

Intelligent Multi-Modal Spam Detection System Using Machine Learning Techniques for AI-Generated Content

Harsh Rakesh Walmiki

Department of Computer Science
MSc/MCA, Academic Year 2025-26
JSPM University, Pune, India
PRN No.: 42456530169
Roll No.:82
harshwalmiki16@gmail.com

Prof. Arshiya A. Khan

Research Supervisor
Department of Computer Science
JSPM University, Pune, India

Abstract

Despite being an old issue, the development of generative AI algorithms made the problem more dangerous. The messages created using LLMs sound very convincing as they mimic human messages. In this paper, we introduce the system for detecting the spam that employs classical NLP text processing algorithms and supervised machine learning considering the existing threat of spam. This solution comprises five steps of the text preprocessing for the SMS messages including lowercasing, noise filtering with the help of regular expressions, removing the stopwords and stemming with the help of Porter Stemmer in order to convert the SMS messages into the feature vectors based on TF-IDF with up to 5,000 features. For the SMS Spam Collection dataset containing 5,572 SMS messages we have used four classifiers including the Multinomial Naïve Bayes, Logistic Regression, Linear SVM and Random Forest. Among all classifiers, the Linear SVM performed the best with 98.2% accuracy, 97.3% precision, 90.6% recall, and F1-score of 93.8%. The part of the problem which we find particularly interesting, and which is discussed mostly in this paper, is whether such a solution can compete with spam produced by the generative AI systems. It is because stemming decreases the diversity of vocabulary which is used by the language models, IDF weight automatically downweights the common tokens that are popular among the language models, and geometric properties of the margins of SVM allow some flexibility in case of perturbation of feature space. We have also developed the web application based on Flask framework for real-time predictions.

I. INTRODUCTION

The significance of text messages can hardly be overestimated. As reported by the findings of the GSMA 2023 annual report, the quantity of SMS in the world is over two trillion per year [1], yet the aforementioned number does not take into account the billions more of text messages delivered via WhatsApp, Telegram, and similar platforms. Besides the gigantic quantity of legit SMS messages, one

should consider spamming, which is consistent, annoying, and sometimes hazardous. Various types of phishing, prize notifications, fake bank messages cost billions to users annually [2]. According to the statistics gathered by Statista, 45% of e-mails are spam, and mobile communication catches up with this trend [3].

Until recently, spam was rather repetitive and predictable so much that one could even regulate their schedule according to the timing of spam messages, and that a decent share of the messages could be easily filtered with the help of keyword detection [4]. However, with the arrival of large language models, including GPT-4 and its open-source variants, things changed drastically [4]. Spam messages created using the above-named model has the capability of generating hundreds of versions of the scam, each of them with flawless grammar and tone. All the signs of the past have gone, and thus the defense against them too [5].

There are three classical approaches to filtering out spam messages. First of all, it is scoring heuristic algorithms, such as SpamAssassin, where a certain amount of points is awarded for each of the signs of the suspect e-mail [6]. This approach works, but it requires constant tuning since spammers are always developing new strategies. Secondly, this is the approach of blacklisting and whitelisting of certain organizations, such as Spamhaus, which blocks any known offenders until they manage to change the IP address [7]. Thirdly, it is collaborative filtering, where the complaints of the users create communal spam signatures database [8].

This is where machine learning comes in because, instead of telling the computer how to recognize spam emails, you would train the computer using appropriately labeled samples [9]. The power of machine learning to generalize, which means to recognize any deviations including those that it has not encountered before, is what makes machine learning interesting in the era of AI-generated phrasing of spam emails [10].

Our paper makes four major contributions that, in our opinion, distinguish it from all previous works on SMS spam detection:

1) We build a preprocessing pipeline that consists of five stages and not only say that “preprocessing helps” but analyze how each stage helps in reducing the variability generated by generative AI systems.

2) We evaluate four classifiers, each belonging to a different algorithmic family on equal terms. The majority of works described here are dedicated to hyperparameter optimization, which could be considered as the measure of how well you are at exploring parameter space and not how different your algorithms are.

3) We provide an analytic explanation of why some of the classical solutions fight AI-generated spam in terms of TF-IDF statistics and geometric properties of SVM margins. We believe that such type of analysis is underrepresented in the current literature.

4) Finally, we present our results in the form of deployed web application that uses REST API. Too many papers analyzing academic spam stop at the confusion matrix.

The rest of this paper is quite traditional in terms of structure. Section II describes existing literature. Section III explains our experiment and model design. Section IV is about our experimental setup. Section V provides the results obtained. Section VI discusses the problem of AI-generated spam.

