USING MACHINE LEARNING TO IDENTIFY PHISHING ATTACKS

Jamie Thorpe, Stephanie Schwartz
Millersville University
jethorpe@millersville.edu, stephanie.schwartz@millersville.edu

Abstract

The term "phishing" refers to a type of cyber attack in which the attacker sends fraudulent emails. What makes this email attack unique is that it often requests that the user follow an included link, where the user is asked to enter private information. This project explores the possibility of using natural language processing and machine learning techniques to differentiate between legitimate and phishing emails. Natural language processing tools used here include word tokenization, stemming, and stopword removal. These processes are used to prepare the emails for analysis by machine learning models by manipulating the text of the emails. Machine learning models look at samples of both types of email, attempt to decide the identifying factors of the two types of emails, and then classify unseen samples. A strong model will be able to identify a phishing email by analyzing the text of the email in addition to the characteristics of the included link.

Keywords

phishing, machine learning, natural language processing

1. Introduction

Those who work to develop computer security measures are faced with the issue of creating a secure but usable system. There is no way to make a device 100% secure without making it unusable. One reason for this is that the user is actually a danger to the integrity of the system. In recent years, a method of attack called "phishing" has become more and more popular, and it relies primarily on human error and lack of awareness.

"Phishing" refers to the use of social engineering to obtain personal information from system users [1]. One of the most popular forms of phishing attack is through email. A "phishing email" is received by a system user, and if that user is not trained to look out for phishing attacks, they may believe the email to be legitimate. The social engineering aspect of phishing involves crafting the language and visual design of phishing emails and websites to make them seem legitimate and to compel the user to submit personal information. If a user receives a phishing email, it likely appears to be from a legitimate source and asks the user to follow a link, which takes the user to a phishing website. Between the email and the site, the user is made to believe that there is some (usually urgent) reason to submit personal information, such as login credentials. The attacker can then use this information to perform future attacks. They now have access to the system, but can use the system while appearing as a legitimate user, thus avoiding detection by the computer's intrusion detection system [2].

Because this kind of attack relies so heavily on social engineering, it can be very difficult to prevent. Phishing emails may be caught by spam filters, but this is certainly not always the case. Even if the email is caught by a spam filter, there is no guarantee that the user won't open it anyway. The best way to prevent an attack it through focused filtering for phishing, and through education for users. Phishing can be particularly dangerous for larger businesses, where more system users means more ways for attackers to access login information. According to the Anti-Phishing Working Group, there were 364,424 phishing websites detected from July to September of 2016, which was actually a 25% decrease from the previous quarter, where they recorded record highs [2]. The number of unique reports of phishing attacks during this time frame also dipped, from 315,524 reports worldwide in the second quarter of this year to 229,251 in the third [2]. Regardless of the decline, hundreds of thousands of attacks in just a few months is still indicative of a major security concern.

The goal of this project is to utilize machine learning and natural language processing techniques in order to develop a classifier to differentiate between phishing and non-phishing emails. The purpose of such a classifier is two-fold: 1) it could be used as part of a tool to filter email, 2) useful information can be gleaned from the attributes used by the classifier and this information can be used to better educate users on how to avoid becoming victims of phishing.

2. Background

Social engineering plays a big role in the success of phishing attacks. In order to provide a better idea of what to look for during text analysis, one must consider the social engineering aspect of the attack. According to Webroot, a computer security services provider, social engineering is an "art", and involves "manipulating people so they give up confidential information" [3]. The aim is generally to obtain bank information or the passwords necessary to access the user's personal or professional computer. In order to convince the user to give up this information, a phishing email may include a request to verify information, a notification stating that the user has won something, or a message asking for help [3]. In any case, the goal of the email is exploitation of the user's trust and curiosity. These trends were important to keep in mind during this research when attempting to distinguish between phishing and legitimate emails. For instance, the phishing email text often contained certain keywords such as “verify” or “bank”.

One of the major problems with text analysis is high dimensionality of the feature space. High dimensionality occurs when there are a large number of variables to consider, which can make analysis more complex. In text analysis, each unique word is an individual variable, so this problem is particularly relevant. Verma and Hossain [4] handled this problem by defining the feature space further using SenseLearner and WordNet together. SenseLearner helps to differentiate the contexts of a word that may have multiple meanings. WordNet can then be used to find a set of synonyms for the word, so that the presence of synonyms between two emails can be detected [4]. These steps reduced the number of individual word variables to consider by equating sets of words as synonyms.

