SMS Spam Detection

ECS 171 Machine Learning

Heidi Trinh, Kazim Jafri, Brandon Hillig, Youssef Qteishat

Group 7 Project Report

1 Introduction and Background

Spam is generally defined as any sort of unsolicited message sent to a large number of people. These messages come in a variety of types including advertisements, phishing, money scams, and malware warnings. The objective of these malicious messages is to mislead someone, often resulting in computers infected with malicious software and stolen personal information. Spam messages are typically distributed via a multitude of online communication platforms such as email, phone calls, social media, and text messaging.

In today's digital age, the usage of mobile devices has become increasingly prevalent. Namely, a survey from the Pew Research Center disclosed that 97 percent of Americans own a cellphone of some kind. Along with the widespread adoption of mobile devices, comes the ubiquitous use of Short Message Service (SMS) as a means of primary communication. While this method of communication is convenient and efficient, it introduces the challenge of managing large amounts of unsolicited messages. In 2022, Americans encountered a staggering 225 billion spam text messages, representing a 157 percent increase from the preceding year (PR Newswire). In the same year, the Federal Communications Commission issued a consumer alert, warning American consumers about the rising threat of spam text messaging scams. Therefore, the development of spam detectors has gained significant attention in recent years.

Without the use of spam filtering software, the average user would be extremely vulnerable to encountering spam messages. In addition to inconveniencing the user, SMS spam may pose a security risk and compromise user privacy. It is important that spam is accurately identified and filtered out to avoid the spread of these malicious messages. Machine learning methods could be used to effectively predict whether or not an incoming message should be flagged as spam. This can assist in reducing the amount of exposure to spam and decrease the likelihood of a user falling victim to downloading malicious software or releasing their personal information.

2 Literature Review

Spam detection has been an ongoing issue for years, with spam content tremendously increasing within the past decade as the use of mobile devices and social media rises. Various machine learning techniques have been developed and used to combat the rise of spam content. This section will present prior work related to spam detection done by different researchers.

Kontsewaya et al. showcased and compared six widely used classification algorithms in machine learning for spam filtering. These algorithms included K-Nearest Neighbors, Naïve Bayes, Decision Tree, Support Vector Machine, Random Forest, and Logistic Regression. The dataset used consisted of emails which were cleaned, tokenized, and passed through a vectorizer. To extract the features, CountVectorizer was used to convert the words within the emails to integers. The researchers found that both Logistic Regression and Naïve Bayes resulted in an accuracy of 99%, achieving the best performance out of the six considered models. Additionally, the models achieved a precision of 98% and 97% respectively. The lowest performing algorithm was Random Forest which had an accuracy of 84% but produced 100% precision. Furthermore, similar work conducted by Maram achieved an accuracy of 98% using the Multinomial Naïve Bayes model. In this case, the dataset consisted of SMS spam and ham as opposed to the previous email format. The process was similar with few exceptions such as the use of the Natural Language Processing techniques of stemming and lemmatization prior to using the CountVectorizer. Finally, Sjarif et al. used a similar dataset but instead implemented spam detection with emphasis on the term frequency-inverse document frequency method (TF-IDF) for feature extraction. Rather than utilizing the CountVectorizer to extract features from the data as in the previous studies, the TF-IDF method was applied to the data. This method considers the importance of a word in the document in comparison to the entire corpus. Sjarif et al. compared their findings across five models which were used in conjunction with the TF-IDF method. Their results demonstrated that Random Forest combined with TF-IDF outperformed the other models, achieving an accuracy of 97.5% and 98% precision.

3 Dataset Description and Exploratory Data Analysis of the Dataset

We chose the SMS dataset by UCI Machine Learning to use as our test subject as we form a model that can flag incoming spam messages as accurately as possible. Our dataset consists of 5,574 messages, some of which are spam and others which are ham (non-spam). Messages labeled spam are malicious messages, while ham are genuine ones. 3,375 SMS messages are randomly chosen ham messages from the NUS SMS Corpus (which is a dataset that contains over 10,000 real messages for research purposes). 425 SMS messages are manually taken from the Grumbletext website. Another 450 are from Caroline Tag's PhD thesis. The last 1,324 messages are from SMS Spam Corpus v.0.1 Big (which has been used for research in the past).

Though the data had already been cleaned and tagged as spam or ham accordingly, we noticed minor discrepancies upon exploration of the dataset. To begin the preprocessing step, we removed NA values so that our data only consisted of valid entries. This involved dropping three columns labeled 'Unnamed: 2,' 'Unnamed 3,' and 'Unnamed 4.' We also renamed the columns 'v1' and 'v2' to 'Class' and 'Text' respectively. In addition to this, the labels 'ham' and 'spam' within the 'Class' column were correspondingly mapped to 0 and 1.

