error rate starts with the highest 21% for the first week and it gradually improves to 0.8% on the last week. Although fresh data eventually helps most classifiers improve over previous experiment where the classifiers are trained only once, we feel that a week's training data is still insufficient. By looking at the trends in our experiments and as shown in Ma et al. (2009b), training on daily data or perhaps using incremental training with individual instance using online algorithms may improve the results on classification of continuously changing phishing URLs. But due to lack of enough data for a lot of individual days in our data sets, we leave these experiments for future work when we'll collect phishing URLs from several feeds on a large scale.

4.4.6 Detecting Top Targets' URLs

In our OldPhishTank-Yahoo (OY) data set (see Section 4.3.2), we include legitimate URLs of top online brands that are frequently targeted by phishers. Since our goal is to detect phishing URLs that forge legitimate URLs, we try to investigate how our proposed method performs against the URLs from legitimate top targets. To that end, we combine all of the data sets (OYND) except the top targets' URLs and use the combined 46,992 instances as a training data set. We use the rest 1,674 instances as a test data set. When trained and tested Random Forests (RF) with these data sets, it yields a 19.4% error rate, meaning only 80.6% of legitimate URLs related to the top targets are correctly classified as non-phishing URLs and the rest are all misclassified as phishing URLs. This big error rate is due to the fact that these target URLs look very similar to the forged phishing URLs, and the model, perhaps, is not trained with the types of URLs that are seen during testing.

In order to address this, we randomly select 75% of legitimate URLs from top targets and include them in the training set and retrain the model. We test the newly trained model with the remaining 418 instances of top target URLs. As expected, the results improve significantly, yielding a 1.7% error rate, misclassifying only 7 non-phishing URLs as phishing. This experiment further emphasizes the importance of judicious selection of training data set with proper representation of all the possible phishing and non-phishing URLs the system will be actually tested against.

Unlike related works (Garera, Provos, Chew, & Rubin, 2007), (Ma, Saul, Safage, & Voelker, 2009a; 2009b), (Whittaker, Ryner, & Nazif, 2010), we explicitly test the validity of our method on the actual phishing targets and show that our method yields good accuracy results in detecting not only the phishing URLs but also in detecting the legitimate URLs of the actual targets.

4.5 Discussion

4.5.1 Limitation

Despite offering low error rates, there are some limitations to our study. First, all of our phishing URLs came from a single source PhishTank; therefore the URLs received may not be representative of all phishing URLs. However, PhishTank not only automatically collects potential phishing URLs from users' email applications, it also allows registered users to manually submit a URL (perhaps received from various attack vectors besides email) for verification.

Search engine-based features contribute to a major performance bottleneck due to the time lag involved in querying search engines. Also the service providers may either permanently block our system citing potential denial of service (DoS) attack originating from our system or temporarily limit the access to their services to manage throttle. Using local caching, our system must manage not to flood search engines with requests.

As our reputation based features rely on blacklists and other historical statistics provided by third parties, we have no control over the quality and reliability of the services and data provided by them. Though absence of these features may not significantly hurt the performance of our approach, we certainly can remove them or look for alternative sources.

PyLongURL script (Basnet, 2010b) couldn't automatically expand some short phishing URLs mostly because the legitimate shortening services blocked and removed them after receiving reports of phishing attacks. Even though these short URLs lack all the lexical and keyword based features, most of them are correctly classified because of the search engines' and reputation-based features. This is very less likely to happen on legitimate URLs shortened using legitimate shortening services as the script will be able to expand them to their final URLs.

4.5.2 Error Analysis

Examining URLs contributing to false positives and negatives can further reveal limitations of the approach and possibly provide potential improvements to the current approach. Some internal links that we harvested from legitimate highly targeted websites have properties similar to phishing URLs, for examples: *http://www.standardbankbd.com/pages_details.php?id=35&phpsessid=39327de0de6269760dc6a1f3fb630*, *http://moneysense.natwest.com/natwest/adults/makingbankingsimple/loans.asp?page=MONEYSENSE/ADULTS/MAKING_BANKING_SIMPLE/LOANS*, etc. Perhaps newly or dynamically generated links or due to the way search engines index URLs, the search engines didn't have these links indexed. Because some target names and some keywords such as login, signoff, etc., are present in them and they are usually lengthy as seen in examples, these non-phishing URLs resemble more like phishing URLs. However, this is as much an issue with the selection of training data as it is with the methodology. Perhaps a second and much shorter work could mitigate these false positives using enhanced benign URL harvesting methods. Similarly, our method may flag new legitimate webpages as phishing in particularly those that have not yet been crawled and indexed by search engines.

Some URLs such as *http://whoblocksyou.net*, *http://www.checkmessenger3.net/en/*, *http://you-areblocked.com*, *http://www.yahooblockchecker.info/*, etc. and their variants are classified as non-phishing. This group of webpages promises users to find if any contacts in MSN, Yahoo or any common instant messenger have blocked you once you provide your email and password. These websites seem to be around for a while, some from as far back as 2007. Even though they have been confirmed as phishing sites by the PhishTank community long back, interestingly, the sites still exist. Moreover, Chrome, Firefox, and Internet Explorer didn't block these websites (as of August 10, 2011) implying that these links were not in their blacklists.

Other group of webpages such as *http://home.comcast.net/~mikeskipper/*, *http://habbomatanza.galeon.com/*, etc. hosted on free legitimate hosting services contributed to more false negatives. These URLs do not have any red flag keywords and their domain and in some instances even the whole URLs were in search results and were not in browsers blacklists.

We believe that by looking into the page contents of URLs, these false negatives can be mitigated.

4.5.3 Tuning False Positives & Negatives

In case of detecting phishing URLs, false positives may be tolerated more than false negatives or vice versa. With false positive URLs, users have to be extra vigilant while loading the URL and manually confirm it before submitting any sensitive personal information. False negatives, on the other hand, may provide false sense of security and users may end up giving up their personal information to a forged webpage. Instead of minimizing the overall error rate, for policy reasons or personal security preferences, users may want to tune the decision threshold to minimize the false negatives at the expense of more false positives or vice versa.

Figure 4-5 shows the tradeoff between false negative and false positive rates as an ROC graph for Random Forests over an instance of whole data set (ONYD) using all the features. The highlight on the figure shows that if the false positives are tuned to 0.2%, the model achieves a false negative rate of 3.2%. If we can tolerate a little higher false positive rate to 0.4%, however, we can achieve a lower false negative rate of 1.1%.

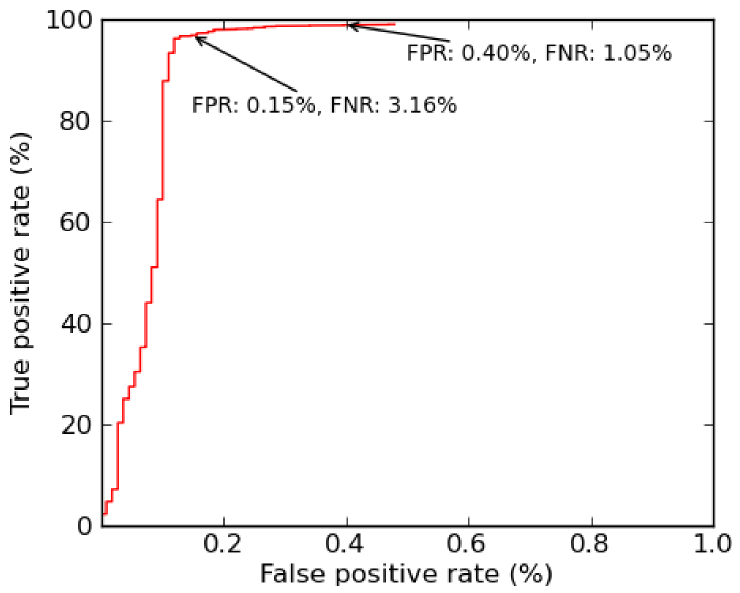

Figure 4-5: ROC graph showing tradeoff between false negatives and false positives. Note the x-axis ranges between 0-1%.

4.5.4 Potential Adversarial Attacks

As search engine based features are highly discriminative, attackers may try to launch distributed DoS attack on search engines. It is not very likely to happen on all three of them simultaneously, however. Though quality and index size of a search engine plays a big role in our approach to correctly classify phishing and legitimate URLs, we can look for an alternative as there are plenty to choose from. Using Blackhat SEO techniques, adversaries can get their phishing websites crawled in a short period of time. Though page ranking is not an issue, our technique may provide a large number of false negatives if phishing links are simply found in search engines' indexes. Adversaries may buy established domain names or establish a new website hosting benign contents for a while. Once the major search engines index the website, they may exploit this fact and start hosting phishing webpages effectively avoiding search engine based features. Consequently, buying a domain name, hosting a website for a long time, employing Blackhat SEO to alter the PageRank of a phishing page require significant investment, which reduces the potential profit from the phishing campaign.

Phishers may try to evade our statistical modeling approach by using new URL shortening services that we are either not aware of or do not know how to utilize the service automatically. Study shows that scammers are now establishing their own fake URL-shortening services (Microsoft Security Intelligence Report, 2011). Under this scheme, shortened links created on these fake URL-shortening services are further shortened by legitimate URL-shortening sites. These links are then distributed via phishing emails, blogs, micro-blogs, and social networking websites. Though we didn't observe it in our data sets, we anticipate this tactic to be used against our proposed system and see a need to address it in our future work. We can always enhance the capabilities of our URL expanding script by actively looking out for new shortening services and incorporating them into our script. Moreover, in order to successfully evade our approach, phishers not only have to use a rare shortening service, but also have to work hard to get those short links indexed by search engines.

Adversaries may try to reduce the information content in the lexical and keyword based features of URLs thus effectively reducing the phishing like features and making their links more legitimate. Another approach is to leverage well-known infrastructure such as hosting phishing page on a legitimate popular domain such as free webhosting services or by breaking into legitimate websites or by exploiting common Cross-site Scripting (XSS) vulnerabilities (Cross-site Scripting - XSS). We can overcome this drawback by looking into the contents of the webpages. This drawback, however, is not particular to our approach, but to all the approaches that rely only on the URL metadata and structures to detect potential maliciousness.

To make reputation based features less suspicious, phishers may try to host their contents in domains and IPs that do not have historically bad reputation of hosting malicious websites. Sites with good reputation, however, are either too difficult to exploit or their administrators typically remove malicious pages under their control promptly, thus limiting the potential audience and profitability of phishing campaign hosted in their web servers.

Furthermore, since we provide equal weight to all the features, phishers do not have an opportunity to target higher weight features in order to invade our classifiers.

4.6 Conclusions and Future Work

In this chapter, we proposed new search engines, reputation, and statistically mined keyword based features for classifying phishing URLs. We empirically demonstrated that the proposed features are highly relevant to the automatic discovery and classification of phishing URLs. We evaluated our approach on real-world public data sets by comparing performance results of several popular supervised learning methods. Experimental results showed that the proposed anti-phishing solution is able to detect phishing URLs with an accuracy of more than

99.5% while maintaining false positive and false negative rates of less than 0.5%. We showed that our experimental prototype, once trained, could classify a given URL as phishing or non-phishing with a turnaround time of about 3.5 seconds.

Most classifiers except Naïve Bayes showed statistically similar performance metrics in our approach. Random Forest (RF) classifier, however, provided the best tradeoff between the classification performance and the training and testing time. RF consistently outperformed all other classifiers in most experiments. We have shown that the steps of selecting representative training set and retraining algorithms continuously with fresh data are crucial aspects in adapting successfully to the ever-evolving URLs and their features.

As future work, we hope to improve the feature collection time by employing parallel processing and local caching. We will also investigate the effectiveness of online algorithms as they have been found to outperform traditional batch algorithms in problem similar to ours (Ma et al., 2009b).

Chapter 5

Detecting Phishing Webpages

5.1 Introduction

Whittaker et al. (2010) define a phishing webpage as "any webpage that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewers would only trust a true agent of a the third party." In this study, we use the same definition of phishing and propose a novel method to detect such webpages using features extracted from URL and webpage contents. The hypothesis is that adding content-based features achieves better false positive and false negative rates achieved by using URL-based features alone.

5.2 Method Overview

Our method for detecting phishing webpages is similar to detecting phishing URLs (see Chapter 4). In this approach, besides using URL-based features, we also look into the contents of phishing webpages and extract a number of features locally and from online sources. Figure 5-1 shows the graphical representation of our phishing webpage detection methodology.

5.2.1 Data Sets

To run our experiments, we use the same data sources described in Chapter 4. Combining these phishing and non-phishing data sources, we generate our 5 data sets which are summarized in Table 5-1.

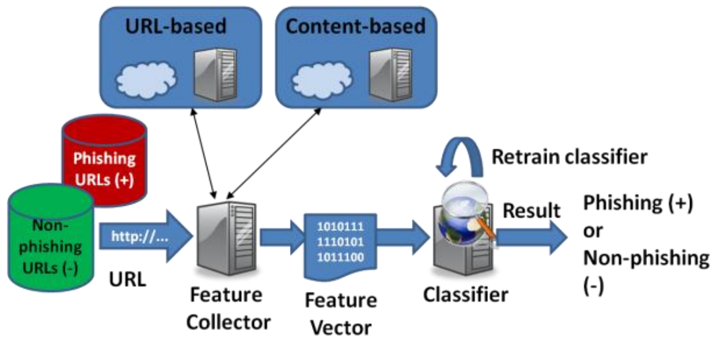

Figure 5-1: Graphical representation of our phishing webpage detection methodology.

Features were collected as soon as a webpage was crawled and successfully downloaded. We discarded the URLs that were no longer valid as the page couldn't be accessed to extract features from their contents. As a result, we have slightly less number of phishing webpages compared to phishing URLs used in Chapter 4.

Data Set	# Phishing Webpages	# Non-phishing Webpages	Total
OldPhishTank-Yahoo (OY)	11,240	21,946	33,186
NewPhishTank-DMOZ (ND)	5,454	9,635	15,089
OldPhishTank-DMOZ (OD)	11,240	9,635	20,875
NewPhishTank-Yahoo (NY)	5,454	21,946	27,400
All data sets (OYND)	16,694	31,581	48,275

Table 5-1: Summary of data sets.

5.2.2 Feature Analysis

In this study, we propose several new server-based and content-based features. The use of relatively small number of fixed set of features makes the decision boundaries less complex, and therefore less prone to over-fitting as well as faster to evaluate.

We group features that we gather into 2 broad categories: URL-based and content-based.

5.2.2.1 URL-Based Features

Our URL-based features are the same ones used in Chapter 4. In reputation-based feature category, we added one new feature based on hpHosts. We use hpHosts to check if the domain of a URL exists in its database. hpHosts is a community managed and maintained hosts file that allows an additional layer of protection against access to ad, tracking and malicious websites (hpHosts, 2005). This is a newly added feature which has 2.57% prominence among phishing URLs and 0.02% among non-phishing URLs.

5.2.2.2 Content-Based Features

URL-based features have shown promising results in classifying phishing URLs (see Chapter 4). Can the content-based features help to improve accuracy in phishing webpage detection? We try to answer this question in this Chapter.

Content-based features have been used in (Whittaker, Ryner, & Nazif, 2010) and in (Zhang, Hong, & Cranor, 2007). Augmenting the features based on a URL, we propose several new features based on the technical (HTML) – as opposed to text-based features used in related works – contents of a phishing webpage. Besides page contents, our web crawlers also download server header information and extract certain features which, we think, can help classifiers in better discriminating phishing webpages from non-phishing ones. Nevertheless, we propose a few novel and discriminative text-based features.

