Spam Detection Model

COMP 333: Data Analytics

Instructor: Dr. Yaser Esmaeili Salehani

Submitted by:

Matteo Robidoux
Student ID: 40282589

Raagav Prasanna
Student ID: 40282749

Date:

April 7th, 2025

Concordia University

Abstract

This study looks into the problem of finding spam in different types of input, such as SMS, email, and YouTube comments. Traditional spam detection commonly employs basic models like text vectorization. However, our study indicates that the identification of URLs embedded within messages plays a critical role in determining spam likelihood. Therefore, in addition to conventional text-based training approaches, we introduced a separate URL-based model to enhance prediction accuracy.

Furthermore, we observed key differences across input datasets. SMS messages consist solely of a body containing text, whereas emails include a subject line. Moreover, YouTube comments also often contained spam related to self-promotion, with messages similar to "Please subscribe to my channel". Additionally, emails and comments provided unique attributes, such as email subjects and the authors of comments, which are also key pieces of data required to identify spam.

To address these variations, we developed three specialized models for SMS, email, and comment spam detection, each catering to the different datasets that we acquired. The individual model's predictions were then combined with the output of the URL spam model whenever a URL was present. This hybrid approach significantly improved classification accuracy, achieving 97.99% accuracy for the URL model, 96.00% for SMS, 96.00% for email, and 95.33% for YouTube comments, significantly outperforming traditional methods.

1 Introduction

The rise of digital communication has led to an increase in the amount of unwanted and potentially harmful spam messages across various platforms. This includes SMS messages, emails, and content within online comment sections. Spam messages can range anywhere from harmless advertisements, to phishing attempts and even malware distribution. This emphasizes the need for effective spam detection, which is needed for ensuring the security of a user, as well as the integrity of a platform. The traditional spam detection techniques, commonly known as "ham or spam", often rely solely on simple text-based models, such as a TF-IDF Vectorizer. While these methods are effective when used to identify many different spam messages given the right input dataset, they often fail to account for additional information catering to the input, leading to both false negatives as well as false positives. In short, on many occasions, messages that are not spam are marked as spam, and messages that are, are not.

One key aspect of spam identification that is not present within traditional methods is the identification of URLs. Many spam messages contain malicious links, requiring models to take these links into account. Furthermore, a message can also contain a valid url sent from a valid user, which means that messages containing urls should not be marked as spam simply because they contain a url. Instead, the url itself should be analyzed to determine whether or not it is spam. Our study demonstrates that integrating URL-based detection with text-based models significantly improves the ability to detect spam, particularly when it comes to wrongful spam detection.

Moreover, spam characteristics differ across various platforms. SMS messages primarily consist of just text, whereas emails include components like subject lines. YouTube comments, on the other hand, often involve engagement-driven spam, such as self-promotion; as well as peculiar author names that could indicate spam. These differences require the need to develop specific models for each input type, as opposed to just one model.

To address these challenges, we propose a hybrid spam detection algorithm consisting of three specialized models for SMS, email, and YouTube comments. Each model is designed to identify the unique characteristics of its respective dataset, while utilizing a separate independent model for the embedded URLs in the event they exist. When a message contains a URL, the URL model's prediction is combined with the text model's prediction, to determine whether the message is spam. Our results demonstrate that this approach significantly enhances detection accuracy, outperforming traditional methods.

The rest of this paper is organized as follows: Section 2 reviews related work in spam detection, and why these models are inefficient. Section 3 describes our methodology, including dataset preprocessing, model architectures, and training procedures. Section 4 presents our experimental results and evaluation metrics. Finally, Section 5 discusses key findings, limitations, and potential future improvements.

2 Related Work

Spam detection has been extensively studied across various communication platforms, including SMS, emails, and online comments. Traditional spam detection approaches, as mentioned previously, use vectorization techniques such as TF-IDF, where all of the English stop words are removed from the text and only the necessary keywords remain for training. While these methods have been successful, they often overlook some important features that are very helpful in spam classification.

A notable study by Gupta et al. follows the conventional ham or spam detection algorithm using TF-IDF vectorization [1]. Their research applies machine learning models to the UCI SMS Spam Collection dataset [2], using text-based features extracted from SMS messages. Their results show strong performance in spam classification, using traditional approaches such as Naïve Bayes and SVM, emphasizing that this is an effective technique.

However, while training models on the same dataset [2], we identified a significant limitation in the way spam messages were labeled, especially when a URL was present in the message body. The dataset shows a strong bias, as nearly all SMS messages containing URLs are labeled as spam. This introduces a risk that models trained on this data may incorrectly generalize, leading to a high false positive rate for messages containing legitimate URLs, significantly impacting simple sms messages such as a getting sent a video link from a friend. Consequently, SMS messages may be disproportionately influenced by the presence of a URL rather than the actual content of the message.

This observation highlights a critical shortcoming of traditional text-based spam detection models, being the lack of a dedicated mechanism for evaluating URLs. Our solution resolves this issue by introducing a separate URL-based classification model, which is then used in conjunction with either the SMS message model, Email model, or YouTube comments model, to predict whether they are spam. By independently determining the likelihood of a URL being spam, this hybrid approach reduces dataset bias and ensures more balanced spam classification, particularly for messages containing URLs.