Verma and Hossain [4] also incorporated a certain amount of Natural Language Processing (NLP) into their work. Performing NLP on a text dataset is a challenging task, but it is necessary in order to help break down the text into something that can be analyzed. It can also be used to further reduce the feature space. The authors used NLP techniques to break the text of each email into individual words; tag parts of speech; recognize named organizations, people and locations; normalize words to lowercase; remove stopwords; and reduce words to their bases through a process called “stemming” [4]. Further research of similar projects shows that this is a fairly standard set of techniques for processing text. Verma worked on a similar project involving phishing email text [5] in which the same NLP was performed.

Another interesting product of [4] was the development of an “Action Detector” and a “Nonsensical Detector”. For classification purposes, the authors utilized pattern matching with these two sub-classifiers. They observed that the phishing emails often focused on some action to be performed by the email recipient, and that phishing emails not caught by this detector are often missed because the text is irrelevant to the subject, thus “nonsensical”. If an email falls into one of these categories, it can be marked by the sub-classifiers as more likely to be phishing.

Similarly, [5] introduced the idea of “actionable” vs “informational” emails. Similar to the observations made in [4] which led to an “action detector”, the authors of [5] looked at the language in the email text to determine whether it is trying to convey a sense of urgency, threat, or concern, or is offering some incentive to performing an action. Because social engineering in phishing emails often uses some kind of story or scenario to convince the user to perform an action [3], this kind of language is more indicative of a phishing email. On the other hand, non-phishing emails tend to be more informative and simple, or less story-based. The difference in language here contributes to the end algorithm.

The analysis done by Basnet, Mukkamala, and Sung in [6] introduces a different approach to the phishing problem. As mentioned above, the authors’ work is based on various features of the websites linked in the phishing emails. Most of the values for the features are binary, and indicate the presence or absence of a certain quality. This data analysis does not require NLP, and is thus much less complex and time-consuming than research involving the actual email text. However, when creating a filter for phishing emails, using the data from the websites linked may not be the safest method to pursue, as it bring the user “closer” to the attacker.

Another major difference between [6] and the other works mentioned here is the use of both supervised and unsupervised algorithms for classification. In supervised classification, a model is trained using a set of sample data, where each sample is associated with a “target” value. This target value is the correct identification for the sample. In unsupervised classification, the sample data is fed to the model without target values. The model then creates groupings in the data without knowledge of the correct identification. Basnet, etc. were studying phishing websites specifically, and their work shows the possibility of a successful classifier when using an unsupervised clustering method. However, the biggest success was with Support Vector Machines, which are supervised methods.

3. Approach

Most of the previous work on this subject focused on creating a phishing email filter. Some researchers used the email text, others used metadata from the link in the text. This paper focuses on the text analysis, with the analysis of link data designated as future work.

A data set of legitimate emails was used alongside the phishing set, in order to compare characteristics of the two different types of emails. A classification model was built for this purpose, with the goal of understanding more about what makes phishing attacks successful. This involved looking at the characteristics that phishing emails tend to have, and determining whether anything can be learned about effective types of social engineering. Finally, most of the researchers mentioned above used statistical analysis as part of their processes alongside common machine learning algorithms. Some statistical-based methods may be used in future work, but the primary focus of this work thus far has been on the machine learning algorithms.

4. Data

The type of analysis performed compares the characteristics of a phishing email to the characteristics of a “regular” email. These non-phishing emails will be referred to as “ham” emails going forward, a term that is used in the dataset to represent non-spam emails. For this application, the features of large sets of both phishing and ham emails were examined, particularly the text of each email. This text data included the email sender, subject line, links, and, most importantly, the content of the email. Once this data was pulled from the email, some natural language processing techniques were used to further manipulate the text data and prepare it for machine learning analysis.

The phishing emails come from the online Phishing Corpus [7]. The corpus was created from spam emails received over several years' time by the corpus author. There are a wide range of emails included in the corpus. It contains a total of 4,550 emails, split into four files that each represent a different span of time during which the author collected the phishing emails. This is one of the few publicly available corpuses of only phishing emails, and many of the papers written about phishing emails since the corpus was published have used this same corpus, including [4], [5], and [6]. It is important to keep in mind that these are emails targeted toward an individual, which may be formatted differently than those sent throughout a company or aimed at gaining access to a larger organization. As such, it was important for the sake of consistency to keep the intended audience of the original email in mind when selecting a ham corpus.