We briefly describe our content-based features next. Each feature and its statistics are extracted from 80% of randomly selected training data (we call it *Training Set*) from the OldPhishTank-Yahoo data set.

TITLE tag: Most webpages (phishing or non-phishing) usually contain a TITLE tag. Essentially, the TITLE tag's text is what is displayed as text for links returned from major search engines (Google, Yahoo and Bing). Zhang et al. select the top 5 words with highest TF-IDF value to generate lexical signature of a page

(Zhang, Hong, & Cranor, 2007). They feed each lexical signature to Google search engine and check if the domain name of the current webpage matches the domain name of the top 30 results. If yes, they consider it to be a legitimate website. We use webpage's TITLE text as the signature of the page and search Google using the title of the page. We then check if the whole URL and domain part of the current webpage matches the URL and domain name of the top 30 results. If yes, we consider it to be a legitimate webpage. We also check if any of the top targets is present in the title. We stick to top 30 results because Zhang et al. showed that going beyond top 30 didn't improve the classification performance. In our *Training Set*, we observed that the whopping 98.6% of title search from phishing webpages didn't result in a matching URL while 55.2% from non-phishing webpages didn't result in a matching URL. Similarly, 95.8% of title search from phishing webpages didn't result in matching domain and 31.34% of non-phishing webpages didn't result in matching domain.

FORM tags: Most phishing webpages have a FORM tag for potential victims to submit their personal information. In fact 99.3% of phishing webpages had one or more FORM tag(s) in them. Phishers tend to bypass the hassle of setting up SSL/TLS and simply follow easy and insecure method to send the form data back to them. Using 'get' method to send form data is insecure way to send password or any sensitive data as the data are encoded by the browser into the URL and which in turn is logged by HTTP(S) server in plain text besides most likely being stored in client's browser history (Korpela, 2003). If a legitimate website is hacked, phishers can modify the form's target to send data to different domain that is under their control. Since, online account password is one of the most common confidential information phishers are after, we check whether a page contains the keyword *password* as an input type attribute inside a form tag.

In the *Training Set*, we observed that 43.8% of phishing pages and 5.1% of the legitimate webpages had a *password* field in them. Furthermore, 15.0% of phishing and 2.1% of legitimate webpages were transmitting their form contents securely using HTTPS protocol. We didn't analyze the security certificate issuing vendor or its encryption strength, however.

Surprisingly, we found 0.9% of non-phishing webpages used HTTP protocol to transmit form data with *password* field. More surprisingly, only 28.4% out of 42.4% of phishing webpages transmitted form data with a *password* field without using encryption as expected because it takes money and time to buy and setup SSL/TLS certificate besides going through some security scrutiny.

Furthermore, 4.1% of phishing webpages transmitted their form data with *password* input using 'get' method. We also observed that 27.4% of phishing webpages contained FORM tag with action destination other than the page URL's domain.

Server and client redirection: Cyber criminals may compromise a legitimate Web server and redirect the traffic to a malicious Web site using HTTP redirect or by Pharming. Pharming is a type of attack which is conducted either by changing the hosts file on a victim's computer or by exploitation of vulnerability in DNS server software (Stamm & Raman, 2006). Phishers may poison DNS entries for a target website on a given DNS server, replacing them with the IP address of a server they control. In order to circumvent anti-phishing technologies, phishers may use META tag's refresh property to make the client's browser automatically load a phishing page that scammer controls. Moreover, phishers may use obfuscated JavaScript to dynamically generate the HTML page with just a META refresh tag with the URL of a phishing webpage as target (Basnet, 2010a).

In the *Training Set*, we observed that 16.6% of all the phishing webpages exploited the META refresh tag while only 4% of legitimate webpages contained this tag. Of all the META tag redirections from phishing webpages, 7.3% were external redirection; meaning the target URL had non-matching domain from the current webpage's domain. 1.7% of the META redirections in legitimate webpages were to external domain. In order to evade detection, phishers may setup a large number of webpages to redirect traffic from one page to another before taking potential victims to the final phishing page. Victims, as in other web based malware attacks, e.g., drive-by-download attacks (Cova, Kruegel, & Vigna, 2010), are often sent through a long chain of redirection operations making it more difficult to track down an attack, notify all the involved parties (e.g., registrars and providers), and ultimately take down the offending sites. We anticipated this tactic and investigated it in our data sets. We record the number of times the browser is redirected, for example, by the server or using META refresh tag. "Number and target of redirections" is one of the 10 features used in detecting drive-by-download attacks (Cova, Kruegel, & Vigna, 2010). We found that 0.3% of legitimate webpages led to 2 or more META redirections while 1.2% of phishing webpages had exploited this feature. Since the META refresh tag's URL is the final destination of phishing scam, our web crawlers download the destination URL and check it against the Google's blacklists. In our data set, 4.9% of destinations URLs from phishing webpages were present in the blacklists while 0.04% of destinations URLs from legitimate webpages were present in the same blacklists. Similarly, we observed that 7.7% of the phishing webpages were redirected by their servers and no legitimate page had server redirection.

Other tags: Due to Cross Site Scripting vulnerability (XSS) (Cross-site Scripting - XSS) or other vulnerabilities in a website, hackers may be able to inject IFRAME tag and force victim's browser to unknowingly load a malicious webpage inside an otherwise legitimate page. We also count anchor tags for both internal (within the same domain) and external (to other domain) links. The idea is that phishers, for convenience, usually copy just one page (normally a login page) of the target website and simply leave all other objects (such as, CSS, scripts, images, etc.) pointing back to the target website. We expected to have more external links than internal links in phishing Webpages.

In the *Training Set*, 23.5% of phishing and 13.8% of legitimate webpages had 4 or more external image sources. Similarly, 56.6% of phishing and 46.6% legitimate webpages had 5 or more external links. Though IFRAMEs are the most prevalent type of attack that use HTML and JavaScript (Messmer, 2008), (Microsoft Security Intelligence Report, 2011), our data set contained nearly the same percentage of IFRAME tags in both phishing and legitimate webpages. 7.7% of phishing and 6.9% of non-phishing webpages had 2 or more IFRAME tags in them.

Furthermore, we check IFRAME's source URLs against the Google's Blacklists and also check if they are indexed by Google search engine by querying search engine with the complete URL and checking if any of resulting links match with the searched URL. 15.5% of IFRAME source URLs from of phishing webpages didn't result in matching results while 14.4% of non-phishing webpages didn't result in the matching results. Most of the IFRAME tags in non-phishing webpages were for displaying banner ads while most of the IFRAME tags in phishing webpages were for displaying part of the webpage perhaps to invade filters. Interestingly, less than 0.01% of IFRAME source URLs from both phishing and non-phishing webpages were in Google's blacklist.

Page structure: Most browsers are forgiving and usually have high tolerance for missing tags or non-standard and invalid markups. Even though phishing webpages need to be rendered well and must resemble very close to their targets visually, we anticipated that sophisticated phishers would exploit this browser feature. In order to break HTML parser and evade detection, phishers may deliberately churn poorly structured obfuscated HTML codes. We modified and used the Beautiful Soup parser (Richardson) to parse HTML content and extract content-based features. One of the most powerful features of Beautiful Soup is that it won't choke on bad markup. It parses a (possibly invalid) XML or HTML document into a tree representation.

Feature Description	**% Phishing webpages**	**% Non-phishing webpages**
Title search results NOT containing webpage's URL	98.56%	55.17%
Title search results don't contain webpage's domain	95.71%	31.34%
Malformed HTML page	27.59%	21.65%
Presence of *Password* field	43.79%	5.11%
Contains PhishTank top target in page	18.72%	10.31%
Contains PhishTank top target in page	64.62%	72.75%
Has external META redirection	7.26%	1.67%
Has META redirection	16.64%	4.02%
META redirected URL in Google malware blacklist	0.04%	0.00%
META redirected URL in Google phishing blacklist	4.80%	0.00%
META redirected URL in Google Regtest blacklist	0.02%	0.00%
META redirected URL in Google RegTest list	0.02%	0.00%
Has Title text	79.34%	93.31%
Non-ASCII character in Title	5.28%	4.17%
Page contains text	94.25%	98.95%
Has chain of 3 or more META redirection	0.00%	0.22%
Page is server redirected	7.69%	0.00%
Server redirected URL in Google phishing blacklist	1.06%	0.00%
Server redirected URL in Google RegTest list	0.01%	0.00%

Table 5-2: Content-based binary features and their statistics on *Training set* OldPhishTank-Yahoo data set.

We observed that our parser failed to parse 28% of phishing webpages and 22% of legitimate webpages. Surprisingly, relatively a large percentage of developers of legitimate websites were not following industry-standard best programming practices. We used regular expressions to collect features from webpages that the parser was unable to parse.

Table 5-2 summarizes the statistics on binary-valued content based features and Table 5-3 summarizes the real-valued content based features on 80% *Training set* from OldPhishTank-Yahoo data set.

Using these features, we encode each webpage into a feature vector with 177 dimensions. Most of the lexical-based and all of the keyword-based features are generated by the "bag-of-words" representation – the order of the word as they

Feature Description	Webpage Type	Max	Min	Mean	Median
Number of short URLs	Phishing	16	0	0.01	0
	Non-phishing	34	0	0.05	0
Number of external image sources	Phishing	296	0	4.98	0
	Non-phishing	877	0	5.79	0
Number of IFrame tags	Phishing	16	0	0.30	0
	Non-phishing	43	0	0.37	0
Number of external links	Phishing	369	0	20.49	9
	Non-phishing	3136	0	21.52	7
Number of IP based URL	Phishing	103	0	0.37	0
	Non-phishing	166	0	0.63	0
Number of URL with ‘username’	Phishing	79	0	0.32	0
	Non-phishing	118	0	0.63	0
Number of URLs with redirection	Phishing	347	0	0.11	0
	Non-phishing	218	0	0.45	0
Number of URL with non-standard port	Phishing	2	0	0.00	0
	Non-phishing	19	0	0.01	0
Number of URLs with path containing ‘login’	Phishing	149	0	1.33	0
	Non-phishing	180	0	0.34	0
Length of META redirection chain	Phishing	4	0	0.13	0
	Non-phishing	3	0	0.04	0
Secure FORM count	Phishing	5	0	0.20	0
	Non-phishing	103	0	0.09	0
Number of FORM actions with external domain	Phishing	11	0	0.45	0
	Non-phishing	103	0	0.24	0
Number of FORMs with ‘GET’ method	Phishing	4	0	0.13	0
	Non-phishing	21	0	0.31	0
Number of FORMs using HTTP	Phishing	26	0	0.13	1
	Non-phishing	64	0	0.79	0
Number of FORM tags	Phishing	27	0	1.07	1
	Non-phishing	106	0	1.00	1
Number of FORMs with ‘POST’ method	Phishing	14	0	0.70	0
	Non-phishing	103	0	0.36	0
Number of internal links	Phishing	1955	0	9.51	3
	Non-phishing	2720	0	66.81	0
Number of IFrame sources NOT in Google top result	Phishing	13	0	0.22	0
	Non-phishing	41	0	0.25	0
Number of IFrame sources in Google blacklists	Phishing	1	0	0.00	0
	Non-phishing	1	0	0.00	0

Table 5-3: Content-based real valued features and their statistics on *Training set* of OldPhishTank-Yahoo data set.

appear on the URL is not taken into consideration. All real-valued features are scaled to fall between 0 and 1, thus giving equal weight to each feature.

5.2.3 Machine Learning Algorithms

In this study, we evaluate a set of 8 batch learning algorithms: Support Vector Machines (SVMs) (Vapnik, 2000), AdaBoost (Freund & Schapire, 1996) with DecisionStump (Ibb & Langley, 1992), LogitBoost (LB) (Friedman, Hastie, & Tibshirani, 2000) with DecisionStump, Bagging (Breiman, 1996) with REPTree, Random Forests (Breiman, 2001), Naïve Bayes (Rish, 2001), C4.5 (Quinlan, 1993), Logistic Regression (LR) (Cessie & Houwelingen, 1992) and a set of 5 online learning algorithms: updatable version of Naïve Bayes (NB-U), updatable version of LogitBoost (LB-U), Perceptron (Rosenblatt, 1958), Passive-Aggressive (PA) (Crammer, Dekel, Keshet, Shalev-Shwartz, & Singer, 2006), and Confidence-Weighted (CW) (Dredze, Crammer, & Pereira, 2008) algorithms.

Note that C4.5 is implemented as J48 in WEKA environment.

5.3 Empirical Evaluation

We use 10 times 10-fold cross-validation (unless otherwise stated) to estimate the test error and report a variety of evaluation metrics to make it easier to compare our results with the existing literature. The experiments were run on a machine with 2 dual-core 2 GHz Intel processors with 4 GB memory. To conduct all the experiments, we used WEKA (Waikato Environment for Knowledge Analysis) (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) and CW libraries (Dredze, Crammer, & Pereira, 2008).

5.3.1 Classifier Evaluation

In these experiments, we evaluate all the classifiers on each of our data sets with full features that we discuss in section 5.2.2. Table 5-4 summarizes the performance results of all the classifiers on four data sets (OY, OD, NY, and ND).

The non-shaded rows are performance results from batch learning algorithms while the shaded rows are results from the online learning algorithms.

The differences in classification results from all the classifiers are statistically insignificant. The best value for each performance metric is boldfaced. Among all the classifiers, Naïve Bayes performed the worst followed by AdaBoost and LogitBoost. Overall, Random Forests (RF) performed the best followed by C4.5 and Bagging. In Table 5-5, we show various performance metrics comparing batch and online learning algorithms on the combined data set (OYND). Among the online algorithms, overall, Passive Aggressive (PA) algorithm yields the best performance results followed by updatable version of LogitBoost (LB-U). When comparing between batch and online learning groups, most batch algorithms yield superior classification results though the differences are not that significant.

Data Set	**OldPhish-Yahoo**			**OldPhish-DMOZ**			**NewPhish-Yahoo**			**NewPhish-DMOZ**		
Classifier	**Error (%)**	**FPR (%)**	**FNR (%)**	**Error (%)**	**FPR (%)**	**FNR (%)**	**Error (%)**	**FPR (%)**	**FNR (%)**	**Error (%)**	**FPR (%)**	**FNR (%)**
RF	0.42	0.26	0.73	**0.04**	0.01	**0.06**	**0.15**	**0.07**	**0.44**	0.23	**0.00**	0.62
C4.5	**0.32**	0.18	**0.58**	0.06	0.02	0.10	0.35	0.16	1.10	0.25	0.01	0.68
Bagging	0.35	**0.15**	0.73	0.05	0.01	0.09	0.19	0.12	0.48	**0.19**	**0.00**	**0.53**
SVM-lin	0.42	0.21	0.81	0.17	0.04	0.28	0.96	0.34	3.45	**0.19**	**0.00**	**0.53**
LR	0.47	0.30	0.81	0.26	0.13	0.37	0.72	0.29	2.44	0.44	0.31	0.66
SVM-rbf	0.86	0.52	1.54	0.35	**0.00**	0.65	1.48	0.48	5.50	2.74	0.65	6.42
LB	1.04	0.69	1.71	0.18	0.03	0.30	1.17	0.65	3.26	0.36	**0.00**	0.99
AB	1.34	1.18	1.67	0.23	**0.00**	0.44	1.68	0.65	5.85	0.69	**0.00**	1.91
NB	2.20	2.22	2.17	1.02	0.22	1.70	4.15	4.44	2.95	0.99	0.47	1.91
PA	**0.57**	**0.37**	0.96	0.17	0.11	**0.21**	1.22	0.48	4.18	0.25	0.04	0.62
LB-U	0.88	0.58	1.48	**0.15**	**0.06**	0.23	**0.34**	**0.14**	**1.14**	0.43	0.05	1.10
Perceptron	0.68	0.59	0.85	0.22	0.22	**0.21**	1.87	0.80	6.18	0.48	0.45	**0.53**
CW	**0.57**	0.46	**0.78**	0.24	0.26	0.22	1.86	1.33	4.00	**0.23**	**0.03**	0.57
NB-U	2.20	2.22	2.17	1.02	0.22	1.70	4.15	4.44	2.95	0.83	0.25	1.85

Table 5-4: Comparison of batch and online learning algorithms on the four data sets (OY, OD, NY, ND). The non-shaded rows are results from the batch and shaded ones are from the online-learning algorithms.

