Learning to Detect Phishing Emails

Ian Fette **Norman Sadeh** **Anthony Tomasic**

June 2006
CMU-ISRI-06-112

Institute for Software Research International
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This report also appears as Carnegie Mellon Cyber Laboratory
Technical Report CMU-CyLab-06-012

The work reported herein has been supported in part under the NSF Cyber Trust initiative (Grant #0524189) and in part under ARO research grant D20D19-02-1-0389 (“Perpetually Available and Secure Information Systems”) to Carnegie Mellon University’s CyLab. The authors would also like to thank Lorrie Cranor, Jason Hong, Alessandro Acquisti, Julie Downs, Sven Dietrich, Serge Egelman, Mandy Holbrook, Ponnurangam Kumaraguru, and Steve Sheng. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

Keywords: phishing, email, filtering, semantic attacks, learning

Abstract

There are an increasing number of emails purporting to be from a trusted entity that attempt to deceive users into providing account or identity information, commonly known as "phishing" emails. Traditional spam filters are not adequately detecting these undesirable emails, and this causes problems for both consumers and businesses wishing to do business online. From a learning perspective, this is a challenging problem. At first glance, the problem appears to be a 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 real emails. We propose a new method for detecting these malicious emails called PILFER. By incorporating features specifically designed to highlight the deceptive methods used to fool users, we are able to accurately classify over 92% of phishing emails, while maintaining a false positive rate on the order of 0.1%. These results are obtained on a dataset of approximately 860 phishing emails and 6950 non-phishing emails. The accuracy of PILFER on this dataset is significantly better than that of SpamAssassin, a widely-used spam filter.

1 Introduction

Phishers launched a record number of attacks in January 2006, as reported by the Anti-Phishing Working Group [1]. These attacks often take the form of an email that purports to be from a trusted entity, such as eBay or PayPal. The email states that the user needs to provide information, such as credit card numbers, identity information, or login credentials, often to correct some alleged problem supposedly found with an account. Some number of users fall for these attacks by providing the requested information, which can lead to fraudulent charges against credit cards, withdrawals from bank accounts, or other undesirable effects.

The first attempts at applying learning to these problems took the form of browser toolbars, such as the Spoofguard [2] and Netcraft [3] toolbars. Our research group is currently conducting a study to determine the accuracy of these and other toolbars more precisely. Preliminary results indicate that a large percentage of phishing emails make it past these toolbars. One of the biggest drawbacks from a learning perspective is that toolbars in web browsers have access to less information. In some sense, users have already partially fallen for the attack by clicking on a link in an email, and this could potentially expose the user to spyware and malware loaded by attacks on insecure web browsers. Furthermore, now that detection is being done in the browser, the contextual information and features from the email are no longer available. In theory, this reduced amount of information should cause a similar reduction in the accuracy that such systems are able to achieve. Aside from accuracy, toolbars suffer from a number of other problems.

Toolbars usually prompt users with a dialog box, which many users will simply dismiss or misinterpret, or worse yet these warning dialogs can be intercepted by user-space malware [4]. Finally, by waiting until the user clicks on a link and goes to a website to address the problem, we have allowed the user to be distracted by illegitimate email, and thereby failed to prevent some loss of productivity and concentration on more important tasks at hand. Newer solutions include attempts to warn users inside of their mail user agent ("client"). Examples of this include Thunderbird 1.5 [5]. However, without formal studies and a fully detailed description of how such filters work, it is difficult to assess their impact. Furthermore, although these solutions are able to make use of the context of the email in which the phishing attack occurs, the user is often still paying the mental price of reading the email, the warning can still be dismissed (either by the user or by malware), and the ISP is still paying the cost of transmission for the junk email.

