2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering

Enhancing the Naive Bayes Spam Filter through Intelligent Text Modification Detection

Linda Huang, Julia Jia, Emma Ingram
2017 Honors Summer Math Camp
San Marcos, TX USA
Email: lindahuang3846@gmail.com,
juliajia7@gmail.com,emmai3120@gmail.com

Wuxu Peng
Department of Computer Science
Texas State University
San Marcos, TX USA
Email: wuxu@txstate.edu

***Abstract*—Spam emails have been a chronic issue in computer security. They are very costly economically and extremely dangerous for computers and networks. Despite of the emergence of social networks and other Internet based information exchange venues, dependence on email communication has increased over the years and this dependence has resulted in an urgent need to improve spam filters. Although many spam filters have been created to help prevent these spam emails from entering a user's inbox, there is a lack or research focusing on text modifications. Currently, Naive Bayes is one of the most popular methods of spam classification because of its simplicity and efficiency. Naive Bayes is also very accurate; however, it is unable to correctly classify emails when they contain leetspeak or diacritics. Thus, in this proposes, we implemented a novel algorithm for enhancing the accuracy of the Naive Bayes Spam Filter so that it can detect text modifications and correctly classify the email as spam or ham. Our Python algorithm combines semantic based, keyword based, and machine learning algorithms to increase the accuracy of Naive Bayes compared to Spamassassin by over two hundred percent. Additionally, we have discovered a relationship between the length of the email and the spam score, indicating that Bayesian Poisoning, a controversial topic, is actually a real phenomenon and utilized by spammers.**

***Index Terms*—Email, Spam, Spam Filter, Bayes Spam Filter, Naive Bayes Classifier, Spamassassin, Text Classification, Bayesian Poisoning**

I. Introduction

As the digitization of communication grows, electronic mail, or emails, has become increasingly popular; in 2016, an estimated 2.3 million people used email [1], [2]. In 2015, 205 billion emails were sent and received daily, which is expected to grow at an annual rate of 3% and reach over 246 billion by 2019 [3]. However, the growth in emails has also led to an unprecedented increase in the number of illegitimate mail, or spam - 49.7% of emails sent is spam [4] - because current spam detection methods lack an accurate spam classifier. Spam is problematic not only because it often is the carrier of malware, but also because spam emails hoard network bandwidth, storage space, and computational power [2], [5], [6]. Additionally, the commercial world has significant interests in spam detection because spam causes loss of work productivity and financial loss [2], [5]. It is estimated that American firms and consumers lose 20 billion annually, even while sustained by the private firms' investment in anti-spam software. On the other hand, spam advertising earns 200 million per year [7].

Although extensive work has been done on spam filter improvement over the years, many of the spam filters today have limited success because of the dynamic nature of spam [2]. Spammers are constantly developing new techniques to bypass filters, some of which include word obfuscation and statistical poisoning [1], [2], [8], [9]. Although these two text classification issues are recognized, research today has largely neglected to provide a successful method to improve spam detection by counteracting word obstruction and Bayesian poisoning, and many common spam filters are unable to detect them [10].

In the remainder of this paper, we will discuss related methods, definitions, our new method, results, and future work. Section II reviews other machine learning spam filter techniques as well as related work and definitions. Section 3 proposes a new algorithm that will effectively increase the accuracy of Naive Bayes and reduce false positives. We describe our methodology in Section III. In Section IV we described our implementation, testing results and performance issues. Concluding remarks and further research work are presented in Section V.

II. Background

Many existing methods of spam detection are ineffective, exemplified by the increase in spam mail. The two categories of email-filtering techniques are knowledge engineering and machine learning [5]. In knowledge engineering, a set of rules to determine legitimacy is established, or rule-based spam filtration. However, rule-based spam detection has slight disadvantages because this method is exclusively based on spammers' previous methods, while spammers' methods are continuously expanding [5]. On the other hand, machine learning methods are customized based on the user and is able to adapt to the changing spamming methods, yet is slower.

