Identifying Phishing Attacks

Brandon Azad
bazad@stanford.edu

Abstract—**Phishing emails attempt to steal a user's personal information by masquerading as legitimate transactional email. Recently, Stanford has been the victim of a large number of phishing attacks, which caused the author to wonder about the difficulty of classifying phishing emails. The author examined the accuracy of several existing algorithms, including naive Bayes, logistic regression, and support vector machine (SVM) classifiers, on the bag of words and augmented bag of words models of phishing and non-phishing emails. Of these, the linear SVM under the augmented bag of words model performed best. Additionally, the author attempted to use an SVM with a nonlinear word subsequence kernel, but this kernel was determined to be too computationally expensive to deploy in a real world setting.**

I. Introduction

Phishing attacks impersonate legitimate transactional electronic communications from trustworthy entities in order to steal a user's personal information, often targeting usernames and passwords, credit card numbers, or social security numbers. Phishing emails are designed to resemble transactional email from the impersonated institution closely so that the user believes she is interacting with the legitimate institution. These attacks have become a serious problem with the rise of digital commerce, since both money and information access are put at risk [1].

Flagging suspicious messages before the user opens them is one way to mitigate the threat posed by phishing email attacks. While the task of identifying phishing emails is similar to that of identifying spam emails, less research has been conducted into the former [2]. Additionally, the difficulty of classifying phishing emails is compounded since the emails are designed to look like legitimate transactional email.

Recently, I have received a surge of phishing attempts to my cs.stanford.edu inbox, which made me curious: How difficult is it to identify an email as phishing? To narrow the scope of the question, I only considered fully offline analysis of the email features. While current research looks at both offline features (such as whether the email contains words such as "verify your account") and online features (such as whether remote JavaScript loaded from a link attempts to modify the status bar [2]), offline analysis is appealing since no network access is necessary to classify the email. This can be advantageous when the additional processing incurred by classification is costly: Eliminating factors such as network access can significantly reduce classification time.

This paper aims to examine techniques that can be used to identify phishing emails with high accuracy. The emails are first examined under the bag of words model, a multinomial event model used for natural language processing. The emails are analyzed using a naive Bayesian classifier to establish a baseline classification accuracy. The bag of words model of the email is then extended to include 14 additional features extracted from the emails in the dataset and reclassified with naive Bayes. Many of these features have already been examined by the literature on phishing email classification [2]. Next, logistic regression and support vector machines (SVMs) are examined as replacements for naive Bayes. Finally, I attempt to use a nonlinear word subsequence kernel based on [3], [4] to capture the meaning inherent in the emails better than the bag of words model.

II. Data Collection

I used a set of public corpora of phishing emails available at [5]. These corpora contain various hand-selected phishing emails. I used two of these corpora, the first of which contains 1423 emails collected between November 15, 2005 and August 7, 2006, and the second of which contains 2279 emails collected between August 7, 2006, and August 7, 2007. It was discovered during the processing and feature extraction phases that some of these emails are best described as traditional spam rather than phishing emails, so such emails were removed from the data sets when found.

I also used two corpora of non-phishing emails. The first is the publicly available SpamAssassin easy_ham ham corpus available at [6]. The easy_ham corpus consists of 2500 emails and was published in 2004. The second corpus was collected from my personal Stanford email account and consists of 9277 non-phishing emails I received in the past year.

This dataset is likely not representative of phishing and non-phishing emails received at a typical inbox since it was compiled from multiple sources at widely different times; however, there are very few publicly available phishing corpora, and (at the time of this writing) none that I could find with both legitimate transactional email and phishing email. It would have been better to have a single corpus containing phishing, legitimate transactional, and ham email, but since no such corpus could be located, this compilation is the best I could achieve. Furthermore, my personal account collected only 29 phishing emails during the course of this project, which was not sufficiently diverse to train a classifier. Subsequent research could be improved by using a unified data set. However, for the purposes of this paper the dataset used here is likely sufficient.

III. Features

The dataset was preprocessed to extract the relevant features. Each email was split into a header and body, and if the email was a multipart MIME message only the part containing textual/HTML content was used. If the body was HTML, it was parsed and decoded to extract the textual content. Once the text had been extracted, it was scanned to identify the following common formats:

ADDRESS an address, e.g. postal;
URL a textual (non-anchor) URL;
EMAIL an email address;
TELEPHONE a telephone number;
IPADDR an Internet Protocol (IP) address;
DATE a Gregorian calendar date;
TIME a time; and
NUMBER a number.