II. RELATED WORK

A. Origin: Bayesian Spam Filtering

However, there is research by Sahami et al., carried out in 1998, that serves as the basis for the entire area of spam filtering using ML [10]. Using bag-of-words features, a Naïve Bayes classifier successfully classified spam and non-spam emails under the very unrealistic assumption about the independence of words. What is most important is that the systematic difference exists between the languages in spam and non-spam messages.

Around ten years later, Cormack [11] gave a detailed analysis of more than 200 papers in the area of spam filtering. Particularly three results of his were extremely significant for our view on spam filtering. First, ML-based filters always worked better than rule-based static ones. Second, evaluation of filters using accuracy criterion is not appropriate since the losses caused by the classification of non-spam as spam are greater for the end-user than the other way round. Third, there is a 'chicken and egg' problem of personalization of the filter: it requires some training data related to the particular user, but the user has no such data when he or she appears for the first time.

B. SVMs and Ensemble Methods for Text

1998, Joachims [12] demonstrated in 1998 that SVM algorithms worked exceptionally well for text categorization tasks, and the explanation of the mechanism behind its success is still applicable today. The inherent high dimensionality (of thousands of features) of text as TF-IDF vectors does not present a challenge for SVM algorithms

since the optimization criterion for SVM algorithms is margin and not least squares loss. This has been theoretically proven by Vapnik in the SVM literature [13]: the maximum-margin hyperplane minimizes the upper bound of the generalization error.

The invention of the Random Forest algorithm by Breiman [14] presented an entirely new idea: use many decision trees created using bootstrap samples of the dataset, let them vote, and hope that their errors will cancel each other through averaging. As far as text categorization is concerned, the strength of the Random Forest algorithm is the ability of this algorithm to work well with the noisy features, which is typical for high dimensionality (of thousands of features) of the text.

There is an interesting theoretical comparison of Naïve Bayes (generative) and Logistic Regression (discriminative) algorithms offered by Ng and Jordan [15]. The authors claim that Naïve Bayes converges faster than Logistic Regression on small datasets but Logistic Regression wins in the long run as the size of the dataset increases. Our dataset is relatively small (4,400 messages after the 80/20 split).

C. Representation Text Matters

It can be safely asserted that the choice of the text representation plays as vital a role in determining the result as the choice of the classifier itself. In the case of term frequency-based vectors, terms like “the” and “is” carry way too much importance since they have nothing to do with spamming. The issue has been resolved with the help of TF-IDF, according to Salton and Buckley [16], which equals the multiplication of the term frequency and the log of its inverse document frequency. Manning et al. [17] describe this concept in their book on information retrieval.

Indeed, in the light of more modern approaches, the issue of dispersed word representations has been considered. Word2Vec [18] is based on dense vectors, whereas BERT [19] uses context-based embeddings to represent words. Both techniques produce highly effective representations, certainly. However, Kowsari et al. [20] provide a good practical insight in their survey; namely, TF-IDF is still quite a good choice when it comes to short-text classification with clear vocabulary, which is substantially less computationally expensive than the two techniques mentioned above. The average length of an SMS message is about 15–25 words.

D. The AI-Generated Text Problem

Machine-generated text identification has become a field of research on its own. Gehrmann et al. [21] developed the GLTR visualizer, which detects a very specific pattern characteristic of machine-generated text – language models tend to predict high-probability tokens much more frequently than humans, resulting in a very specific uniformity of token distribution. In another line of research, Mitchell et al. [22] came up with the DetectGPT algorithm that relies on the pattern that machine-generated text is generated in regions of negative curvature of the probability surface of the generating model.

The work by Jawahar et al. [23], where it was demonstrated that even basic paraphrasing or back-

translation would cause severe deterioration of AI-generated text detection system accuracy, is very relevant to the topic of this paper. Indeed, if the spammer uses one large language model to generate the template and then uses another language model to paraphrase it, both AI content detector and the spam filter should be able to handle this.

E. SMS Spam Specifically

The Dataset used in this work and generally utilized by other researchers to study SMS spam was created by Almeida, Gómez Hidalgo and Yamakami [24]. The dataset includes 5,572 SMS text samples obtained from several sources including UK Grumbletext site and Singaporean NUS SMS Corpus. Metsis et al. [25] performed benchmarking of Naïve Bayes algorithms on term frequency features and came to conclusion that Multinomial NB performed best.

Augmentation of TF-IDF with manually engineered features (capitalization of the message, number of special characters, mean length of the words) implemented by Gupta et al. [26] resulted in minimal performance boost, which supports an idea that feature set based solely on content is not enough. Ramachandran and Feamster [27] concentrated on completely different side of the problem – network behavior of spam, such as bursty messages from temporary IP-addresses, which is helpful when working with email spam but not with SMS.