Another major issue with knowledge engineering spam detection is that, although some of the rules are often characteristics of spam emails, they do not necessarily imply that the message is spam. Since emails are text in the form of strings, they must be converted into objects such as vectors

2324-9013/18/31.00 ©2018 IEEE
DOI 10.1109/TrustCom/BigDataSE.2018.00122

of numbers, or feature vectors, so that there is some measure of similarity between the objects [5] In this conversion process, there may be loss of information. Feature selection is a prominent yet neglected issue in modern spam filtering because spam and ham emails with the same feature vector will be incorrectly classified, resulting in a high false positive rate, or ham emails being misclassified as spam [5], [11]. Thus, most effective spam detection methods utilize some form of machine learning [6].

Non-machine learning methods of spam detectors include using the IP numbers of senders, calculating the correlation of text to a preset list of words used to find spam, and many etc. The differentiating characteristic between machine learning techniques and non-machine learning methods is that after being trained using a data set, the machine is able to make more accurate predictions on its own instead of constantly requiring human programming. Spam detection methods employing machine learning include the Naive Bayes, Vector Space Models, clustering, neural networks, and rule-based classification.

A. Naive Bayes

The most effective spam detection methods utilize some form of machine learning [5], [6]. Some machine learning spam filtration methods include Naive Bayes, Vector Space Models, clustering, neural networks, and rule-based classification. The Naive Bayes spam detection method is a supervised machine learning probabilistic model for spam classification based on Bayes Theorem [2], [12], [13], [14]. Supervised machine learning is a method of teaching computers without direct programming, or machine learning, that uses a known data set (a training set), which contains input and response values, to make predictions on another dataset, the testing set. We have chosen Naive Bayes for its speed, multi-class prediction ability, and small training set [5]. Additionally, since Naive Bayes is the baseline for most spam filters, improving Naive Bayes will inevitably improve most spam filters overall [2].

B. Current Issues

Although the Naive Bayes classifier has high accuracy in testing, there are several major flaws that surface in real life spam detection as exemplified by the increase in spam. One way spammers can bypass spam filters easily is through using tokenization attacks. A tokenization attack disrupts the feature selection process by inserting forms of word obfustication. Some different forms of word obfustication include embedding special characters, using HTML comments, character-entity encoding, or ASCII codes [2], [15], [16].

One important aspect to note here is that although humans are able to discern the actual words, the computer is incompetent [2]. Common Spam senders are able to bypass spam detectors by using leetspeak and diacritics. Leetspeak is an alternative alphabet that is primarily used on the Internet. Diacritics are the accents placed on words to modify the appearance [15]. Leetspeak allows the spam senders to change letters into symbols or a series of symbols. For example, "A" can be written as "/-\". When a word is modified using leetspeak, spam detectors are not able to identify the email as spam, which creates a false positive. Naive Bayes filters are also susceptible to a form of statistical attack called Bayesian Poisoning; Bayesian poisoning occurs when random but harmless words are inserted into spam messages, causing a spam email to be incorrectly classified as ham [2], [8], [12].

Additionally, there was, and is, speculation about the existence of Bayesian Poisoning because it requires knowledge of which words are considered ham. However, studies have shown that in both a passive attack, or when the spammers speculates about ham words, and an active attack, when the spammer knows which words are ham words, the performance of Naive Bayes severely decreases [2], [12]. Bayesian poisoning results in a high false negative rate, or spam emails incorrectly classified as ham. Furthermore, there are some inherent flaws within Bayes Theorem and the Naive Bayes classifier itself. Naive Bayes makes the naive assumption that all feature vectors are independent of one another; in other words, Naive Bayes will not be able to detect if certain words or phrases are related [17], [18].

C. Previous Work

Some advanced methods of text classification that have been proposed include word stemming and preprocessing to Naive Bayes [2], [11], [19], [20].

Data preprocessing is a data mining technique that transforms raw data into a more understandable format. It is applied to the dataset of emails to account for feature obstruction, tokenization attacks, and Bayesian Poisoning. Three of the most popular preprocessing techniques include word stemming, lemmatization, and stop word removal.

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Stemming is a heuristic process that removes derivational affixes, while lemmatization is a more refined process that considers the context to return the lemma of the word. Another method is the removal of stop words. The stoplist is a list of 100 of the most common words in the English language, such as the, and, of, etc. Since Naive Bayes is a probability model, these stop words would act the accuracy of the emails.