The ham emails come from the SpamAssassin public mail corpus [8]. This corpus was built to compare spam emails to non-spam, or "ham", emails. Only the ham emails will be used, so there are no spam or phishing objects mixed in with the ham set. The data set includes 3,900 "easy" ham emails, which are easy to differentiate from spam. It also contains an additional 250 "hard" ham emails, which are more difficult to differentiate. This is a total of 4,150 ham emails, and each are viewed as individual text files. Since the difficulty of filtering different types of emails is not being studied, it is not necessary to keep the sets separated. However, it is beneficial that this set provides a wide range of email samples. This corpus was selected because the emails are not exclusively ones that were originally sent within a particular company, which is the case with some other ham corpuses. In this way, we can be sure that the original phishing and ham emails had similar intended audiences, and that our final results will not be skewed by dissimilar data in this way.

For machine learning classification, it is important to build models using approximately the same number of objects for each class. The phishing set had about 400 more emails than the ham set, but a difference of only 400 emails isn't significant considering the size of the datasets overall. However, at the end of the parsing stage, the size of the dataset was reevaluated to determine whether the sets were still balanced. Another factor to consider is the time and size burden of the dataset. Text data takes much longer to process than numerical data. Having to handle nearly 9,000 email samples may be out of the question for certain applications simply because of how long it takes for the text to be processed and quantified in preparation for the machine learning process.

5. Methods

5.1 Parsing - Before the emails can be analyzed, the relevant information was parsed from the email files. The data extracted from the emails included the sender, subject, included links, and the email text. The sender, subject, and links, or references, could be gathered from the headers of each email. The email text was the last part of each email, although the emails were recorded differently in the ham and phishing corpuses.

The ham emails were in individual text files. Parsing the email involved looking for keywords on each line of the header to find the pertinent information and then, once the message started, reading to the end of the text file. Since the message is the last component to appear in the file, the end of the file is also the end of the message content.

The phishing emails were slightly more complicated. Each phishing file contained not one, but hundreds if not thousands of emails. The headers could be read similarly to the ham headers, but the email text contained some HTML. The BeautifulSoup package in Python was used to extract the important text data from the HTML. Again, the message was the last component of the email listed in the file before continuing to the next email's header. So, once the end of the HTML had been reached, this also signified the end of the email.

During the original parsing process for the phishing emails, it seemed as though about 3,300 emails were being ignored because they were not parsing cleanly. This still left enough samples to do some preliminary analysis, but it left a big gap in the dataset. Upon further inspection, it was discovered that the phishing parser was not always correctly identifying the end of each email. So, it was collecting as the "message" part of one the correct message, the header of the next email, and the message of the next email. After adjustments were made to the parser, about 3,700 could be cleanly parsed, and the parser could more easily detect the end of an email sample. Because of this, the dataset is not only larger, it is more accurate.

In total, 3859 ham emails and 3719 phish emails were successfully parsed. From these, 3719 ham emails were used so that the phishing and ham samples were balanced. This set was then split into a training set (90%, or 6694 emails) and a testing set (10%, or about 744 emails).

5.2 Natural Language Processing - Because human languages are made up of so many complex rules, it can be difficult to design a model that can analyze written or spoken language. To help solve this problem, natural language processing (NLP) can be used to simplify the text. This simplification can also help reduce the feature space, or number of individual variables, to be considered by any classification algorithms. In the case of text data, each unique word is a variable, so reducing the number of unique words in the data can make analysis much less complex. Several NLP techniques were used to process the text of the phishing and ham emails. These were also popular techniques used in existing work on phishing email detection (see [4], [5], [6]).

Word tokenization [9] is a simple form of processing that makes the text easier to work with. Tokenization breaks down the text data into individual words or word parts. The rules dictating how the words are split will depend upon the algorithm used for tokenization. In this research, the algorithm breaks words apart using primarily whitespace. Apostrophes in the text are also used to separate common contractions from the "base" word. For instance, words like "couldn't" become "could" and "n't".

