Spam, or Ham?

Charlotte Jin
University of Colorado, Boulder
yuji5852@colorado.edu

Erica Lee
University of Colorado, Boulder
erle3969@colorado.edu

Jack Santaniello
University of Colorado, Boulder
cisa6492@colorado.edu

Cindy Yu
University of Colorado, Boulder
xiyu2891@colorado.edu

Abstract

Email is a popular mode of communication nowadays. However, due to its popularity, spam emails are sent out every day. These spam emails can impact people negatively and even threaten online safety. This project aims to find a method to utilize machine learning and artificial intelligence to filter spam emails in email boxes. Solving this problem can effectively save time by not looking at spam emails, and at the same time, it can also protect online safety and wellness. In this project, we focus primarily on the Multinomial Naive Bayes Classifier. Besides the Multinomial Naive Bayes Classifier, this problem has also been solved by a Support Vector Machine, spaCy, and Recurrent Neural Network. By solving this problem properly, the Multinomial Naive Bayes Classifier was chosen because it provides the best recall rate for ham. The recall rate for the ham label is the core element to select the proper model because important emails should not be missed. Although the Multinomial Naive Bayes Classifier may miss spam emails sometimes, it can ensure no important emails will be identified as spam emails. For the next step, our Multinomial Naive Bayes Classifier should be optimized by having higher accuracy and higher recall for spam to correctly identify spam emails. It may also be worthwhile to explore the spaCy or the Support Vector Machine models more to provide better results.

1. Introduction

1.1 Statement of Problem

The nature of communicating through electronic messages has proven to take over our modern lives. The inception of electronic mail, more commonly known as email, in the early 1960s, changed the way the world would communicate with each other through technology forever. Since then, it has become one of the most important and standard modes of communication within our modern society. Everyone has an email account, or two, or five, for anything from their personal to business and work purposes. Unlike instant messaging, it has the power to transmit large files and is more accessible for users of all different types of backgrounds. However, alongside this rise in popularity of this platform comes the increase in spam messages. A spam email can mean anything from a less harmful unwanted marketing promotion to more harmful types of content such as computer malware. “Spam emails bring financial damage to companies and annoying individual users” [1]. Given the sheer amount of emails we receive on a day-to-day basis, developing a method to accurately predict whether an incoming email is spam or not spam is crucial to the usability of the platform.

1.2 Importance and Scope of Work

As a team, we chose this topic because of its relevance in our technological lives. We believe that spam emails are harmful and annoying. So filtering spam emails can bring benefits to our daily lives. Our focus was centered on optimizing a machine learning model that uses natural language processing methods to ensure a user’s inbox is free of unsolicited messages. The recall, or the percent of actual positives that were identified, is incredibly important to us as we want to be mindful of the risk of not accidentally classifying an important message as spam, and consequently not letting messages that contain malware slip through the classifier and potentially harm a user’s privacy. After exploring the problem further and considering further research, we have developed a Multinomial Naive Bayes Classifier with perfect recall for ham emails, which is 100%, and the overall accuracy of 91%. While we had

discovered other models with higher accuracy, ultimately prioritizing the score metric of precision is more important for the context of the problem. We may be more likely to accept that there are a few spam emails in our email box, but may not be likely to accept that our important credit-related emails are identified as spam emails.

2. Related Work

As stated in the introduction, combating "spam" is a major issue that we deal with every day. As such, there has been plenty of research done with varying levels of complexity with the two most popular techniques being Content-Based and Case-Based filtering. Content-Based filtering involves analyzing the word features, frequency, and distributions of words and phrases in an email to generate and then test on automatic filtering rules to detect spam emails. They typically use implementations such as Support Vector Machine, K-Nearest Neighbor, Neural Nets, and Naïve Bayesian classifiers. Case-Based filtering involves extracting all spam and non-spam emails from each user. Then, the emails are pre-processed, to evaluate their data in order to split emails into 2 vector sets, one representing spam and one representing non-spam. Incoming emails are then compared to these two vector sets to determine which they resemble the most. For the purposes of our project, we decided to focus mostly on Content-Based filtering.

One of the methods that we decided to do further research on was a Naive Bayesian classifier which led us to Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras' paper *Spam Filtering with Naive Bayes – Which Naive Bayes?* [2]. As the title states, they discuss the different types of Naive Bayes theorems and their application specifically for spam filtering. With their Multinomial Naive Bayes models they are able to obtain a recall of up to 97% which is what we will be attempting to recreate with our data.