A keyword based statistical method like Naive Bayes relies on the strict lexical correctness of the words [19]. In other words, the Naive Bayes algorithm is unable to detect all forms of word obfuscation [2]. Very minimal research has been done on the forms of word obfuscation. One previous research paper created a very simple algorithm for word stemming, essentially removing the non-alpha characters, vowels, and repeated characters to get the phonetic syllables. However, although the efficiency of the Spam-Assassin and their addition was approximately 20%, the study failed to account for word boundary detection or optimization.

The research in [21] proposed a two-step feature selection method for text classification using Naive Bayes.

III. Methodology

In order to minimize false positives and increase the accuracy of Naive Bayes, we created an addition to the existing Naive Bayes method. Our addition will be able to convert symbols inside words to possible letters and use a spell check function to ensure the corrected symbol is a word and then run the word through the Naive Bayes spam filter.

A. Naive Bayes Classifier

Naive Bayes is known for its simplicity, multi-class predictive ability, and small training set requirement [13]. Naive Bayes uses probabilistic models based on Bayes Theorem, which states that:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

where

- $P(c|x)$ is the posterior probability that document c is in a class given evidence x;
- $P(x|c)$ is the conditional probability that evidence x is in document c;
- $P(c)$ is the class prior probability that any document is in the certain class;
- $P(x)$ is the predictor prior probability that evidence x is true.

B. Bayesian Classifier

For spam detection, we employ the Bayesian Classifier [22], [1]. There are two classes: let S stand for spam, and L stand for ham. $P(c|x)$ is the a-priori probability, given a message x, what category c produced it, if a word is in x [5], [12]:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x|S)P(S) + P(x|L)P(L)}$$

- $P(c)$ is the overall probability that a message is spam;
- $P(x|c)$ is the probability that the word appears in spam messages;
- $P(L)$ is the probability that a given message is legitimate mail;
- $P(x|L)$ is the probability that the word is in ham.

C. Multinomial Naive Bayes

We chose the Multinomial Naive Bayes optimization of the Naive Bayes classifier mainly for its multi-class predictive ability. Multinomial Naive Bayes is also more accurate than the other optimization methods [22]. Multinomial Naive Bayes(MNB) is the conditional probability of the evidence t_k, or words, within a message d given a class of the message, spam or ham. It assumes that the message is a bag of tokens or words, such that the order of the tokens is irrelevant. Multinomial Naive Bayes essentially counts the relative occurrences of a particular token within the message to determine the conditional probability:

$$P(c|d) = P(c) \prod_{1 \leq k \geq n_d} P(t_k|c)$$

where

- d is the document;
- c is the class, either spam or ham;
- t_k is the evidence, where t_1 to t_{n_d} are tokens.

Theoretically, the best class is determined by multiplying all the probabilities that each individual word is spam together as shown in the equation to get an overall probability, with probabilities closer to 1 being spam. However, there are instances where the spam word does not occur at all in a message; Laplacian Smoothing may ameliorate this problem [20], [23].

D. Our Algorithm

Besides preprocessing the data, we developed another algorithm that combines semantic-based, key-word based, and machine learning in Python. First, we preprocess the messages using Algorithm 1, the leetspeak and diacritics preprocessing. Then, we utilize algorithms 2 and 3 as explained by algorithm 4. The keyword search command used in algorithm 4 searches for common spam key words [16]. Our addition to Naive Bayes is described in the 4 pseudo-code algorithms below.

Algorithm 1 Leetspeak and Diacritics Preprocessing

```
If a = leetspeak
    then replace(a, c)
If b = diacritc
    then replace(b, c)
```

Algorithm 2 Multinomial Naive Bayes $(C; D)$

```
V ExtractVocabulary(D)
N CountDocs(D)
for each c ∈ C do
    N_c ← CountDocsInClass(D; c)
    prior[c] ← N_c/N
    text_c ← ConcatenateTextOfAllDocsInClass(D; c)
    for each t ∈ V do
        T_ct ← CountTokensOfTerm(text_c; t)
    for each t ∈ V do
        condprob[t][c] ← (T_ct + 1) / Σ_tf (T_ctf + 1)
return v; prior; condprob
```

Algorithm 3 ApplyMultinomialNB(C;C; prior; condprob; d)

```
W ← ExtractTokensFromDoc(V; d)
for each c ∈ C do
    score[c] ← log prior[c]
    for each t ∈ V do
        score[c] += log condprob[t][c]
return arg maxc2Cscore[c]
```