These generic tokens replaced the original text of the message when they were identified. All non-alphabetic characters were then removed to yield the "text" of each email. Finally, each word in the text was stemmed using the Snowball stemmer [7] available through the Natural Language Toolkit [8] to produce the "stemmed text".

The stemmed text was further processed in order to create the bag of words representation. The frequency of each word in the stemmed text for each email was determined, and then for each corpus the most frequently occurring words and the words appearing in the most number of emails were recorded. This resulted in a vocabulary of 1126 words for the bag of words model; all other words were discarded to form the "word frequencies" for each email. This list of 1126 word frequencies formed the feature vector for the bag of words representation.

During subsequent analysis, this feature vector was augmented with 14 additional features extracted from the email headers, the original HTML, and the text:

CHARSPERWORD the average number of characters per word in the email text;
DIFFFROMREPLYTO whether the `From:` and `Reply-To:` header fields are different;
DOMAINNUMDOTS the maximum number of dots in the domain of any URL;
INREPLYTO whether the email header contains the `In-Reply-To:` field;
IPINURL the number of URLs containing IP addresses;
JAVASCRIPT whether the email contains JavaScript;
NUMSCRIPTS the total number of script environments in the email;
NUMURLS the total number of URLs in the email;
UNIQUEWORDCOUNT the number of unique words in the stemmed text;
URLHTMLENTITIES how much the URLs shorten when HTML entities are decoded;
URLLINKMISMATCH how many link target domains do not match link text domains;
URLNUMTLDS the number of different top-level domains in the URLs of the email;
URLPORT the number of explicit port references in URLs; and
WORDCOUNT the total number words in the text.

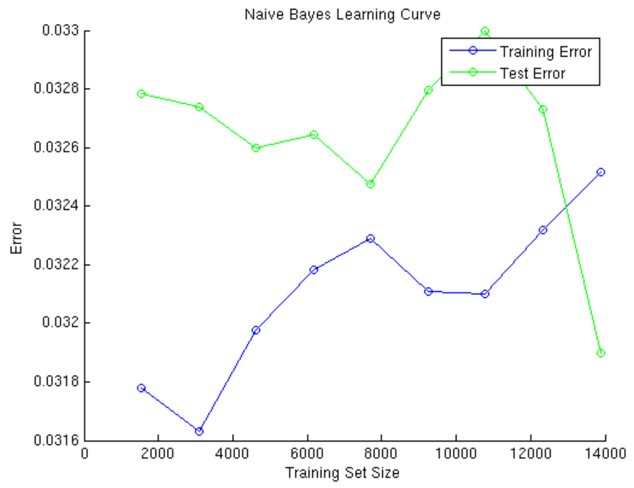

Fig. 1. The learning curve for naive Bayes using the bag of words model. Although the learning curve was generated with 30 Monte-Carlo repetitions per training set size, significant irregularities remain.

Many of these features have been used before in phishing classification, while some I have not found in previous literature.

IV. Results

Using the bag of words model, a naive Bayesian classifier was trained with Laplace priors on a stratified random sample of 70 percent of the data. When classifying the remaining 30 percent, it achieved an accuracy of 96.67 percent, establishing a baseline empirical error of 3.33 percent. The training error of the classifier was 3.25 percent. These high classification accuracies indicate that the data is readily separable based on the most frequent words observed in each corpus. Using the bag of words model augmented with the additional features, the naive Bayesian classifier was again trained on a stratified random sample of 70 percent of the data. Empirical error on the remaining 30 percent of the data was 2.88 percent; thus, the addition of the extra features improved classification accuracy by 0.45 percent. This improvement is considerable given the high classification accuracy without the additional features.

Learning curves for all models and classifiers were generated using 30 Monte-Carlo repetitions per training set size. The learning curve for the naive Bayesian classifier using the augmented bag of words model (Figure 2) suggests that the classifier is encountering an issue with bias. Unfortunately, the confusion matrix (Table I) reveals that the classifier missed an unacceptably high 10.35 percent of phishing emails, even though it only incorrectly labelled 0.57 percent of non-phishing emails.