3. Data

As stated in the introduction, the project aims to filter spam emails. So the data has been worked on highly related to spam email, and the original dataset was obtained from Kaggle. There are four columns in the dataset, which include "emailID", "label", "text", and "label_num". To explain it in detail, "emailID" is unique for each instance in the dataset, which ensures each instance is unique. "Label" contains two types of categories, which are "ham" and "spam". "Ham" means this email or this instance is not a spam email, and "spam" means this instance is a spam email. So based on this information, "label_num" is directly correlated to "label". The "label num" feature is a binary indicator that assigns feature assigns 0 to "ham" and 1 to "spam". The last feature is called "text", which is the major variable that will be used in the model building process. It is described as a combination of the email's title and email subject. "EmailID" feature and "label_num" feature are integer features and the "label" feature and "text" feature are object features.

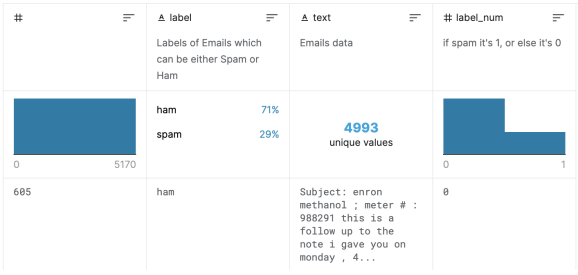

Fig 1. Image of the Kaggle Data

There are 5,171 instances in the whole dataset. To build the model accurately, the whole dataset has been separated into the training set and the testing set where 70% of data are in the training set and 30% of data are in the testing set, which means 3,619 data are in the training set and 1,552 data are in the testing set. In the modeling process, the training set will be used to train the model, and the testing set will be used to test the model. We did the train-test split because it "can help to prevent models from overfitting and to accurately evaluate models" [10]. Besides, the random state is set to 888 to get the same training set and testing set at each time. So if we change the model setting, the results will not be influenced due to the fact that the model takes a different training and testing set.

Besides the train set and test set split, the independent variable and the dependent variable need to be set up. Particularly, the "text" feature will be used as the independent variable, and the "label" will

be used as the target variable. "Email ID" and "label_num" will not be used in the modeling process. Furthermore, because the original dataset we got from Kaggle is clean, we don't need to do extra work to clean the data and remove NA values. In this case, after the train-test split, the data is clean enough to be used for the model building process.

4. Methods

4.1 Approach

When trying to deal with problems related to the language, the first approach that comes to mind is Natural Language Processing, which "helps computers understand, interpret and manipulate human language" [3]. As stated in the related work, there are many different directions we can take such as Content-Based filtering and Case-Based filtering. By researching on and developing models based on the given data, which mainly focus on identifying human language, four different models were decided to build on, including a Support Vector Machine model, Multinomial Naive Bayes Classifier, spaCy, and Recurrent Neural Network.

These four models have their specific strategies for doing Natural Language Processing tasks. "SVM is a supervised algorithm that classifies or separates data using hyperplanes" [4]. "Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in NLP. The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article" [5]. "spaCy is designed specifically for production use and helps you build applications that process and 'understand' large volumes of text" [6]. "RNNs effectively have an internal memory that allows the previous inputs to affect the subsequent predictions. It's much easier to predict the next word in a sentence with more accuracy if you know what the previous words were" [7]. Since different models have their own advantages and disadvantages for building model processes based on these specific tasks, we choose the Multinomial Naive Bayes Classifier eventually after the comparison.

4.2 Process

The model of our choice was the Multinomial Naive Bayes classifier. This model is a probabilistic method that uses conditional probability on independent words. After preprocessing the data by lowercasing words, removing stopwords, stemming words, and tokenizing each one, we converted each word in the corpus into a TF-IDF vector. The data was split into 70% for training the classifier and 30% for testing the model. While the accuracy of this model was slightly lower than some of the other ones at only 91%, we opted to choose the model with the highest recall. Recall considers the number of true positives that were correctly identified, meaning that in the context of this situation, we want to prioritize recall because it is better to correctly classify a spam email as ham than to classify a ham email as spam. The Naive Bayes classifier is classifying ham with 100% accuracy so it is not missing any important emails and sending them to the junk folder. However, since the recall rate of spam is 70%, we are only predicting spam emails correctly some of the time. As a team, we decided the tradeoff of this was worth it as classifying spam emails as ham and letting them slip into a user's inbox is not as harmful as incorrectly classifying ham as spam.

