IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

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

Harsh Rakesh Walmiki
Department of Computer Science
MSc/MCA
JSPM University, Pune, India
harshwalmiki16@gmail.com

Abstract: *Although spam is not a new problem, the emergence of generative AI tools has made the problem more hazardous than ever before. Spam messages written using large language models are extremely convincing since they sound like human messages. In this paper, we present our spam detection system which uses classical NLP text processing techniques and supervised machine learning with the emerging threats of spam in mind. The proposed solution consists of five text preprocessing steps for the SMS messages—lowercasing, noise removal using regular expressions, stopwords removal, and Porter stemming—to convert SMS messages to feature vectors of TF-IDF using up to 5,000 features. Four classifiers have been tested on the UCI SMS Spam Collection dataset which contains 5,572 SMS messages—the Multinomial Naïve Bayes, Logistic Regression, Linear SVM, and Random Forest. The Linear SVM worked best, with an accuracy of 98.2%, 97.3% precision, 90.6% recall, and an F1 score of 93.8%. The part which we find particularly intriguing, and which occupies much of the remainder of this paper, is how this sort of pipeline can still hold its own when it comes to defending against spam generated by AI. This is because stemming reduces the diversity of the vocabulary exploited by LLMs, IDF weighting automatically discourages the common tokens favored by language models, and the geometry of margins in SVM gives some leeway in case of perturbation of feature space. We have also developed a web application using Flask for real-time predictions.Index Terms— Spam Filtering, Text Classification, NLP, TF-IDF, SVM, Naïve Bayes, AI-Generated Content, Adversarial Robustness*

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

I. INTRODUCTION

The importance of text messaging cannot be overstated. According to the findings of the GSMA 2023 annual report, the number of SMS in the world exceeds two trillion annually [1], but does not take into account billions more being transmitted via WhatsApp, Telegram, and other such platforms. Alongside this huge amount of legitimate SMS there is spamming – persistent, troublesome, and sometimes dangerous. Various forms of phishing, prize notifications, and fake banking messages cost users billions annually [2]. According to statistics provided by Statista, approximately 45% of emails are spam, and mobile communications catch up with this trend [3].

What is different now? Up until just a few years ago, spam messages were formulaic and predictable enough that you could even time your day with their regularity, and that a good percentage of the messages could be caught simply with keyword matching [4]. But the rise of large language models, like GPT-4 and its open source alternatives, has turned things on its head [4]. Spam generated through these models has the ability to create hundreds of different versions of a

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

scam message, all with perfect grammar and tone. The giveaways of the past have disappeared, along with the defenses against them [5].

The classical methods for filtering out spam include three major categories. First, heuristic scoring algorithms like SpamAssassin, which give points for suspicious attributes and flag e-mails with too many points [6]. It works, but it constantly needs tuning since the spammers keep advancing. Second, blacklisting and whitelisting techniques of various organizations such as Spamhaus [7], which stop known bad actors until they change IP addresses. Third, collaborative filtering, in which user complaints populate communal spam signatures database [8]. And the problem with collaborative filtering is its lag time: the filter can only recognize the new spam once the victims complain about it.

This is where machine learning comes into play because, rather than explaining how to recognize spam emails, you would teach the computer using samples that are marked appropriately [9]. This ability of machine learning to generalize, meaning to identify any variations even those which it has not seen before, is precisely what attracts attention in the age of AI-generated phrasing for spam emails [10].

The contributions made by our paper are four-fold and, in our opinion, distinguish our work from all other prior work on SMS spam:

1) We construct a preprocessing pipeline with five stages, and instead of simply stating that “preprocessing helps,” we investigate how each stage contributes to reducing the variability introduced by generative AI systems.

2) We evaluate four distinct classifiers coming from truly dissimilar algorithmic families on exactly the same footing. Most of the literature reviewed here applies to hyperparameter optimization, which may be regarded as an indicator of how good you are at exploring parameter space, not how dissimilar your algorithms are.