III. SYSTEM DESIGN AND METHODOLOGY

A. Pipeline Architecture

The architecture of the system was designed as a pipeline of five components: data ingestion, text preprocessing, feature engineering using TF-IDF, classification, and deployment. Fig. 1 gives an overview of the pipeline.

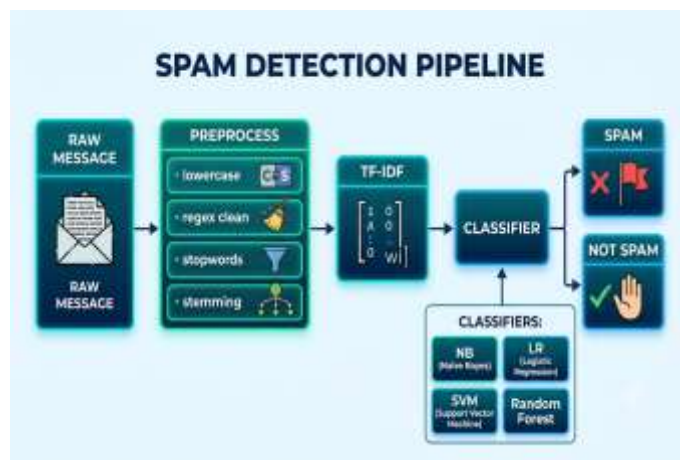


Fig. 1. System pipeline from raw input to deployed prediction interface.

This concept is quite easy to grasp. Everything is meant to have its role. If you want to change a vectorizer for another one or include another classifier to the pipeline, you can do that very easily without altering anything else in the pipeline.

B. The Dataset

We use the UCI SMS Spam Collection [24], probably the most widely used benchmark in this area. Table I has the key statistics for the given dataset.

TABLE I: DATASET CHARACTERISTICS

Property	Value
Total messages	5,572
Legitimate ("ham")	4,827 (86.6%)
Spam	747 (13.4%)
Median ham length	~61 characters
Median spam length	~133 characters
Avg. ham word count	~15 words
Avg. spam word count	~25 words
Language	English

There are two interesting observations. The first one is that ratio in class distribution is 6.5:1 for ham, which does reflect the real world (since most of the emails we receive are real), but, at the same time, it makes classification testing harder because the classifier predicting "ham" would score 86.6% of accuracy. The solution to this problem is stratified split and emphasis on F1-score.

The second one is that spam messages are almost twice longer than non-spam ones. The logic behind this observation is obvious - spam message has to convince user to buy something, therefore, it will always include some kind of sales pitch, URL and/or phone number, and language of urgency, while messages written by friends are usually short. This difference in lengths is not encoded in dataset directly, but it is picked up by TF-IDF.

C. Text Preprocessing in Detail

The pre-processing we conduct is a five stage process. Every stage of the process is associated with handling particular noise, which if not eliminated, will either increase the number of vocabularies we have or create confusion between spam and ham.

Stage 1 – Lowercasing. Everything is lowercased. "FREE", "Free", and "free" are all "free". This is an easy process to conduct, however, if ignored, the vocabulary increases twofold as the same word written in different cases is perceived as another feature. In the context of spam detection using AI technology, the capitalization is randomized; this is addressed in this very first stage.

Stage 3 – Cleaning whitespaces. We reduce multiple whitespaces to single whitespaces and remove the trailing spaces. Small action but has big influence on the process of tokenization.

Stage 4 – Removing stopwords. Using the list of English stopwords from NLTK [28], we remove all the unnecessary words like "the", "is", "and", "but", which are independent of message's type. They are just taking unnecessary space in feature space. And thanks to this step only we have reduced our vocabulary by about 25-30%.

Stage 5 – Porter Stemming. This stage can be called the most important one for defense from paraphrasing produced by artificial intelligence. The algorithm is reducing suffixes in order to decrease words to their stems: "winning" becomes "win", "congratulates" becomes "congrat", "urgently" becomes "urgent". When paraphrases are created by LLM, for example, "claim your prize" becoming "claiming prizes",

all the paraphrases will have the same set of stems at this stage.

Here's a worked example to make this concrete:

```
Input: "CONGRATULATIONS!! You've WON a
$1,000 prize! Call NOW!!!"
Step 1: "congratulations!! you've won a
$1,000 prize! call now!!!"
Step 2: "congratulations you've won a prize
call now"
Step 3: "congratulations you've won a prize
call now"
Step 4: "congratulations you've won prize
call"
Step 5: "congratul youv won prize call"
```

The visual presentation is not that great, but the output is brief, normalized, and, most importantly, looks very similar no matter how LLM changes the message underneath

D. From Words to Numbers: TF-IDF

After the pre-processing stage, the next thing that comes up is to prepare the text for classification. The process employed in this case is TF-IDF, where each term in the text is assigned a weight based on its frequency in the document and its rarity in the entire corpus.

$$weight(t, d) = tf(t, d) \times \log(N / df(t)) \quad (1)$$

where N denotes the total number of documents, and df(t) denotes the number of documents containing the term t. IDF acts as a filter of relevancy, putting to zero all those terms that occur in virtually all the documents and highlighting the ones that occur in specific parts of the collection of documents.