To address the bias issue, a logistic regression classifier was trained on the dataset using the augmented bag of words model. However, the logistic regression classifier did not reach convergence within 48 hours, the maximum time a job is allowed to run on the computational clusters used

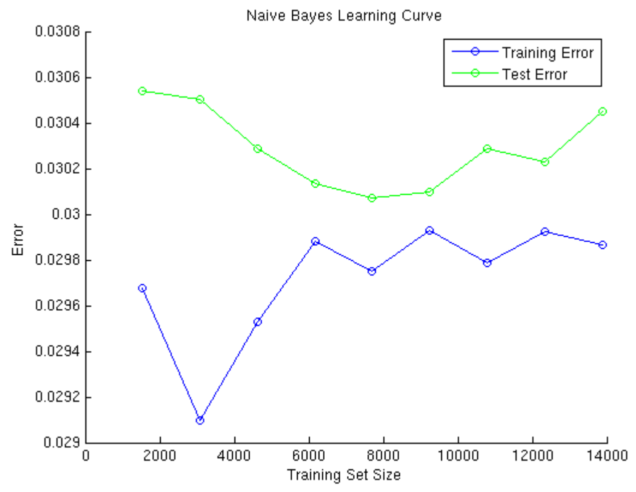

Fig. 2. The learning curve for naive Bayes using the augmented bag of words model.

TABLE I
CONFUSION MATRIX FOR NAIVE BAYES

	As Phishing	As Not Phishing
Phishing	979 (89.65%)	113 (10.35%)
Not Phishing	20 (0.57%)	3514 (99.43%)

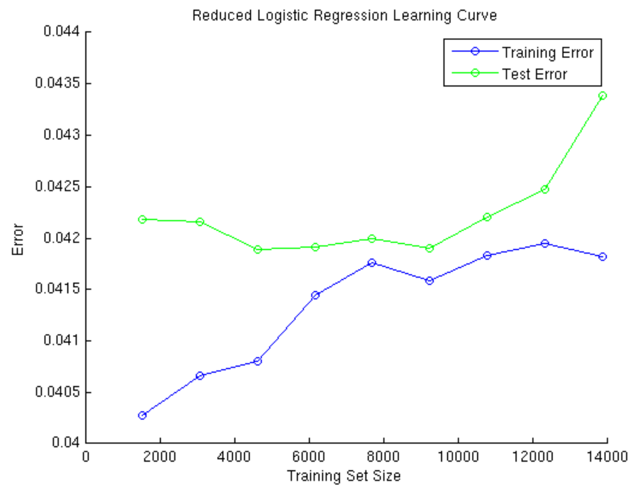

Fig. 3. The learning curve for Logistic Regression using a subset of just 16 features. The features chosen were identified during forward feature selection using an SVM with a linear kernel.

for this project. Thus, it was decided that logistic regression would be run on a smaller feature set. Using just 16 features identified during forward feature selection on the linear SVM (see Table II), the logistic regression classifier reached convergence with a test error of 4.18 percent, which is higher than that of either naive Bayesian classifier. However, the reduction of the feature space reintroduced the bias problem observed in the naive Bayes classifiers (Figure 3).

Using an SVM with a linear kernel produced a classifier with a test error of 1.21 percent and a training error of 0.47 percent. The learning curve for this classifier (Figure 4)

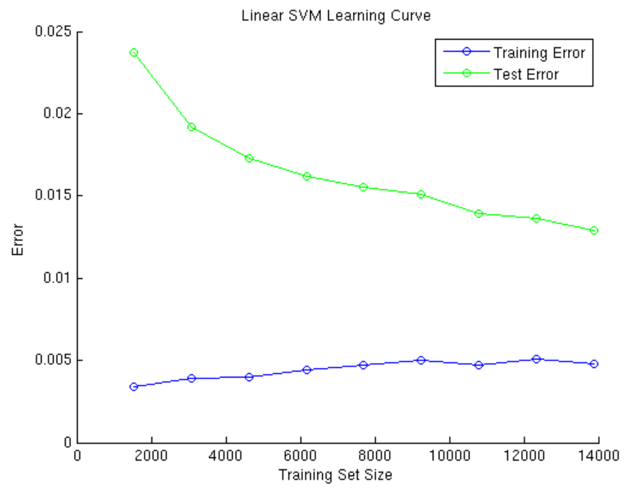

Fig. 4. The learning curve for an SVM using a linear kernel and the augmented bag of words model.

suggests that there is now a problem with variance, although classification accuracy is still very high at 98.8 percent. In order to narrow down the features used, forward feature selection was used to identify the top 50 features, the first 20 of which are listed in Table II. However, training a linear SVM on just these 50 features identified by forward feature selection produced a classifier with 2.10 percent test error and 1.93 percent training error. Experimenting with using a quadratic kernel as opposed to a linear kernel produced a classifier with significant overfitting: although training error was just 0.33 percent, the test error was 4.6 percent. This high variance is also visible in the learning curve (Figure 5).