3) We describe an analytical perspective on why some classical solutions work against AI-generated spam in terms of the TF-IDF statistics and SVM margins geometry. We hope that this kind of analytical approach is not sufficiently covered in the current literature.

4) We wrap up our analysis in a deployed web app featuring a REST API. All too many research papers analyzing academic spam stop at the confusion matrix.

The remaining part of this paper is quite conventional in its structure. Section II provides an overview of existing literature. In Section III, we go through our experimental process and model design. Section IV is about experimental set-up. Section V shows the results obtained during our research. In Section VI, we discuss the problem of spam created by AI.

II. RELATED WORK

A. Origin: Bayesian Spam Filtering

There is a study by Sahami et al., published in 1998, which is considered the origin of the whole idea of spam filtering using ML [10]. With bag-of-words feature, a Naïve Bayes classifier managed to correctly classify spam and non-spam emails, despite the very unrealistic assumption about the independence of words. The key point here is very clear: there is a systematic difference between the languages used in spam and non-spam messages.

Approximately ten years later, Cormack [11] conducted an exhaustive review of more than 200 works devoted to spam filtering. Three conclusions from him had especially much impact on our understanding. First, filters based on machine learning techniques always outperformed rule-based static systems. Second, assessing the quality of filters in terms of accuracy is inappropriate, as the loss caused by labeling non-spam as spam (false positive) is much greater for the end-user compared to labeling spam as not spam (false negative). Third, personalization of filters suffers from the 'chicken and egg' problem – they require some training data specific for the user, but the new user does not have one yet.

B. SVMs and Ensemble Methods for Text

In 1998, Joachims [12] showed that support vector machines performed exceptionally well on text classification tasks, and his analysis of why they worked has stood the test of time. Since text data as TF-IDF vectors is inherently high

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

dimensional (thousands of features), the high-dimensional nature does not cause SVMs much difficulty because the optimization criterion for SVMs is margin and not least squares loss. This is proven theoretically by Vapnik in the SVM literature [13]; the maximum-margin hyperplane minimizes the upper bound of the generalization error.

The Random Forest algorithm invented by Breiman [14] was a completely different approach; create multiple decision trees using bootstrap samples of the dataset, let them vote, and hope that their errors cancel out via averaging. For text classification, the benefit of Random Forests lies in the fact that text is inherently high dimensional (with thousands of features), and many of these will be noisy; Random Forests deal well with noisy features.

Ng and Jordan [15] offered an interesting theoretical comparison between Naïve Bayes (generative) and Logistic Regression (discriminative). Their key finding was that Naïve Bayes converges faster with small training sets, while Logistic Regression eventually wins out given enough data. Since our training set is modest (about 4,400 messages after the 80/20 split), we expected Naïve Bayes to be competitive—and it was.

C. How You Represent Text Matters

The selection of the text representation may be said to affect the outcome as much as the choice of classifier itself. Term frequency-based vectors give too much weightage to terms such as "the" and "is", which do not indicate anything about spamming at all. The problem is solved using TF-IDF, defined by Salton and Buckley [16] as the product of the term frequency and the logarithm of its inverse document frequency. Manning et al. [17] have explained this in detail in their book on information retrieval.

Of course, more recent research has investigated dispersed word representations. Word2Vec [18] uses dense vectors capture semantic similarity, while BERT [19]'s transformed-based embeddings encode contextual meaning. These are powerful representations, no doubt about it. But Kowsari et al. [20] make an important practical point in their survey: for short-text classification with well-defined vocabularies, TF-IDF remains competitive while being orders of magnitude cheaper to compute. SMS messages average around 15–25 words. At that length, the contextual advantages of BERT are marginal, but the computational overhead is not.