We limit the number of features to 5,000, anything over that is hapax legomena, meaning a one-off occurrence term, and most probably noise, not a relevant term. The resulting representation is a 5,000-dimensional sparse vector for each message having less than 2% non-zero entries.

However, there is a lesser-known, but quite useful feature of the IDF weighting approach that is concerned with the problem of artificial intelligence, yet did not receive adequate attention from the spam research community. As stated by Gehrmann et al. [21], language models usually favor high-probability and thus high-frequency tokens. But IDF weighting inherently down-weights precisely such tokens. In other words, the TF-IDF vectors, as it were, are calibrated right out of the box with regard to the linguistic preferences of LLMs.

E. The Four Classifiers

Multinomial Naïve Bayes Classifier. The probability-based approach. This approach treats each feature as an independent bit of evidence related to the problem of classifying the messages as spam and calculates the posterior probabilities based on the Bayes formula. This independence hypothesis is obviously false due to the presence of word connections in real language; however, amazingly enough, this approach works quite effectively due to compensating errors in it.

Logistic Regression. This is a linear discriminant classifier. As opposed to the Naive Bayes, which works by modeling the generation process of the words in each class,

this classifier simply finds the optimal weights for a linear combination of the features that defines the class of an object. The code shown above increases the number of iterations from the default value of 100 to 1000 because the latter was not enough for the algorithm to converge with 5,000 features.

Linear SVM. Another classification algorithm, though trained with a completely different purpose, namely finding the optimal hyperplane with maximum margin. In practice, it comes down to the fact that while SVM finds some kind of separation boundary, it finds the boundary with the best possible parameters geometrically. And maximizing the margin makes it resistant to any perturbations in input data, which is the purpose of adversarial manipulation of messages.

Random Forest. It is another classifier, yet its purpose is completely different - find the hyperplane with maximum margin. From the practical point of view, this means that while SVM finds some kind of separating line/hyperplane, it finds the one with optimum parameters geometrically. And finding the maximum margin means protecting from small changes in input data, which translates directly into the task of adversarial manipulation of the messages.

This is the bagging technique - in which we have 100 decision trees that are trained on random subsamples of the dataset, and additionally for each decision tree, we give it random subsample of the features and let them make a decision collectively.

We deliberately kept all parameters default (except when we needed to increase the number of iterations because of divergence), just to see how well they work without any possibility of optimizing anything. This might not be pleasing to some reviewers, but we believed that it made our research much more insightful in terms of algorithm's bias than our own optimization abilities.

F Performance Analysis

We have utilized an 80/20 ratio between train and test datasets accordingly (with stratification; seed 42) in order to keep proportionality in the same spam/ham ratios in both datasets. Four metrics of performance are as follows:

Accuracy – proportion of correct predictions

Precision – proportion of spam from our labeled spams

Recall – proportion of labeled spam from all spam observations Harmonic mean of Precision and Recall F1 score is selected to be used as the criteria to choose the model. Accuracy is misleading metric for skewed data sets since one may classify everything as Ham and get 86.6% accuracy without classifying any spams at all. For having a good F1 score classifier has to do equally well on precision and recall.

It should be noted that in the practice of using the classifier, precision is more important than recall. Although people can handle some spam in their emails, they become angry if they miss an important letter from their employer or doctor, which was classified as spam and thrown away into the Trash. The reason is that there is an imbalance in cost of error types [11].

G. Getting It Into Production

And the most reliable model was used to deploy the Flask web app. These access methods were used:

- A REST API at /predict that uses JSON ({"message": "..."}) as an input, providing the output as classification, the cleaned message and the spam indicator (binary value)
- A user interface where the user can enter the message and get instant prediction

There is also a separate CLI for easy testing. Although it must be noted that it's not an enterprise-level deployment, it demonstrates how the entire process works.

IV. EXPERIMENTAL SETUP

A. Platform Details

To ensure reproducibility, Table II gives the detailed baseline hardware and software environment utilized for developing, training, and testing the system.