Classifier	Error	FPR	FNR	Precision	Recall	F-Measure	MCC	BER
C4.5	**0.41**%	**0.24**%	0.75%	**99.55**%	99.25%	**99.40**%	**0.991**	**0.49**%
RF	0.44%	0.30%	**0.70**%	99.43%	**99.30**%	99.36%	0.990	0.50%
Bagging	0.46%	0.26%	0.83%	99.51%	99.17%	99.34%	0.990	0.54%
LR	0.75%	0.42%	1.36%	99.20%	98.64%	98.92%	0.984	0.89%
SVM-lin	0.77%	0.38%	1.50%	99.28%	98.50%	98.89%	0.983	0.94%
SVM-rbf	1.17%	0.54%	2.38%	98.97%	97.62%	98.29%	0.974	1.46%
LB	1.35%	0.73%	2.54%	98.61%	97.46%	98.03%	0.970	1.63%
AB	2.25%	1.39%	3.86%	97.33%	96.14%	96.73%	0.950	2.63%
NB	2.95%	3.08%	2.69%	94.34%	97.31%	95.80%	0.936	2.89%
PA	**1.00**%	**0.61**%	**1.74**%	**98.84**%	**98.26**%	**98.55**%	**0.978**	**1.18**%
LB-U	1.25%	0.84%	2.02%	98.41%	97.98%	98.19%	0.972	1.43%
Perceptron	1.35%	1.10%	1.84%	97.93%	98.16%	98.05%	0.970	1.47%
CW	1.36%	1.00	2.03%	98.10%	97.97%	98.04%	0.970	1.52%
NB-U	2.95%	3.08	2.69%	94.34%	97.31%	95.80%	0.936	2.89%

Table 5-5: Comparison of batch and online learning algorithms on the combined data set (OYND). The non-shaded rows are results from the batch and shaded ones are from the online learning algorithms.

Additionally, in order to investigate the tradeoff between the accuracy and the time taken to build the model, we compare training time of each classifier on all the data sets in Table 5-6.

Data Set	C4.5	RF	Bagging	LR	SVM-lin	SVM-rbf	LB	AB	NB	PA	LB-U	Perce-ptron	CW	NB-U
OY	31.7 m	10 m	54 m	4.6 m	3.2 m	6 m	77 m	30.3 m	**2.8 m**	**0.8 s**	11.2 m	**0.8 s**	2.7 s	3.4 m
OD	6.9 m	3.7 m	15.9 m	1.8 m	**0.9 m**	1.8 m	49.9 m	20.1 m	1.9 m	**0.5 s**	6.8 m	**0.5 s**	1.7 s	1.9 m
NY	45 m	9.6 m	46.8 m	8.9 m	5.7 m	7.6 m	79.1 m	33.8 m	**3.1 m**	**0.6 s**	8.8 m	**0.6 s**	2.1 s	2.7 m
ND	10.8 m	3.9 m	18 m	**0.6 m**	14.5 m	21.6 m	35.3 m	14 m	1.3 m	0.4 s	4.7 m	**0.3 s**	1.2 s	1.0 m
OYND	127.6 m	25.7 m	122.2 m	16.1 m	12.5 m	18.1 m	151.1 m	67.1 m	**6.1 m**	**1.1 s**	15.4 m	**1.1 s**	4.2 s	5.6 m

Table 5-6: Time taken to build model (training time) for batch and online learning algorithms on all data sets. The shaded classifiers are online learning and non-shaded ones are batch learning algorithms.

Naïve Bayes is widely used in spam filters and related security applications partly because the training and testing time is fast. We see similar results in our experiments with Naïve Bayes showing the fastest time in most of the data sets to build models. Among online algorithms, Passive Aggressive (PA) and Perceptron take the least time among all the data sets. Overall, online algorithms take less time compared to batch algorithms to build models.

Overall, we see that Random Forests (RF) has the best tradeoff for training time and overall accuracy among all the data sets.

5.3.2 Feature Evaluation

In these experiments, we explore the discriminative strength and importance of each feature and feature category along with various combinations of feature sets. Chapter 4 shows that URL-based features have been proven effective in detecting phishing URLs. Particularly, as the focus of this Chapter is on the content based features, we investigate the benefit of using content-based features along with the URL-based features. Figure 5-2 compares the classification error rates on the four data sets using 6 different sets of features. For brevity, we only show detailed results from Random Forests (RF) classifier, which yields overall both low error rates and also relatively faster training time (see section 5.3.1). However, the other classifiers also produced similar results on the data sets. Table 5-7 shows the total number of features in each feature set for the whole (OYND) data set. For these experiments, we also report the number of relevant features that received non-zero F-score values (more on this next).

Search engines-based feature set yields the best error rates among all the individual feature categories in all the data sets. Content-based feature set yields the second best results. Combined full features, however, yield the best overall error rates in all the data sets. Content-based features significantly help in improving error rates by improving both false positive and negative rates when combined with each individual URL based feature set. The combined URL-based feature sets yield 0.67% error rate which is much better than each URL-based feature set. All feature sets but search engine-based feature category yields 2.23% error rate which is better than that yielded by content-based feature set but worse than that given by the combined URL-based features. This further emphasizes the importance of search engine-based features. URL-based features combined with the content-based features (full features), however, yield the best error rates combined with the best false positive and false negative rates, emphasizing the importance of content-based features.

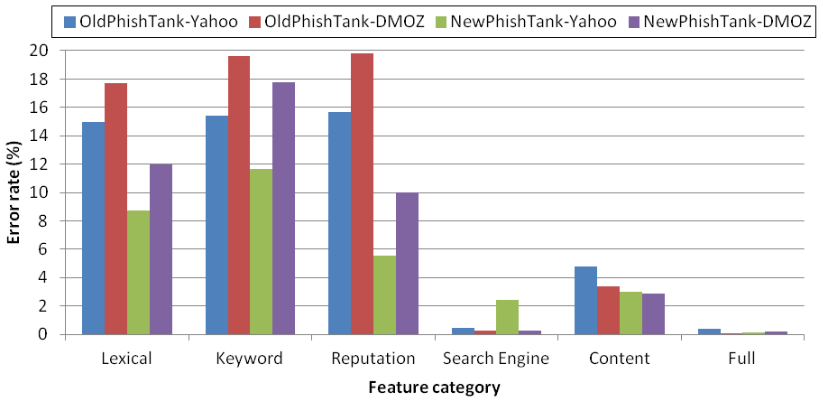

Figure 5-2: Error rates for RF classifier with five feature categories on each of the four data sets. Overall, using more features improves classification accuracy.

Next, in order to find the importance of each individual feature, we compute the F-score (Fisher score) of all the features. F-score is a simple but effective criterion to measure the discrimination between a feature and the label. Based on statistic characteristics, it is independent of the classifiers (Chen & Lin, 2003). For brevity, we list the top 20 features based on F-score in Table 5-8.

A larger F-score value indicates that the feature is more discriminative. One known disadvantage of F-score is that it considers each feature separately and does not say anything about the mutual co-relation among features.

Though the top 20 feature list is dominated by URL-based features, there are 5 content-based features that are ranked 7, 9, 10, 18, and 19th. Mostly keyword-based features made up the bottom of the list. This shows that the proposed content-based features are highly relevant to phishing webpages and are important in classifying phishing and non-phishing webpages.

Feature Category	Feature Count		Error	FPR	FNR
	Total	Relevant			
Lexical based	24	23	14.34%	8.00%	26.40%
Keyword based	101	98	16.42%	5.20%	37.60%
Reputation based	8	8	14.4%	1.50%	38.90%
Search Engine based	6	6	1.62%	1.00%	2.70%
URL (lexical + keyword+ reputation + search engine) based	139	135	0.67%	0.40%	1.20%
Content based	38	38	4.49%	3.60%	6.20%
Lexical + Content based	62	61	3.53%	2.90%	4.70%
Keyword + Content based	139	137	3.45%	2.50%	5.30%
Reputation + Content based	46	46	3.55%	2.70%	5.20%
Search Engine + Content based	44	44	0.75%	0.05%	1.20%
Lexical + Keyword + Reputation + Content	171	167	2.23%	1.60%	3.40%
Full features	177	173	0.44%	0.30%	0.70%

Table 5-7: Error rates for RF classifier with various combinations of features on the combined (OYND) data set.

Rank	Feature Description	F-score	Rank	Feature Description	F-score
1	Domain NOT in Google top results	8.48	2	URL NOT in Google top results	7.77
3	URL NOT in Yahoo top results	3.46	4	Domain NOT in Yahoo top results	2.47
5	URL NOT in Bing top results	1.95	6	Domain NOT in Yahoo top results	2.05
7	Title search NOT matching domain	0.68	8	URL in Google phishing blacklist	0.62
9	Title search NOT matching URL	0.31	10	Password field in page	0.26
11	Digit [0-9] in Host	0.22	12	IP in PhishTank top 10	0.20
13	Number of dots in URL path	0.11	14	PhishTank top target in URL	0.11
15	‘log’ in URL	0.10	16	PhishTank top domain in URL	0.10
17	Length of URL	0.08	18	Number of internal links	0.08
19	Has server redirection	0.08	20	‘pay’ in URL	0.07

Table 5-8: Top 20 features based on F-score using the combined OYND data set.

5.3.3 Training on One Data Set and Testing on Another

Features extracted by observing a particular data set can yield impressive low classification error rates when trained and tested on disjoint sets of the same data source using the right classifier. However, experiment results shown in Chapter 4 and (Ma, Saul, Safage, & Voelker, 2009a) show that when training and testing sources are completely mismatched, a classifier's accuracy decreases significantly. To investigate if this phenomenon holds in this context, we experiment with training and testing on various combinations of phishing and non-phishing sources of webpages. We use the abbreviations defined in section 5.2.1 to refer to each combination of data sets, e.g., OY for Old PhishTank-Yahoo.

Table 5-9 shows classification results of training and testing on mismatched data sets using Random Forests (RF) classifier.

As expected, when trained and tested RF classifier with same data set, the overall error rates are normally better compared to when trained and tested with mismatched – possibly different sources – data sets (see the diagonal values in Table 5-9). When newer phishing webpages are tested against the model trained from older phishing webpages, the error rate is 3.8% contributed mostly by false negatives. When only the non-phishing webpage source is mismatched (e.g., OD and OY), error rate increases due to higher false positives. This observation is similar to the ones observed in Chapter 4 and (Ma, Saul, Safage, & Voelker, 2009a). However, when only the phishing webpage data sets are mismatched (e.g., OD, ND), we do not see a big increase in error rates. The worst error rate in this category is 3.9% (NY and OY). This small increase in error rate may be due to the fact that the sources of phishing webpages are not technically different but they differ by the time when phishing URLs were submitted for verification. We look into this in detail in the next section. When classifier is trained with combination of any phishing data sets plus DMOZ (OD, ND), and tested with combinations of phishing and Yahoo data set (OY, NY), the error ranges between 9–38%, contributed mostly by high false positives. This is due to the fact that DMOZ data set is about half the size of Yahoo data set and Yahoo data set also includes non-phishing webpages from top targets which look very similar to phishing webpages that DMOZ may not have. When trained with the combined data set, the model generalizes well and yields good performance results across all test data sets (last row) except for testing OY data set which yields a higher error rate of 5.3%. This experiment concludes that the training data set needs to be selected properly that represent the actual test environment in the real world in order to achieve the best performance from a classifier.

Training	**Testing (Error rate)**			
	OY	**OD**	**NY**	**ND**
OY	0.42%	0.44%	3.82%	4.27%
OD	9.84%	0.04%	38.30%	0.55%
NY	3.96%	19.11%	0.87%	1.53%
ND	11.52%	0.77%	16.34%	0.45%
All data sets (OYND)	5.28%	0.00%	0.00%	0.16%

Table 5-9: Overall error rates when training on one data set and testing on another (possibly different source) using RF classifier.

5.3.4 Data Drift

As phishers change their attack tactics over time, so must the classifier's model employed in detecting phishing webpages. It is evidently clear from previous experiments that classifiers performance drifts over time (see Table 5-9) when classifiers are trained with older data set (OY) and tested with newer data set (NY). Moreover, measuring how performance drifts over time can help us determine how often we need to retrain our classifier to keep it up to the minute. In order to experimentally demonstrate how our approach would work if deployed in real world, we use OldPhishTank phishing data set and 22,480 (twice the number of phishing webpages) randomly selected non-phishing webpages from Yahoo and DMOZ data sets as our "base" training set. Figure 3 and Figure 4 show the comparison of batch and online learning algorithms respectively after training the classifiers on different intervals. The *x*-axis shows number of days in the experiment with the phishing webpages collected from January 1st to May 3rd of 2011, and the *y*-axis shows the cumulative error rates on testing the classifiers. To generate test data for non-phishing webpages, twice the number of phishing webpages was randomly selected from the remaining 9,101 non-phishing webpages from Yahoo and DMOZ data sets.

5.3.4.1 Comparison of Batch Algorithms

Though batch learning algorithms are ultimately limited by the training time and memory limitation in a large-scale, real-time application, we experimented with the top 2 batch algorithms from the experiments in 5.3.1 as our combined data set (OYND) is small enough to hold in the memory. Figure 5-3 compares classification accuracies of Random Forests (RF) and J48 using different training intervals. RF-once and J48-once curves represent training once on “base” training set and using that models for testing on all other days. RF-weekly and J48-weekly curves represent training the classifiers weekly with the data collected up to that week and testing on the data collected in the subsequent week. Similarly, RF-daily and J48-daily curves represent training and testing the models on the daily basis. Overall, RF outperforms J48 among all interval-based training with the best cumulative error approaching 1.5% when training the classifiers daily.

As most batch algorithms take anywhere from several seconds to minutes to train models, retraining them after every instance may not be practical in real-world applications where tolerably low turnaround time is expected for a successful implementation besides high accuracy rates. Therefore, we try to exploit the advantage of online algorithms in the next section.

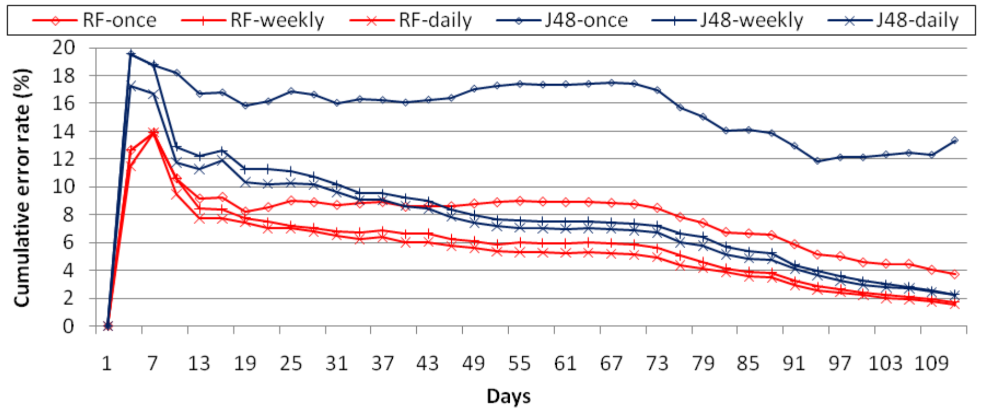

Figure 5-3: Error rates for batch algorithms using all features and various interval-based training.