5. Empirical Applications, Experiments, and Results

5.1 Overview

To approach this problem, we explored a few different models: an SVM model, a Naive Bayes Classifier, a spaCy model, and an RNN model. Out of these four, the spaCy model had the highest accuracy at 99%, Multinomial Native Bayes Classifier had the highest recall of Ham at 100%, and spaCy had the highest recall of Spam at 98%.

	Accuracy	**Recall: Ham**	**Recall: Spam**
SVM	93%	94%	89%
Multinomial Naive Bayes	91%	100%	70%
spaCy	96%	99%	89%
RNN	91%	90%	92%

Fig 2. Table displaying accuracy and recall for each model.

5.2 Multinomial Naive Bayes Classifier Results

The final accuracy of the Multinomial Naive Bayes classifier model was 91%. The recall for ham emails is perfect, meaning we never misclassify an important email. The model also has a perfect precision for spam emails, meaning that the emails that were flagged as spam were all correctly identified. Compared to all other models, this classifier had the lowest overall accuracy and the lowest overall recall for predicting spam emails. Below are the performance metrics for the classifier we built.

	precision	recall	f1-score	support
ham	0.89	1.00	0.94	1096
spam	1.00	0.70	0.82	456
accuracy			0.91	1552
macro avg	0.94	0.85	0.88	1552
weighted avg	0.92	0.91	0.91	1552

Fig 3. Classification matrix for Multinomial Naive Bayes Classifier.

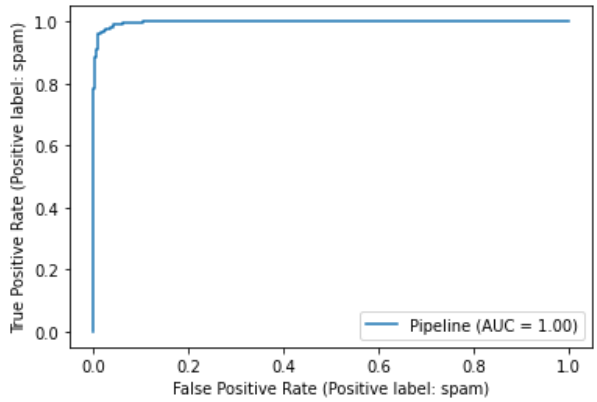

Fig 4. ROC Curve graph for Multinomial Naive Bayes Classifier.

Despite this, we cannot overlook the importance of not misclassifying a ham email as spam. However, we still explored other models to see if there were other methods that could be considered for our problem.

5.3 Connections to Related Work

The Naive Bayes model built in the related work *Spam Filtering with Naive Bayes – Which Naive Bayes?* had a ham recall of 99.9%. As shown in the chart above, our Multinomial Naive Bayes model achieved a perfect recall for ham emails, proving that we were able to match the paper's performance. However, their recall for spam was 97% while our model had a lower spam recall of 70%. The drawbacks of our lower spam recall is addressed in later sections as we are still optimizing for the highest ham recall for our project.

5.4 Support Vector Machine Model Results

A common alternative to using Naive Bayes in Natural Language Processing is a Support Vector Machine (SVM) which takes a vector of numbers (which can be calculated from word frequencies). From here it uses the vector of word frequencies to classify our email. We have included an SVM for comparison purposes to see how our data performs against another competing classifier. As you can see in Figure 5, our SVM outperforms the Multinomial Naive Bayes in accuracy (93%) but we sacrifice our re-call only obtaining a recall of 94%.

	precision	recall	f1-score	support
0	0.96	0.94	0.95	1102
1	0.87	0.89	0.88	450
accuracy			0.93	1552
macro avg	0.91	0.92	0.91	1552
weighted avg	0.93	0.93	0.93	1552

Fig 5. Classification matrix for Support Vector Machine model.

5.5 spaCy Model Results

Another method we used was called spaCy, which is "an open-source software library for advanced natural language processing, written in programming languages Python and Cython" [6]. "spaCy is designed specifically for production use and helps you build applications that process and 'understand' large volumes of text. It can be used to build information extraction or natural language understanding systems or to pre-process text for deep learning" [6]. The model firstly converts target variables into the format that spaCy likes, where they combine two categorical variables into a list. Since the project aims to identify if emails are spam or not, the value for spam is False and the value for ham is True. By building the model and training the model with training data, the model is prepared for testing. To increase the processing accuracy, the model first

finds the accuracy based on the validation set, and then finds the classification report based on the testing set. “spaCy can recognize various types of named entities in a document, by asking the model for a prediction” [6].

	precision	recall	f1-score	support
0	0.96	0.99	0.97	1090
1	0.98	0.89	0.94	461
accuracy			0.96	1551
macro avg	0.97	0.94	0.95	1551
weighted avg	0.96	0.96	0.96	1551

Fig 6. Classification matrix for spaCy model.