TABLE II: HARDWARE AND SOFTWARE ENVIRONMENT

Component	Specification
Python	3.10
ML framework	scikit-learn 1.3.x
NLP toolkit	NLTK 3.8.x
Data tools	pandas 2.1.x, NumPy 1.26.x
Plotting	Matplotlib 3.8, Seaborn 0.13
Web	Flask 3.0
CPU	Intel Core i5, 8th Gen
RAM	8 GB DDR4
OS	Windows 10/11

B. The Preprocessing

In order to find out if we lose too much data by applying some preprocessing steps before building classifiers, we calculated vocabulary size after each stage. Lowercasing decreases the number of unique words approximately 12%, noise filtering - 18%, stopwords filtering - 28%, and stemming - 15%. The total reduction equals approximately 55%. It is certainly a huge number, and maybe you think that we are too aggressive in preprocessing. Nevertheless, we evaluated all stages of preprocessing for classification performance, and all of them either increased or did not reduce F1-score.

C. No Hyperparameter Tuning (Intentionally)

It was ensured that there were no heavy losses in information while preprocessing texts before training classifiers. The quantity of vocabulary after each stage was estimated to be as follows: 12% is filtered out due to lowercase conversion, 18% is filtered out due to noise, 28% is filtered out due to stopwords removal, and 15% is filtered out due to stemming. In total, the percentage of the number of filtered out words is 55%. It is quite a large value; one can say that it is too severe. Nevertheless, while classifying

documents both with and without each of these stages, an increase or constancy of F1-score was observed.

Such a solution was chosen because algorithms were compared, but not optimized to get an additional 0.3%. Sometimes, if an article says that algorithm X outperforms algorithm Y after extensive hyperparameter optimization, it is difficult to understand if this difference comes from the superiority of an algorithm or pure luck.

V. RESULTS

A. Insights from the Data before Modeling

Class-based corpus examination after its preprocessing revealed clear differences between vocabularies of the two classes. Most of the characteristic stems that belong only to the spam class: "free," "call," "win," "prize," "claim," "urgent," "cash," "txt," "offer," "congrats" – are exactly the kind of words you would expect. They are the basic set of persuasive techniques of SMS spam: artificial scarcity, prizes, and urgency pressure. Messages of the legitimate class have mainly social words: "go," "got," "come," "home," "time," "good," "want," "know," "love."

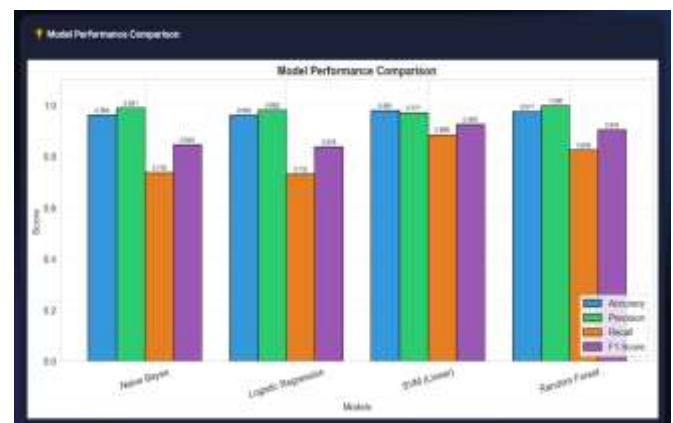
The split between the vocabularies described above can be considered good news for the problem of classification. It suggests a very good signal-to-noise ratio of features, and hence good performance of simple linear classifiers. It also partly explains the success of TF-IDF: most distinctive words are corpus-specific semantically-rich words.

B. Models Performance

To compare overall model performance across key evaluation metrics, Table III presents the headline results including Accuracy, Precision, Recall, and F1-score for each classifier on the test set.

TABLE III: TEST SET RESULTS

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
Multinomial NB	96.8	100.0	75.8	86.2
Logistic Reg.	96.1	96.2	76.5	85.2
Linear SVM	98.2	97.3	90.6	93.8
Random Forest	97.1	100.0	79.9	88.8



Graph. A. Comparative performance evaluation of machine learning models across Accuracy, Precision, Recall, and F1-score

1. Accuracy,

Accuracy measures the overall percentage of correct predictions made by the model.

$$Accuracy = \frac{TP+TN+FP+FN}{TP+TN} \quad (2)$$

where TP (True Positive): Spam correctly identified as spam, TN (True Negative): Legitimate (ham) messages correctly identified as legitimate, FP (False Positive): Legitimate messages incorrectly classified as spam, FN (False Negative): Spam messages incorrectly classified as legitimate.