Stopwords are words that appear often in certain languages. In the english language, these words include "and", "the", "it", etc. Because these words would appear so often in any email, regardless of its classification, these words were removed entirely [9]. This is another way to reduce the overall feature space of the data. In addition, word tokenization aids stopword removal. Splitting words on apostrophes breaks the positive and negative forms of a word into similar pieces. As Figure 1 depicts, word tokenization before stopword removal actually makes the stopword list shorter and simpler.

The same method for removing stopwords can also be used to remove punctuation [9]. Like the stopwords, the same punctuation would appear in an email whether it was a phishing or ham sample. Removing all punctuation reduces the feature space.

Original String	"test should shouldn't would wouldn't could couldn't"
Desired String	"test"
Stopword List 1	should, shouldn't, would, wouldn't, could, couldn't (size = 6)
String After Tokenization	"test", "should", "should", "n't", "would", "would", "n't", "could", "could", "n't"
Stopword List 2	should, would, could, n't (size = 4)

Figure 1 - Visualization of the benefit of word tokenization before stopword removal. The chart shows the stopword lists required to reduce the original string to the desired string without and with word tokenization.

Stemming is also a popular method for reducing feature space [9]. The stemming algorithm selected for this work was the Porter Stemmer [10]. The algorithm was used to remove any suffixes from the words in the text data. This returned words in a similar tense. For example, "run", "runner", and "running" all became "run" when passed through the Porter stemmer. These three words were then considered the same word, and the number of unique word variables was reduced by two. It is important to note that this process does not negatively affect the text analysis of the email by making it more difficult to understand. It may be more difficult for human eyes to read, but to the model, the presence of the word in any tense should have the same effect on how the email is analyzed. It is also important to note that not all words become another proper word. For instance, "computer" becomes "comput" in the Porter stemmer, which is technically the root of the word. However, since all instances of "computer" are treated this way (along with words like "compute" and "computing"), this also does not negatively affect the analysis.

5.3 Vectorization - The vectorization of the data is an important step because in general, classification models can only handle some kind of numerical data, not text data. Vectorization transforms a string of words into a numerical representation that can be analyzed. "Term Frequency-Inverse Document Frequency" Vectorization, or TFIDF, was chosen for this purpose. This means that for each word in a document, or email, the number of times a word appeared in the email was compared to the number of times it appeared across the data as a whole. Use of this method relies on a single vocabulary to be used for all analysis.

The training documents were first passed through a separate vectorizer in order to create a single usable vocabulary. The training documents were then vectorized using a TFIDF vectorizer with the vocabulary created by the first vectorizer. This same vocabulary was used to vectorize the test documents as well. This was done so that the training and testing vectors were comparable. Unfortunately, this means that any "new" words in the test documents were effectively ignored because the training documents do not contain those words. This highlights one benefit of having a large training set: it reduces the number of new, unseen words that the testing set could contain, simply by containing a wider variety of words in the training set.

5.4 Classification Algorithms - Classification algorithms are used to build the classification model. The point of the classification model in this project is to be able to analyze similarities and differences between the vectorized phishing emails and the vectorized ham emails in the training set, and to build a model based on this analysis. The model can then be used to identify new emails in the testing set as either "Phishing" or "Ham", which in this project, corresponds to model responses of "True" or "False" respectively.

Logistic Regression is a classification algorithm that deals easily with binary classification, where there are only two categorical target values, such as true/false. The binary logistic statistical equation is used to determine the probability that a sample will fall into each possible category. It is classified based on the target value with the highest probability of being correct.

Support vector machines work well when samples have many variables, which makes it a good choice for text classification. The specific type of support vector machine is a linear support vector classifier, or linear SVC. This algorithm will basically draw a linear boundary between the two target values based on the characteristics and corresponding target values of the training samples. A test sample is then classified by identifying where it falls in relation to this boundary.

The random forest classifier is an ensemble method, which means multiple individual classifiers are used together in order to compound their classifying power. A random forest is a combination of multiple decision trees. A decision tree is a tree-like structure which partitions the training data using various tests, creating branches at each new test. The leaf nodes represent the final classification of each partition of samples. In a random forest classifier, multiple decision trees "vote" on the correct classification of each test sample.