3 Methodology

3.1 Datasets and Preprocessing

For this study, in order to ensure that each of the models have more than enough data to train and eventually perform inference on, we used a wide variety of different datasets. Specifically, we used datasets for URLs, SMS messages, Emails, and YouTube comments

URL Dataset

For the URL's we decided to have a dedicated dataset containing plenty of addresses that were classified to be either spam or non spam. This way, the model had more than enough data to decide which urls were malicious, preventing common links such as from platforms like YouTube and TikTok from being marked as spam. The kaggle dataset that we used, titled Malicious and Benign URL's [3], contained over 450,000 url's, that were marked as either being 0 for benign, or 1 for malicious.

Table 1: URL Dataset Structure

	url	label	result
0	https://www.google.com	benign	0
1	https://www.youtube.com	benign	0
2	https://www.facebook.com	benign	0
	⋮		
410975	http://mytorsmired.ru/gate.php	malicious	1
410976	http://narbit.com/rss/feed/stream/	malicious	1
410977	http://narbit.com/rss/feed/stream	malicious	1

In terms of preprocessing, we modified the names of the result and url columns to is_spam and text

respectively. Furthermore, we dropped all other columns as these were the only 2 that were needed. Finally, we normalized the text column by removing trailing whitespaces, as well as non unicode characters identified by ¿.

Table 2: URL Dataset post Preprocessing

	text	is_spam
0	https://www.google.com	0
1	https://www.youtube.com	0
2	https://www.facebook.com	0
	⋮	
410975	http://mytorsmired.ru/gate.php	1
410976	http://narbit.com/rss/feed/stream/	1
410977	http://narbit.com/rss/feed/stream	1

SMS Datasets

For SMS messages, we utilized 2 datasets being the UCI SMS Spam Collection Dataset [2], containing a total of 5575 messages, as well as another kaggle dataset named SMS SPAM DATASET [4], containing 10286 messages. Both datasets contained the headers v1 and v2, where v1 categorized the row as either ham or spam, and v2 contained the text of the SMS message.

Table 3: UCI SMS Spam Collection Dataset

	v1	v2
0	ham	Go until jurong point, crazy only in ...
1	ham	Ok lar... Joking wif u oni
2	spam	Free entry in 2 a weekly comp to win FA Cup ...
		⋮
5574	spam	WINNER!! As a valued network customer you have ...
5575	spam	Had your mobile 11 months or more? Then ...

Table 4: SMS SPAM DATASET

	v1	v2
0	spam	Congratulations! You've been selected for a luxury vacation getaway...
1	spam	URGENT: Your account has been compromised. Click here to reset your password immediately...
2	spam	You've won a free iPhone! Claim your prize by clicking on this link now...
		⋮
9526	ham	Okie...
9527	ham	"Aight, I'm chillin in a friend's room so text me when you're on the way"

Email Dataset

When extracting the email data, we went for a total of 2 datasets. The first kaggle dataset titled Email Spam Dataset [5], contained 3 data subsets. The second dataset was titled Spam Mails Dataset [6]. In total, these 4 csv files provided us with over 23000 rows of data.

For preprocessing most of the methods stayed exactly the same as prior. However now we had the added challenge of needing to extract the subject from the email. This was done through simple regex matching and adding a subject column to our normalized dataset.

Table 5: TODO: ADD NORMALIZED DATA TABLE

subject	text	is_spam
⋮		

YouTube Comments Dataset

For comments we used a single kaggle dataset titled YouTube Comments Spam Dataset. This dataset contained the following headers: COMMENT_ID,AUTHOR,DATE,CONTENT,VIDEO_NAME,CLASS, and contained 1962 rows of data.

When preprocessing, all previous methods were also applied. Furthermore, we decided to also add the author column in the normalized dataset, as we figured that it could potentially be useful in predicting spam.

Table 6: TODO: ADD NORMALIZED DATA TABLE

author	text	is_spam
⋮		

3.2 Model Features and Training

In this section, we discuss the specific architecture and training process of each of the four models used in our spam detection system: the URL model, SMS model, Email model, and YouTube Comments model. Each model is designed to capture specific characteristics of its respective data type while integrating with the URL classifier to improve overall accuracy.

3.2.1 URL Model

Feature Extraction

We extract the following features from each URL:

- Structural features
  - Length of the URL, number of dots, number of query parameters, etc.
- Specific keywords
  - Presence of words like “free,” “win,” “click,” “offer,” and brand names such as “PayPal” or “Amazon.”
- Domain and subdomain features
  - Length of the domain name, presence of suspicious top-level domains (TLDs) such as .xyz or .biz
- Redirect and path features
  - Presence of redirects, subdomain keywords like ”auth” or ”login,” and path length.

