Building on *SpamCop* and Graham: How varying preprocessing techniques can aid spam classification

Kevin O'Kane
University of Southern California
Mentor: John Heidemann
krokane@usc.edu
https://github.com/krokane/spam_classification_project

1. ABSTRACT

Email remains a crucial part of human communication in the age of the internet. Because of its widespread use and low start-up cost, it provides an ideal platform for spammers to communicate with masses of people. This puts email users at risk of malicious spammers. Despite strong progress made in spam classification and machine learning in general, spam continues to be a profitable enterprise for malicious actors [5]. This is primarily due to the adversarial nature of the problem, with ever changing technologies at the disposal of spammers proving more and more difficult to defend against. This project aims to further aid in solving the issue of email spam by running several experiments that could improve algorithms used for these tasks. The first experiment will determine the minimum training set size required in order to achieve strong model performance using Naive Bayes and Support Vector Machine models. The second experiment explores changing the amount of "important words" used to create predictions from these models, and determines whether or not the entire training corpus is required at prediction time. The third experiment deploys modern NLP toolkits to add parts-of-speech tags to words in the training corpus, to determine if providing additional information beyond surface text to one-token classifiers improves performance. These experiments aim to build upon related work in the field of spam classification, most specifically the work of Pantel and Graham discussed below, and hope to answer some of the crucial, unanswered questions of their work to further improve spam classification algorithms. This work was interesting to conduct because of the real-world applications to any active user of email. It also was interesting as it builds directly upon discussions regarding the previous work in this realm, specifically in regards to Naive Bayesian techniques for this task. This work also allowed for expanding beyond the discussed Naive Bayesian techniques, and engaging in modern work in this field regarding different models such as SVMs and deep learning methods.

2. RELATED WORK

Spam classification has been a popular topic in machine learning fields due to vast usage of email for communication and the ever changing nature of email spam. Works in this field primarily differ in two main categories – the models used for the task and the preprocessing steps prior to model training.

The first machine learning models for this task came in the 1990's, and replaced rules-based methods that could not scale to the breadth of emails received nor the adversarial nature of spammers. Pantel and Sahami [6, 7] presented some of the first of these machine learning models for spam detection, both deploying Naive Bayesian classifiers, though with differing preprocessing techniques. Pantel differed in two crucial ways. The first is the use of a Porter stemmer for tokens before training the model. This was designed to reduce the amount of potential tokens in the training set, which would work well given a smaller email training corpus. Pantel also tried using trigrams as tokens, as opposed to full words, but did not outperform using single words as tokens. Sahami used word tokenization techniques similar to Pantel, but also added several domain-specific features to aid in training the model. These features included a variety of hand-crafted phrases, not too dissimilar to rules-based techniques that preceded them.

Graham [3, 4] furthered the work of Pantel and Sahami using Naive Bayesian techniques for this task, but added more sophisticated preprocessing steps that were workshopped over time. These

preprocessing techniques were used to enhance the word-tokenization as well as add additional information to the model training, including using subject lines as part of the email data. Graham also only used the top 15 “most important” words to predict the class of an email. This concept involves finding the top 15 words most probabilistically indicative of a spam or ham email, and only determining a class outcome based on those 15 words. Lastly, Graham saw it rather important to bias the Naive Bayes models to try and reduce the amount of false positive outcomes. This is due to the nature of email spam classification, as classifying a ham email as spam could have drastic outcomes for the user. Using these techniques, Graham was able to produce better accuracy outcomes for classification, as well as strong false-positive rates.

As machine learning capabilities have enhanced over time, newer models have been deployed for this classification task that produced stronger outcomes. Drucker [2] deployed a support vector machine model on this task, demonstrating stronger results than the previous Naive Bayes models in terms of accuracy. Drucker also discussed the concept of “important words” as discussed by Graham, but ultimately did not use less than the complete set of words for prediction. Drucker argues that the SVM model should be capable of automatically selecting these words out, something that the Naive Bayes models could not.

More recently, deep learning models have been deployed on email spam classification tasks to good effect. Basyar [1] showed strong performance from LSTM neural networks for the task, which are capable of learning beyond the individual word tokens.