5.3.4.2 Comparison of Online Algorithms

The major advantage of online algorithms is that they allow the model to update incrementally without having to retrain them from the scratch which saves time and memory resources. Online algorithms have been shown to perform better compared to some batch algorithms when trained continuously with variable-feature sets in the context of detecting malicious URLs (Ma, Saul, Safage, & Voelker, 2009b). In order to investigate if they achieve similar superior performances in our context, we compare classification performances of five online algorithms such as updatable version of Naïve Bayes (NB-U), Perceptron, Passive-Aggressive (PA), Confidence-Weighted (CW), and the updatable version of LogitBoost (LB-U) algorithms and show the performance results in Figure 5-4.

When trained on the daily interval, the updatable version of LogitBoost (LB-U) performs the best among all the online learning algorithms with cumulative error approaching to 2.9%. Interestingly, LB-U performs worse when trained incrementally. When trained in interval or incrementally, the online version of Naïve Bayes, surprisingly, outperforms the remaining three online algorithms with the cumulative error approaching to 4.2% as compared to 16.1% by Perceptron, 10.4% by PA and 9.8% by CW. Naïve Bayes is also comparatively the most robust to data drift as its performances remain very similar (15-4%) when trained either daily or continuously. Though the cumulative error rates are much worse (more than 10%) for other three classifiers (PA, Perceptron and CW), they show improvements in accuracy results when trained continuously over training on the daily basis. Moreover, performances of these 3 classifiers are as bad as random guessing when they were retrained weekly (the results are not shown). Nevertheless, all online classifiers were much faster in training models as opposed to all the batch algorithms.

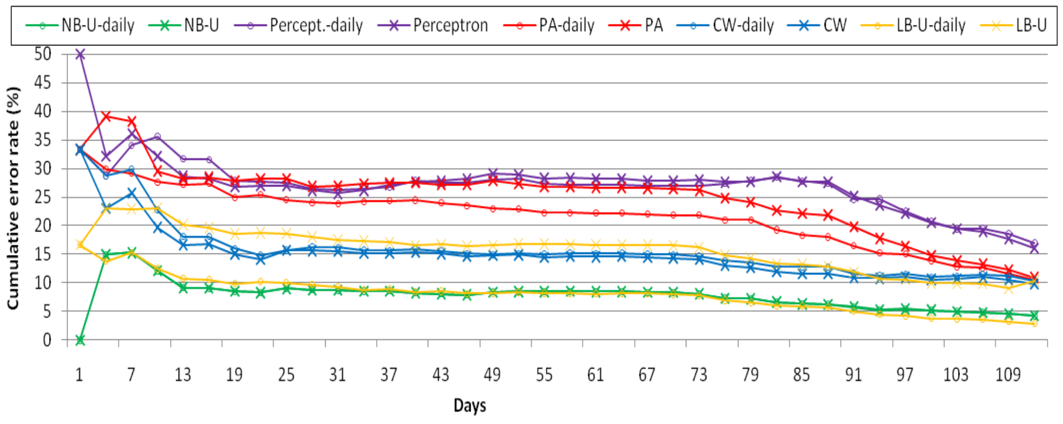

Figure 5-4: Error rates for online algorithms using all features and continuous and interval-based training.

Interestingly, in this context, all the online algorithms yield lower classification accuracies compared to the batch algorithms that are trained on the daily basis.

5.3.5 Detecting Top Targets

In these experiments, we investigate how our proposed method performs in detecting the legitimate webpages from top targets as non-phishing. We combine all the data sets (OYND) except the webpages of the top targets and use the combined 46,992 instances as a training data set. We use the rest 1,671 instances as a test data set. Random Forests (RF) yields a 13.8% error rate, meaning only 86.2% of legitimate webpages related to the top targets are correctly classified as non-phishing webpages and the rest are all misclassified as phishing webpages. This big error rate is due to the fact that these target webpages look very similar to the forged phishing webpages, and the model, perhaps, is not trained with the types of webpages that are seen during testing.

In order to address this, we randomly select 66% of legitimate webpages from top targets and include them in the training set and retrain the model. We test the newly trained model with the remaining 569 instances of webpages. As expected, the results improve significantly, yielding a 1% error rate, misclassifying only 6 non-phishing webpages as phishing. This experiment further emphasizes the importance of judicious selection of training data set with proper representation of all the possible phishing and non-phishing webpages the system will be tested against in real world.

Unlike related works (Ma, Saul, Safage, & Voelker, 2009a), (Whittaker, Ryner, & Nazif, 2010), and (Zhang, Hong, & Cranor, 2007) we explicitly test the validity of our method on the actual phishing targets and show that our method yields good accuracy results in detecting not only the phishing webpages but also in detecting the legitimate webpages of the actual targets.

5.3.6 Analyzing False Positives and Negatives

In Chapter 4, we proposed a hypothesis that false positives and negatives in classifying phishing websites could be lowered by including features from webpage contents. In this chapter, we experimentally confirmed our hypothesis as both the false positive and false negative rates have been lowered by using the content-based features (see Table 5-7). Despite low false positive and false negative rates achieved by this approach, examining misclassified webpages may help us to understand the limitations of our approach and possibly reveal trends that can suggest refinements to our current approach. We examined the test results by Random Forests (RF) on the combined (OYND) data set.

Webpages that we downloaded from the links harvested from our initial seed URLs of most targeted brands contributed to some false positives. These URLs were long in nature containing some of the red-flagged keywords. Interestingly, these URLs also didn't exist in search engines' indexes. Also, poorly written legitimate webpages that didn't use HTTPS to transmit FORM data with *password* field were classified as phishing webpages. Though these groups of non-phishing webpages are not phishing, but they sure are insecure in the sense that unencrypted confidential data can be sniffed during the transmission.

Phishing webpages hosted on free legitimate hosting services that have been around for a while, whose URLs didn't have any red flagged keywords and whose URLs and domain were in search results were classified as legitimate webpages contributing to false negatives.

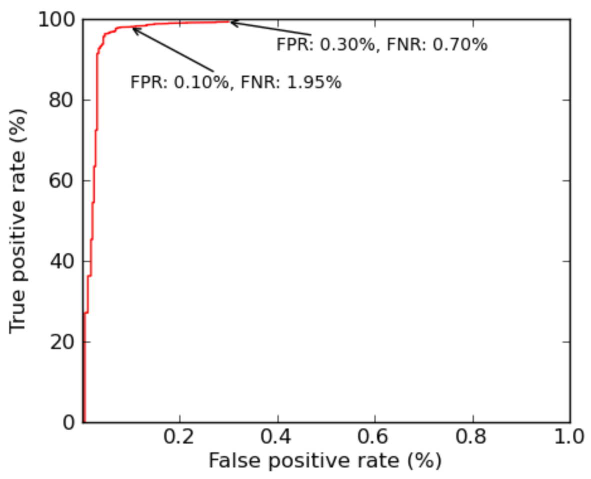

Figure 5-5: ROC showing tradeoff between false positives and false negatives using RF on the combined data set (OYND). Note the false positive rate on the x-axis ranges between 0 and 1%.

One of the advantages of using machine learning approach is that their learned models allow us to tune tradeoff between false positives and negatives. Figure 5-5 shows the results of this experiment as an ROC graph with respect to threshold *t* over an instance of the combined data sets using (OYND) using RF classifier. By tuning false positives to a very low 0.1%, we can achieve 1.9% false negative rate. By tolerating slightly higher false positives of 0.3%, however, we can achieve significantly lower false negative rate of 0.7%.

5.4 Related Work

The work by Whittaker et al. (2010) is most closely related to our work. They describe the design and performance characteristics of a scalable machine learning classifier to detect phishing websites, which has been used in maintaining Google’s phishing blacklist automatically. Their proprietary classifier analyzes millions of pages a day, examining the URL and the contents of a page to determine whether or not a page is phishing. Their system classifies webpages submitted by end users and URLs collected from Gmail’s spam filters. Though some features are similar, we propose several new URL and webpage content-based features and evaluate our approach with publicly available batch and online learning algorithms using public data sets. While they use features from the terms appearing in the text of a page, most of our content-based features are based on technical properties of a webpage. Unlike their approach, we also do not use DNS entries and geo-locations of pages’ host and nameservers.

Zheng et al. (2007) present CANTINA, content-based approach to detect phishing websites, based on the TF-IDF information retrieval algorithm. By using a weighted sum of 8 features (4 content-related, 3 lexical, and 1 WHOIS-related), they show that CANTINA can correctly detect approximately 95% of phishing sites. Though similar in motivation, our approach differs significantly in both methodology (comparing a number of batch and online learning algorithms) and scale (considering a large number of features and an order-of-magnitude more training examples).

Miyamoto et al. (2008) used AdaBoost algorithm to 8 heuristics proposed by Zheng et al. (2007) and achieved 97% accuracy (a 2% improvement) in detecting phishing websites. Our approach differs in the say way as it differs from Zheng et al.

Other papers have attempted to automatically classify malicious and phishing URLs from using the information and meta-data on URLs alone, see for e.g., (Garera, Provos, Chew, & Rubin, 2007), (Ma, Saul, Safage, & Voelker, 2009a).

5.5 Conclusions

In this chapter, we proposed novel server-based and content-based features for classifying phishing webpages. Augmenting with the URL-based features, we demonstrated that the proposed features are highly relevant to the automatic discovery and classification of phishing webpages. We tested our approach on a real-world temporal data sets using a number of popular batch and online learning methods. We experimentally demonstrated that phishing webpages can be classified with an accuracy of 99.9% and very low false positive rate of 0% and false negative rates of 0.3% using features from URLs, web servers, and the contents of the phishing pages.

Most classifiers show statistically similar performance metrics. Overall, Random Forests (RF) performed the best in terms of classification accuracy and time required to build the model in this context. Online algorithms, however, showed poorer classification performance compared to batch algorithms. As expected, the performances of all the classifiers decrease while training them with older data set and testing them with newer data set. It is, thus, recommended to retrain classifiers with newer data sets as and when they become available if deployed in real-world.

Chapter 6

Detecting Phishing Webpages: A Rule-Based Approach

6.1 Introduction

In this chapter, we propose a rule-based approach to detect phishing webpages and present our experimental results on temporal data sets using non-machine learning approach.

Though different in goal, our approach is particularly inspired by the approach introduced by the open source intrusion detection and prevention system (IDS/IPS), Snort. Snort monitors networks by matching each packet it observes against a set of rules (Oresch). As the phishing attacks have been growing rapidly by the day, we feel that there is a need for Snort like phishing attack detection technology at the application level. We try to investigate such an approach.

Just like a network IDS signature, a rule is a pattern that we want to look for in a webpage. The idea behind the rule-based approach is to make the process of phishing attack detection as intuitive, simple, and user-friendly as possible. Besides achieving high detection accuracy, a major goal of our approach is to make the framework flexible and simple to extend the rule set by incorporating new and emerging phishing tactics as they are encountered. We generate our rule set primarily relying on our observations and the machine learning features proposed in various existing literatures (Basnet, Mukkamala, & Sung, 2008), (Ma, Saul, Safage, & Voelker, 2009a), (Zhang, Hong, & Cranor, 2007), (Garera, Provos, Chew, & Rubin, 2007) on phishing attack detection. We gather various techniques and tricks used by phishers to lure their potential victims to a forged website and use those heuristics to develop our initial rule set.

A rule is usually written in the following form:

IF *conditions* THEN *actions.*

If the *conditions*, also known as *patterns*, are satisfied then the *actions* of that particular rule are fired.

A rule may range from very simple – checking a particular value in the URL – to highly complex and time-consuming that may require to analyze meta-data, query search engines and blacklists and combine several conditions with *AND* and *OR* operators. Depending on their characteristics and the methods used to extract the rules, we broadly group them into the following categories.

6.2 Rules

In this section, we briefly describe various rules that we employ in detecting whether a given webpage is phishing.

6.2.1 Search Engine-Based Rules

The idea behind using results from top search engines is to leverage their power and effectiveness in continuously crawling and indexing a large number of webpages. In Chapter 4, we show that search-engine based features are very effective in determining phishing URLs and essentially demonstrate that search engines' large and continuously growing indexes act as a rudimentary white-list against the phishing webpages. We develop two rules using search engines.

Rule 1: IF a webpage's URL is not present in all search engines' indexes, THEN the webpage is potentially phishing.

Rule 2: IF a webpage's domain is not present in all search engines' indexes, THEN the webpage is potentially phishing.

To generate Rule 1, we check if a URL exists in the search engines' (Google, Yahoo!, and Bing) indexes. Our rule generator automatically queries the search engines and retrieves top 30 results. If the results do not contain the URL, this rule considers the webpage as potentially a phishing attack. We observed that all three search engines returned the URL as the first result if they have indexed the URL. Intuitively, it makes sense because we search the URL itself not ranked relevant URLs based on keywords. But to be on the safe side, we use top 30 results as it has been shown that going beyond the top 30 results had little effect (Zhang, Hong, & Cranor, 2007).

Similarly, Rule 2 is generated by querying the search engines with the domain of a URL. If the top 30 results do not contain the domain, this rule says that the given webpage is potentially phishing.

6.2.2 Red Flagged Keyword-Based Rule

By examining 80% of randomly selected URLs on DS1 data set, we found that certain groups of words seem to be more popular among phishers, perhaps, to lure unsuspecting users to the forged webpage. Using substring extraction algorithm, we generated a list of 62 word stems that frequently occur in our training data set. We iterate through this keyword list and check if any of the word is found in the URL. Thus, we generate our next rule:

Rule 3: IF a keyword is present in the URL, THEN the webpage is likely phishing.

6.2.3 Obfuscation-Based Rules

Phishers often obfuscate URLs to trick users into thinking that the malicious URL belongs to a legitimate website users are familiar with. Obfuscating URLs with certain characters such as "-", soft hyphen, Unicode, and visually similar looking characters are very common techniques employed by phishers. We try to identify these tactics and generate rules from them. For example, we check if certain characters such as “-”, “_”, “=”, “@”, digits, and non-standard port, etc. are present in a webpage’s URL.

These tactics used by phishers lead us to our next set of rules.

Rule 4: IF a webpage’s URL is IP based (hex-based, octal, or decimal-based), THEN the webpage is potentially a phishing attack.

Rule 5: IF a URL contains any of the following characters [-, _, 0-9, @, “,”, ;] OR contains a non-standard port, THEN the webpage is potentially phishing.

Rule 6: IF host part of a URL has 5 or more dots OR length of the URL is longer than 75 characters OR length of the host is longer than 30 characters, THEN the webpage is potentially a phishing attack.

6.2.4 Blacklist-Based Rule

We employ Google Safe Browsing API (Google Safe Browsing API) to check URLs against Google’s constantly updated blacklists of suspected phishing and malware pages and generate our next rule.

Rule 7: IF a URL is in Blacklist(s), THEN it is potentially a phishing webpage.

6.2.5 Reputation-Based Rule

We generate our next set of rules from historical stats on top IPs and domains that have a bad reputation of hosting the most phishing webpages. We use 3 types of statistics: Top 10 Domains, Top 10 IPs, and Top 10 Popular Targets published by PhishTank (Statistics about phishing activity and PhishTank usage). We also use top 50 IP address stat produced by StopBadware.org (StopBadware).