The SVM with the linear kernel was the best classifier of the data. Interestingly, backward feature selection revealed that the augmenting features WORDCOUNT and UNIQUEWORDCOUNT actually were the least helpful in classifying emails, and removing them improved training error slightly. The confusion matrix is given in Table III. The classifier missed 2.66 percent of the phishing emails and incorrectly flagged 0.76 percent of the non-phishing emails. This is a significant improvement on the false negative rate of naive Bayes. Additionally, if we remove the augmenting features from the SVM, the test error rises to 2.33 percent and the classifier misses 5.86 percent of the phishing emails, indicating that the augmenting features are extremely important in correctly classifying emails as phishing.

Examining the decision boundary for the first two features identified during forward feature selection (Figure 6), it is apparent that the SVM is simply drawing a line separating words that are common to each type of email. This hypothesis is also supported by the most important features found during forward feature selection in Table II: Words such as “account”, “ebay”, and “bank” are found almost exclusively in phishing emails while words such as “stanford”, “science”, and “linux” are found almost exclusively in non-

TABLE II
FORWARD FEATURE SELECTION ON AUGMENTED BAG OF WORDS MODEL

Feature	Cumulative Error
account	0.104379
URLLINKMISMATCH	0.074213
stanford	0.059552
ebay	0.053649
TIME	0.047097
anni	0.042361
into	0.039572
der	0.037496
del	0.035615
IPINURL	0.034447
some	0.033214
0x3	0.032371
INREPLYTO	0.031009
paul	0.028998
attend	0.028154
untitl	0.027506
scienc	0.026857
bank	0.026208
linux	0.025689
submit	0.025235

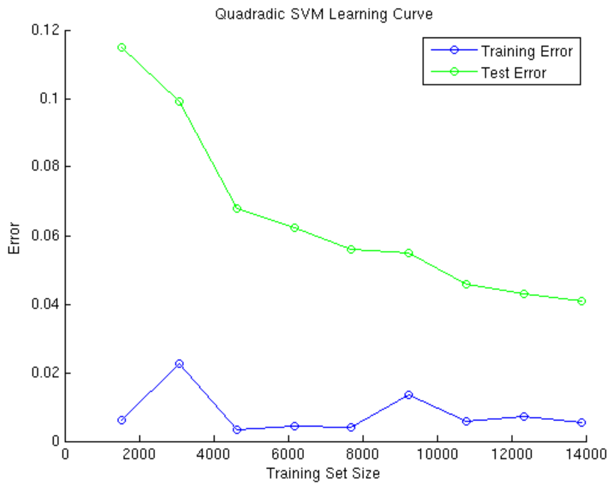

Fig. 5. The learning curve for an SVM using a quadratic kernel and the augmented bag of words model.

TABLE III
CONFUSION MATRIX FOR LINEAR SVM

	As Phishing	As Not Phishing
Phishing	1063 (97.34%)	29 (2.66%)
Not Phishing	27 (0.76%)	3507 (99.24%)

TABLE IV
CONFUSION MATRIX FOR LINEAR SVM WITHOUT AUGMENTING FEATSURES

	As Phishing	As Not Phishing
Phishing	1028 (94.14%)	64 (5.86%)
Not Phishing	44 (1.25%)	3490 (98.75%)

phishing emails. Features such as URLLINKMISMATCH and INREPLYTO play a similar role. Additionally, many of the words identified in forward feature selection are common words that carry little content or meaning, words such as