Upon further investigation of the dataset, we had found that out of 5572 observations, 403 of them were duplicates and needed to be dropped to avoid inaccuracies. From there, we converted all characters in each message to lowercase and removed punctuation as part of our tokenization step. After tokenizing the data, we removed all stopwords from each observation, as these words are abundant but do not offer significant value to the data and add unnecessary processing time. To do this, we utilized the list of English stopwords from the Natural Language Toolkit (NLTK) library. Lastly, we used the Natural Language Processing technique of stemming to reduce the inflection of words by removing any affixes on them. For example, the word 'texting' would be reduced to 'text' after stemming.

To further analyze the data, we decided to look at the contents of the ham messages in comparison to the spam messages. For this, we utilized the WordCloud library to make two different WordClouds which provided us with a visual representation of the text data. The larger the word is, the more frequent it is within the document. The three most frequent words found in ham messages were 'u,' 'im,' and 'ok.' In contrast, the three most frequent words found in spam messages were 'free,' 'call,' and 'text.'

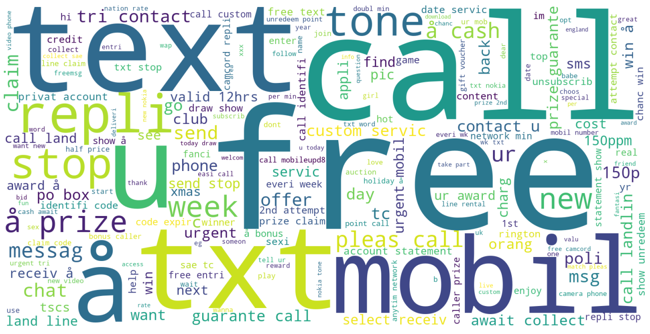
Figure 1: Worldcloud for most frequent words in ham messages

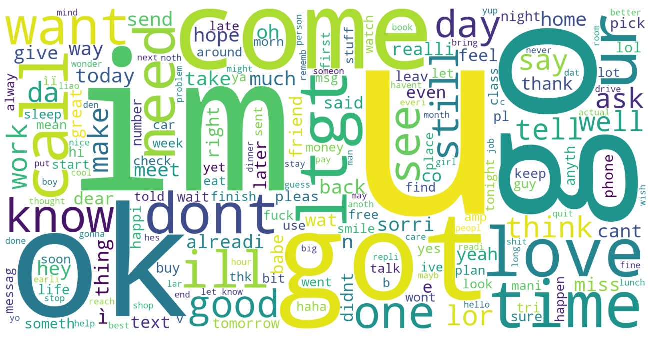
Figure 2: Worldcloud for most frequent words in spam messages

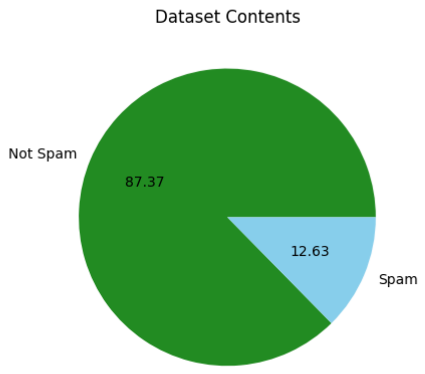

Figure 3: Dataset contents of Ham and Spam

Spam Dataset	Total
Grumbletext website	425(spam)
NUS SMS Corpus (NSC)	3,375(ham)
Caroline Tag's PhD Thesis	450(ham)
Corpus v.0.1 Big	1,002(ham) 322(spam)

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win...	NaN	NaN	NaN
3	ham	U dun say so early hor...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf...	NaN	NaN	NaN

4 Proposed Methodology

After the data was cleaned, it was ready to be vectorized (convert text data to numerical data). For the feature extraction stage, we initially utilized CountVectorizer from the Scikit-learn library. However, we found that this vectorizer simply counted the frequency of a word appearing in a document without considering any other factors. We decided that this vectorizer underperformed in comparison with others so we quickly moved to the term frequency-inverse document frequency (TF-IDF) method. The TF-IDF Vectorizer takes into account the importance of a word in comparison to the entire corpus in addition to the frequency of the word within a document. After passing the textual data through the TF-IDF Vectorizer, we were left with vectors rather than words. These vectors would act as features that we could then pass into our models for training. Our data was divided into training and test sets, with a ratio of 90:10 respectively.