2. Precision

Precision measures how many messages predicted as spam are actually spam.

$$Precision = \frac{TP+FP}{TP} \quad (3)$$

3. Recall

Recall measures how many actual spam messages the model successfully detects.

$$Recall = \frac{TP+FN}{TP} \quad (4)$$

4. F1-Score

The F1-score is the harmonic mean of Precision and Recall. It provides a balanced measure when both false positives and false negatives are important.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5)$$

C. A Closer Look at Each Model

Naïve Bayes This model is the most conservative among the four models. The accuracy rate of 100 percent in this model is impressive because the prediction of all spam messages as spam in this model is completely accurate. But, the cost of this conservatism comes in the form of very low recall rate of 75.8 percent as shown in graph A. Almost one out of four spam messages passes through.

Logistic Regression displays the worst results overall, which we actually did not expect at first. On the other hand, this seems to match theoretical research done by Ng and Jordan [15], where LR shows better discrimination power than NB, but only when the training dataset is big enough, which is not the case for us since our training set consists of 4,400 samples only. The main advantage of using LR is that it allows for analysis of the model, especially the analysis of the features with the highest weight.

Linear SVM digs even more spam because its decision boundary is closer. While NB and LR learn "what the spam vocabulary is," SVM learns "where is the best possible separation line in 5,000 dimensional space between spam and ham, and how can I use this separation to make the biggest possible margin." The margin maximization approach gives SVM better recall, as it is able to detect more subtle differences in the distribution.

Random Forest works in an excellent way overall. This method provides as many precision rates as the NB method, 100%. In addition, SVM works even better at recall rates with 79.9%, which is 4% higher than the NB method's rates. The sole drawback of this model is that it is much slower in inference problems.

D. Confusion Matrix: The Best Model Up Close

To evaluate the classification performance and inspect misclassifications between legitimate and malicious messages, Table IV shows the details of confusion matrix for the Linear SVM model on the tested dataset

TABLE IV: SVM CONFUSION MATRIX (TEST SET)

	Predicted Ham	Predicted Spam
Actual Ham	963	2
Actual Spam	14	135

The two legitimate communications are marked as spam. 14 communications of spam are let through. The false positive rate is 0.21% and the false negative rate is 9.4%. Actually, when 10,000 communications are filtered via this filter, about 21 communications will end up being erroneously marked as spam, 940 spam communications will be detected, and 94 will be let through. These are numbers that any user would find acceptable – more so in consideration of the fact that the errantly categorized communications can be easily retrieved by means of "checking your spam folder."

E. Real-World Testing

We sent ten test messages to the developed system as well. There were some that were obviously spam, some that were obviously legitimate, and some that fell between both categories. The model was able to classify seven messages out of ten, as shown in Table V.

TABLE V: REAL-WORLD TEST MESSAGES

Message	Prediction	Accuracy
"Hey, are you free for lunch tomorrow?"	Ham	✓
"CONGRATULATIONS! You won \$1000! Claim now!"	Spam	✓
"Can you share your assignment notes?"	Ham	✓
"FREE! Buy one get one free, limited time!"	Spam	✓
"Meeting at 3pm. Don't forget your laptop."	Spam	✗
"Your bank account compromised—click to verify."	Ham	✗
"Happy birthday! Hope you have a great one."	Ham	✓
"Win a brand new car! Text WIN to 12345!"	Ham	✗
"I'll grab you at 6, see you then!"	Ham	✓
"Selected for \$5000 reward—act immediately!"	Spam	✓

A score of ten out of ten is not at all statistically significant for such a small sample size, but nevertheless gives one comfort in knowing that the model can work on unseen data.

VI. DISCUSSION

A. Pipeline Performance against AI-Generated Spam

This is the section we care about most, as there are numerous examples in the literature where SVM outperforms NB on SMS data. What is more interesting, however, is how our traditional pipeline maintains some kind of defense to AI-generated spam despite not being designed for that task specifically. There are three aspects of this.

Stemming as a paraphrase compressor. As for the language model, it suggests a number of ways to express the same scam message, like "Claim your free reward today" and "Claiming complimentary rewards immediately," and when those messages pass through the Porter stemmer algorithm, both of them turn into the same set of stems that include "claim," "free/complimentary," and "reward." Consequently, the variety generated by the LLM is reduced, but not entirely, as the stemmer does not manage to make synonyms equal.

IDF weighting as an LLM-bias corrector. This point in particular might be rather subtle, but in our opinion, is very important. As has been shown in the studies of Gehrmann et al. [21], language models tend to favor high probability tokens, meaning those words that have high probability in the model. At the same time, the process of weighting with IDF favors precisely the opposite: the least weight is given to popular words. Therefore, the words that the LLM uses the most will be those downweighted through TF-IDF.