Rule 8: IF a URL contains a top phishing target OR its IP or domain is in the statistical reports produced by PhishTank, Stopbadware, etc., THEN the webpage is potentially a phishing attack.

6.2.6 Content-Based Rules

The rules in this category are rooted in the HTML contents of the phishing webpages. An ingenious phishing webpage resembles the look and feel of the target legitimate website. Nevertheless, the same tactics employed by phishers also give us opportunities to discover our content-based rules. By observing HTML structures of hundreds of phishing webpages, we've generated the following rules:

Rule 9: IF a webpage contains *password* input field AND (the corresponding form content is sent in plain text without using Transport Layer Security (TLS)/Secure Sockets Layer (SSL) OR the form content is sent by using 'get' method), THEN the webpage is potentially phishing.

Rule 10: IF a webpage contains *password* input field AND the corresponding form content is sent to external domain regardless of TLS/SSL, THEN the webpage is potentially phishing.

Rule 11: IF a webpage contains META tag AND the refresh property's destination URL is in external domain OR it belongs to a blacklist, THEN the webpage is potentially phishing.

Rule 12: IF a webpage is redirected by its server AND the page contains *password* field, THEN the webpage is potentially phishing.

Rule 13: IF a webpage has IFrame tag AND its source URL belongs to a blacklist, THEN the webpage is potentially a phishing attack.

Rule 14: IF a webpage contains *password* input field AND the webpage has more external than internal links, THEN the webpage is potentially phishing.

Rule 15: IF a webpage has bad HTML markups AND contains *password* input field, THEN the webpage is potentially a phishing attack.

Figure 6-1 shows the histograms of Rules 1-15 obtained on data set DS1. The histogram confirms that Rule 1 and Rule 2 have high prominence in phishing webpages and are very strong indicators of whether a webpage is phishing. These rules by themselves can detect more than 97% of phishing webpages, while correctly classifying 100% of legitimate webpages. Rule 3 has high presence in phishing webpages as well compared to non-phishing webpages.

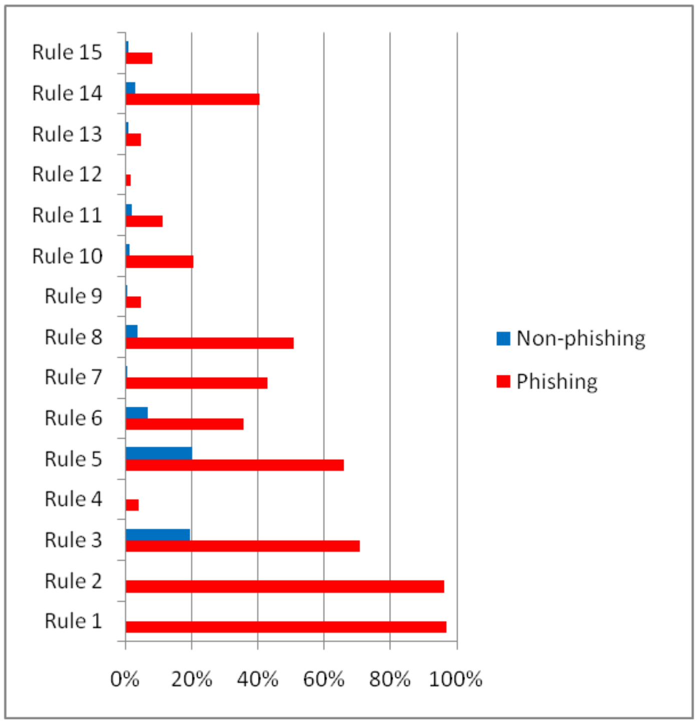

Figure 6-1: Histogram of Rules 1-15 on DS1 data set.

Some phishing webpages (~4%) satisfy Rule 4, while not a single non-phishing webpage satisfies it. Though very sparsely present, this rule can be a good indicator of whether a webpage is phishing. Roughly 66% of phishing webpages and surprisingly 20% of non-phishing webpages satisfy Rule 5.

43% of phishing webpages satisfy Rule 7, while 0% of non-phishing webpages satisfy the same. This suggests that Rule 7 is a strong indicator of whether a page is phishing. Rule 8 is present in about 51% of phishing webpages and in roughly 4% of non-phishing webpages. Rule 9 has relatively small presence among phishing webpages but no presence on non-phishing webpages, indicating that this rule is not universally applicable, but still a strong indicator of phishing webpage.

About 21% of phishing and 1% of non-phishing webpages satisfy Rule 10. Relying on this rule alone would miss a large percentage of phishing webpages while it would also misclassify some legitimate webpages as phishing. Rule 12 is satisfied by a very small number of phishing webpages (~1%). However, no single non-phishing webpage satisfy the same suggesting that this rule may not aid in false positives.

Relatively more phishing webpages satisfy Rule 13, 14, and 15 compared to non-phishing webpages.

We point out that these are not the exhaustive list of rules. One of the major advantages of rule-based approach is to be able to quickly tune the rules to ones' needs and easily modify or add rules as and when needed to detect new and ever changing phishing attacks.

6.3 Experimental Evaluation

In this section, we briefly describe the data sets we use in this chapter and present the results of experimental validation of our approach on these data sets. The experiments were carried out on a machine with Core 2 Duo 2 GHz Intel processors and 3 GB RAM.

6.3.1 Data Sets

We use the same data sources described in Chapter 4. Based on the date on which phishing URLs were submitted to PhishTank for verification, we generated two data sets. The first data set, we refer to it as DS1, contains 11,341 phishing webpages submitted before October 31, 2010 and 14,450 legitimate webpages from Yahoo! and seed URLs. The second data set, we refer to it as DS2, contains 5,456 phishing webpages submitted for verification between January 1 and May 3 of 2011 and 9,636 randomly selected legitimate webpages from DMOZ.

We discarded the URLs that were no longer valid as the page couldn't be accessed to test rules from their contents.

6.4 Counting Rules

In order to detect a phishing webpage based on rules, one simple approach is to give equal weight to each rule and count the number of rules satisfied by the page. Using a carefully chosen threshold, if the total number of rules satisfied by an instance is more than the threshold value, we can alert that the webpage is phishing. However, as the histogram shows (see Figure 6-2), choosing the best threshold value that would give the balanced and best false positive and negative rates is not a trivial task. Histogram in Figure 6-2 shows rule count from 0 up to 10 as only a very few phishing instances in the data set satisfied more than 10 rules.

Nonetheless, we experiment with a few thresholds and present the results, True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR), and True Negative Rate (TNR) in Table 6-1. Threshold of 4 rules has the best tradeoff between false positives (1.8%) and false negatives (8.1%). Depending on the users security preference, one may choose the threshold of 3 rules which has better FNR of 2.4% compared to 8.9% FPR.

Flexibility in adding and removing rules as and when required is one of the goals of our rule-based approach. To demonstrate this flexibility, we remove Rule 3 and Rule 5 as the histogram in Figure 6-1 shows that about 20% of non-phishing webpages satisfy these rules besides more than 60% of phishing webpages also satisfying them. We then use various thresholds to calculate the performance metrics and achieve much better results for threshold of 2 rules. By using the threshold of 2 rules to predict whether a webpage is phishing, we achieve 98.14% TPR, 97.62% TNR, 2.38% FPR, and 1.86% FNR. These results are comparable to the results achieved by machine learning approaches which we discuss next.

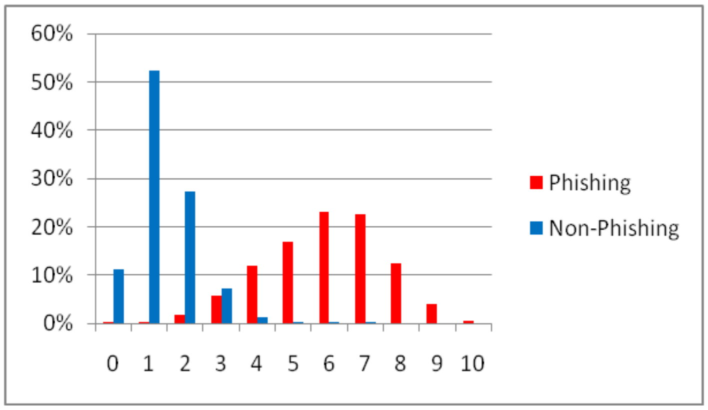

Figure 6-2: Histogram of rules count on the data set DS1. Horizontal axis is rule count and vertical axis is instance count.

Rule Count Threshold	TPR	TNR	FPR	FNR
2	99.55%	63.60%	36.40%	0.45%
3	97.64%	91.04%	8.96%	2.36%
4	**91.87%**	**98.20%**	**1.80%**	**8.13%**
5	79.96%	99.63%	0.37%	20.04%

Table 6-1: Performance results for various threshold values.

6.5 Training with Rules

In these experiments, we used the rules identified in Section 6.2 as binary features and applied them to train classifiers. We used C4.5 (Quinlan, 1993) decision tree and Logistic Regression (LR) algorithms implemented in the WEKA data mining library (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009). We employed 10-fold cross validation method to test the models and performed our analysis.

Using C4.5 (which is implemented as J48 in WEKA), we obtained the pruned decision tree of size 19 with 10 leaf nodes (see Figure 6-3).

Rules 2, 4, 9, 10, 11, and 12 are removed from the model as a result of pruning.

Besides simple to understand and interpret, decision tree is robust and uses a white box model. Decision tree-based model can be converted to rules using boolean logic as:

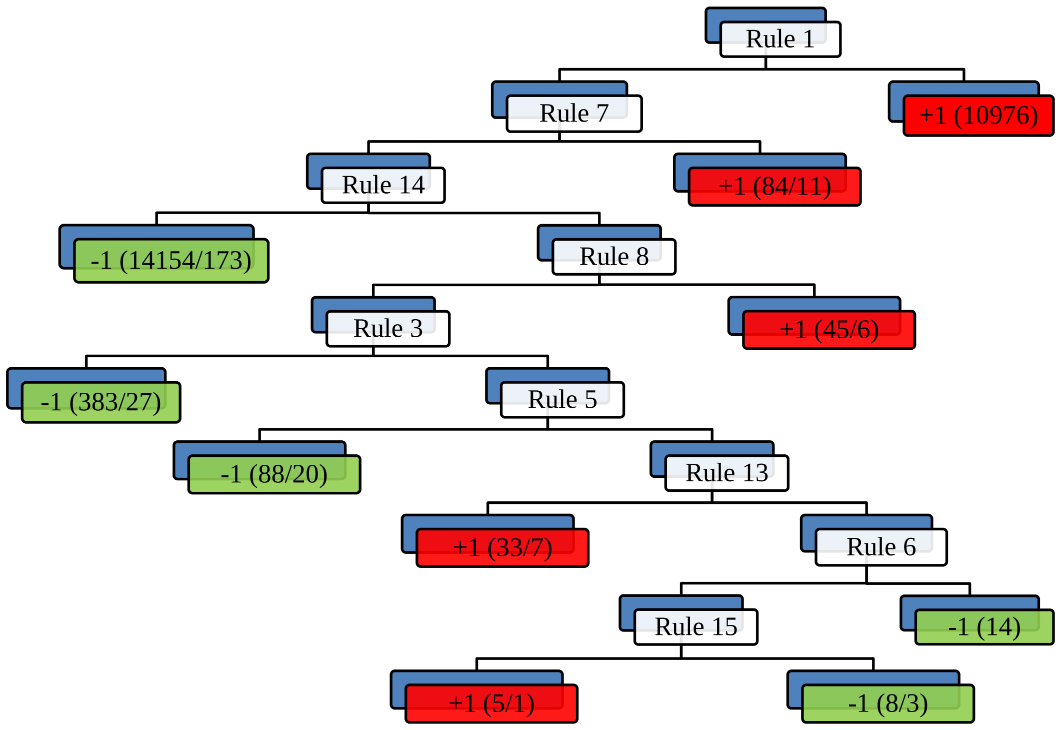

Figure 6-3: DT model using C4.5 on data set DS1. Left edge corresponds to the value of each rule <=0 and the right edge corresponds to the value > 0. “-1” green leaf node represents the non-phishing classification and “+1” red leaf node represents phishing classification.

IF (Rule_1 <= 0) AND (Rule_7 <= 0) AND (Rule_14 <= 0) THEN phishing = No. Using this sequence of tests, 14,154 of non-phishing samples from DS1 data set are correctly classified, while 173 phishing webpages are misclassified. The rest of the rules can be interpreted in the similar manner.

It took about 3.4 seconds to build C4.5 model. The detailed test results are shown in Table 6-3.

We also applied Logistic Regression (LR) classifier as it has been proven effective in problems similar to ours (Ma et al., 2010a). In LR because the output of a linear model depends on the weighted sum of the features, the sign and magnitude of the individual parameter vector coefficients can tell us how individual features contribute to a ‘phishing’ or a ‘non-phishing’ prediction. Positive coefficients correspond with phishing features while negative coefficients correspond with legitimate non-phishing features. A zero coefficient means that the corresponding feature will not contribute to the prediction outcome.

The coefficients and the odds ratio obtained from LR model on data set DS1 are displayed in Table 6-2.

As indicated by the high odds ratio, Rule 4 is found to be the most useful in detecting whether a webpage is phishing. Similarly, Rules 1, 2, and 12 are strong indicators that a webpage is phishing attack. Interestingly, Rules 10, 13, and 15, on the other hand, seem to indicate that a webpage is non-phishing. LR took 2.33 seconds to build model and gave classification error rate of 1.02%, TPR of 97.88%, and FPR of 0.15%. C4.5 and LR achieve statistically similar results.

Feature	Logistic Coefficient	Odds Ratio
Rule 1	80.4784	8.94E+34
Rule 2	63.5746	4.07E+27
Rule 3	1.4302	4.1796
Rule 4	138.2439	1.09E+60
Rule 5	1.5243	4.5917
Rule 6	0.3788	1.4606
Rule 7	5.008	149.6037
Rule 8	2.2699	9.6781
Rule 9	0.9032	2.4675
Rule 10	-0.155	0.8564
Rule 11	0.9984	2.714
Rule 12	58.7186	3.17E+25
Rule 13	-0.3982	0.6715
Rule 14	3.2524	25.852
Rule 15	-0.497	0.6084
Constant	-5.8523	

Table 6-2: Features and their coefficients and odds ratios using Logistic Regression classifier.

6.6 Training with One Data Set and Testing with Another

Phishing tactics and attack techniques keep changing as attackers come up with novel ways to circumvent the existing filters. Rules developed from observing a particular data set can yield a highly accurate classification results when trained and tested on disjoint sets of the same data source. But do these results hold when testing new phishing webpages using the same rule set extracted from old phishing webpages? To investigate this question, we tested new phishing data set DS2 against the model obtained by training the C4.5 classifier on old data set DS1. Table 6-3 shows classification results using C4.5 classifier on various combinations of temporal data sets.

As expected, when trained and tested using the same data set DS1, the error yielded is the lowest due to low FPR (0.21%) and FNR (1.9%). When training and testing sources are completely mismatched, the error ranges from 1.2% to 4.8%. Surprisingly, the error received on training and testing using DS2 is comparatively higher (4.5%) with 1.3% FPR and 10.2% FNR. Although, error rate doesn't significantly decrease with the newer phishing data, the disparity in accuracy emphasizes that rules and training data should be selected judiciously. Thus, it is important to collect data that is representative and retrain the deployed classifier with new data often. Finding an optimal time interval to retrain the system for optimum performance represents an interesting direction for future study but is beyond the scope of this study.

