Filtering Spam From Legitimate Messages: Data Challenge 2

Jacob Farner
Computer Science, COMP331
Occidental College
Los Angeles, California
jfarner@oxy.edu

Leopold Ringmayr
Computer Science, COMP331
Occidental College
Los Angeles, California
lringmayr@oxy.edu

***Index Terms*—Email Classification, Naive Bayes, NLTK, Sentiment Analysis, Spam Filtering.**

I. Introduction

Anyone who uses online messaging clients such as email or texting services is familiar with, and likely frustrated by, spam. Spam messages are unsolicited and sent out in bulk, often used to advertise, phish for user information, and target victims in scams. Modern messaging clients are good at filtering spam out of our inboxes, so what may surprise some is that spam emails accounted for approximately 55% of all email traffic in 2019, according to Statista.com [1]. This may seem like an overwhelming proportion, but it is down significantly from an estimated 90% in 2014 [2]. Because spam is sent to users' inboxes so relentlessly, it is critical for messaging clients to filter it out from legitimate communications so that users may focus on relevant information.

To explore and better understand how this works, we implemented our own spam detection system on a dataset of text messages which classifies texts as either spam (illegitimate) or ham (legitimate) based on the content of each message. This was done using Naive Bayes and logistic regression models through the Natural Language ToolKit (NLTK).

II. Related Work

Similar work in language analysis and classification is used across many online platforms to discourage spam and eliminate bot accounts, which we are able to compare our work against to confirm legitimacy of our own strategies. In his article *Don't Follow me: Spam Detection in Twitter*, Alex Hai Wang outlines how he was able to detect spam and bot accounts on twitter with 90% accuracy using a Bayesian model similar to the one used in our paper [3]. His paper Twitter classifies spam using a Bayesian model, similar to our own.

Spam Detection on Twitter Using Traditional Classifiers by Chuah and McCord is a paper with a near similar goal and conclusion to the one by Alex Hai Wang, but uses a random forest classifier instead of a Bayesian model [4]. This spam detector proved more robust with a precision of 95.7%. For our purposes, we felt that a Naive Bayes Classifier would suffice, as our model has an accuracy of 95% after cross validation.

III. Methodology

The main components for this classification problem are pre-processing of the raw CSV data, reading in and segmentation of the text data, feature engineering, and classification of text messages into Spam or Ham.

A. Project Data and Data Preparation

For this work, the text files are read in locally; the data is stored within a PyCharm project. The project data consists of 5,579 text messages labeled as either spam or ham. The data was originally presented in CSV format but was converted to a dataframe using the Pandas Data Analysis Library for Python to allow for easier use.

For building the classifier using Python's NLTK, the text messages are mapped to the corresponding label and stored in a shuffled list. The models employed in this project are built and tested using ten-fold cross-validation, allowing for a more accurate assessment of the models' performance than a simple test-training split of the data.

B. Data Cleaning and Feature Selection

To account for the inconsistencies in the raw dataset of messages, the Python pandas library was used to separate label from the message and to remove unnecessary characters. In this step, various tokens were removed in order to improve training of the models and to improve the predictive accuracy. Many of the messages contained punctuation and colloquial/texting language–these tokens were removed. Other non-alphabetic tokens were also eliminated from the larger list of tokens.

C. Classification

For the classification portion of the project, a Naive Bayes approach and a logistic regression model were employed for comparison. The Naive Bayes model relied on NLTK's built-in Bayes classifier, while the regression model made use of the scikit-learn and NLTK regression function. For each of the models, precision, recall, and F1-measure were collected and averaged over the ten folds in the cross-validation. For the overall assessment of the models' performance, a ten-fold cross-validation was used. Aside from accuracy, precision, recall, and F1-measure were collected for each of the folds and ultimately averaged for the final results.

IV. Results and Analysis

For this work, model performance statistics were collected for both the Naive Bayes model and the logistic regression model. Over the ten cross-validation folds, the Naive Bayes had an average classification accuracy of 92.89 percent. The average precision on the 'spam' class was 0.77 with a recall of 0.69 and an F1-measure of 0.72 on the 'spam' class. For the 'ham' class, the average precision was 0.95, and the average recall was 0.97. The F1-measure for 'ham' class was 0.95.

As a second model, this project used a logistic regression model on the same dataset. For the logistic regression model, the average accuracy over 10 folds was 92.85 percent. For the 'spam' class, the average precision was 0.77, with a recall of 0.68, and an F1-measure of 0.72. On the 'ham' class, the precision was 0.95, recall of 0.97 and an F1-measure of 0.96. The two models had very similar performance statistics overall.

V. Threats to Validity

Our model passes data into Naive Bayes and Logistic Regression models after it has been cleaned and edited, including the removal of stopwords, punctuation, and numerical values. One could argue that these components could be useful features in analyzing text messages to separate spam from ham, because the style and punctuation used in texting is very different from other forms of written communication. We still had high rates of success in our models, but for future projects it may be worth comparing the tokenized version of our edited data to tokenized data left in its original state.

VI. Conclusion

This project was able to classify text messages as spam or ham with an accuracy of " ". We used a variety of approaches including " ", eventually settling on " " as it provided a very precise categorization on our dataset. This project served as a continuation of our work into natural language processing through the Natural Language ToolKit (NLTK), and provided us with further experience in data cleaning, data processing, and natural language processing. More spefically for the Spam detection problem, our model confirmed some of our assumptions on tokens that are generally associate with Spam-type messages such as "award" or "selected". Based on the findings on the given dataset, one can conclude that some tokens very strongly indicate whether a message falls into the Spam category, and that there are general characteristics for these types of messages.

Acknowledgment

Thank you to Professor Chen, who taught us NLTK fundamentals and the theoretical and practical underpinnings of natural language processing.

References

[1] Clement, J. "Spam Statistics: Spam e-Mail Traffic Share 2019." Statista, 4 Dec. 2019, www.statista.com/statistics/420391/spam-email-traffic-share/.

[2] M3AAWG, www.m3aawg.org/for-the-industry/email-metrics-report. www.jstor.org/stable/24710076.

[3] A. H. Wang, "Don't follow me: Spam detection in Twitter," 2010 International Conference on Security and Cryptography (SECRYPT), Athens, 2010, pp. 1-10.

[4] McCord M., Chuah M. (2011) Spam Detection on Twitter Using Traditional Classifiers. In: Calero J.M.A., Yang L.T., Mármol F.G., García Villalba L.J., Li A.X., Wang Y. (eds) Autonomic and Trusted Computing. ATC 2011. Lecture Notes in Computer Science, vol 6906. Springer, Berlin, Heidelberg