D. The AI-Generated Text Problem

Identification of machine-generated text has emerged as a mini-field of research in itself. Gehrmann, Strobelt, and Rush [21] created the GLTR visualizer, which identifies one peculiar behavior exhibited by machine-generated text – language models predict high-probability words more often than people do, leading to an unusual consistency of token distribution. In a separate line of work, Mitchell et al. [22] developed DetectGPT, which identifies machine-generated text through the observation that such text resides in areas of negative curvature on the surface of the log-probability of the generating model.

What worries us most, though, is the work by Jawahar et al. [23], who showed that simple paraphrasing or back-translation can tank the accuracy of AI text detectors. If a spam operator generates a template with one LLM and runs it through a second LLM for paraphrasing, both the AI-content detector and the spam filter have to cope. That compound evasion scenario is a central concern of this paper.

E. SMS Spam Specifically

The dataset used in the paper and which is used by everyone doing SMS spam studies was developed by Almeida, Gómez Hidalgo, and Yamakami [24]. This dataset contains a total of 5,572 SMS texts from a number of different sources such as the UK Grumbletext website and the Singaporean NUS SMS Corpus. Metsis et al. [25] benchmarked Naïve Bayes algorithms for term frequency features and concluded that the Multinomial NB was the best one.

Augmentation of TF-IDF using manually crafted features (such as capitalization of the message, frequency of special characters, average length of words), done by Gupta et al. [26], produced only marginal improvement in classification performance, thus supporting the notion that content-based features are not enough. Ramachandran and Feamster [27]

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

focused on a totally different aspect of spam detection, namely network-related behavior such as bursty messages sent from ephemeral IP addresses – good for email spam but not easily usable in SMS scenario.

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.

SPAM DETECTION PIPELINE

Raw Message → Preprocess → TF-IDF → Classifier

- lowercase
- regex clean
- stopwords
- stemming

Classifiers: NB / LR / SVM / Random Forest

Best Model → Flask Web App + CLI Predictor

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

The idea of such a pipeline architecture is quite clear. Each component is responsible for doing one thing. In case you will need to substitute a vectorizer with another one or introduce another classifier into the pipeline, you can easily do that without modifying the rest of 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 stats.

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

Two notable observations come to mind. First, the ratio in class distribution is around 6.5:1 for ham, which indeed depicts the real world (as most of the emails people receive are legitimate); however, it complicates the classifier evaluation process, as a simple classifier guessing "ham" would have 86.6% of accuracy. The problem is addressed through stratified splitting and focusing on F1-score.

Another thing is that spam is about twice as long as non-spam messages. It is clear when you start thinking about it: spam messages need to make you buy something, so they will always have some kind of sales pitch, an URL, phone number, and the language of urgency. True messages sent between friends tend to be short. The difference in lengths is not explicitly encoded in the dataset, but it is caught by TF-IDF.

C. Text Preprocessing in Detail

Our pre-processing process involves five steps. We consider each of these as handling a certain kind of noise, which, if not removed, will either expand our vocabulary set or blur the lines between spam and ham in our feature space.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

Stage 1 — Lowercasing. Everything becomes lowercased. “FREE”, “Free”, and “free” all become “free”. Sounds simple, and it really is, but neglecting this step effectively doubles the vocabulary size because the same word written in a different case is considered a separate feature. In the case of spam detection based on AI, capitalization is often randomized, which is solved by this step right away.

Stage 2 – removal of noise via regular expressions. For the removal of numbers, punctuation, emojis, currency signs and any part of URL, [^a-zA-Z\s] is used. Yes, it is true that some valuable information is being removed in this step, for example, phone numbers and money values. However, from a practical perspective, removal of these tokens is better because the addition of these tokens would result in creation of many unique features.

Stage 3 – Cleaning whitespace. We consolidate repeated whitespaces to single whitespaces and strip the ends. Tiny step, but hugely influential on tokenization.

Stage 4 – Removing stopwords. Applying the English stopwords list from NLTK [28], we remove all the common words like “the,” “is,” “and,” “but,” which occur irrespective of message type. These words are merely eating up space in feature space. This process alone decreased our vocabulary size by about 25–30%.