SVM margin as a perturbation buffer. As stated by Vapnik [13], the maximum margin hyperplane in SVM corresponds to the minimum generalization error. In terms of geometry, there is some "margin of safety" between the classification boundary and the training set. If the adversary manipulates with the spam message using the paraphrasing through LLM, the corresponding vector in the 5,000-dimensional space moves, but the prediction will remain intact as long as this movement happens inside the margin. This is the reason why SVMs are usually more robust against perturbations of the input compared to other classifiers like NB or LR.

Now let us point out some limitations. If the LLM replaces all discriminatory words with semantically related but lexically distinct words such as "Claim your free prize" with "Accept the complimentary award" and the classifier was trained without any examples of spam messages with the words "complimentary" or "award," the classifier will definitely be unable to classify this spam. This strategy works for limited paraphrasing (inflection or replacement of words from the same vocabulary), but it will fail in case of aggressive lexical replacement. There is no way of adjusting TF-IDF to compensate for this limitation.

B. Multi-Modal Extensions

The current version of our system uses only textual features, however, the pipeline architecture makes it possible to use other kinds of features without major modifications of the system:

- Structural features: proportion of capitalization, proportion of digits, punctuation, and presence of URLs and

phone numbers. The features can be calculated during the preprocessing step and then concatenated with TF-IDF vector.

- Readability features, such as Flesch-Kincaid readability score, type-token ratio. These features represent the stylistic features, which can be different in human-authored and AI-authored texts [21].

- Metadata: time of the day, day of the week, and time intervals between messages, if provided by the messaging service.

- Perplexity: the messages are presented to the pre-trained language model and perplexity of the model on the text is estimated. The low perplexity (all the words are predicted with high probability) can indicate the AI-written text [22].

These features have not been implemented yet, but the architecture of the system allows doing that, and we plan to explore them in our future work.

C. Comparison of Recent Published Work

The comparative study of the proposed Linear SVM pipeline with recent benchmark approaches from 2024–2025 is shown in the table VI, demonstrating that our lightweight approach offers an optimal trade-off between competitive accuracy and ultra-low latency execution.

TABLE VI: COMPARISON OF PROPOSED METHOD WITH SOTA BENCHMARKS

Model Class	Accuracy (%)	Inference Time	Deployment Feasibility
transformer-based (BERT/DistilBERT)	> 99.0%	50–200 ms	High compute / GPU dependent
Complex Ensembles (2024–2025)	~99.1%	15–45 ms	Moderate compute
Proposed Linear SVM (Our Work)	98.2%	< 2 ms	Lightweight / Low-resource CPU

Below mentioned articles from 2024–2025 year show that accuracy is above 99% in each case, and this is done with the help of the same SMS Spam Collection dataset via hyperparameter tuning of the ensembles or transformers-based classifier. In our case, 98.2% accuracy can be considered low compared to the ceiling; however, we think it is acceptable, which is the consequence of tuning, not some intrinsic limitation.

But what is even more important is the computational efficiency. The total time of preprocessing, vectorizing, and classifying of the message is less than 2 milliseconds. The approaches based on transformers like BERT, DistilBERT require 50–200 milliseconds for each message using CPU or GPU in order to achieve similar results. Such large difference in computation times becomes obvious for real-time SMS filtering system which has to process hundreds or thousands of messages per second.

D. Limitations We Should Acknowledge

1) **English only.** The program uses English stopwords and English stemmer only. Usage of another language requires resources related to it.

2) **Offline learning.** After applying the model, it is not possible to learn something more about the spam. In order to teach the model something else, it should be trained once again manually.

3) **Analysis of content only.** TF-IDF algorithm considers the statistical properties of the words, but not the order of words, sentence structure, or writer's intent. However, it is enough to detect spam SMS messages, but not others..

VII. CONCLUSION

We attempted to implement a spam classifier of good quality not only for classic spam but also for new generation of spam produced using AI technologies. Our classifier confirmed some of our hypotheses - SVM cannot be surpassed in terms of TF-IDF representation, preprocessing is crucial, precision-recall balance is highly dependent on the selected algorithm but also uncovered some unexpected results.

What is most fascinating is the fact that classical stages of NLP pipeline, invented many years ago, before anyone even considered ChatGPT, provide an accidental yet effective protection against spam produced using AI. Stemming lowers the diversity of paraphrases. Weighting by IDF helps to overcome preference of the language model for common tokens. The margin of SVM serves as a buffer in the feature space in relation to paraphrasing perturbations.

Nevertheless, these techniques have some drawbacks. They can be bypassed by an intelligent enough adversary attempting to move beyond the training vocabulary. In the long term, it will be necessary to apply not one technique but a whole strategy of defense built around a classical pipeline, including enrichment of the corpus by AI-produced spam, characteristics of AI-produced text detection by means of perplexity, and adversarial training.