The experiments for this project were designed to further the work completed by these mentioned research projects. Graham mentions several times that increasing the training set size could be beneficial for model prediction, and, with Drucker, also hypothesizes about altering the amount of “important words” used in prediction. The experiments also build upon the deployment of Naive Bayes and SVM models for this task as used by several of the mentioned projects above.

3. PROJECT GOALS

Project A set out an overarching goal to improve techniques for spam classification through different preprocessing techniques for Naive Bayes models. It set out two clear objectives for the project that would be accomplished with two experiments. The first experiment would be to vary the size of the training set in order to determine a minimum threshold of training emails required to produce strong model performance. This objective was fulfilled in the final work, and is discussed as the first experiment. The second experiment would be to deploy modern NLP techniques to increase the amount of information the model was receiving from surface text to determine if overall performance would improve. The two tools mentioned in Project A to achieve this were parts-of-speech (POS) tagging and named-entity resolution. This objective was fulfilled by the third experiment in the final work, though named-entity resolution was not deployed to achieve this. The main reason for this was the initial poor performance of the POS tagging.

Project B added two additional goals to the previous goals mentioned in Project A. The first was a third experiment that varies the amount of “important words” used in model prediction. This goal was achieved by the second experiment below, which includes a further description of the concept of “important words” and how it was deployed. The second goal added in Project B was to train a long short-term memory (LSTM) neural network to classify the emails. The LSTM was going to be used to compare to the Naive Bayes model overall, as well as within the previous three experiments. This last goal did not come to fruition however, and the LSTM was instead replaced by a support vector machine (SVM) model for the final report for Project C. The replacement of the SVM was made after consulting related works (which proved that SVMs were strong for this classification task), as well as considering compute and time constraints for the project.

4. METHODOLOGY

4.1 Dataset

Each experiment for this work required a vast email dataset. In practice, ideally this dataset would be personalized to the user of the spam detection

technology. However, for this work, a dataset compiled from several publicly available email spam databases was used to produce more general results. The data used was randomly selected from three different sources. Each email in the dataset contains a body of email text. The first source was the SpamAssassin dataset, which provided 6,000 emails for the compiled dataset. The second source was the Enron email dataset, which contains roughly 33,000 emails, of which 8,500 emails were selected for the compiled dataset. The last source was a phishing email dataset from Kaggle user *CyberCop*, which contained roughly 19,000 emails. Again, 8,500 emails were taken from the third source. In total, the compiled dataset contained 23,000 emails – 20,000 used for model training and 3,000 used a test set to determine the effectiveness of the models. The 3,000 test set emails were randomly selected from each of the three sources, and includes 1,000 emails from each source. The overall class breakdown of the final dataset was roughly 58/42 ham to spam, which provides a relatively balanced dataset to work with.

4.2 Token Definition

In order to train the models for this task, each email was tokenized into single word tokens. Several steps were taken in order to simplify this tokenization process. Some punctuation marks were excluded from the tokens, including periods, commas, dashes, apostrophes, parentheses, and brackets. What remains in the tokens is more rarely used punctuation (including dollar signs, hashtags, percentages, etc.). The logic behind keeping these punctuation marks was that these marks could be used more frequently in spam emails, compared to typical punctuation marks that won't be sharing any additional information at the single-word level. Spaces were also removed in the tokenization process, as was the case. While case might be indicative of something in the email, all words were changed to lowercase as one of the initial data sources only provided email body text in lowercase. The most drastic change in the tokens is that digits are replaced with a "<digit>" tag. The logic here is that it should not matter the specifics of what numbers are included in the text, but rather that the mere existence of numbers in the text. The length of the number is maintained, so for example 1000 would be represented as "<digit><digit><digit><digit>" in the corpus. Once this preprocessing was completed, the tokens were vectorized and prepared for model training.

4.3 Models

For each experiment, two types of models were used for the task. The two models include a Naive Bayes model and a Support Vector Machine model. These models were chosen based on their strong performance to this specific task based on previous work conducted in the space, as mentioned above.