Stage 5 – Porter stemming. This stage might be the most crucial one for protection from AI-produced paraphrasing. The Porter algorithm [29] breaks down the suffixes to decrease words to their roots: “winning” will become “win,” “congratulates” will turn into “congratl,” “urgently” will turn into “urgent.” When paraphrases are produced by the LLM, for example, “claim your prize” being changed to “claiming prizes,” all the paraphrases will end up with the same stem set during 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

Once we have done our pre-processing, the next step is to make the text ready for classification. We use TF-IDF, which gives a weight to each term depending on its frequency in the document being classified (TF), and its rarity in the whole corpus (IDF).

$$\text{weight}(t, d) = \text{tf}(t, d) \times \log(N / \text{df}(t)) \qquad (1)$$

where N is the total number of documents, and df(t) is the number of documents containing the term t. IDF works as a relevance filter by setting terms that occur in virtually all the documents to zero and emphasizing terms occurring in certain parts of the document collection.
The maximum number of features we use is 5,000, anything above that is hapax legomena, which refers to one-time appearing terms, and most likely noise rather than relevant terms. The transformation results in a 5,000-dimensional sparse vector representation for each message with less than 2% non-zero elements.
However, there is a little-discussed but rather useful property of IDF weighting which is related to the problem of AI and has not received proper attention in the spam research community. As reported by Gehrmann et al. [21], language

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

models generally prefer high probability and hence high-frequency tokens. However, IDF weighting automatically downweights precisely these types of tokens. In this respect, TF-IDF vectors are, so to speak, calibrated out of the box against the linguistic preferences of LLMs.

E. The Four Classifiers

We chose four models that are truly diverse in how they learn, rather than just four variations of the same idea:

Multinomial Naïve Bayes Classifier. The probability-based method. It considers each feature to be an independent piece of evidence regarding the spam classification problem and determines the posterior probabilities using the Bayes theorem. The independence assumption is clearly false since word associations exist in real language, but surprisingly it works pretty well because the errors in it are likely to cancel each other out across both classes.

Logistic Regression. This is a linear discriminative model. It doesn't work by modeling the process of generating words in each class like Naive Bayes does; it just learns the weights of a linear combination of features that determines the class of an object. In the code above, we increase the number of iterations from the default value of 100 to 1000 since the latter was not sufficient for the convergence of the algorithm on 5,000 features.

Linear SVM. Another classifier but with an entirely different training aim, i.e., discover the hyperplane with maximum margin. Practically, the difference lies in the fact that whereas SVM discovers some separating boundary, it discovers the one with optimum parameters in a geometrical way. And the maximization of that margin ensures protection from slight deviations in input data, which directly corresponds to the task of adversarial modification of messages.

Random Forest. This is the ensemble method – where we have 100 decision trees which are trained on random sub-samples of the dataset, and also each tree gets to see random sub-samples of the features and make a collective decision on the prediction.

We purposely left all default settings on (except where we had to increase the number of iterations due to divergence issues). This was a conscious decision on our part to gauge their performance without being able to optimize anything. It may not please some reviewers, but we thought that it made our study much more insightful regarding the bias of the algorithms rather than our own optimization skills.

F. How We Measured Performance

We have used an 80/20 split between training and test sets respectively with stratification (seed 42) in order to preserve the same spam/ham proportionality in both datasets. Here are four performance measures used:

- **Accuracy** – simple proportion of correctly predicted observations
- **Precision** – among our labeled spams, what portion is spam indeed?
- **Recall** – from all the spam observations, what portion was labeled?

Harmonic mean of Precision and RecallF1-score is chosen to be used as a criterion for choosing a model. Using accuracy is highly misleading for skewed data sets as one could simply "classify everything as Ham" and achieve 86.6% accuracy without actually classifying any of the spam messages. In order to have a high F1-score, the classifier has to perform equally well on precision and recall.