Training

We trained an XGB classifier splitting the dataset into 80

- 150 estimators
- Max depth of 6
- Learning rate of 0.1

- Subsampling of 0.8

The model achieved the following results after training 450,000 rows on a Lenovo Laptop with a Ryzen 7 4700U and 16gb of ram, taking 2.86 secs:

Table 7: URL Model Classification Report

	Precision	Recall	F1-Score	Support
0	0.97	1.00	0.98	69148
1	0.98	0.89	0.93	20888
Accuracy			0.97	
Macro Avg	0.98	0.94	0.96	90036
Weighted Avg	0.97	0.97	0.97	90036

3.2.2 SMS Model

Feature Extraction

We applied TF-IDF vectorization to capture word patterns and important terms while filtering out common English stopwords. The vectorization settings included:

- Maximum features: 7000 (expanded from 5000 for better feature representation)
- Minimum document frequency: 3 (to filter out rare words)
- Maximum document frequency: 0.90% (to remove common words)

Once again training was done via an 80/20 train-test split with the same laptop, using an XGB classifier with the following optimized hyperparameters:

- 400 estimators
- Max depth of 6
- Learning rate of 0.05
- Subsampling of 0.8
- Early stopping rounds of 10 to prevent overfitting

The model achieved the following results after training 5575 rows, taking 0.25 secs:

Table 8: SMS model Classification Report

	Precision	Recall	F1-Score	Support
0	0.92	0.99	0.95	2676
1	0.96	0.62	0.75	631
Accuracy			0.92	
Macro Avg	0.94	0.81	0.85	3307
Weighted Avg	0.92	0.92	0.92	3307

3.2.3 Email Model

Feature Extraction

We applied TF-IDF vectorization separately to both the email subject and body, allowing the model to leverage textual patterns in each section.

Text vectorization settings:

- Unigrams and bigrams
- 5000 max features
- Stopwords removed

Subject vectorization settings:

- Unigrams and bigrams
- 1000 max features
- Max document frequency: 0.95 (to retain common subjects)

Training

We trained an XGBoost classifier, with the same hardware but with a 70/30 train-test split:

- 500 estimators
- Max depth of 10
- Learning rate of 0.05
- Subsampling of 0.8
- Early stopping at 30 rounds

The model achieved the following results after training 23000 rows, taking 249 secs (4.15 mins):

Table 9: Email Model Evaluation Report

	Precision	Recall	F1-Score	Support
0	0.99	0.96	0.98	4498
1	0.94	0.98	0.96	2649
Accuracy		0.97		
Macro Avg	0.96	0.97	0.97	7147
Weighted Avg	0.97	0.97	0.97	7147

3.2.4 YouTube Comments Model

Feature Extraction

Here we once again utilize a TF-IDF vectorizer for text, as well as one for author.

- Textual Features: TF-IDF vectorization applied to comment text
- Author Based Features: TF-IDF vectorization applied to usernames to detect repeated spam patterns from specific users, as well as recognize patterns in usernames that look suspicious

Training

Here we train using a Gradient Boosting Classifier mainly so we can capture hierarchical relationships within the text data, since they are comments. This is all done using the same hardware. The hyperparameters include:

- 150 estimators
- Max depth of 5
- Learning rate of 0.1
- Subsampling of 0.8

The model achieved the following results after training 1962 rows, taking 0.75 secs:

Table 10: Comments Gradient Boosting Model Evaluation Report

	Precision	Recall	F1-Score	Support
0	0.85	0.98	0.91	285
1	0.97	0.83	0.90	302
Accuracy			0.90	
Macro Avg	0.91	0.90	0.90	587
Weighted Avg	0.91	0.90	0.90	587

4 Inference

The inference process across SMS, email, and comments, generally follows the same execution steps While the specifics vary slightly for each type of message, the overall approach remains the same.

4.1 Preprocessing and Feature Extraction

1. URL Cleaning: Any URLs present in the text are extracted and normalized.
2. Vectorization: The cleaned text is transformed using pre-trained vectorizers specific to each message type (SMS, email, comments). Email subjects and comment author names are also vectorized where applicable.
3. URL Feature Extraction: If URLs are found in the text, a feature extraction function analyzes them based on characteristics such as length, presence of suspicious keywords, and domain reputation, as done within the training scripts feature extractor.

4.2 Probaility Estimation

1. Text Classification: The vectorized text is passed through a trained classification model that returns a spam probability score.
2. URL Spam Detection: If URLs are present, they are sent into the URL model to predict the likelihood of them being malicious or spam-related.
3. Weighted Combination: The probability scores from text classification and URL analysis are combined using a weighted sum approach. The weight distribution varies depending on the message type, with URL spam probability often given higher significance if URLs are detected.

4.3 Final Decision

The final spam probability score is then compared against a predefined threshold (typically 0.5). If the score meets or exceeds this threshold, the message is classified as spam; otherwise, it is classified as non-spam.

References

[1] S. D. Gupta, S. Saha, and S. K. Das, "SMS Spam Detection Using Machine Learning," *J. Phys. Conf. Ser.*, vol. 1797, no. 1, p. 012017, Feb. 2021, doi: 10.1088/1742-6596/1797/1/012017.

[2] UCI Machine Learning Repository, "SMS Spam Collection Dataset," [Online]. Available: https://www.kaggle.com/data spam-collection-dataset. [Accessed: 2025-03-24].