Having split our data, it was time to execute our models. To ensure a comprehensive evaluation, we employed three different models: Logistic Regression, Naive Bayes, and Random Forest. We chose these models as they seemed prevalent according to previous research regarding spam detection. Based on the literature review, we learned that there was an overall high success rate for the Logistic Regression and Naive Bayes models in previous studies. As for the Random Forest algorithm, it was observed that it had performed well in addition to the TF-IDF Vectorizer in past research. For each of these models, we initially cleaned the data and mapped the labels ‘ham’ and ‘spam’ to 0 and 1 respectively as previously mentioned.

Once our models were constructed, we proceeded with hyperparameter tuning to optimize their performance. By fine-tuning the hyperparameters, we aimed to achieve the highest possible accuracy without falling into the trap of overfitting, which would result in a model that performs well only on the training data but poorly on new, unseen data. For Multinomial Naive Bayes for instance we iterated through 100,000+ values to get an ideal alpha. We then chose an alpha value based on the best test accuracy.

5 Experimental Results

As mentioned earlier, we decided to stick with three primary models to put our project against: Naive Bayes, Random Forest, and Logistic. The only way to be able to process these values was to use a vectorizer. Once converted into a matrix-style table of frequencies, we were able to initialize the model. As the words were all converted into a matrix, we found it rather difficult to try to create a classification visual data table with a line splitting types of words. Instead, we opted to go for a word cloud visual to represent the most common or popular words in both the list of spam messages and the list of non-spam messages.

Once we had our models ready to run, we found that all of the models were pretty accurate from the start. On average, all of our models hit a 95%+ accuracy rating which we believed to be pretty good. We decided to try to run hyper-parameter tuning on the Naive Bayes model to see how it would affect our results, and we found that the best values put it somewhere between logistic and random forest. Considering this, we decided it was sufficient to use the default values for the remaining two models as their accuracies were already relatively high.

The other two key data points we collected were precision and recall. Precision is the comparison of the true positives to all of the values that the model marked as being positive (true and false positives) in effect, the lower the precision, the more often it would mark a post being falsely part of a category that it does not belong to. For example.if a message was spam, but it marked it as not spam, the not spam precision would decrease as this is a false positive for not spam. On the other hand, recall is the comparison of true positives to all of the values that should have been marked positive (true positives compared to true positives and false negatives). If we look at a spam message one more time, if the model marks a spam message as not spam, this negatively impacts our recall.

As we can see in the data table below, our accuracy was on average 96-97%. We considered this a sufficient metric. Moving onto our precision, we found that only the logistic model seemed to struggle with precision for spam. Comparing it to the other models in terms of precision, the other two seemed to do pretty well at 96 and 100% respectively.

Looking now at the precision for the non-spam detection, we can see a pretty consistent result with an average of 96% for all three models. There doesn't seem to be any discrepancy here and we were satisfied with the data collected. We found similarly high results for the recall for non-spam messages. Considering that our dataset was over 80% non-spam messages, we do not see this as very surprising, if anything an indication that our data may have been slightly biased towards non-spam messages.

The final value we need to consider is the recall rate for spam messages. This is the notable value that was lacking throughout the board for all three models. As mentioned earlier, we had the issue with data being biased towards non-spam data leading to a possible lack of training with spammy data. The low recall rate indicates that there were a lot of spam messages that were detected as being not spam. When we run our model on new data that we create on the spot - similarly to how we did it for our demo - we see that our models do struggle to detect when a message is intended to be a spam message. In order to trigger the spam flag, we found that we would have to make unnecessarily large messages and jam-pack them with a bunch of high-frequency spam words.

DEMO:

To implement the GUI portion, we decided to use a python graphics library called Tkinter. The way we imagined our program to work would be to have some ability to load in the model that we want to use and submit any custom message to it. Since we had three relatively accurate models, we decided it would be useful to be able to load in any of the three models and put them against each other, trying to figure out which model was able to catch a message and which model was more likely to miss it. To do this, we load in the logistic model by default, but the user has the option to load in any model and submit the message again to see if the other models flag the message as spam or not.

For additional information, we included the classification report statistics below each of the loading buttons. For additional utility, we included a little textbox that displays the most recent message sent to the system below the textbox that displays the model that is currently being displayed. We also colored the text to tell the user whether or not the text is a spam message or not. We believe our GUI contains a good level of functionality and information.