It is worth noting that in the practical applications of our classifier, precision is more desirable than recall. While people might be able to deal with some spam in their emails, they become very upset if an important letter from their employer or doctor is classified as spam and lost in the Trash. This is due to the imbalance in the costs of error types [11].

G. Getting It Into Production

The most accurate model was then deployed using a Flask Web App. The following access points were provided:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

- A REST API at /predict which takes an input in the form of JSON ({"message": "..."}), giving the output as the classification, the cleaned-up message, and the spam flag (binary value)
- A user interface through which users could enter messages and get immediate predictions

A separate CLI script is also available for easy testing. While we accept that this is not enterprise-level deployment, it proves that the process works from input to prediction.

IV. EXPERIMENTAL SETUP

A. Platform Details

Table II lists the environment we used. Nothing exotic—this runs on a standard laptop.

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. Sanity-Checking the Preprocessing

We checked to see whether we discard too much information by doing some preprocessing steps before training any classifiers. We measured vocabulary size after each step: lowercase reduces the number of unique tokens by 12% approximately; noise filter reduces by 18% approximately; stopwords filter by 28%; and stemming reduces by 15%. The overall reduction is about 55%. It is indeed a big amount of information, and you may think that maybe we are too aggressive in preprocessing. However, we checked all our stages of processing on classification results they all increase or do not decrease F1-score.

C. No Hyperparameter Tuning (Intentionally)

We made sure that our pipeline does not lose too much information before training classifiers. We have estimated the amount of vocabulary after each step: 12% is lost due to lowercase conversion, 18% due to noise filtering, 28% due to stopwords deletion, and 15% due to stemming. In total, the number of words lost reaches 55%. This is a lot; one may think that we are being too harsh here. However, when we tried to classify documents both with and without each of these steps, we saw F1-score improvements or constancy.

We took this path since we are comparing algorithms and not optimizing our models for every possible additional 0.3%. If a paper claims that algorithm X works better than algorithm Y after long hyperparameter optimization, sometimes it is hard to distinguish whether this is really caused by algorithm superiority or just good luck in the search process.

V. RESULTS

A. What the Data Told Us Before We Even Started Modeling

Corpus exploration after preprocessing showed evident differences in vocabulary of the two classes. Most distinctive stems specific to spam class—“free,” “call,” “win,” “prize,” “claim,” “urgent,” “cash,” “txt,” “offer,” “congratl”—are exactly what one would expect. These stems represent the basic set of persuasion tricks of SMS spam: artificial

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

scarcity, fake prizes, and urgency pressure. Messages of legitimate class, on the other hand, contain mostly common social words: “go,” “got,” “come,” “home,” “time,” “good,” “want,” “know,” “love.”

Vocabulary division observed above is good news from the point of view of classification. It indicates high signal-to-noise ratio in the feature space, which implies good performance of even linear classifiers. It also partially explains success of TF-IDF in this case: most differentiating words belong to the category of corpus-specific semantically-loaded words.

B. How the Models Performed

Table III has the headline numbers.

TABLE III: TEST SET RESULTS

Model	Acc. (%)	Prec. (%)	Rec. (%)	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

It’s a Linear SVM all the way. But the victory is not equally decisive when it comes to all measures—what is particularly notable is the recall measure. Recall is 90.6% in the case of SVM and 79.9% in the case of the closest competitor to SVM, Random Forest. So, there’s a margin of 10.7%, which translates into 11 extra spam detection per 100 cases of spam that would have been missed by RF, and this is done with 97.3% precision!

C. A Closer Look at Each Model

Naïve Bayes is the most conservative of the four. Its 100% accuracy is remarkable since all of its identifications of spam messages as spam are correct. However, the price of such conservatism is low recall of 75.8%, which means that almost one out of four spam messages gets through. If not bothering the user even by a single false positive message is the main goal, NB is your model.