6. Results

The preliminary results of our analysis can be seen in Figure 2. The numbers represent the percentage of test samples correctly classified by the three tested classification models. Each classification model was recreated and given test samples five times (five trials in Figure 2). The results of each individual trial is recorded, with the results of all five trials being averaged together for each model type. This is done because certain model algorithms create different models with each run. For instance, a Random Forest algorithm will create a different set of trees with each run, depending on how the algorithm divides the samples each time. To really gauge the accuracy of such a model, the model must be tested multiple times. However, for Logistic Regression and Linear SVC, the results remain consistent through each trial. This is not actually surprising, knowing the way these algorithms work. Logistic regression uses an unchanging statistical equation to perform classification. Linear SVC virtually graphs the characteristics of each sample to create its boundary. Since the models were trained on the same data each time, it is reasonable to assume that the resulting models did not vary from trial to trial.

	Logistic Regression	**Linear SVC**	**Random Forest**
Trial 1	92.06%	93.27%	92.19%
Trial 2	92.06%	93.27%	92.60%
Trial 3	92.06%	93.27%	92.33%
Trial 4	92.06%	93.27%	92.19%
Trial 5	92.06%	93.27%	92.19%
Average	92.06%	93.27%	92.30%

Figure 2 - Results table, before tuning.

Logistic regression and random forest are achieving approximately 92% accuracy results, and linear svc is about 93% accurate on average. These initial results, using the default settings for the algorithms, are promising even though no parameter tuning has been done to mold the model algorithms to the problem at hand. Parameter tuning is an important step in the modelling process, and the concept is accepted as standard practice by those in the field [11]. Tuning occurs when the changeable parameters of a model are adjusted to their optimal values for the data set. It is not common for the optimal parameter settings for the models to be the default values. Of the models used in this research, the simplest to tune are the Logistic Regression and Linear Support Vector Machine models.

For these two models, there is only one parameter that is usually tuned: C. In the linear support vector classifier, the parameter C represents how large of a margin should be around the hyperplane, or, how much misclassification of training samples should be avoided. Larger values of C mean smaller margins. Most of the other parameters for this model do not apply to the problem posed in this research. C has a similar purpose as a parameter in logistic regression models, except instead of "penalty", the parameter is called "regularization strength". In both cases, the C parameter helps to reduce the likelihood of overfitting, where the model is fit so closely to the training set, that it does not predict well when given something new in a test set. Also similarly to linear support vector classification, many of the other parameters in the logistic regression model do not apply to this type of problem.

Tuning the random forest model is a bit more difficult and time-consuming, simply because there are so many available parameters to tune that affect not only the accuracy of the model, but also the amount of time that it takes to build the model. Adjustable parameters in this model include the number of trees in the final model, the number of features considered when looking for the best way to split the samples at a branch, the maximum depth at each tree, and the minimum number of samples required to split a node. Each of these parameters has an influence on how complex the end model is, and therefore how well it performs and how long it takes for the model to be built. This also makes parameter tuning more complex. Generally, in order to tune one parameter, a set of potential values for that parameter is created. Then, the model is rebuilt for each value in the set, and the best-performing model by some metric is reported. When multiple parameters are being tuned, the model is rebuilt for each combination of parameter values listed. This means that tuning three parameters using only three values each causes the model to be built 27 times. For a model like random forest, which is already more complex and time-consuming to build, tuning multiple parameters can take a very long time. Therefore, it is sometimes best to tune only one or two parameters at a time, and see which parameters or parameter combinations seem to have the most positive effect on the final results. After several rounds of attempting to tune four different parameters, tuning concluded with the realization that the model still performed best under its default parameter values, with the exception of one parameter. Increasing the number of trees produced by the model by about 50 trees improved accuracy without producing a very large increase in the amount of time taken to build the model.

The tuning methods reported performed five-fold cross validation in the background in order to determine which parameters led to the best model performance with each trial. In five-fold cross validation, the training set is partitioned into five subsets. Then, a model is trained and tested ten times, with a different subset acting as the testing set each time. The sets of possible values for each model were manually adjusted after each trial, in an attempt to hone in on a small range of optimal values. That results for the tuned models are as follows in Figure 3.

Overall, the models are performing at about 93% accuracy. The model tuning efforts had an effect on the results for the random forest and logistic regression models, but little to no effect on the linear support vector classifier. These results are nearly on-par with the results from much of the background research referenced in [4], [5], and [6]. However, these researchers included several additional measures for improving results (particularly in the natural language processing step) which this research does not yet include. While it is possible that past researchers saw similar results at this stage, and that the additional steps in language processing were done to close that final five percent gap (the most accurate models in [4], [5], and [6] performed at approximately 98% accuracy), it is still necessary to reflect on the current results and address any existing issues. Some additional steps have therefore been taken in order to validate the accuracy of the current results.