Ideally, phishing detection algorithms require minimal user interaction, either as a server-side filter to compliment existing spam filters, or as a filter running locally in the client. This approach has several benefits over other methods. First, by removing user interactions, there is no chance for the user to dismiss warning dialogs and proceed to provide information to an attacker. Second, contextual information is available in the email. Finally, by operating on the email rather than in the browser, server-side detection is possible. Server-side detection avoids paying a transmission cost for sending the email (and subsequent requests for images linked to in an HTML image) to the user, as well as the cost of evaluating certain features (such as doing WHOIS lookups, whose results can be cached for re-use). We present in this an algorithm, which we call "PILFER" - phishing identification by learning on features of email received. Our implementation is not optimal. It does not make use of all the information potentially available to a server-side filter. However, we obtain high accuracy rates, and posit that further work in this area is warranted.

The remainder of this paper is organized in the following manner. Section 2 discusses previous approaches at filtering spam email, while section 3 gives an overview of learning and how we apply it to the task of classifying phishing emails. Section 4 covers the results of empirical evaluation, as well as some challenges presented therein, and in section 5 we present concluding remarks.

2 Background

Many people have proposed ways in which to eliminate spam emails (see, for example, [6, 7, 8, 9, 10, 11]) Many of these approaches use a naïve methodology, ranging from "bag-of-words" approaches, where the features of an email are the presence or absence of highly frequent and rare words, to analysis of the entropy of the messages. While these approaches looking at the text of the email appear to do well for spam, phishing messages still get through these filters. On some level, this makes sense, as phishing emails are designed to look as close as possible to a real, non-spam email that a legitimate company would (or already has) sent out. As such, it is our belief that to stop phishing emails, we need to look at features selected specifically to detect this class of emails.

Looking at class-specific features is not a new approach in email filtering. SpamAssassin [12], for instance, has a number of rules that try to detect features common in spam email that go beyond just the text of the email. Such tests include things like the ratio of pixels occupied by text to those occupied by images in a rendered version of the mail, presence of certain faked headers, and the like. Spamato [13] is another extensible filtering platform that ships with a number of advanced filters, such as Vipul's Razor [14] (a collaborative algorithm using both URLs and message hashes), that work in tandem to detect spam emails. Our contribution is a new approach focused on learning to detect phishing, aka semantic attacks. Our solution can easily be used in conjunction with existing spam filters. The solution significantly reduces the amount of phishing emails with minimal cost in terms of false positives.

3 Method

3.1 Learning Overview

PILFER is a machine-learning based approach to classifying emails. Classification is a heavily studied problem within the learning community, and classification problems typically consist of the following components. First, there is a notion of "classes", to which instances of data belong. In this context, we have two classes, namely the class of phishing emails, and the class of good ("ham") emails. Next, there is a notion of features. Features are properties of the instances being classified, and are the source of information upon which the classification decision is made. If we were trying to distinguish between birds and fish, we might use features such as whether or not the creature had wings, whether it had legs, and so forth. A learning algorithm is then used to create a model, which accepts input in terms of a set of assignments to the known features, and returns a class label as an output. This model is first trained by supplying a set of "training data", which is comprised of pairs of feature assignments and class labels. A separate set of "test data" is then

supplied to the model, and the predicted class of the data (phishing or ham) is compared to the actual class of the data to compute the accuracy of the model.

In PILFER, we first run a set of scripts to extract a set of ten features, which are described in the next section, subsection 3.2. Once the features are extracted, we train and test a classifier using 10-fold cross validation. (The dataset is divided into ten distinct parts. Each part is then tested using the other nine parts of the data as the training data. This ensures that the training data is separate from the test data, and is called "cross-validation".) For our reference implementation, of PILFER, we use a support vector machine as a classifier. A support vector machine creates a multi-dimensional decision boundary that tries to separate the margin (distance) between the closest instances of opposite classes, while penalizing misclassifications. Our implementation uses libsvm [15] - a freely available SVM library, with a slack penalty of 10, as our classifier. We also tried a number of different classifiers, including rule-based approaches, decision trees, and Bayesian approaches, but the overall accuracies of most of the classifiers were not different with statistical significance. Accuracies for some of these other classifiers are shown in appendix A. For a complete discussion of SVM and other classifiers, the reader is directed to a machine learning text such as [16] or [17].