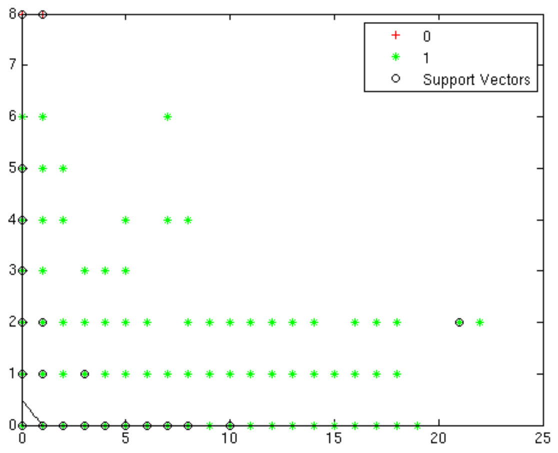

Fig. 6. The SVM decision boundary plot for the first two features found during forward feature selection. Most negative (non-phishing) training examples reside at $(0,0)$, and the decision boundary simply separates that point from the rest of the graph.

“into” and “some”. This motivates the search for a model that better captures the intrinsic meaning of the contents of an email than the (augmented) bag of words model.

V. WORD SUBSEQUENCE KERNEL

The bag of words model assumes that the class of each email is drawn independently and identically from some distribution of email classes, and then for a fixed class the words of each email are drawn independently and identically from a distribution that depends on the email class. While this model performs well in many practical applications, it does not account for word ordering or other indicators of meaning. Thus, it is possible that using a nonlinear kernel that can better measure the similarity in meaning between two emails might produce better results.

One such kernel K is the word subsequence kernel described in [3], [4]. This kernel treats each input s and t as a sequence of words, and finds the number of word subsequences of length p common to both s and t. A subsequence can be penalized for gaps by a factor λ to prefer subsequences that are close together. This kernel could better capture the inherent degree of similarity in meaning between two emails since similar communications are more likely to share words and phrases than dissimilar communications.

An efficient solution can compute $K_p(s,t)$ in time $O(p|s||t|)$ and can be used as a nonlinear subsequence kernel. To correct for the fact that longer sequences have more potential subsequences, the kernel was normalized:

$$K_p^{\text{norm}}(s,t) = \frac{K_p(s,t)}{\sqrt{K_p(s,s)\cdot K_p(t,t)}}.$$

However, training an SVM classifier with this kernel did not complete within the 48 hour time limit for any value of p greater than 1, and subsequently it was discovered that computing the unnormalized kernel function between emails took an average of 5 seconds. Thus, this kernel is too computationally expensive to be feasible for real world use under current technology.

VI. Conclusions

While it was discovered that the word subsequence kernel was too expensive to be a viable real world solution to phishing classification, other classifiers performed adequately for this task. All the classifiers examined achieved over 95 percent accuracy, although naive Bayes and the SVM with the linear kernel performed best. Naive Bayes misclassified over 10 percent of phishing emails, however, while the linear SVM misclassified only 2.66 percent of phishing emails. Without augmenting the bag of words model with additional features, the linear SVM misclassifies 5.86 percent of phishing emails, indicating that the additional features significantly improve classification accuracy. Performance of the SVM was on par with naive Bayes and logistic regression operating on the reduced feature set, making the linear SVM a practical real world method of flagging potential phishing emails before they reach the user.

References

[1] P. P. Stavroulakis and M. Stamp, Eds., *Handbook of Information and Communication Security*. Springer, 2010.

[2] W. Gansterer and D. Pölz, "E-mail classification for phishing defense," in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science, M. Boughanem, C. Berrut, J. Mothe, and C. Soule-Dupuy, Eds. Springer Berlin Heidelberg, 2009, vol. 5478, pp. 449–460. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00958-7_40

[3] N. Cancedda, E. Gaussier, C. Goutte, and J.-M. Renders, "Word-sequence kernels," *JOURNAL OF MACHINE LEARNING RESEARCH*, vol. 3, p. 2003, 2003.

[4] J. Rousu and J. Shawe-Taylor, "Efficient computation of gapped substring kernels on large alphabets," *J. Mach. Learn. Res.*, vol. 6, pp. 1323–1344, Dec. 2005.

[5] J. Nazario, "Phishing corpus," Published online, 2008. [Online]. Available: http://monkey.org/~jose/wiki/doku.php

[6] SpamAssassin, Published online, 2004. [Online]. Available: http://spamassassin.apache.org/publiccorpus/

[7] M. F. Porter, "Snowball: A language for stemming algorithms," Published online, October 2001. [Online]. Available: http://snowball.tartarus.org/texts/introduction.html

[8] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly, 2009. [Online]. Available: http://www.nltk.org/book