The phishing parser was reexamined, as described in Section 5.1. Before parser adjustments were made, the models were performing at about 98% accuracy without any parameter tuning, which was very suspect. Once more phishing emails were cleanly parsed, these results dropped to about 92% without parameter tuning. This is a much more realistic result, although it still warrants some examination. However, the now-corrected issues with the parser did explain some other problems with the models. First of all, there were some unusual words appearing in the vocabulary for the models. Some of the words seemed to be more like what might be found in the header of an email, rather than the message. The vocabulary has since been cleaned up quite a bit with the adjustments to the parser. In addition, the issues helped explain the extremely positive results that the models were previously producing. The models work by analyzing the frequency with which certain words appear in the message of the email. If the "message" data for an email sample after parsing including not only the correct message text, but also the header data and message for the email that followed it, then one sample would actually contain the message data of two emails. It follows then that certain phishing-related keywords could appear with as much as twice the frequency in a single sample as they should. This would make the phishing classification of this email pretty obvious, thus skewing the results positively.

The vocabulary of the vectorization models was also revisited. It is possible that a particular word would appear with high frequency in one type of email, but this word would not be removed by the stopword processing. For instance, "eBay" is not a common enough word in everyday conversation to appear in most stopword lists, and may not even be a significantly common word in phishing emails in general. But if a large proportion of the phishing emails in this data set were created to appear as if they came from eBay, then the model would have no trouble identifying an email as a phishing email if it contained the word "eBay" an unusually large number of times. This, however, does not make it an effective model. If the model were to be deployed as an email filter, it wouldn't be quite so successful because it would have greater difficulty identifying phishing emails that did not contain "eBay" a disproportionately large number of times. In order to investigate the possibility that this phenomenon

	Logistic Regression	**Logistic Regression Value of C**	**Linear SVC**	**Linear SVC Value of C**	**Random Forest**	**Random Forest Value of n_estimators**
Trial 1	93.41%	1000	93.41%	1.00	93.00%	70
Trial 2	93.41%	750	93.41%	1.00	93.14%	60
Trial 3	93.41%	1250	93.41%	1.00	92.19%	50
Trial 4	93.41%	750	93.41%	0.75	93.00%	60
Trial 5	93.41%	750	93.41%	0.75	92.73%	70
Average	93.41%	-	93.41%	-	92.81%	-

Figure 3 - Results table, after tuning.

had occurred with this data, the training set was split into ham and phishing, and the vectorizer was run on each set separately to create a vocabulary specific to each type of email. The top one hundred words in the vocabularies were then examined for unusual stopwords to add to the stopword list. Words like “eBay” and “Paypal” were added to the stopword list for the reasons explained above, but this did not affect the model results.

Creating these top vocabulary lists did offer an interesting way to compare the emails by sight. Looking at the lists of the top vocabularies for each type of email, it can be seen that some of the social engineering patterns mentioned in Section 2 are being reflected in this data set. Some of the top words in the ham email vocabulary included “work”, “people”, “wrote”, “state”, “world”, and even words as benign as “mailman”. It is easy to imagine how these words would appear often in workplace emails, or casual emails that may relate some information about politics or world news. The top words in the phish vocabulary include “account”, “bank”, “access”, “protect”, “update”, and “login”. These are certainly words that fall in line with the description of phishing emails in Sections 1 and 2. A phishing email will often have the user follow a link to update or verify account information, and will try to convey a sense of urgency or that that user’s account may be vulnerable. It is a positive sign to see that the top vocabulary results for each type of email do make sense on their respective lists.

Cross validation was incorporated to determine whether the models would change significantly with a slightly different training set. Cross validation is a method generally used to tune model parameters, but it can also be used to further investigate the training set. For this research, 10-fold cross validation was used. Ideally, the model will give consistent results regardless of which subset is used for testing. Otherwise, it could indicate that a certain subset of the training set is skewing the final results. Inconsistency between the cross validation test results and the final test set prediction accuracy could also indicate that the model is being overfit to the test set. When 10-fold cross validation was performed on this set, the results did vary a reasonable amount, but were overall consistent not only with the other folds, but also with the results for the predictions on the actual testing set. This could indicate that the positive results from the final classification model are actually accurate.