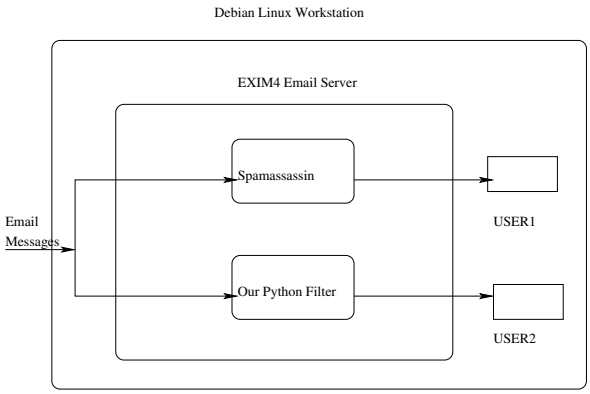

Fig. 1. Real-Time Testing Environment

Algorithm 4 A Novel Algorithm for Naive Bayes
1: Convert all text to lowercase using lower()
2: Replace leetspeak and diacritics
3: Determine $P(x\|c)$ through training set
4: Use Fuzzy Matching
5: Use Multinomial Naive Bayes

IV. Implementation, Testing, and Performance

A. Real-Time Test Environment

We set up a real-time testing suite to verify the viability of our spam filtering algorithm. Our goal is to observe and compare the behavior of the default spam filtering functionality with our new spam filtering algorithm. The test environment consists of a Linux server running the latest Debian Linux. The Linux server hosts an Exim4 email server [24] integrated with the well known Spamassissin spam detection/filtering software [25]. Exim4 is the default mail transfer agent in Debian Linux, and 56% of all publicly reachable mail servers run Exim4 [24].

The test environment is shown in Figure 1. Each email message will be sent twice, once through the normal reception path (which has Spamassassin built in) to User 1, and once again through our Python script to User 2. The Python script that implements the new spam filtering algorithm is embedded into the Exim4 email server. The testing set up allows us to easily evaluate the performance of our spam algorithm in a real Internet environment.

All of our data comes from the Ling-Spam corpus, a popular dataset consisting of common spam and ham emails [26]. The Ling-Spam dataset presents the most realistic evaluation suite that best mimics real world situations. We are using half of the pre-processed version of the Ling-Spam dataset as our training set, and another half as our testing.

B. Spamassasin

Leetspeak spam, diacritics spam, and ham emails were first sent through the Spamassasin server. From Figure 2, the majority of spam scores fall under the orange threshold of 5. Spamassasin only classified 26.9% of the leetspeak emails as

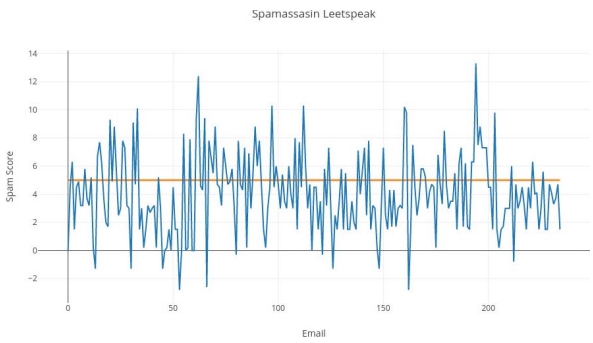

Fig. 2. Spamassasin Leetspeak Spam Scores.

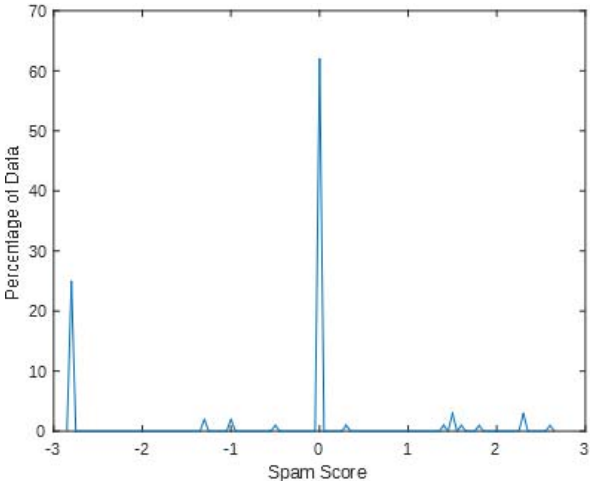

Fig. 3. Spamassasin Diacritics Results

spam even though all of the emails were spam. For diacritics emails, Spamassasin assigned a mode score of 0.

