Spam Detection

James Elliott
Purdue University Fort Wayne
Fort Wayne, IN USA
ellijr07@pfw.edu

Abstract

This paper aims to look at the field of spam detection by employing a traditional and state-of-the-art language model to classify an email as spam or not. Specifically, I implemented a Naive Bayes model, known for it's simplicity and effectiveness in text classification, along side a state-of-the-art RoBERTa-based model. Initial results showed overfitting was occurring, so both models were tuned and trained to overcome it. The final results showed the RoBERTa-based model outperformed the Naive Bayes model based on accuracy. Finally the two model's robustness was tested using TextAttack showing both models were susceptible to Targeted Classification.

1 Introduction

Spam detection is a critical challenge and has evolved significantly over the years [2]. With the advent of machine learning, the ability to detect spam has greatly increased. Machine learning models are able to be trained on large datasets, and then provide a prediction on whether a specific email is spam or not [3].

Once a model has been trained with an adequate dataset, the model's robustness should be looked at to determine if it can withstand adversarial attacks such as in [7]. For this paper I used TextAttack [4] to conduct a Targeted Classification attack. The goal of the Targeted Classification attack is to get the model to increase the score of the targeted label, ultimately getting the target label to be the predicted label. This means that if we can get our model to classify a spam message as not being spam, then the attack succeeded.

2 Problem

The original problem with spam was that emails are typically sent in large numbers to recipients who did not request or opt to receive them [2]. They don't normally consist of much personalization and usually have some type of promotional content. While spam is annoying, the overall goal was to sell a product, service, or drive traffic to certain websites.

Nowadays, spam has evolved into more harmful content to include malware, viruses, or phishing for some type of theft of credentials or sensitive data [1]. According to the 2023 Verizon Data Breach Investigation Report (DBIR) [6], 83% of breaches involved an actor that was external to an organization, with one of the primary ways of gaining a foothold into a company being phishing. This is why it is critical to find ways to detect spam.

3 Motivation

The primary motivation for this paper is to have a deeper understanding of how language models are used to detect spam emails, in the hopes of continuing research in the future. As a security professional, finding ways to prevent attacks has always been a passion. One topic that has been in the news recently is how hard it's become to detect spam now that attackers are using Generative AI to create these emails [5]. The typical techniques of looking for grammar errors or contextual issues are in the past. With tools like Google's Gemini and OpenAI's ChatGPT, attackers can be less sophisticated and less knowledgeable when writing spam or phishing emails.

4 Datasets

Both the Naive Bayes model and the RoBERTa model were trained and tested with the *SetFit/enron_spam* dataset from HuggingFace. The dataset consists of 31,716 rows for training and 2,000 rows for testing. The training dataset was split 80/20 to create a validation set. As shown in Figure 1, the dataset was fairly balanced between spam and not spam for both the training and test

dataset. Instead of using the *text* field from the dataset, which included both the subject and the message, I chose to use the message to train and test the models. The reason for this was to reduce any outliers in the subject which may cause overfitting.

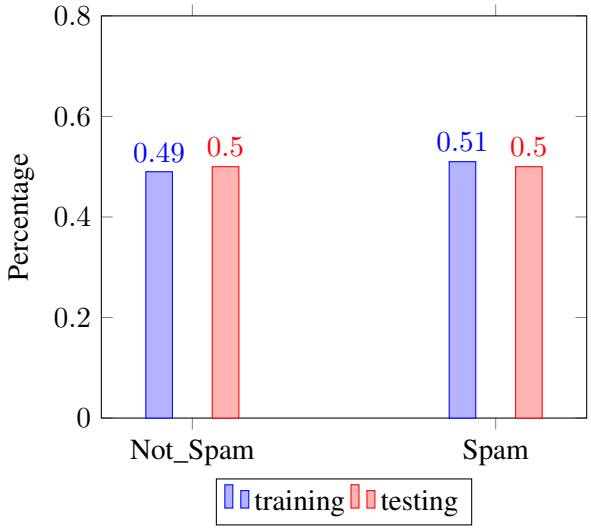

Figure 1: Not Spam/Spam distribution across training and test dataset

5 Experiments

The preprocessing techniques used differed between the Naive Bayes Model and the RoBERTa based model. For Naive Bayes a couple different options were experimented with to include simple tokenization with no punctuation's or spaces, and using the lemma form of words while also removing punctuation's, spaces and stop words. I also tested with only alphabetic words with no puctuation's, spaces, stop words, or numbers. I had the best results with using a blank SpaCy tokenizer with no punctuations, spaces, or stop words. It was decided to keep the numbers as they can be good indicators of spam. The preprocessing for the RoBERTa based model had a pretrained tokenizer that used batching, truncation, padding, and a max length of 256.

During the initial training of both models, the accuracy and F1 score were very high as if the models were overfitting the dataset. Table 1 and Table 2 show the initial accuracy scores after training the models. To address overfitting in the Naive Bayes model, I used Random Under Sampling which is a technique for training models on small datasets. I also used Boost which is a ensemble technique that trains several models sequentially and then aggregates the results.