Training	**Testing (Error rate)**	
	DS1	**DS2**
DS1	0.98%	4.86%
DS2	1.22%	4.51%
DS1+DS2	0.96%	4.18%

Table 6-3: Overall error rates on training on one data set and testing on another data set.

6.7 Discussion

In this section, we discuss some of the limitations of rule-based approach and some possible ways to address them.

6.7.1 Limitations

The rule set is preliminary and we emphasize its needs for expansion and thorough scrutiny. The system, if deployed, will likely produce some false alarms, while also missing a good number of phishing webpages. Attackers may thwart the system by minimizing the phishing tricks matching none or a small number of rules on their crafted phishing webpage. One such scenario is when attackers hack a legitimate webpage to host their phishing campaign. Such legitimate webpages are highly likely to appear on search engines' results. Phishers may design flash-based webpages virtually hiding all the HTML contents for analysis. However, expanding our rule set may address such potential attacks. We can add visual similarity-based rules, for instance. An interesting research area would be to expand the rule set using the tactics used in phishing emails and use it to detect phishing attacks carried out by emails.

URL-shortening services are growing in popularity thanks to micro-blogging websites such as Twitter[3]. In order to take advantage of the popularity and the obscurity provided by these shortening services, scammers are now establishing their own fake URL-shortening services (MessageLabs Intelligence Reports). Under this scheme, shortened links created on these fake URL-shortening services are further shortened by legitimate URL-shortening sites. These links are then distributed via phishing emails, blogs, micro-blogs, and social networking websites. We use the Python library (Basnet, 2010b) to automatically detect and expand shortened URLs.

To address this, additional rule could be determined such as: IF a URL is shortened by unsupported URL-shortening service, THEN the webpage is potentially a phishing attack. Because our data set doesn't have any webpage satisfying this rule, we do not include it in our current rule set.

6.7.2 Tuning False Positives & Negatives

Figure 6-4 shows the results of this experiment as an ROC graph with respect to the decision threshold *t* over an instance of DS1 data set using C4.5 classifier.

[3] http://twitter.com

Instead of using decision threshold *t* to minimize the overall error rate, Internet users may want to tune the threshold to have very low false positives at the expense of more false negatives or vice versa. By tuning false positives to conservatively low 0.1%, we can achieve false negatives of 12.8%. By tolerating slightly higher false positives of 0.2%, however, we can achieve significantly lower false negatives of 3.2%.

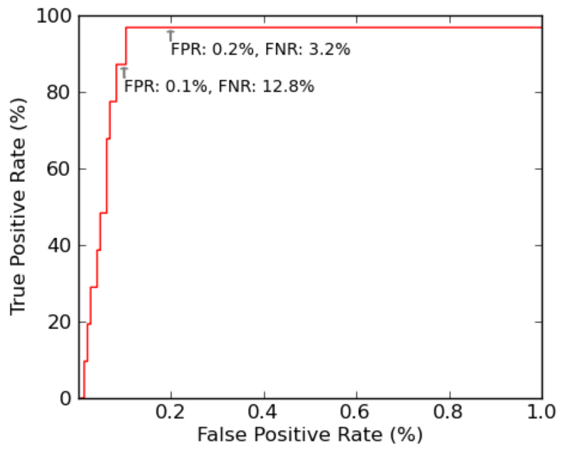

Figure 6-4: ROC showing tradeoff between false positives and false negatives using C4.5 on DS1 data set.

6.8 Related Work

There is an extensive recent literature on automating the detection of phishing attack, most importantly, detection of phishing emails and phishing webpages. Phishing attack detection techniques based on the machine learning methodology has proved highly effective, due to the large phishing data set available and the advances in feature mining and learning algorithms; see e.g., (Basnet, Mukkamala, & Sung, 2008), (Basnet & Sung, 2010), (Fette, Sadeh, & Tomasic, 2007), (Garera, Provos, Chew, & Rubin, 2007), (Ma, Saul, Safage, & Voelker, 2009a), (Whittaker, Ryner, & Nazif, 2010), (Zhang, Hong, & Cranor, 2007), (Miyamoto, Hazeyama, & Kadobayashi, 2008), etc.

Anomaly detection has been used to detect phishing webpage (Pan & Ding, 2006) where a number of anomaly-based features are extracted from webpages. Applying SVMs on a data set with 279 phishing and 100 legitimate webpages, their approach achieves 84% classification accuracy.

Visual similarity based methods have been explored for detecting phishing webpages. Weynin et al. (2005) compare legitimate and spoofed webpages and define visual similarity metrics. The spoofed webpage is detected as a phishing attack if the visual similarity is higher than its corresponding preset threshold. Medvet et al. (2008) consider three key page features, text pieces and their style, images embedded in the page, and the overall visual appearance of the page as rendered by the browser.

6.9 Conclusions and Future Direction

In this chapter, we proposed and evaluated a rule-based phishing attack detection technique. By analyzing tens of thousands of phishing webpages, we generated our 15 initial rule set. By weighting each rule equally and using a threshold of number of rules satisfied by a webpage, we achieved a FPR of 2.4% and FNR of 1.9% in detecting phishing webpages.

To compare the results with that achieved by machine learning approach, we used rules as features for C4.5 and Logistic Regression learning algorithms. Both the classifiers gave statistically similar results with FPR of 0.5% and FNR of 2.5%.

The experimental results indicate that even using a set of preliminary rules, the approach is nearly as accurate as the machine learning approach in detecting phishing webpages. Therefore, using more thoughtfully developed rules and applying possible rule chaining, the conventional rule-based approach possesses great potential for developing highly effective phishing detection systems.

As the rules are preliminary, they need to be refined and improved in order to achieve better false positives and false negatives.

Chapter 7

Ensemble-Based Systems for Phishing Detection

7.1 Introduction & Motivation

There are several mathematically sound and psychologically strong reasons to use ensemble-based systems (Polka, 2006). It is a common practice to seek a second opinion or third, and sometimes many more before making an important decision in matters that have financial, medical, social or other implications. Similarly, since no single classifier is perfect and its generalization performance depends on the application and the data sets, it may be beneficial to resort to ensemble-based technique. The strategy in ensemble systems is therefore to create many classifiers, and combine their outputs such that the combination improves the performance of a single classifier. The strategy, therefore, works the best when the individual classifiers make errors on different instances. The idea is that if each classifier makes different errors, then a strategic combination of these classifiers can reduce the total error.

In this chapter, we evaluate the efficacy of ensemble-based systems for detecting phishing URLs and websites. We evaluate various ensemble-based hybrid systems on real-world data sets of more than 16,000 phishing instances and 31,000 non-phishing instances.

7.2 Ensemble-Based Systems

Two interrelated questions need to be answered in designing an ensemble system: 1) how will individual classifiers (base classifiers) be generated? 2) how will they differ from each other? The answers ultimately determine the diversity of the classifiers, and hence affect the performance of the overall system (Kuncheva & Whitaker, 2003).

We evaluate the following most widely used ensemble-based approaches: AdaBoost.M1 (Freund & Schapire, 1996), LogitBoost (LB) (Friedman, Hastie, & Tibshirani, 2000), and MultiBoosting (MB) (Webb, 2000) which are variations of AdaBoost. For binary data AdaBoost.M1 is essentially the same as the original AdaBoost. We use Decision Stump (Ibb & Langley, 1992) as weak classifiers for all the versions of Boosting-based classifiers. A decision stump is a machine learning model consisting of a one-level decision tree. We use default number of rounds $T = 10$ for both the classifiers.

Besides, we also evaluate Bagging (Breiman, 1996) and Random Forests (Breiman, 2001) . We call these boosting and bagging based classifiers as *meta* classifiers from henceforth. In our experiments, we use the default REPTree as the base classifier for Bagging and use default values as parameters for the classifier.

7.2.1 Combining Classifiers

Combining classifiers is a key component of any ensemble system. Various strategies are employed for optimal performance of the ensemble system. Basically, there are two ways to combine the classifiers based on their output (Polka, 2006). One is combing the class labels that are available from the classifier outputs. And the other is combining continuous output provided by a classifier for a given class which is interpreted as the degree of support given to that class.

We evaluate the following seven learning algorithms as *individual* classifiers that are used in our various ensemble-based systems: Support Vector Machines

(SVMs) (Vapnik, 2000) with linear and rbf kernels, C4.5 (Quinlan, 1993), Logistic Regression (LR) (Cessie & Houwelingen, 1992), REPTree, Classification and Regression Trees (CART) (Breiman, Friedman, & Olshen, 1984), and Multilayer Perceptron (MLP). We refer this group of classifiers as *individual* classifiers.

7.2.1.1 Combining Class Labels

When only the class labels are available from classifiers' outputs, we can combine them in various ways (Polka, 2006). Let us define the decision of the *t*th classifier as $d_{t,j} \in \{0, 1\}$, $t = 1 \ldots, T$ and $j = \{1, -1\}$, where T is the number of classifiers and *j* is the actual label of an instance (phishing or non-phishing in our case). If *t*th classifier chooses class ω_j,, then $d_{t,j} = 1$, and 0, otherwise. We employ the simplest and most commonly used majority voting technique to combine classifiers' outputs. Majority voting-based ensemble makes the correct decision if at least $\lfloor T/2 \rfloor + 1$ classifiers choose the correct label, where the floor function $\lfloor \cdot \rfloor$ returns the largest integer less than or equal to its argument. Detail analysis of majority voting approach can be found in (Kuncheva, 2005).

7.2.1.2 Combining Class Continuous Output

When each base classifier in ensemble system gives continuous output as the degree of support given to that class, then the output can be combined using algebraic combiners such as, mean rule, weighted average, max rule, sum rule, median rule, product rule, etc. We use the sum rule which appears often in the literature (Kotsiantis & Pinetlas, 2004).

Sum Rule: The support for ω_j is the sum of all classifiers' *j*th outputs,

$$\mu_j(x) = \sum_{t=1}^{T} d_{t,j}(x) \tag{7-1}$$

The candidate class ω_j which has the largest total support $\mu_j(x)$ is considered the final decision of the ensemble system.

7.2.2 Measuring Diversity

Diversity is the cornerstone of ensemble systems. A successful ensemble system can be realized by making each base classifier as unique as possible, particularly with respect to misclassified instances. There are several ways to achieve classifier diversity a detail survey on which can be found in (Kuncheva & Whitaker, 2003), (Polka, 2006). In this research study, we use different parameters for instance, *linear* and *rbf* kernel types for SVMs and entirely different type of classifiers (see Section 7.2.1) to achieve diversity.

Similarly, there are several measures to quantitatively assess diversity among the classifiers involved in ensemble system. Empirical studies by Kuncheva and Whitaker (2003) on 10 diversity measures exhibit reasonably strong relationships among themselves. We use the same notation described in (Kuncheva and Whitaker, 2003). For T classifiers, we can calculate $T(T\text{-}1)/2$ pair-wise diversity measures, and an overall diversity of the ensemble can be obtained by averaging these pair-wise measures. Given two hypotheses h_i and h_j, we use the notations:

	h_j is correct (1)	**h_j is incorrect (0)**
h_i is correct (1)	N^{11}	N^{10}
h_i is incorrect (0)	N^{01}	N^{00}

Table 7-1: A 2X2 table of the relationship between a pair of classifiers.

where N^{11} is the number of instances that are correctly classified by both classifiers, N^{10} is the number of instances correctly classified by h_i but incorrectly classified by h_j, and so on. Total $N = N^{00} + N^{01} + N^{10} + N^{11}$. Then we use the following two diversity measures.

Correlation coefficient ρ is measured as the correlation between two classifiers, T_i and T_j, defined as

$$\rho_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11} + N^{10})(N^{01} + N^{00})(N^{11} + N^{01})(N^{10} + N^{00})}} \tag{7-2}$$

Yule's ***Q*-Statistics** (Yule, 1900) diversity measure between two classifiers, T_i and T_j, is

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \quad (7\text{-}3)$$

For statistically independent classifiers, maximum diversity is obtained for $Q = 0$. Q varies between -1 and 1. Classifiers that tend to recognize the same instants correctly will have positive values of Q, and those which commit errors on different instances will have negative values of Q. Kuncheva and Whitaker (2003) recommend Q-Statistic based on ease of interpretation and easy to calculate over other measures.

For an ensemble of T classifiers, the averaged α_{avg} diversity measure over all pairs of classifiers is,

$$\alpha_{avg} = \frac{2}{T(T-1)} \sum_{i=1}^{T-1} \sum_{j=i+1}^{T} \alpha_{i,j} \quad (7\text{-}4)$$

For any two classifiers, Q and ρ have the same sign and it can be proved that $|\rho| \leq |Q|$ (Kuncheva & Whitaker, 2003).

7.3 Experiments and Results

Ensemble systems are generated from *individual* and *meta* classifiers using various techniques to combine outputs from the base classifiers. Figure 7-1 shows the graphical representation of our phishing URL and webpage detection methodology using ensemble based systems.

We used WEKA library (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) to carry out our experiments on a machine with 2 dual-core 2 GHz Intel processors and 4 GB RAM.

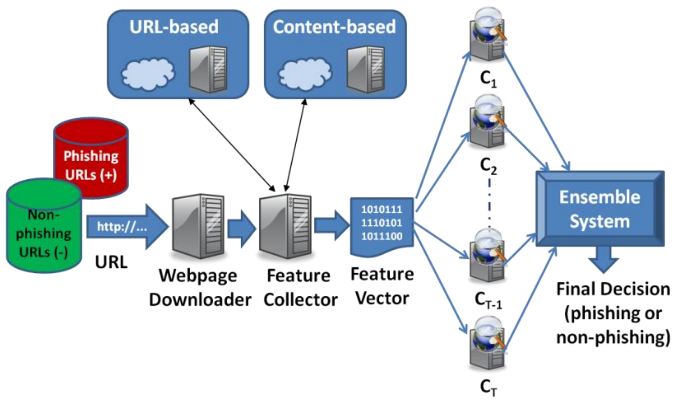

Figure 7-1: Graphical representation of ensemble-based phishing webpage detection methodology.

7.3.1 Data Sets

To run the experiments, we use the same data sources that we discuss in Chapter 4. Based on the features used to classify phishing and non-phishing websites, we generated two data sets. The first data set, we refer to it as DS1, contains 16,694 phishing URLs and 31,581 non-phishing URLs. DS1 data set uses only URL-based features. The second data set, we refer to it as DS2, contains the same phishing and non-phishing instances. However, DS2 augments the URL-based features with the content-based features. Most of the experiments are carried using 66/34 percentage split, where 66% of data set is used for training the classifiers and 34% for testing the models generated. Various ensembles are then evaluated to improve the test results.

7.3.2 Classifier Evaluation

While developing an effective ensemble based systems, it is important to evaluate the individual base classifiers that will potentially be used in the ensemble systems. The goal of the ensemble system is to combine classifiers that are as diverse as possible while also individually performing well in the given application. If the base classifiers used are themselves weak and have poor generalization capabilities, the final ensemble system produced from these classifiers – no matter what method used to combine the classifiers – may likely be not as robust and have a poor generalization capability.

In these experiments, we compare performance accuracies of *individual* classifiers and *meta* classifiers on our two data sets DS1 and DS2. Table 7-2 compares the performance results of *individual* classifiers on the two data sets and Table 7-3 compares the performance results of *meta* classifiers. The best values for each performance criteria are highlighted.