In addition, Figure 3 depicts how the vast majority of spam emails are given a score of 0, falsely indicating non-spam.

Next, we tested ham emails and encountered a similar issue. The distribution of Figure 4 shows about half the emails received scores indicating ham while another half received scores indicating spam; furthermore, there are a couple outliers of extremely high spam scores. Thus, the false positive rate, or the percentage of misclassified ham emails, is 44.66%.

C. Optimal Threshold: Spam Score

Most spam servers have a pre-programmed value that determines whether the email should be spam or ham. Depending on the elements incorporated within different servers, that spam threshold varies. Thus, when proposing a new addition to the spam servers, we tested for the optimal threshold. To do this, we took a test set of 750 leetspeak and diacritic spam emails and sent them through the new spam filter programmed in the server. The spam threshold that correctly detected the most spams was then identified as the our default (and optimal) spam threshold. Our test identified threshold value 5 as our optimal threshold.

D. Optimal Threshold: Fuzzy Matching

Although many elements of accuracy are implemented such as text reversal and detection, we still have to account for spelling mistakes that cannot register as spam words through the spam servers. For example, our Python code will not be able to detect a small change in the spelling of a key spam word, and then not count that as a spam word. Thus we have incorporated the Fuzzy Matching algorithm that allows for a greater degree of mispellings to identify spam words. We found this code matches about 80% of letters in corresponding order to word in the spam dataset. These values to fuzzy match words are set as a default in our spam filter.

E. Error Rate and Accuracy

$$ErrorRate = \frac{N_{S\to L} + N_{L\to S}}{N_S + N_L}$$

$$Accuracy = 1 - ErrorRate$$

Where $N_{S\to L}$ is false negative, $N_{L\to S}$ is false positive, N_S is total spam, and N_L is total ham. Error rate is probability of misclassification of a classifier and accuracy is defined as the percentage of the dataset correctly classified by the method. Using our Python filter we achieved an error rate of 38% and an accuracy of 62%. This is a drastic increase to the accuracy and error rate of the original Spamassassin, which has an error rate of 76.1% and accuracy of 23.9%.

V. Conclusion and Further Work

We proposed a novel algorithm for enhancing the accuracy of the Naive Bayes Spam Filter. The algorithmm was implemented and tested in real-time environments over the Internet. We carried out our testing to two main types of spam encryptions: leetspeak and diacritics. The algorithm significantly improved the accuracy of the Spam Server Spamassassin by coding a new addition to the current servers. We analyzed a particular aspect of spam emails that causes many challenges for individuals and companies. We demonstrated that our algorithm consistently reduced the amount of spam emails misclassified as ham email. The algorithm not only increased the accuracy of email sorting but also proved to be a valuable addition to the current systems.

Through the Naive Bayes classifier, we were able to improve upon the baseline for spam filtering. Naive Bayes has a very fast processing speed and allows for a small training set, hence is suitable for real-time spam filtering. Previous research showed that most spam classifications failed when exposed to text modifications. By creating an addition to Naive Bayes, we were able to improve the multi-class prediction ability. We also found that our new addition helped improve ham classification due to the high recall and precision rates.

Using our improved Spamassassin spam server, we tested hundreds of emails both spam and ham on the servers. Our test results showed that the addition did, indeed, drastically improve accuracy from about 23.9% to 62%. This is almost