	Precision	Recall	F1
Training	0.99	0.99	0.99
Validation	0.99	0.99	0.99
Testing	0.99	0.99	0.99

Table 1: Initial Naive Bayes Results

	Precision	Recall	F1
Training	0.99	0.99	0.99
Validation	0.99	0.99	0.99
Testing	0.99	0.99	0.99

Table 2: Initial RoBERTa Results

The RoBERTa based model required tuning during training through batch size and learning rate in order to address overfitting. The methodology used here was to set a higher batch size and higher learning rate and then slowly decrement the learning rate to fine tune the model. I was able to get an acceptable and consistent training and validation loss using a batch size of 32, a learning rate of 1e-4 and 3 epochs. The Training Loss and Validation Loss are shown in Figure 2.

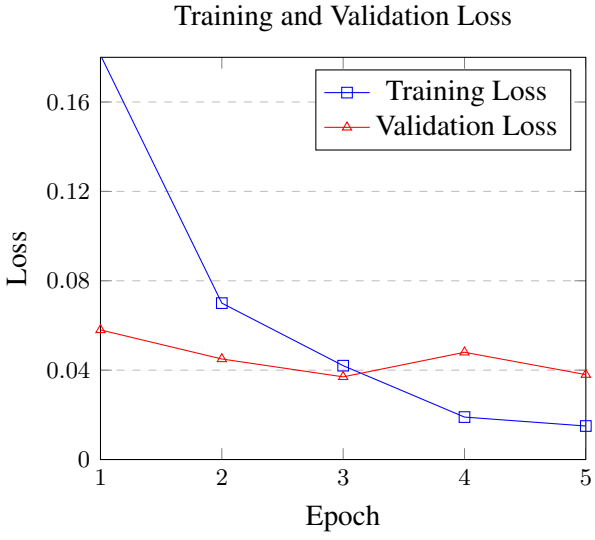

Figure 2: Comparison of training and validation loss over epochs.

After finalizing the training of the two models, I used TextAttack to test the robustness using the Targeted Classification goal. The target class for this was *not spam* (0). This meant that TextAttack would only attack the dataset where it was labeled as spam and skip any documents that were already labeled as not spam.

6 Results

The final results for both the Naive Bayes model and the RoBERTa based model are shown in Table 3 and 4.

	Precision	Recall	F1
not-spam	0.98	0.89	0.93
spam	0.90	0.98	0.94

Table 3: Final Naive Bayes Results

	Precision	Recall	F1
not-spam	**1.00**	0.99	0.99
spam	0.99	**1.00**	0.99

Table 4: Final RoBERTa Results

Table 5 shows the overall results of the adversarial attack between the Naive Bayes model and the RoBERTa based model.

7 Analysis

Analysing the performance of the Naive Bayes model and the RoBERTa model, the RoBERTa based model had a higher accuracy. This could be for a number of reasons, one being the RoBERTa model is still overfitting the data. One way to overcome this is by training the model on additional datasets so it can generalize better. Using additional datasets could have also helped the Naive Bayes model to better generalize.

As additional proof the RoBERTa model may have been overfitting the data was in the TextAttack results. TextAttack had an attack success rate of 85.71% against the RoBERTa based model. This is an indicator the model doesn't generalize as well as it should.

Though the Naive Bayes model had lower attack success rate, it also had a lower average number of queries needed for a successful attack which makes it more desirable from an attack perspective. Attackers would send almost two thirds less queries attacking Naive Bayes as they would the RoBERTa based model.

8 Conclusion

We discussed the problems faced with spam detection and how machine learning models can assist with classifying spam. Specifically we tested a traditional Naive Bayes model and a state of the art RoBERTa based model. We worked through the models overfitting the dataset and addressed the issues during preprocessing or tuning hyperparameters during training.

Model	Success Rate	Avg Num Queries
Naive Bayes	**57.14**%	736
RoBERTa	85.71%	**2076**

Table 5: TextAttack Results

The results showed the RoBERTa based model had a higher accuracy than the Naive Bayes model when it came to spam classification. Testing both model's robustness we saw the RoBERTa based model performed poorly as it was probably still overfitting the dataset. Although, the RoBERTa model did require three times more queries than Naive Bayes in order to get a successful attack.

** *The code for this project can be found here: https://github.com/w4rr4nt/nlp-project*

9 References

Roderic Broadhurst and Harshit Trivedi. 2020. Malware in spam email: Risks and trends in the australian spam intelligence database. *Trends and Issues in Crime and Criminal Justice*, (603):1–18.

Lorrie Faith Cranor and Brian A LaMacchia. 1998. Spam! *Communications of the ACM*, 41(8):74–83.

Suhaima Jamal and Hayden Wimmer. 2023. An improved transformer-based model for detecting phishing, spam, and ham: A large language model approach. *arXiv preprint arXiv:2311.04913*.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

Assem Utaliyeva, Millati Pratiwi, HyeGyoung Park, and Yoon-Ho Choi. 2023. Chatgpt: A threat to spam filtering systems. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 1043–1050. IEEE.

Verizon. 2023. 2023 data breach investigations report. Available online. Accessed: 02/16/2024.

Steve Webb, Subramanyam Chitti, and Calton Pu. 2005. An experimental evaluation of spam filter performance and robustness against attack. In

2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, pages 8–pp. IEEE.