3.2 Features

Some spam filters use hundreds of features to detect unwanted emails. We have tested a number of different features, and present in this paper a list of the ten features that are used in PILFER, which are either binary or continuous numeric features. As the nature of phishing attacks changes, additional features may become more powerful, and PILFER can easily be adapted by providing such new features to the classifier. At this point, however, we are able to obtain high accuracy with only ten features, which makes the decision boundaries less complex, and therefore less prone to over-fitting and faster to evaluate. We explain these features in detail below. Some of these features are already implemented in spam filters (namely the presence of IP-based URLs); these features are also a useful component of a phishing filter.

3.2.1 IP-based URLs

Some phishing attacks are hosted off of compromised PCs. These machines may not have DNS entries, and the simplest way to refer to them is by IP address. Companies rarely link to pages by an IP-address, and so such a link in an email is a potential indication of a phishing attack. As such, anytime we see a link in an email whose host is an IP-address (such as `http://192.168.0.1/paypal.cgi?fix_account`), we flag the email as having an IP-based URL. As phishing attacks are becoming more sophisticated, IP-based links are becoming less prevalent, with attackers purchasing domain names to point to the attack website instead. However, there are still a significant number of IP-based attacks, and therefore this is still a useful feature. This feature is binary.

3.2.2 Age of linked-to domain names

Phishers are learning not to give themselves away by using IP-based URLs. Name-based attacks, in which a phisher will register a similar or otherwise legitimate-sounding domain name (such as playpal.com or paypal-update.com) are increasingly common. These domains often have a limited life, however. Phishers may register these domains with fraudulently obtained credit cards (in which case the registrar may cancel the registration), or the domain may be caught by a company hired to monitor registrations that seem suspicious. (Microsoft, for instance, watches for domain name registrations involving any of their trademarks.) As such, the phisher has an incentive to use these domain names shortly after registration. We therefore perform a WHOIS query on each domain name that is linked to, and store the date on which the registrar reports the domain was registered. If this date is within 60 days of the date the email was sent, the email is flagged with the feature of linking to a "fresh" domain. This is a binary feature.

3.2.3 Nonmatching URLs

Phishers often exploit HTML emails, in which it is possible to display a link that says paypal.com but actually links to badsite.com. For this feature, all links are checked, and if the text of a link is a URL, and the HREF of the link is to a different host than the link in the text, the email is flagged with a "nonmatching URL" feature. Such a link looks like `<a href="http://www.badsite.com"> http://www.paypal.com</a>`. This is a binary feature.

3.2.4 "Here" links to non-modal domain

Phishing emails, often contain text like "Click here to restore your account access". In many cases, this is the most predominantly displayed link, and is the link the phisher intends the user to click. Other links are maintained in the email to keep the authentic feel, such as the link to a privacy policy, a link to the user agreement, and others. We call the domain most frequently linked to the "modal domain" of the email. If there is a link with the text "link", "click", or "here" that links to a domain other than this "modal domain", the email is flagged with a "here" link to a non-modal domain feature. This is a binary feature.

3.2.5 HTML emails

Most emails are sent as either plain text, HTML, or a combination of the two in what is known as a multipart/alternative format. The email is flagged with the HTML email feature if it contains a section that is denoted with a MIME type of text/html. (This includes many multipart/alternative emails). While HTML email is not necessarily indicative of a phishing email, it does make many of the deceptions seen in phishing attacks possible. For a phisher to launch an attack without using HTML is difficult, because in a plain text email there is virtually no way to disguise the URL to which the user is taken. Thus, the user still can be deceived by legitimate-sounding domain names, but many of the technical, deceptive attacks are not possible. This is a binary feature.

3.2.6 Number of links

The number of links present in an email is a feature. The number of links is the number of links in the html part(s) of an email, where a link is defined as being an ¡a¿ tag with a href attribute. This includes mailto: links. This is a continuous feature.