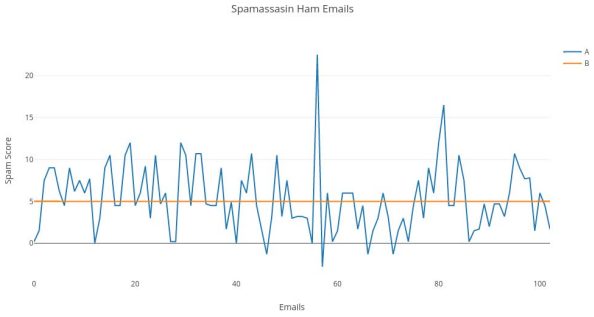

Fig. 4. Spamassasin Ham Emails

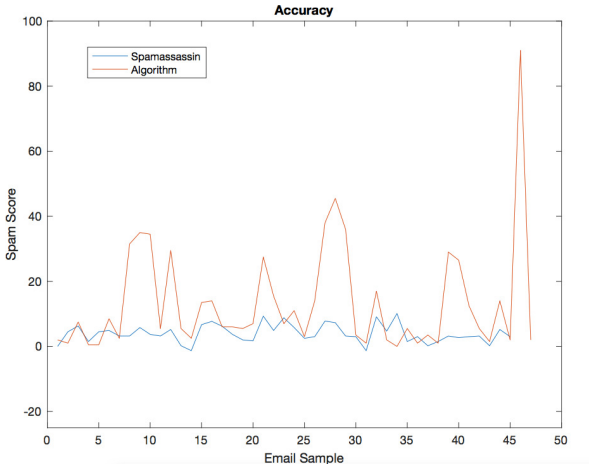

Fig. 5. Comparison of Spamassassin Accuracy to Spamassassin with new algorithm

a 259% increase in accuracy and can save corporations millions of dollars. Thus, the discovery of an addition not only improves the overall detection abilities of Spamassassin, but has enormous impacts on the customers of email servers, potentially preventing millions of dollars in malware damages.

A. Bayesian Poisoning Phenomenon

As discussed earlier, viability of Bayesian Poisoning, or the addition of harmless stop words to decrease the spam score, is still a debatable topic [8]. In our research we have found an interesting corelation between the length of the email and the spam score. As shown in Figure 6, there is an exponential regression between email length and spam score, meaning that as the length increases, the score decreases. This phenomenon partially confirms that Bayesian Poisoning does influence the spam score negatively. Dealing with Bayesian Poisoning is still a major challenge.

One of the key features of our research is applying text classification techniques to spam filtering/detection. While email spam detection has fundamental differences from text classification because the opponents of spam detection are dynamic, our work on machine learning text classification techniques could also be applied to other natural language

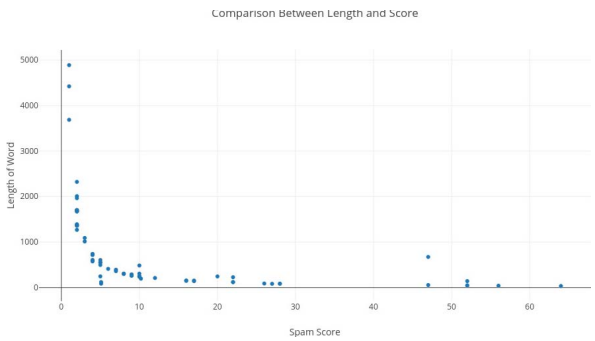

Fig. 6. Comparison and Length to Score

processing tasks such as news filtering or user preference prediction [14].

Our findings show that the improvements to the spam email servers can drastically enhance spam filtering and, in turn, create less of a burden on corporations to hand sort any unwanted emails. In addition, the simulation of the spam servers has significant implications on preventing malicious activity and viruses from entering the computer system and costing a company thousands of dollars.

B. Future Work

Since our addition successfully enhances the Naive Bayes spam filter, we will try to implement the addition onto other machine learning spam filters such as Vector Space Models, clustering, and artificial neural networks. Combining these other methods will allow the improvement of spam detection across many different systems to ultimately create a well-developed spam detector for text modifications. Currently we have two additonal research topics:

1) Speed and Efficiency: While our novel spam algorithm does improve the accuracy of the Naive Bayes classifier, we are currently expanding our research to include speed and efficiency optimization. These two factors are crucial simply because of the sheer number of processed emails which induces a high energy consumption; lowering energy consumption not only increases environmental well-being, but also reduces the cost of these spam filters while maintaining the same accuracy.

2) Expansion to HTML and XML: We plan to extend our techniques from detecting text modification in pure texts to other forms of email messages. For example we will try to improve spam detection for messages encrypted in HTML and XML. Currently, many spam emails are sent in HTML/XML to bypass spam detectors. We plan on proposing new methodologies that will have the ability to turn encrypted HTML into a computer-friendly message, which will be then sent through our proposed filtering algorithm.

REFERENCES

[1] "A bayesian approach to filtering junk e-mail," https://robotics.stanford.edu/users/sahami/papers-dir/spam.pdf, Stanford University, accessed: 2017, 2018.