5.6 Recurrent Neural Network Model Results

The last method we tried is called RNN model. It is made up of feedback links, and this model could deal with both entire data sequences and single data points. RNN takes inputs from the previous stage and gives its output as input to the next stage.

We first made the message content and subject combined into one feature, then turned the text into a sequence of tokens ids, and pad ids sequence so that the length could be kept the same. Lastly, put them into the RNN model. The result is shown below (ham is 0 and spam is 1)

	precision	recall	f1-score	support
0	0.96	0.90	0.93	1090
1	0.80	0.92	0.85	461
accuracy			0.91	1551
macro avg	0.88	0.91	0.89	1551
weighted avg	0.91	0.91	0.91	1551

Fig 7. Classification matrix for Recurrent Neural Network model.

5.7 Model Comparisons

Compared to the Multinomial Naive Bayes model, the SVM model was strong in that it was very balanced but was unfortunately still having trouble with accurately capturing ham emails correctly. The RNN model was also stronger overall, but with a 9% difference between the ham recall for the Multinomial Naive Bayes classifier, it was not optimal. Lastly, we explored the spaCy model as another unique method that had the highest accuracy and spam precisions. We did not choose this model because we believe we need to research it further to understand its complexities, and only chose to explore it as it was a different method. Despite building other models using varying methods, our Multinomial Naive Bayes classifier is the best model for prioritizing ham email recall rate.

6. Conclusion

By applying the concepts of conditional probability through a Naive Bayes Classifier, we can predict through an email’s contents whether the message is spam or ham with a 100% recall rate. We also believe that further hyperparameter tuning and cross-validation concepts from Machine Learning could improve the predictive ability of the classifier. Other models such as the SVM and spaCy model are excellent for balanced classifications, but once again we opted to prioritize recall for ham so that no important emails are lost to the spam folder. The RNN model performed relatively well, but due to its low recall for ham, it may not be the best model type to select for this specific business problem.

When choosing a priority model, model accuracy does not mean everything, just like our model. Although the Multinomial Naive Bayes classifier was not the highest accuracy model, our team decided to make the tradeoff that classifying spam emails as ham and letting them slip into a user’s inbox is not as harmful as incorrectly classifying ham as spam. However, one of our next steps is to optimize the model further to get a higher recall for spam.

6.1. Social and Business Impact

If we are able to classify spam correctly 100% of the time, it could potentially have a positive impact on many people and reduce risks such as financial loss, since many spam emails contain links that steal personal information from personal computers. Further optimization can also reduce other types of negative impacts on certain brands or companies, as many spam email subjects are disguised as advertisements and can harm the reputation of the business. Lastly, other types of spam email include explicit content, such as sexually explicit material or violent content. These have a significant effect on the well-being of individuals as they can be exposed to sensitive groups such as children.

Since our model performs so well at correctly identifying ham emails, we can guarantee the user that their important emails will not be lost in the void of the spam folder. We still recognize that due to our model selection, there will be more spam emails in the user's inbox, resulting in the downsides of spam listed above. Ham emails are still more important as they can contain information such as bank statements, bill reminders, and other reminders and we did not want to sacrifice this for higher overall performance.

7. References

[1] Enrico, Anton. A survey of learning-based techniques of email spam filtering, 2009.

[2] Vangelis, Iron, Georgios. Spam Filtering with Naive Bayes - Which Naive Bayes?

[3] SAS. Natural Language Processing (NLP) What is it and why it matters.

[4] PythonWife. The Support Vector Machines (SVM) algorithm for NLP.

[5] Shriram. Multinomial Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2022.

[6] spaCy 101. spaCy 101: Everything you need to know

[7] Christopher Thomas. Recurrent Neural Networks and Natural Language Processing, 2019.

[8] Emmanuel, Joseph, etc. Machine learning for email spam filtering: review, approaches and open research problems.

[9] LaBianca, Ivan. How Spam Filtering Works (And How To Stop Emails From Going To Spam)

[10] Jacob Solawetz. Train, Validation, Test Split for Machine Learning, 2020.