3.2.7 Number of domains

For all URLs that start with either `http://` or `https://`, we extract the domain name for the purpose of determining whether the email contains a link to a "fresh" domain. For this feature, we simply take the domain names previously extracted from all of the links, and simply count the number of distinct domains. We try to only look at the "main" part of a domain, e.g. what a person actually would pay to register through a domain registrar. It should be noted that this is not necessarily the combination of the top- and second-level domain. For instance, we consider the "main" part of `www.cs.university.edu` to be `university.edu`, but the "main" part of `www.company.co.jp` would be `company.co.jp`, as this is what is actually registered with a registrar, even though technically the top-level domain is .jp and the second-level domain is .co. This feature is simply the number of such "main" domains linked to in the email, and is a continuous feature.

3.2.8 Number of dots

There are a number of ways for attackers to construct legitimate-looking URLs. One of which is to have a subdomain, like `http://www.my-bank.update.data.com`. Another is to use a redirection script, such as `http://www.google.com/url?q=http://www.badsite.com/update.cgi`, which to the user may appear like a site hosted at google.com, but in reality will redirect the browser to badsite.com. In both of these examples, either by the inclusion of a URL into an open redirect script or by the use of a number of subdomains, there are a large number of dots in the URL. Of course, legitimate URLs also can contain a number of dots, and this does not make it a phishing URL, however there is still information conveyed by this feature, as its inclusion increases the accuracy in our empirical evaluations. This feature is simply the maximum number of dots ('.') contained in any of the links present in the email, and is a continuous feature.

3.2.9 Contains javascript

JavaScript is used for many things, from creating popup windows to changing the status bar of a web browser or email client. It can appear directly in the body of an email, or it can be embedded in something like a link. Attackers can use JavaScript to hide information from the user, and potentially launch sophisticated attacks. An email is flagged with the "contains javascript" feature if the string "javascript" appears in the email, regardless of whether it is actually in a `<script>` or `<a>` tag. This might not be optimal, but it makes parsing much simpler, especially when dealing with attacks that contain malformed HTML. This is a binary feature.

3.2.10 Untrained SpamAssassin Output

Many mail clients already have a spam filter in place, and as such it seems natural to leverage the ability of existing solutions in combating the phishing problem. We therefore include as a feature the result of whether SpamAssassin labels an email as spam, using the default threshold of 5 and an untrained SpamAssassin installation.

3.3 Testing SpamAssassin

SpamAssassin is a widely-deployed freely-available spam filter that is highly accurate in classifying spam emails. For comparison against PILFER, we classify the exact same dataset using SpamAssassin version 3.1.0, using the default thresholds and rules. The results reported for "untrained" SpamAssassin are obtained by simply treating the entire dataset as a test set, and not training on any emails. This represents an out-of-the-box install of SpamAssassin. (To be sure that SpamAssassin was untrained, we ran `sa-learn --clear` before testing the emails). The results reported for the "trained" SpamAssassin are from the same version, except that we now use 10-fold cross validation, where before each fold we clear SpamAssassin's internal database (by calling `sa-learn --clear`), and then train on all the emails in the train part of the fold and test on those in the test part of the fold.

4 Empirical Evaluation

4.1 Overview

In this section, we present the results of running PILFER and SpamAssassin on a dataset of emails, described in subsection 4.2. Certain challenges are present when trying to do post-hoc analysis of phishing attacks, the specifics and impact of which are discussed in subsection 4.3. Subsection 4.4 shows our results in classifying the dataset, and subsection 4.5 discusses the impact of the choice of classifier in PILFER.

4.2 Datasets

Two publicly available datasets were used to test our implementation: the ham corpora from the SpamAssassin project [18] (both the 2002 and 2003 ham collections, easy and hard, for a total of approximately 6950 non-phishing non-spam emails), and the publicly available phishingcorpus [19] (approximately 860 email messages). We use a series of short scripts to programmatically extract the above features, and store these in a database for quick reference. We label emails as being non-phishing if they come from the SpamAssassin ham corpora, and as phishing if they come from the phishingcorpus. It should be noted that the content of the phishingcorpus is less than ideal, and depending on your particular definition of phishing, it may contain emails that might not be considered phishing. For these experiments we used the entire dataset and did not re-label any of its contents.