[2] A. Bhowmick and S. Hazarika, "Machine learning for e-mail spam filtering: review, techniques and trends," https://arxiv.org/abs/1606.0104, 2016, accessed: 2017.

[3] "An analyst review of hotmail anti-spam technology," https://www.lifewire.com, Radicati Group, Inc, 2010, accessed: 2017.

[4] H. Tschabitscher, "How many emails are sent every day?" https://www.lifewire.com/how-many-emails-are-sent-every-day-117121, 2017, accessed: 2017.

[5] K. Tretyakov, "Machine learning techniques in spam filtering," in *Data Mining Problem-Oriented Seminar*, 2004.

[6] A. Aski and N. Sourti, "Proposed efficient algorithm to filter spam using machine," in *Pacific Science Review A: Natural Science and Engineering*, vol. 18, 2016, pp. 145–149.

[7] J. Rao and D. Reiley, "The economics of spam," in *Journal of Economic Perspectives*, vol. 26, no. 3, 2012.

[8] J. Graham-Cumming, "Does bayesian poisoning exist?" https://www.virusbulletin.com/virusbulletin/2006/02/does-bayesian-poisoning-exist/, 2006.

[9] B. Biggio, P. Laskov, and B. Nelson, "Poisoning attacks against support vector machines," in *Proc. of the 29th International Conference on Machine Learning*, 2012, pp. 1467–1474.

[10] J. Eberhardt, "Bayesian spam detection," in *University of Minnesota Morris Digital Well*, vol. 2, no. 1, 2015.

[11] K. Asa and L. Eikvil, "Text categorisation: a survey," https://www.nr.no/ eikvil/tm_survey.pdf, 1999.

[12] M. Sprengers, "The effects of different bayesian poison methods on the quality of the bayesian spam filter," in *B.S. thesis, Radboud University Nijmegen, Nijmegen, Netherlands*, 2009.

[13] S. Raschka, "Naive bayes and text classification i: introduction and theory," https://arxiv.org/abs/1410.5329, Cornell University Library, 2014, 2015, 2017, accessed: 2017, 2018.

[14] N. NaveenKumar and A. Saritha, "Effective classfication of text," in *International Journal of Computer Trends and Technology*, vol. 11, no. 1, 2014.

[15] "The shifting tactics of spammers: What you need to know about new email threats," https://secureemailplus.com/wp/WP10-01-0406_Postini_Connections.pdf, Postini, Inc, 2004, accessed: 2017, 2018.

[16] "The shifting tactics of spammers: What you need to know about new email threats," https://emailmarketing.comm100.com/email-marketing-ebook/spam-words.aspx, accessed: 2017, 2018.

[17] C. Li and L. Li, "Research and improvement of a spam filter based on naive bayes," in *Proc. of 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2015.

[18] K. Netti and Y. Rahdika, "A novel method for minimizing loss of accuracy in naive bayes classifier," in *Proc. of IEEE International Conference on Computational Intelligence and Computing Research*, 2015.

[19] S. Ahmed and F. Mithun, "Word stemming to enhance spam filtering," in *Proc. of First Conference on Email and Anti-Spam (CEAS)*, 2004.

[20] G. Qiang, "An effective algorithm for improving the performance of naive bayes for text classification," in *Proc. of IEEE International Conference on Computational Intelligence and Computing Research*, 2010.

[21] S. Sarkar, G. S., A. A., and J. Aktar, "A novel feature selection technique for text classification using nave bayes," *International Scholarly Research Notices*, vol. 2014, 2014.

[22] "Text classification and nave bayes, the task of text classification," https://web.stanford.edu/ jurafsky/slp3/slides/7_NB.pdf, Stanford University, 2011, accessed: 2017, 2018.

[23] G. Cormack, "Email spam filtering: a systematic review, foundation and trends," in *Journal Foundations and Trends in Information Retrieval Archive*, vol. 1, no. 4, 2007, pp. 335–455.

[24] "Exim internet mailer," https://www.exim.org, accessed: 2017, 2018.

[25] "Spamassassin: Welcome to spamassassin," https://spamassassin.apache.org, accessed: 2017, 2018.

[26] csmining.org, "Ling-spam datasets - csmining group," http://csmining.org/index.php/ling-spam-datasets.html, accessed: 2017, 2018.