4.4 Experiment 1: Differing Training Size

The first experiment for this project altered the amount of training data available for the model to be trained on. The hypothesis of this experiment was that there would exist some minimum amount of training emails required to produce strong model performance. As mentioned previously, our base training set included 20,000 emails. For this experiment, both a Naive Bayes model and SVM model were trained at several subsets of that 20,000 email training data to find accuracy at each training data size. The sizes of these sampled training sets included 500, 1000, 2000, 3000, 4000, 5000, 7500, 10,000, and the full 20,000. The results are slightly different for the Naive Bayes model and the SVM model. The results for the Naive Bayes models indicate that a training set of merely 2,000 emails will produce similar results (less than 1% difference in accuracy) to any larger training set. This is demonstrated in Figure 1, which slows a plateaued light blue line around 2,000 training emails and 95% accuracy. The results for the SVM model differ however. The SVM model struggles to compete with the Naive Bayes performance until 7,500 emails are used to train the model. At that point, with each increased increment of training data, the SVM further improves its accuracy, as shown by the dark blue line in Figure 1.

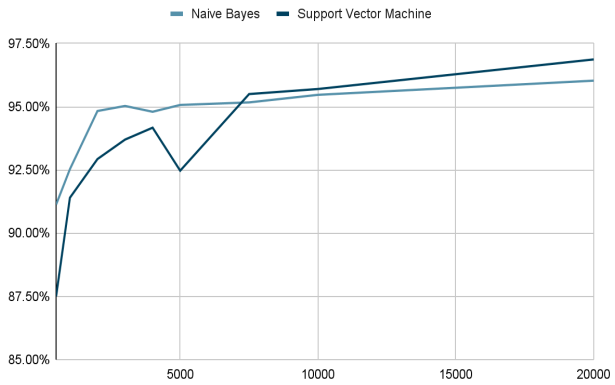

Figure 1. Classification Accuracy by Different Training Set Size

This experiment shows several outcomes. In small data environments, the Naive Bayes model clearly outperformed the SVM. However, due to the expansive capabilities of the SVM, additional data proves more beneficial for ultimate prediction accuracy. This work can be used practically in a spam detection technology. In the ideal world, a spam detection algorithm would be specific to each user, as each individual receives different emails. However, when a new user signs up for an email service, they do not have a large corpus of previously received emails to train an algorithm, so they would have to start with a more generic training corpus. By establishing a minimum training size for strong model performance, the algorithm can know when to switch to a personalized corpus from that initial generalized training set, improving the new user's spam detection outcomes as soon as possible.

4.5 Experiment 2: Differing "Important Words"

The second experiment conducted focuses on the concept of "important words." This concept is taken from the work previously discussed by Graham and Drucker. The idea behind an "important word" is that not every word in an email should be used to predict the classification of an email. For example, the word "the" is likely used evenly across both spam and ham emails. Also, it is probable that spam emails repeat very specific words more frequently than an average email. This is why Graham argues that only specific words, the most "important words" should be used for spam prediction. In a Naive Bayes context, importance is measured by the probability that a word indicates a spam/ham email. Returning to the example of "the," it is likely that word would not be very predictive of an email class, so it would have a probability of each class close to 0.5. Important words would be the opposite of this, which means their probabilities are as far away from 0.5 as possible. In the context of SVM, this importance measure is built into the algorithm itself, which produces coefficients of importance to determine the weight each feature should have in determining a classification. Thus, important words in this context refer to the words with the highest magnitude weights produced by the SVM.

The hypothesis of this experiment is that there is an ideal amount of "important words" to use for predicting classes. To test this hypothesis, a Naive Bayes and SVM model were trained using the full training dataset. Once the model was trained, each word in the training dataset was assigned a measure of importance as described above. For prediction, the number of important words used varied from 1-200, and accuracy was calculated. The results are shown in Figure 2.

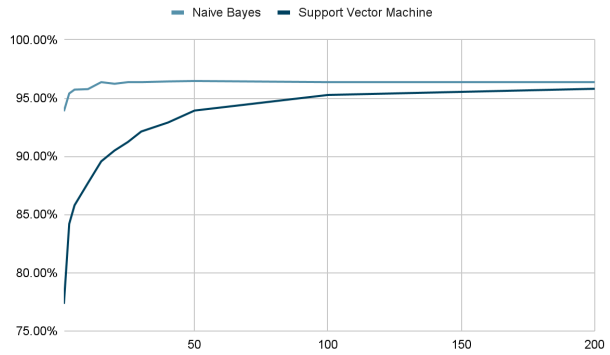

Figure 2. Classification Accuracy by Different Amount of "Important Words"

For Naive Bayes models, there seems to be no strongest amount of "important words" used. At each interval, the prediction accuracy on the test set stayed around roughly 95%, similar to the outcome of the baseline Naive Bayes model using the full training set in the first experiment. The SVM model shows that it prefers to have as much information as possible, as Drucker suggested. The SVM is designed to automatically select out the less relevant words, so it makes sense that the model performs worse with less overall information.