4.3 Additional Challenges

There are a number of challenges posed by doing post-hoc classification of phishing emails. Most of these challenges only apply to the phishing emails in the dataset and materialize as a form of missing information, which has the net effect of increasing the false negative rate. Without the challenges outlined below, which are mostly artifacts of tesing after the fact as opposed to live in a real system, even better accuracy should be possible.

The age of the dataset poses the most problems, which is particularly relevant with the phishing corpus. Phishing websites are short-lived, often lasting only on the order of 48 hours. Some of our features can therefore not be extracted from older emails, making our tests difficult. For instance, in one of our features, we are interested in the age of domains linked to. We perform a WHOIS query to determine the date a domain was registered, and subtract this date from the date the email was sent according to its headers to determine its age. In many cases of phishing attacks, however, these domains are not still live at the time of our testing, resulting in missing information.

The average phishing site stays live for approximately 2.25 days [20]. After the phishing site is discovered and taken down by web hosts or authorities, the domain name may be turned over to the spoofed company, it may be allowed to simply expire, or it may be canceled by the registrar. In some cases, the domain used may have been purchased with a stolen credit card, in which case there is a good chance that the charge will be reversed and the domain name registration canceled. In any case, it becomes very hard to get accurate WHOIS data from a domain used in an attack so far in the past. The most recent email in the phishing dataset is from November 2005, fully six months prior to the writing of this paper.

Matters are further complicated by the fact that different registrars report different amounts of data for domains, and all the registrars seem to use a different format. For instance, a WHOIS query on example.de will usually return the date of last change, but will usually not give the date on which the domain was registered. A WHOIS query on example.com will usually return a create date, but the format of the result is entirely different than that used for example.de. This has a definite effect of introducing noise into our feature set. Of the 870 distinct domain names referenced in our data set, we were only able to programmatically extract registration dates for 505 of these as of when this paper was written.

4.4 Results

On our dataset, PILFER achieves an overall accuracy of 99%, with a false positive rate (non-phishing emails marked as phishing) of less than 1%. PILFER's false negative rate (missing a phishing email) on the dataset is on the order of 7-8%, which is approximately half that of SpamAssassin's. These results are compared in detail with those of SpamAssassin in Table 1. As seen in the table, either with or without the input from SpamAssassin, our classifier has orders of magnitude fewer false positives on the dataset, and approximately half the false negatives. Curiously, SpamAssassin seems to perform slightly worse when trained and tested under 10-fold cross validation. This might be due to the fact that the phishing emails SpamAssassin catches are triggering a number of sufficiently-weighted rules without explicit training, such as blacklist rules like DNS FROM RFC ABUSE, and the learning rules like Naïve Bayes might not actually help the classifier

Table 1: Accuracy of classifier compared with baseline spam filter

CLASSIFIER	False Positives	False Negatives
PILFER, with S.A. feature	0.12%	7.35%
PILFER, without S.A. feature	0.30%	8.40%
SpamAssassin (Untrained)	10.96%	14.35%
SpamAssassin (Trained)	11.13%	15.87%
PILFER OR SpamAssassin (Untrained)	11.12%	2.08%

Table 2: Percentage of emails matching the binary features

Feature	Non-Phishing Matched	Phishing Matched
Has IP link	0.06%	45.04%
Has "fresh" link	0.98%	12.49%
Has "nonmatching" URL	0.14%	50.64%
Has non-modal here link	0.82%	18.20%
Is HTML email	5.55%	93.47%
Contains JavaScript	2.30%	10.15%
Untrained S.A. Output	10.96%	85.65%

on this dataset as the text of phishing emails are often similar to the text of legitimate emails.

Table 2 shows the exact percentages of emails (by class) matching each of the seven binary features. All of the binary features are matched more frequently by phishing emails than by non-phishing emails. For the three non-binary features, their averages and standard deviations per-class are shown in Table 3. These features have higher mean values for phishing emails.