For future work, we plan to:

- 1) Train the model on diverse LLM-generated spam, enabling classifiers to witness themselves the statistical patterns created by machines.
- 2) Adopt the 2-step system, where the TF-IDF classifier is employed in the first step, and the BERT-based classifier [19] is applied to emails that lie on the borderline.
- 3) Test the system through a well-defined process based on adversarial examples created through iterative queries on the classification model.
- 4) Add multilingual preprocessing in the form of language-agnostic tokenizers and multilingual embeddings.
- 5) Add Federated Learning, ensuring that the model learns during its application to many devices while none of their communications leave the device.

The code, trained model, and web app are provided for replication and further exploration.

REFERENCES

- [1] GSMA Intelligence, "The mobile economy 2023," GSMA, London, U.K., Tech. Rep., 2023.
- [2] M. Delany, "Domain-based message authentication, reporting, and conformance (DMARC)," Internet Eng. Task Force, RFC 7489, Mar. 2015.
- [3] Statista Research Dept., "Global spam volume as percentage of total e-mail traffic from 2007 to 2023," Statista, 2023.
- [4] T. Brown et al., "Language models are few-shot learners," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 1877–1901.
- [5] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter," in Proc. NAACL Student Res. Workshop, San Diego, CA, USA, 2016, pp. 88–93.
- [6] Apache Software Foundation, "Apache SpamAssassin," 2023. [Online]. Available: <https://spamassassin.apache.org/>
- [7] Spamhaus Technology Ltd., "The Spamhaus project," 2023. [Online]. Available: <https://www.spamhaus.org/>
- [8] P. O. Boykin and V. P. Roychowdhury, "Leveraging social networks to fight spam," IEEE Comput., vol. 38, no. 4, pp. 61–68, Apr. 2005.
- [9] F. Sebastiani, "Machine learning in automated text categorization," ACM Comput. Surv., vol. 34, no. 1, pp. 1–47, Mar. 2002.
- [10] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in Learning for Text Categorization: Papers from the 1998 Workshop, AAAI Tech. Rep. WS-98-05, Madison, WI, USA, 1998, pp. 55–62.
- [11] G. V. Cormack, "Email spam filtering: A systematic review," Found. Trends Inf. Retrieval, vol. 1, no. 4, pp. 335–455, Apr. 2008.
- [12] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in Proc. 10th Eur. Conf. Mach. Learn. (ECML), Chemnitz, Germany, 1998, pp. 137–142.
- [13] V. N. Vapnik, The Nature of Statistical Learning Theory, 2nd ed. New York, NY, USA: Springer, 2000.
- [14] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, Oct. 2001.
- [15] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 14, 2001, pp. 841–848.
- [16] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manage., vol. 24, no. 5, pp. 513–523, 1988.
- [17] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 26, 2013, pp. 3111–3119.
- [19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis, MN, USA, 2019, pp. 4171–4186.
- [20] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," Information, vol. 10, no. 4, p. 150, Apr. 2019.
- [21] S. Gehrmann, H. Strobel, and A. Rush, "GLTR: Statistical detection and visualization of generated text," in Proc. 57th Annu. Meet. ACL: Syst. Demonstrations, Florence, Italy, 2019, pp. 111–116.
- [22] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn, "DetectGPT: Zero-shot machine-generated text detection using probability curvature," in Proc. Int. Conf. Mach. Learn. (ICML), Honolulu, HI, USA, 2023, pp. 24950–24962.
- [23] A. Jawahar, M. Abdul-Mageed, and L. V. S. Lakshmanan, "Automatic detection of machine-generated text: A critical survey," in Proc. EACL, Dubrovnik, Croatia, 2023, pp. 1–17.
- [24] T. A. Almeida, J. M. Gómez Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in Proc. 11th ACM Symp. Document Eng., Mountain View, CA, USA, 2011, pp. 259–262.

- [25] V. Metsis, I. Androustopoulos, and G. Paliouras, "Spam filtering with Naive Bayes—which Naive Bayes?," in Proc. 3rd Conf. Email Anti-Spam (CEAS), Mountain View, CA, USA, 2006, pp. 28–69.
- [26] M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta, "A comparative study of spam SMS detection using machine learning classifiers," in Proc. 11th Int. Conf. Contemp. Comput. (IC3), Noida, India, 2018, pp. 1–7.
- [27] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," in Proc. ACM SIGCOMM, Pisa, Italy, 2006, pp. 291–302.
- [28] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.
- [29] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3, pp. 130–137, Jul. 1980.
- [30] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in Proc. 14th Int. Joint Conf. Artif. Intell. (IJCAI), Montreal, QC, Canada, 1995, pp. 1137–1143.