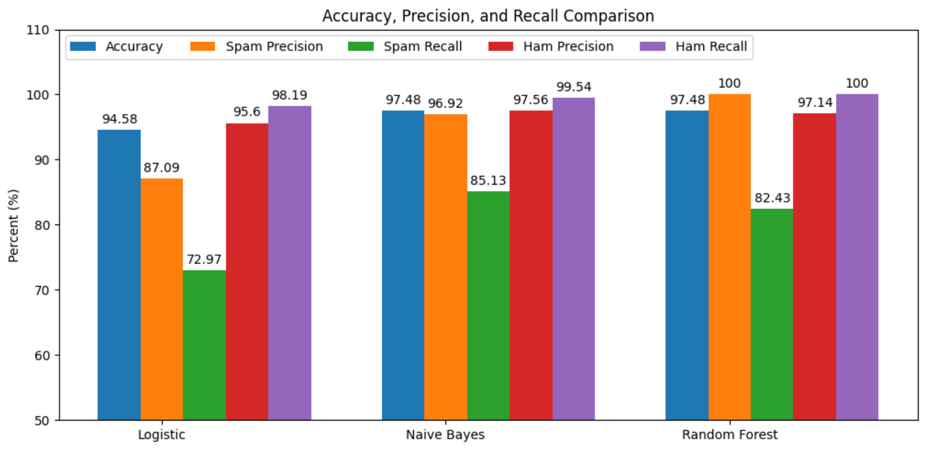

Figure 4: Accuracy, Precision, and Recall results for all three models

6 Conclusion and Discussion

In conclusion, we found that Naive Bayes yielded the best precision and recall, both at 98

One of the ways we can improve the precision and accuracy of our models is by selecting a better dataset. During preprocessing, we realized that our dataset was slightly unbalanced. This can be addressed by either oversampling or selecting a more robust and balanced dataset. In addition, selecting a more modern dataset would also improve the effectiveness of all of our models. Given UCI's SMS Spam dataset was collected in 2012, it is not reflective of what spam messages look like in 2023.

In order to improve the implementation of our models, we could have utilized lemmatization instead of stemming prior to vectorizing the raw text. Since stemming removes prefixes and suffixes from words, it can cause some of the text to lose meaning or proper spelling. On the other hand, lemmatization considers the context of the word and converts it into its base dictionary form (ie. lemma). This would result in better detection of words in spam/ham messages. Another way to improve the implementation of our models is more extensive hyperparameter tuning. One method that comes to mind is utilizing Gridsearch to find the best combination of the vectorizer's hyperparameters. We can also consider attributes such as sentence structure instead of simply relying on the frequency of "spammy words".

7 References

Dataset:
https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

Introduction:
“Mobile Fact Sheet.” Pew Research Center: Internet, Science Tech, 7 Apr. 2021, www.pewresearch.org/internet/fact-sheet/mobile/.

“Robocall Growth Slows, Spam Texts Explode According to Robokiller 2022 Phone Scam Insights Report.” PR Newswire, 6 Mar. 2023.

United States, Congress, Wiquist, Will. FCC Issues Consumer Alert on Robotext Scams, 28 July 2022. https://www.fcc.gov/robotext-scams-rise. Accessed 6 June 2023.

Literature Review:
Kontsewaya, Yuliya, et al. “Evaluating the Effectiveness of Machine Learning Methods for Spam Detection.” Procedia Computer Science, vol. 190, 22 July 2021, pp. 479–486, https://doi.org/10.1016/j.procs.2021.06.056.

Maram, Sai Charan Reddy. “SMS Spam and Ham Detection Using Naïve Bayes Algorithm.” SSRN Electronic Journal, 22 Sept. 2021, https://doi.org/10.2139/ssrn.3908998.

Sjarif, Nilam Nur Amir, et al. “SMS Spam Message Detection Using Term Frequency-Inverse Document Frequency and Random Forest Algorithm.” Procedia Computer Science, vol. 161, 2 Jan. 2020, pp. 509–515, https://doi.org/10.1016/j.procs.2019.11.150.

Literature Review:

Kontsewaya, Yuliya, et al. - Evaluating the Effectiveness of Machine Learning Methods for Spam Detection
Discusses 6 widely used machine learning algorithms (KNN, Naive Bayes, Decision Tree, SVM, Random Forest, Logistic Regression) **Dataset**: Email
Flow: Preprocessing, Tokenization, CountVectorizer, Train, Test
Compares performance (classification report) of all algorithms
Logistic Regression and Naive Bayes performed the best (most efficient) with 99% accuracy, 98% and 97% Precision respectively, 96% and 99% Recall respectively
Random Forest had 84% Accuracy, 100% Precision, 28% Recall

Sai Charan Reddy Maram - SMS Spam and Ham Detection Using Naïve Bayes Algorithm.
SMS dataset
Removed stopwords, stemming + lemmatization
Countvectorizer
Multinomial naive bayes often used in NLP
98% accuracy
Sjarif, Nilam Nur Amir, et al. "SMS Spam Message Detection Using Term Frequency-Inverse Document Frequency and Random Forest Algorithm."