We also tested to see if the emails SpamAssassin classified as spam were a subset of those that PILFER detected, or separate emails. If one were to classify an email as phishing if identified by either our classifier (PILFER) or SpamAssassin (shown in Table 1 as "PILFER OR SpamAssassin"), one would detect 97.9% of the phishing emails, but would suffer a false positive rate of 11.1%. This comparison suggests that the emails detected by one classifier are not a strict subset of those detected by the other.

In summary, PILFER can be either deployed in a stand-alone configuration to catch a large percentage of phishing emails with very few false positives, or in conjunction with an existing spam filter such as SpamAssassin for even fewer false negatives. If a filter like SpamAssassin is already deployed, then adding PILFER has the advantage of significantly reducing the number of phishing emails making it to the user, while having no significant effect on the number of emails erroneously caught by the filtering system.

Table 3: Mean, standard deviation of the continuous features, per-class

Feature	$\mu_{\textbf{phishing}}$	$\sigma_{\textbf{phishing}}$	$\mu_{\textbf{non-phishing}}$	$\sigma_{\textbf{non-phishing}}$
Number of links	3.87	4.97	2.36	12.00
Number of domains	1.49	1.42	0.43	3.32
Number of dots	3.78	1.94	0.19	0.87

5 Concluding Remarks

In this paper, we have shown that it is possible to detect phishing emails with high accuracy by using a specialized filter, using features that are more directly applicable to phishing emails than those employed by general purpose spam filters. Although phishing is a subset of spam (after all, who asks to receive emails from a person pretending to be their bank for the purpose of fraud and identity theft?), it is characterized by certain unique properties that we have identified.

One might be inclined to think that phishing emails should be harder to detect than general spam emails. After all, phishing emails are designed to sound like an email from a legitimate company, often a company with which the attacker hopes the user has a pre-existing relationship. Models based on naïve assumptions, such as certain words like "viagra" being indicative of a class of un-desirable emails, no longer hold when the attackers are using the same words and the same overall "feel" to lure the user into a false sense of security. With that said, phishing emails present unique opportunities for detection that are not present in general spam emails.

In general spam emails, the sender does not need to misrepresent their identity. A company offering to sell "viagra" over the Internet does not need to convince potential buyers that they are a pharmacy that the user already has a relationship with, such as CVS or RiteAid. Instead, a spammer can actually set up a (quasi-)legitimate company called Pharmacy1283, and identify themselves as such, with no need to try to convince users that they are receiving a communication from their bank, or some other entity with which they have an established relationship. It is this mis-representation of sender identity that is key to the identification of phishing emails, and further work in the area should concentrate on features to identify this deceptive behavior.

There are a number of emerging technologies that could greatly assist phishing classification that we have not considered. For instance, Sender ID Framework (SIDF) [21] and DomainKeys [22], along with other such sender authentication technologies, should help to both reduce false positives and make detection of spoofed senders much simpler in the time to come. In the meantime, however, we believe that using features such as those presented here can significantly help with detecting this class of phishing emails. We are currently in the process of building a live filtering solution based around PILFER, which we will start making available to a small number of users for testing for further data collection and analysis.

References

[1] "Phishing activity trends report," Anti-Phishing Working Group, Tech. Rep., Jan. 2005. [Online]. Available: http://www.antiphishing.org/reports/apwg_report_jan_2006.pdf

[2] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, "Client-side defense against web-based identity theft." in *NDSS*, 2004. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Chou.pdf

[3] "Netcraft toolbar," 2006. [Online]. Available: http://toolbar.netcraft.com/

[4] A. Alsaid and C. J. Mitchell, "Installing fake root keys in a pc." in *EuroPKI*, 2005, pp. 227–239. [Online]. Available: http://dx.doi.org/10.1007/11533733_16

[5] "Mozilla thunderbird," 2006. [Online]. Available: http://www.mozilla.com/thunderbird/

[6] B. Leiba and N. Borenstein, "A multifaceted approach to spam reduction," in *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. [Online]. Available: http://www.ceas.cc/papers-2004/127.pdf