	DS1			**DS2**		
	Error	**FPR**	**FNR**	**Error**	**FPR**	**FNR**
C4.5	**0.75%**	0.46%	1.30%	**0.68%**	0.40%	1.20%
REPTree	0.94%	0.60%	1.60%	0.83%	0.44%	1.55%
CART	0.79%	0.46%	**1.13%**	0.75%	0.52%	**1.18%**
MLP	0.85%	**0.36%**	1.78%	1.06%	**0.37%**	2.36%
LR	0.99%	0.59%	1.74%	0.82%	0.48%	1.44%
SVM-rbf	1.45%	0.83%	2.63%	1.29%	0.61%	2.56%
SVM-lin	0.98%	0.49%	1.90%	0.80%	0.40%	1.57%

Table 7-2: Comparison of *individual* classifiers on DS1 and DS2 data sets.

	DS1			**DS2**		
	Error	**FPR**	**FNR**	**Error**	**FPR**	**FNR**
RF	**0.77%**	0.64%	**1.02%**	**0.76%**	0.53%	**1.20%**
Bagging	1.46%	**0.53%**	1.46%	0.77%	**0.38%**	1.50%
LB	1.74%	1.21%	2.73%	1.49%	0.91%	2.57%
AB	2.09%	1.15%	3.88%	1.98%	0.95%	3.93%
MB	2.56%	1.10%	5.30%	2.50%	0.94%	5.44%

Table 7-3: Comparison of *meta* classifiers on DS1 and DS2 data sets.

On both DS1 and DS2 data sets, C4.5 performed the best among the *individual* classifiers. SVM with rbf kernel achieved the worst results on both the data sets. Among the *meta* classifiers, Random Forests (RF) outperformed the rest in both the data sets. Bagging consistently yielded the best false positive rates on both the data sets. MultiBoosting (MB) performed the worst in both the data sets among the entire *individual* and *meta* classifiers. Interestingly, *individual* classifier C4.5 outperformed the best *meta* classifier Random Forests.

7.3.3 Diversity among Classifiers

In these experiments we calculate pair-wise diversity measures correlation coefficient ρ and Q-Statistic using Equations (7-2) and (7-3), respectively among all the classifiers. Table 7-4 and Table 7-5 show the pair-wise diversity measures among *individual* classifiers on data sets DS1 and DS2 respectively. Table 7-6 and Table 7-7 show the diversity measures among *meta* classifiers on data sets DS1 and DS2, respectively.

C4.5 and SVM-rbf have the best pair-wise diversity measures among all the pairs on both data sets. Essentially, these are also the best and worst performing classifiers for both the data sets.

	REPTree		**CART**		**MLP**		**LR**		**SVM-rbf**		**SVM-lin**	
	ρ	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***
C4.5	0.6927	0.9980	0.7334	0.9985	0.6046	0.9958	0.5487	0.9940	**0.5213**	**0.9933**	0.5576	0.9944
REPTree	1.0	1.0	0.7303	0.9984	0.6142	0.9958	0.5191	0.9916	**0.4888**	**0.9889**	0.5400	0.9925
CART			1.0	1.0	0.6045	0.9958	0.5403	0.9934	**0.4663**	**0.9892**	0.5421	0.9935
MLP					1.0	1.0	0.7212	**0.9981**	**0.6760**	0.9982	0.7571	0.9987
LR							1.0	1.0	**0.6323**	**0.9958**	0.8530	0.9995
SVM-rbf									1.0	1.0	**0.7068**	**0.9981**

Table 7-4: ρ and Q-Statistic diversity measures among individual classifiers on DS1 data.

	REPTree		**CART**		**MLP**		**LR**		**SVM-rbf**		**SVM-lin**	
	ρ	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***
C4.5	0.6047	0.9963	0.6707	0.9977	0.5189	0.9933	0.5435	0.9945	**0.4964**	**0.9925**	0.5654	0.9971
REPTree	1.0	1.0	0.6389	0.9969	0.5746	0.9945	0.5369	0.9935	**0.5265**	**0.9924**	0.5637	0.9946
CART			1.0	1.0	0.5153	0.9925	0.5495	0.9943	**0.5043**	**0.9920**	0.5775	0.9953
MLP					1.0	1.0	**0.5261**	**0.9925**	0.8014	0.9991	0.7033	0.9981
LR							1.0	1.0	**0.5366**	**0.9930**	0.7423	0.9986
SVM-rbf									1.0	1.0	**0.6740**	**0.9980**

Table 7-5: ρ and Q-Statistic diversity measures among individual classifiers on DS2 data set.

Random Forests (RF) and MultiBoosting (MB) have the best pair-wise diversity measures on both DS1 and DS2 data sets among the *meta* classifiers. Notice that these are also the best and worst performing classifiers among *meta* classifiers on both data sets (see Table 7-2 and Table 7-3).

	Bagging		**LB**		**AB**		**MB**	
	ρ	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***
RF	0.5589	0.9945	0.4194	0.9856	0.3954	0.9836	**0.3512**	**0.9778**
Bagging	1.0	1.0	0.4999	0.9912	0.4728	0.9902	**0.4129**	**0.9853**
LB			1.0	1.0	0.8545	0.9994	**0.7553**	**0.9984**
AB					1.0	1.0	**0.8880**	**0.9998**

Table 7-6: ρ and Q-Statistic diversity measures among *meta* classifiers on DS1 data set.

	Bagging		**LB**		**AB**		**MB**	
	ρ	***Q***	***ρ***	***Q***	***ρ***	***Q***	***ρ***	***Q***
RF	0.5383	0.9940	0.4409	0.9877	0.4051	0.9849	**0.3856**	**0.9839**
Bagging	1.0	1.0	0.5141	0.9927	0.5036	0.9938	**0.4690**	**0.9937**
LB			1.0	1.0	0.8173	0.9993	**0.7159**	**0.9984**
AB					1.0	1.0	**0.8796**	**0.9999**

Table 7-7: ρ and Q-Statistic diversity measures among *meta* classifiers on DS2 data set.

7.3.4 Hybrid Systems Using Majority Voting

Combining *Individual* Classifiers: In these experiments, we combine *individual* classifiers using majority voting method. Hybrid systems using majority voting method works best with odd number of classifiers. We experiment with the top 3, 5, and 7 classifiers based on their classification performance (as shown in Table 7-2) and show the results in Table 7-8.

Classification accuracies given by 3 different combinations of classifiers are statistically similar on both the data sets. Interestingly, ensemble of top 3 and top 5 classifiers yielded the same best error rates on both the data sets. The best error rate (0.71%) achieved by an ensemble of top individual classifiers is slightly better than the best error rate (0.75%) achieved by C4.5 on DS1 data set. Similarly, the best error rate (0.58%) achieved by an ensemble of top 3 classifiers is better than the best error rate of 0.68% achieved by C4.5 on DS2 data set.

	DS1					DS2				
	Error	**FPR**	**FNR**	ρ_{avg}	Q_{avg}	**Error**	**FPR**	**FNR**	ρ_{avg}	Q_{avg}
All 7 Classifiers	0.76%	0.35%	1.53%	**0.623**	**0.995**	0.65%	**0.27%**	1.37%	0.589	**0.995**
Top 5 Classifiers	**0.71%**	**0.34%**	1.43%	0.641	0.996	**0.58%-**	0.30%	1.13%	0.600	0.996
Top 3 Classifiers	**0.71%**	0.37%	**1.36%**	0.658	0.997	**0.58%-**	0.30%	**1.11%**	**0.580**	**0.995**

+,- statistically significant degradation or improvement over the best result given by the base classifier

Table 7-8: Comparison of hybrid systems using majority voting on different number of individual classifiers on data sets DS1 and DS2.

Combining *Meta* Classifiers: In these experiments, we combine top *meta* classifiers based on their classification performance (as shown in Table 7-3) using majority voting technique and compare the results in Table 7-9.

	DS1					DS2				
	Error	**FPR**	**FNR**	ρ_{avg}	Q_{avg}	**Error**	**FPR**	**FNR**	ρ_{avg}	Q_{avg}
All 5 Classifiers	1.74%	1.16%	2.82%	0.561	0.991	1.50%	0.87%	2.70%	0.567	0.993
Top 3 Classifiers	**0.81%**	**0.52%**	**1.36%**	**0.493**	**0.990**	**0.82%**	**0.44%**	**1.55%**	**0.498**	**0.992**

Table 7-9: Comparison of hybrid systems using majority voting on different number of *meta* classifiers on data sets DS1 and DS2.

The ensemble results using top 3 and top 5 classifiers based on their classification performance, however, didn’t yield better error rates compared to the best error rates of 0.77% and 0.76% by Random Forests (RF) alone on DS1 and DS2 data sets, respectively.

7.3.5 Hybrid Systems Using Sum Rule

Combining *Individual* Classifiers: In these experiments, we combine *individual* classifiers using the sum rule and compare their results in Table 7-10.

Ensemble of top 4 classifiers yielded at best 0.69% error rate on DS1 data set. Difference in error rates using ensemble of all other top classifiers are statistically insignificant on both data sets, however. Interestingly, the average diversity measures are not the best for this group of classifiers. The best diversity measure is achieved with the ensemble of all 7 classifiers which yielded the worst error rate of 0.76%, however. On DS2, ensembles of top 3 and top 5 classifiers achieve the best error rate of 0.58%. Top 3 classifiers also have the best correlation coefficient on this data set. Note that the best results achieved with this approach are better than the results from *individual* classifiers on both the data sets.

Combining *Meta* Classifiers: In these experiments, we combine *meta* classifiers using the sum rule. Based on the performance results of each *meta* classifier, we combine all possible top classifiers and compare their results in Table 7-11.

	DS1					DS2				
	Error	**FPR**	**FNR**	ρ_{avg}	Q_{avg}	**Error**	**FPR**	**FNR**	ρ_{avg}	Q_{avg}
All 7 Classifiers	0.76%	0.35%	1.53%	**0.623**	**0.995**	0.65%	0.27%	1.37%	0.589	**0.995**
Top 6 Classifiers	0.72%	0.37%	1.37%	0.640	0.996	0.61%	**0.25%**	1.29%	0.589	**0.995**
Top 5 Classifiers	0.71%	**0.34%**	1.43%	0.641	0.996	**0.58%-**	0.30%	1.13%	0.600	0.996
Top 4 Classifiers	**0.69%**	0.38%	**1.29%**	0.669	0.997	0.60%	0.28%	1.20%	0.609	0.996
Top 3 Classifiers	0.71%	0.37%	1.36%	0.658	0.997	**0.58%-**	0.30%	**1.11%**	**0.580**	**0.995**
Top 2 Classifiers	0.73%	0.43%	**1.29%**	0.733	0.999	0.71%	0.45%	1.20%	0.671	0.998

+,- statistically significant degradation or improvement over the best result given by the base classifier

Table 7-10: Comparison of hybrid systems using sum rule on different number of *individual* classifiers on data sets DS1 and DS2.

Ensemble of top 2 *meta* classifiers achieved the best error rates of 0.71% and 0.66% on data sets DS1 and DS2, respectively. Interestingly, we noticed this pattern on both the data sets: as the number of base classifiers increased, the error rates of the ensemble system also increased. Also, these results are better than that achieved by each *meta* classifiers.

	DS1					DS2				
	Error	**FPR**	**FNR**	ρ_{avg}	Q_{avg}	**Error**	**FPR**	**FNR**	ρ_{avg}	Q_{avg}
All 5 Classifiers	1.74%	1.17%	2.82%	0.5608	**0.991**	1.52%	0.87%	2.77%	0.567	0.993
Top 4 Classifiers	1.15%	0.63%	2.11%	0.5335	0.991	0.97%	0.45%	1.97%	0.537	0.992
Top 3 Classifiers	0.81%	0.52%	1.36%	**0.4928**	0.991	0.82%	0.44%	1.55%	**0.498**	**0.992**
Top 2 Classifiers	**0.71%-**	**0.44%**	**1.23%**	0.5589	0.995	**0.66%-**	**0.32%**	**1.32%**	0.538	0.994

+,- statistically significant degradation or improvement over the best result given by the base classifier

Table 7-11: Comparison of hybrid systems using sum rule on different number of *meta* base classifiers on data sets DS1 and DS2.

We also combined the classifiers based on the best pair-wise diversity measures. Surprisingly, the results yielded were worse. For ensemble of *individual* classifiers, we combined C4.5 and SVM-rbf. Using the sum rule, this pair yielded error rates of 1.57% on DS1 and 1.41% on DS2 data set. For ensemble of *meta* classifiers, we combined Random Forests (RF) and MultiBoosting (MB) using sum rule which resulted error rates of 1.81% on DS1 and 1.84% on DS2 data set.

7.3.6 Hybrid Systems Using Model-Based Features

Certain classifiers may consistently correctly classify or consistently misclassify certain instances. Instead of using the typical majority voting on the output of the classifiers in the ensemble system, one may wonder if we can automatically learn this phenomenon using a classifier. Or more specifically, given an ensemble of classifiers operating on a set of data, can we map the outputs of these classifiers to their true classes? We investigate this question in these experiments.

In this approach, which is similar to Wolpert's stacked generalization (Wolpert, 1992), an ensemble of classifiers are first created, whose outputs are used as inputs to a second level classifier to learn the mapping between the ensemble outputs and the actual correct classes.

We experiment with various number and combinations of base classifiers and use C4.5 as the second level classifier to learn from the input/output pairs obtained from base classifiers. The input vector thus generated for the second level classifier are simply binary features stating whether the corresponding base classifier classified the given instance as +1 (phishing) or -1 (non-phishing). We present these results in Table 7-12.

Base Classifiers	**DS1**					**DS2**				
	Error	**FPR**	**FNR**	ρ_{avg}	Q_{avg}	**Error**	**FPR**	**FNR**	ρ_{avg}	Q_{avg}
7 *Individual* Classifiers	0.71%	**0.40%**	1.30%	0.623	0.995	**0.55%-**	0.33%	0.97%	0.589	0.995
5 *Meta* Classifiers	0.85%	0.65%	**1.21%**	**0.561**	**0.991**	0.62%	0.49%	**0.96%**	0.567	**0.993**
12 (7 *Individual* + 5 *Meta*) Classifiers	**0.70%**	0.38%	1.30%	0.571	0.993	0.58%-	**0.29%**	1.15%	**0.556**	0.993

+,- statistically significant degradation or improvement over the result given by the top base classifier

Table 7-12: Performance results on hybrid systems using model based features with varying number of base classifiers.

Ensemble of 12 classifiers (including both *individual* and *meta*) achieves at best 0.7% error rate on DS1 data set. Whereas, ensemble of 7 *individual* classifiers achieves at best 0.6% error rate on DS2. These results are statically significant improvement over the results by *individual* classifiers.

7.4 Related Work

Most of the related works described in previous chapters are also related to this study.

Miyamoto et al. (Miyamoto, Hazeyama, & Kadobayashi, 2008) used AdaBoost algorithm to 8 heuristics proposed by Zheng et al. (2007) and achieved 97% accuracy in detecting phishing websites.

Ensemble based systems have been used in phishing email and webpage classification, see e.g., (Toolan & Carthy, 2009), (Sanglerdinglapachai & Rungsawang, 2010), and (Saberi, Vahidi, & Bidgoli, 2007).

Though the goals are different, ensemble approaches have been applied to intrusion detection systems (Mukkamala, Sung, & Abraham, 2003), (Chebrolu, Abraham, & Thomas, 2004).

7.5 Conclusions

In this chapter, we evaluated ensemble-based expert systems for classifying phishing websites. All the classifiers employed achieved consistently better results on the data set with full features (DS2). Though ensemble-based systems didn't achieve drastic improvement, we demonstrated that the ensemble-based systems, when combined strategically, outperform *individual* base classifiers in the application of phishing webpage classification.