The takeaway from this experiment is that the concept of manually feature selecting the "important words" is not backed up by the results of this experiment. This experiment suggests that using all of the words/information available in the email corpus is the best course of action for SVM models, and that only using the "important words" for Naive Bayes models does not drastically change the prediction accuracy of the model.

4.6 Experiment 3: Using NLP Tools

The last experiment conducted utilized modern NLP tools to add information to the training corpus regarding the parts-of-speech of each word. The hypothesis for this experiment was that including the parts-of-speech information would improve prediction accuracy of the models, given that information is being added regarding the words' usage in the larger sentence. This theoretically would give the model more information than the single-word token, potentially aiding in machine understanding of the actual e-mail.

To test this hypothesis, each token was automatically assigned a POS tag using NLTK's toolkit for this purpose. Each token was then altered to add this information into the token set. This was done by adding an underscore and the POS tag after the token itself. An example of this would be altering the word "dog" used as a noun, which would alter the token to "dog_N". These tokens were then vectorized and used to train a Naive Bayes and SVM model.

	Naive Bayes	SVM
Baseline	96.03%	96.87%
w/ POS Tag	69.07%	80.23%

Table 1. Accuracy by Model and Use of Parts-of-Speech Tagging

The results of the third experiment can be found in Table 1 compared with the baseline performance of the Naive Bayes and SVM models. Counter to the stated hypothesis, the models performed drastically worse. The SVM did better than the Naive Bayes model, suggesting that if the dataset was further increased it could potentially perform even better as more information is added. However, the main takeaway from this experiment is that adding POS tags did not improve the accuracy and that the models did not gain further information in order to better their predictions.

5. FUTURE WORK

Potential future improvements to this work include training a more sophisticated deep learning model for similar experiments, such as an LSTM neural network, and using personalized email data for the training task. Also, the dataset used in this work was fairly balanced between spam/ham. In the future, studying more realistic, unbalanced datasets could change overall results. These improvements are likely valuable to real-world applications, and would be important to conduct given additional resources and time.

6. REFERENCES

[1] Basyar, I., Adiwijaya, M. D., & Murdiansyah, D. (2020). *Email spam classification using gated recurrent unit and long short-term memory.* Journal of Computer Science (Vol. 16, no. 4, pp. 559-567). https://thescipub.com/pdf/jcssp.2020.559.567.pdf

[2] Drucker, H., Wu, D., & Vapnik, V. N., (1999). *Support Vector Machines for Spam Categorization*. IEEE Transactions on Neural Networks (Vol 10, no. 5, pp. 1048-1054). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=788645&casa_token=2z68SKmY3o4AAAAA:T_v1Y0DO3edzxqq-2DPe_4jiuvkeDgCKYSsdJPiHKkSI_HXTEIrMhDkTMHtCRXv-2hYwkPgY&tag=1

[3] Graham, P., (2002). *A Plan for Spam*. https://paulgraham.com/spam.html

[4] Graham, P., (2003). *Better Bayesian Filtering*. https://paulgraham.com/better.html

[5] Levchenko, K., Pitsillidis, A., Chachra, N., Enright, B., Félegyházi, M., Grier, C., ... & Savage, S. (2011). *Click Trajectories: End-to-end analysis of the spam value chain.* IEEE Symposium on Security and Privacy (pp. 431-446). https://ieeexplore.ieee.org/document/5958044

[6] Pantel, P., & Lin, D. (1998). *SpamCop: A Spam Classification & Organization Progra*. AAAI Technical Report WS-98-05. https://cdn.aaai.org/Workshops/1998/WS-98-05/WS98-05-017.pdf

[7] Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E., (1998). *A Bayesian approach to filtering junk e-mail.* Learning for Text Categorization: Papers from the 1998 workshop (Vol. 62, pp. 98-105).https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b4498c71651f0327b5d51c8f8008d5a1804a084a