[7] W. Cohen, "Learning to classify English text with ILP methods," in *Advances in Inductive Logic Programming*, L. De Raedt, Ed. IOS Press, 1996, pp. 124–143. [Online]. Available: citeseer.ist.psu.edu/cohen96learning.html

[8] P. Graham, "Better bayesian filtering," in *Proceedings of the 2003 Spam Conference*, Jan 2003. [Online]. Available: http://www.paulgraham.com/better.html

[9] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," in *Learning for Text Categorization: Papers from the 1998 Workshop*. Madison, Wisconsin: AAAI Technical Report WS-98-05, 1998. [Online]. Available: http://robotics.stanford.edu/users/sahami/papers-dir/spam.ps

[10] I. Rigoutsos and T. Huynh, "Chung-kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam)," in *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. [Online]. Available: http://www.ceas.cc/papers-2004/153.pdf

[11] T. Meyer and B. Whateley, "Spambayes: Effective open-source, bayesian based, email classification system," in *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. [Online]. Available: http://www.ceas.cc/papers-2004/136.pdf

[12] "Spamassassin homepage," 2006. [Online]. Available: http://spamassassin.apache.org/

[13] K. Albrecht, N. Burri, and R. Wattenhofer, "Spamato - An Extendable Spam Filter System," in *2nd Conference on Email and Anti-Spam (CEAS), Stanford University, Palo Alto, California, USA*, July 2005.

[14] "Vipul's razor," 2006. [Online]. Available: http://razor.sourceforge.net

[15] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[16] T. M. Mitchell, *Machine Learning*. McGraw-Hill Higher Education, 1997.

[17] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, 2000.

[18] "Spamassassin public corpus," 2006. [Online]. Available: http://spamassassin.apache.org/publiccorpus/

[19] "phishingcorpus homepage," Apr. 2006. [Online]. Available: http://monkey.org/%7Ejose/wiki/doku.php?id=PhishingCorpus

[20] "Putting an end to account-hijacking identity theft," FDIC, Tech. Rep., Dec. 2004. [Online]. Available: http://www.fdic.gov/consumers/consumer/idtheftstudy/identity_theft.pdf

[21] "Sender ID framework," 2006. [Online]. Available: http://www.microsoft.com/senderid

[22] "Domainkeys," 2006. [Online]. Available: http://antispam.yahoo.com/domainkeys

A Accuracies of other classifiers

Table 4 shows the accuracies in terms of false positive and false negative rates when different classifiers are used instead of the SVM used in PILFER. For almost all of the high-performing classifiers, the difference in accuracy is not statistically significant, and none are statistically significantly better than SVM.

Table 4: Accuracy of different classifiers on same features

Classifier	**False Positive Rate**	**False Positive Std. Dev**	**False Negative Rate**	**False Negative Std. Dev**
SVM, C = 10	0.12%	0.15%	7.35%	2.67%
RIPPER	0.19%	0.18%	7.48%	2.79%
Decision Table	0.12%	0.13%	8.91%	3.00%
Nearest Neighbor w/ Generalization	1.25%	3.20%	6.68%	3.12%
1R	0.92%	0.35%	9.02%	2.83%
PART	0.20%	0.17%	7.43%	2.63%
Alternating Decision Tree	0.23%	0.23%	7.56%	2.96%
Decision Stump	5.14%	0.72%	6.93%	2.41%
Pruned C4.5 Tree	0.23%	0.1%	7.28%	2.75%
Hybrid tree w/ Naïve Bayes leaves	0.22%	0.18%	7.18%	2.92%
Random Tree (1 random attribute/node)	0.15%	0.15%	7.16%	2.75%
AdaBoosted C4.5 tree	0.23%	0.18%	7.28%	2.75%
AdaBoosted Decision Stump	0.24%	0.17%	10.99%	3.11%
Voted Perceptron	0.76%	0.36%	13.28%	3.45%
Bayes Net	3.96%	0.75%	6.89%	2.43%
Naïve Bayes	1.06%	0.31%	7.83%	2.55%