Using sum rule, an ensemble of top 4 *individual* classifiers achieved at best 0.69% error rate on the data set using URL-based features (DS1). An ensemble of model-based features with 7 *individual* classifiers achieved the best error rate of 0.55% on the data set using URL and content-based features (DS2). Though base classifiers' diversity is the cornerstone of ensemble-based system and has been the focus of several studies – (Optiz & Maclin, 1999), (Kuncheva, 2005), (Kuncheva & Whitaker, 2003), etc. – neither the ensemble systems with the best diversity produced the best classification accuracy nor the ensemble system with the best classification accuracy provided the best diversity measure in our context. Classifiers with the best accuracy results when combined provided consistently better performance results compared to each base classifier used in the ensemble system.

Some ingenious phishing websites are extremely difficult to detect even for human experts (Dhamija, Tygar, & Hearst, 2006); while some are fairly easy. It might be an interesting research area to categorize the phishing webpages based on the complexity of attack tactics used and focus on the complex ones.

Bibliography

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. *In Proceedings of 2nd Annual eCrime Researchers Summit (eCrime '07)* (pp. 60-69). Pittsburgh, USA: ACM.

Anderberg, M. R. (1973). *Cluster analysis for applications.* London: Academic Press.

APWG. (2006). *Phishing Activity Trends Report.* APWG.org: http://www.antiphishing.org/reports/apwg_report_mar_06.pdf.

APWG. (2009). *Phishing Activity Trends Report.* APWG.org: http://www.antiphishing.org/reports/apwg_report_Q4_2009.pdf.

APWG. (2010). *Phishing Activity Trends Report.* apwg.org: http://www.antiphishing.org/reports/apwg_report_Q1_2010.pdf.

APWG. (2011). *Phishing Activity Trends Report- 2nd Half 2010.* APWG.org: http://apwg.org/reports/apwg_report_h2_2010.pdf.

AVG Security Toolbar. (n.d.). Retrieved June 10, 2011, from http://www.avg.com/product-avg-toolbar-tlbrc#tba2

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C., & Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* , 412-424.

Basnet, R. B. (2010a, September 19). *Interpreting JavaScript.* Retrieved from Google Site: http://sites.google.com/site/rambasnet/anatomy-of-phishing-attacks/javascript-deobfuscation-technique

Basnet, R. B. (2010b, May). *PyLongURL - Python library for longurl.org*. Retrieved from Google Code: http://code.google.com/p/pylongurl/; 2010

Basnet, R. B., & Sung, A. H. (2010). Classifying phishing emails using confidence-weighted linear classifiers. *International Conference on Informaiton Security and Aritificial Intelligence* (pp. 108-112). Chengdu, China: IEEE.

Basnet, R. B., Mukkamala, S., & Sung, A. H. (2008). Detection of phishing attacks: A machine learning approach. In B. Prasad, *Studies in Fuzziness and Soft Computing* (pp. 373-383). Springer.

Basnet, R. B., Sung, A. H., & Liu, Q. (2011). Rule-based phishing attack detection. *In Proceedings of the International Conference on Security and Management (SAM '11).* Las Vegas, USA.

Basnet, R. B., Torres, G. J., Sung, A. H., & Ribeiro, B. (2009). Translation-based foreign language text categorization. *Unpublished.*

Bergholz, A., Beer, J. D., Glahn, S., Moens, M.-F., Paab, G., & Strobel, S. (2010). New filtering approaches for phishing email. *Journal of Computer Security , 18* (1), 7-35.

Berners-Lee, T., Masinter, L., & McCahill, M. (1994, December). *Uniform Resource Locators (URL).* Retrieved Januar 17, 2011, from faq.org: http://www.faqs.org/rfcs/rfc1738.html

Brannick, M. (n.d.). *Logistic regression*. Retrieved 02 27, 2011, from http://luna.cas.usf.edu/~mbrannic/files/regression/Logistic.html

Breiman, L. (1996). Bagging Predictors. *Machine Learning , 24*, 123-140.

Breiman, L. (2001). Random Forests. *Machine Learning , 45* (1), 5-32.

Breiman, L., Friedman, J. H., & Olshen, R. A. (1984). *Classification and Regression Trees.* Belmont, California: Wadsworth International Group.

Cessie, S. l., & Houwelingen, J. C. (1992). Ridge Estimators in Logistic Regression. *Applied Statistics , 41* (1), 191-201.

Chan, C., & King, I. (2004). Using biased support vector machine to improve retrieval result in Image Retrieval with self-organizing map. *In Proceedings of International Conference on Neural Information Processing* (pp. 714-719). Heidelberg: Springer.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM -- A Library for Support Vector Machines*. Retrieved 8 10, 2010, from CSIE@NTU: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Chang, C.-C., & Lin, C.-J. (2003). *LOOMS: leave-one-out model selection for support vector machines*. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/looms/

Chapelle, O., & Vapnik, V. (2000). Model selection for support vector machines. *Advances in Neural Information Processing Systems 12*. Paris, France.

Chebrolu, S., Abraham, A., & Thomas, J. P. (2004). Feature deduction and ensemble design of intrusion detection systems. *Computers & Security* .

Chen, Y., & Lin, C. (2003). *Combining SVMs with Various Feature Selection Strategies.* Retrieved January 25, 2010, from http://www.csie.ntu.edu.tw/~cjlin/papers/features.pdf

Cherkassy, V. (2002). Model complexity control and stastistical learning theory. *Journal of Natural Computing* , 109-133.

Chou, N., Ledesma, R., Teraguchu, Y., Boneh, D., & Mitchell, J. C. (2004). Client-side defense against web-based identity theft. *In Proceedings of NDSS.*

Cova, M., Kruegel, C., & Vigna, G. (2010). Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. *International World Wide Conference* (pp. 281-290). Releigh, North Carolina: ACM.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research* (7), 551-585.

Cross-site Scripting - XSS. (n.d.). Retrieved March 12, 2011, from http://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. *In Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 581-590). Montreal, Canada: ACM.

Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-Weighted linear classification. *In Proceedings of the International Conference on Machine Learning (ICML)* (pp. 264-271). Omnipress.

Egan, J. (1975). *Signal Detection Theory and ROC Analysis*. New York: Academic Press.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). *LIBLINEAR -- A library for large linear classification*. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to Detect Phishing Emails. *In Proceedings of the 16th International Conference on World Wide Web (WWW'07)* (pp. 649-656). Banff, Alberta, Canada: ACM.

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. *Thirteenth International Conference on Machine Learning*, *14*, pp. 148-156. San Francisco.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: A Statistical view of Boosting. *The Annals of Statistics , 28* (2), 337-407.

Garera, S., Provos, N., Chew, M., & Rubin, A. (2007). A framework for detection and measurement of phishing attacks. *In Proceedings of the 5th ACM Workshop on Recurring Malcode (WORM '07)* (pp. 1-8). New York: ACM.

Google Safe Browsing API. (n.d.). Retrieved June 12, 2010, from Google Code: http://code.google.com/apis/safebrowsing/

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter , 11* (1), 10-18.

hpHosts. (2005). *hpHosts Online - Simple, Searchable & FREE!* Retrieved June 2011, from hpHosts: http://hosts-file.net/

Ibb, W., & Langley, P. (1992). Induction of one-level decision trees. *In Proceedings of the 9th International Conference on Machine Learning* (pp. 233-240). Aberdeen: Morgan Kaufmann.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *In Proceedings of the Tenth European Conference on MachineLearning (ECML-98)* (pp. 137-142). Springer.

Korpela, J. “. (2003, September 28). *Methods GET and POST in HTML forms - What's the difference?* Retrieved June 19, 2010, from http://www.cs.tut.fi/~jkorpela/forms/methods.html

Kotsiantis, S. B., & Pinetlas, P. E. (2004). Combining Bagging and Boosting. *International Journal of Computational Intelligence , 1* (4), 324-333.

Kuncheva, L. I. (2005). *Combining Pattern Classifiers, Methods and Algorithms.* New York, NY: Wiley Interscience.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of Diversity in Classifier Ensembles and Their Relationship with Ensemble Accuracy. *Machine Learning , 51*, 181-207.

Lee, J., & Lin, C. (2000). *Automatic model selection for support vector machines.* Taiwan: Department of Computer Science and Information Engineering, National Taiwan University.

Lewis, D. D., Yand, Y., Rose, T., & Li, F. (2004). A new benchmark collection for text categorization research. *Journal of Machine Learning Research (JMLR)* , 361-397.

Ludl, C., McAllister, S., Kirda, E., & Kruegel, C. (2007). On the Effectiveness of Techniques to Detect Phishing Sites. *DIMVA '07 - the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 20-39). Springer-Verlag.

Ma, J., Saul, L. K., Safage, S., & Voelker, G. M. (2009). Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs. *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 1245-1253). Paris, France.

Ma, J., Saul, L. K., Safage, S., & Voelker, G. M. (2009). Identifying suspicious URLs: An application of large-scale online learning. *In Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, (pp. 681-688). Montreal, Canada.

Manning, C., Raghavan, P., & Schutze, H. (2008). *Introduction to information retrieval.* Cambridge University Press.

McAfee SiteAdvisor Software. (n.d.). Retrieved July 15, 2011, from Website Safety Ratings and Secure Search: http://www.siteadvisor.com

McCall, T. (2007, 12 17). *Press Releases.* Retrieved 10 6, 2010, from Gartner: http://www.gartner.com/it/page.jsp?id=565125

Medvet, E., Kirda, E., & Kruegel, C. (2008). Visual-similarity-based phishing detection. *In Proceedings of the 4th International Conference on Security and Privacy in Communication Networks.* Istanbul, Turkey: ACM.

MessageLabs Intelligence Reports. (n.d.). Retrieved May 25, 2011, from Symantec.cloud: http://www.messagelabs.com/mlireport/MLI_2011_05_May_FINAL-en.pdf

Messmer, E. (2008, April 24). *iFrame attacks surge, security firm says.* Retrieved May 15, 2011, from Network World: http://www.networkworld.com/news/2008/042408-iframe-attacks-surge.html

Microsoft Security Intelligence Report. (2011). Retrieved June 5, 2011, from http://www.microsoft.com/security/sir/default.aspx

Milletary, J. (n.d.). *Technical Trends in Phishing Attacks.* Retrieved 07 15, 2010, from US-CERT: http://www.us-cert.gov/reading_room/phishing_trends0511.pdf

Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. (2008). An evaluation of machine learning-based methods for detection of phishing sites. *In Proceedings of the 15th International Conferences on Advances in Neuro-Information Processing*, (pp. 539-546). Auckland, New Zealand.

Mukkamala, S., Sung, A. H., & Abraham, A. (2003). Intrusion detection using ensemble of soft computing paradigms. *Third International Conference on Intelligent Systems Design and Applications, Advances in Soft Computing* (pp. 239-248). Springer.

Nazario, J. (n.d.). *Phishingcorpus homepage*. Retrieved February 10, 2010, from monkey.org: http://monkey.org/~jose/wiki/doku.php?id=PhishingCorpus

Netcraft Toolbar. (n.d.). Retrieved October 5, 2010, from NETCRAFT: http://toolbar.netcraft.com/

NLTK. (n.d.). Retrieved 2010, from Natural Language Toolkit: http://www.nltk.org/

Optiz, D., & Maclin, R. (1999). Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research , 11*, 169-198.

Oresch, M. (n.d.). *Snort - Lightweight Intrusion Detection for Networks*. Retrieved from http://assets.sourcefire.com/snort/developmentpapers/Lisapaper.txt

Pan, Y., & Ding, X. (2006). Anomaly Based Web Phishing Page Detection. *In Proceeedings of the 22nd Annual Computer Security Application Conference (ACSAC '06)*, (pp. 381-392). Miami Beach, FL, USA.

PhishTank. (n.d.). Retrieved 2010, from PhishTank - Out of the Net, into the Tank: http://www.phishtank.com/developer_info.php

Polka, R. (2006). Ensemble Based Systems in Decision Making. *Circuits and Systems Magazine* , *6* (3), 21-45.

Quinlan, J. R. (1993). *C4.5 programs for machine learning.* San Mateo, USA: Morgan Kaufmann Publishers.

Richardson, L. (n.d.). *Beautiful Soup.* Retrieved May 19, 2010, from crummy.com: http://www.crummy.com/software/BeautifulSoup/

Rish, I. (2001). *An empirical study of the Naïve Bayes classifier.* Retrieved October 25, 2010, from IBM Research Report: http://www.research.ibm.com/people/r/rish/papers/RC22230.pdf

Rosenblatt, F. (1958). The Perceptron: A Probalisitic Model for Information Storage and Organization in the Brain. *Psychological Review* , 686-408.

Saberi, A., Vahidi, M., & Bidgoli, B. M. (2007). Learn to Detect Phishing Scams Using Learning and Ensemble Methods. *International Conferences on Web Intelligence and Intelligent Agent Technology Workshops* (pp. 311-314). IEEE.

Sanglerdinglapachai, N., & Rungsawang, A. (2010). Web phishing detection using classifier ensemble. *2th International Conference on Information Integration and Web-based Applications & Services* (pp. 210-215). ACM.

Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., & Zhang, C. (2009). An empirical analysis of phishing blacklists. *In Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS 2009).* Mountain View, CA, USA.

SmartScreen Filter. (n.d.). Retrieved 2011, from Microsoft Windows: http://windows.microsoft.com/en-US/internet-explorer/products/ie-9/features/smartscreen-filter

SpamAssassin for Win32. (n.d.). Retrieved July 1, 2010, from SAwin32: http://sawin32.sourceforge.net/

SpamAssassin public corpus. (n.d.). Retrieved August 17, 2010, from The Apache SpamAssassisn Project: http://spamassassin.apache.org/publiccorpus/

SpoofStick Home. (n.d.). Retrieved July 25, 2011, from http://www.spoofstick.com

Stamm, S., & Raman, Z. (2006, December). *Drive-By Pharming.* Retrieved November 18, 2010, from http://www.cs.indiana.edu/pub/techreports/TR641.pdf

Statistics about phishing activity and PhishTank usage. (n.d.). Retrieved from PhishTank: http://www.phishtank.com/stats.php

StopBadware. (n.d.). Retrieved June 12, 2010, from IP Address Report - Top 50 by number of reported URLs: Available from: http://stopbadware.org/reports/ip

Toolan, F., & Carthy, J. (2009). Phishing detection using classifier ensembles. *eCrime Researchers Summit*, (pp. 1-9). Tacoma, WA.

URL Syntactic Components. (n.d.). Retrieved January 17, 2011, from An Internet Enclyopedia: http://www.freesoft.org/CIE/RFC/1808/3.htm

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory.* Springer.

Webb, G. I. (2000). MultiBoosting: A Technique for Combining Boosting and Wagging. *Machine Learning , 40* (2), 159-196.

Weynin, L., Huan, G., Xiaoyue, L., Min, Z., & Deng, X. (2005). Detection of phishing webpages based on visual similarity. *In Proceedings of the 14th International Conference on World Wide Web (WWW '05)* (pp. 1060-1061). New York, USA: ACM.

Whittaker, C., Ryner, B., & Nazif, M. (2010). Large-scale automatic classification of phishing pages. *In Proceedings of 17th Annual Network and Distributed System Security Symposium.* San Diego, USA.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks , 5* (2), 241-259.

Yule, G. (1900). On the association of attributes in statistics. *Phil. Trans. A* , 257–319.

Zhang, Y., Hong, J., & Cranor, L. (2007). CANTINA: A content-based approach to detecting phishing web sites. *In Proceedings of the 16th International Conference on World Wide Web (WWW'07)* (pp. 639-648). Banff, Canada: ACM.