Assuming that the results are accurate, one interesting trend is that errors tend to come from phishing emails being misclassified as ham emails (since a phishing identification is considered “positive”, such errors are called false negatives). In addition, the same test samples are consistently misclassified. This indicates that these particular emails may be especially well-crafted to resemble legitimate emails. Further analysis of these emails is warranted.

7. Future Work

The natural language processing section of this project is the most complex. It is here that the text data can be manipulated in order to improve results. For the projects referenced in Section 2, the natural language process involved part-of-speech tagging, searching for named entities, and some reduction of the feature space by identification of synonyms. None of these steps have yet been incorporated into this research. The natural language processing steps currently being used may also be improved. For instance, the stopword list can be adjusted, or alternative algorithms for stemming can be explored.

Much of the previous research done on this subject has included analyzing data about the link(s) included in the emails. This is not something that has yet been addressed in this research, but will likely be incorporated in the future. Whether phishing emails are misclassified as ham or vice versa, it could be helpful to analyze the link in the email, or to see whether the email contains an outside link at all. If there is no link, this is likely not a phishing email. There are also certain characteristics that tend to indicate that a link is illegitimate. For example, the data set created by Muhammad, McCluskey, and Thabtah [12] for the purpose of analyzing phishing websites specifies several key features that tend to characterize phishing websites. By combining the results of an analysis of the text along with analysis of features of any included links, more effective filters and educational tools could be developed.

Future work may also include some analysis to determine the most common features of a phishing email. In addition to being able to detect phishing emails using the classification models, another part of the original project goal is to gain some insight on what makes a believable phishing email. The specific social engineering methods utilized present in this data set may be further investigated, along with the vocabulary lists from both types of email. Additional analysis of this kind could provide unique insights into the methods most popularly utilized by phishing attackers, and could lead to improvements in phishing educational tools.

8. Conclusion

This paper presents preliminary results that indicate that machine learning techniques can be effectively utilized to build a classifier that distinguishes phishing emails from non-phishing emails based on the text content. Several machine learning techniques have been applied to a corpus of emails with fairly consistent and promising results. The future phases of this project will include the incorporation of analysis of links contained within the emails, improvements to the text processing steps of the data analysis, and evaluation of the social engineering methods being utilized in the phishing attacks in the corpus.

References:

[1] United States Computer Emergency Readiness Team, *Incident Reporting System*, Official Website of the Department of Homeland Security, https://www.us-cert.gov/report-phishing

[2] Anti-Phishing Working Group, *Phishing Activity Trends Report, Quarter 3*, 2016.

[3] L. Criddle, *What Is Social Engineering?*, Webroot, https://www.webroot.com/us/en/home/resources/tips/online-shopping-banking/secure-what-is-social-engineering

[4] R. Verma, N. Hossain, Semantic Feature Selection for Text with Application to Phishing Email Detection, *16th. Annual International Conference on Information Security and Cryptology*, Seoul, Korea, 2013.

[5] R. Verma, N. Shashidhar, N. Hossain, Detecting Phishing Emails the Natural Language Way, *Computer Security–ESORICS*, Pisa, Italy, 2012, 824-841.

[6] R. Basnet, S. Mukkamala, A. Sung, Detection of Phishing Attacks: A Machine Learning Approach, *Studies in Fuzziness and Soft Computing*, 226, 2008, 373–383.

[7] J. Nazario, The online phishing corpus, 2004, http://monkey.org/~jose/wiki/doku.php

[8] Apache SpamAssassin, The public mail corpus, 2006, https://spamassassin.apache.org/publiccorpus/

[9] D. Jurafsky, J. Martin, *Speech and language processing, 2nd Edition* (Upper Saddle River, NJ: Pearson - Prentice Hall, 2009).

[10] C.J. van Rijsbergen, S.E. Robertson, M.F. Porter, New models in probabilistic information retrieval, *British Library Research and Development Report*, no. 5587, 1980.

[11] F. Hutter, H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, *5th International Conference on Learning and Intelligent Optimization*, Rome, Italy, 2011, 507-523.

[12] R. Mohammad, L. McCluskey, F. Thabtah, Phishing Websites Data Set, *UCI Machine Learning Repository*, 2015, https://archive.ics.uci.edu/ml/datasets/Phishing+Websites