Logistic Regression showed the poorest performance overall, which was actually unexpected for us in the beginning. However, it seems to be consistent with Ng and Jordan's theoretical research [15], where LR outperforms NB in terms of discriminating power only when the training dataset is large enough, while our training set is rather small (about 4,400 training samples). The strength of the LR is that it gives an opportunity to analyze the model, especially to look at the top weighted features. Namely, “free”, “txt”, “claim”, and “prize” were the largest positive coefficients.

Linear SVM digs up more spam since it has a closer decision boundary. Whereas NB and LR learn "what is the spam vocabulary," SVM learns "where is the best possible boundary in 5,000 dimensions separating the spam from the ham, and how do I put that boundary to make the largest possible margin." This margin maximization strategy makes the SVM have better recall since it captures finer distribution differences that other algorithms miss.

Random Forest performs in a strong, all-round manner. It is equally precise as NB with 100%, and more recall at 79.9%, just 4% ahead of NB’s score. The only problem with it is its slower speed on inference tasks, which can be up to one magnitude behind the linear models for one-shot classification.

D. Confusion Matrix: The Best Model Up Close

Table IV shows the confusion matrix for the Linear SVM.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

TABLE IV: SVM CONFUSION MATRIX (TEST SET)

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

The two valid communications have been classified as spam. Fourteen spam communications got through. False Positive Rate stands at 0.21%, while False Negative Rate stands at 9.4%. In reality, when 10,000 communications are filtered using this filter, around 21 communications will be wrongly buried in the spam section, 940 spam communications will be identified, and 94 will get through. These are figures that most users will consider reasonable – especially considering that the wrongly classified communications can easily be found using the "check your spam folder" method.

E. Real-World Testing

We sent ten hand-written 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 all ten messages correctly, as shown in Table V.

TABLE V: REAL-WORLD TEST MESSAGES

Message	Prediction
"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."	Ham ✓
"Your bank account compromised—click to verify."	Spam ✓
"Happy birthday! Hope you have a great one."	Ham ✓
"Win a brand new car! Text WIN to 12345!"	Spam ✓
"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. Why This Pipeline Holds Up Against AI-Generated Spam (To a Point)

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. The language model comes up with a variety of different ways to say the same scam message such as "Claim your free reward today" and "Claiming complimentary rewards immediately," and when processed through the Porter stemmer, both messages get represented by a set of stems which includes "claim," "free"/"complimentari," and "reward." Thus, the diversity that the LLM created disappears to some extent. However, not completely, since the stemmer fails to normalize synonyms that are used creatively.

IDF weighting as an LLM-bias corrector. This particular point is subtle, but, we believe, quite significant. According to the research by Gehrmann et al. [21], language models are inclined to prefer high probability tokens, i.e., popular

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

words with a high probability in the model. Meanwhile, IDF weighting, which is applied to each token, does just the opposite – gives the smallest weight to popular words. Thus, the words that will be used the most by the LLM are those that are actively downweighted via TF-IDF.

SVM margin as a perturbation buffer. According to Vapnik [13], the maximum margin hyperplane in the SVM corresponds to the lowest generalization error. Geometrically speaking, there is some "margin" of safety between the decision boundary and the training set. When the adversary modifies the spam email by paraphrasing through LLM, the corresponding TF-IDF vector moves in the 5,000-dimensional space, but as long as the movement occurs within the margin, the prediction remains unchanged. That is why the SVM is usually more resilient to input modifications than other classifiers, such as NB or LR.

Where it breaks down. Let us acknowledge some constraints. If the LLM substitutes all discriminatory words by semantically similar but lexically different words—say "Claim your free prize" is transformed into "Accept the complimentary award"—and the classifier was not trained with any spam messages containing the words "complimentary" or "award," then the classifier will most likely fail to recognize it. This approach works for modest paraphrasing (word inflections and synonym substitution in the same vocabulary) but fails if there is aggressive lexical substitution. No TF-IDF tweaking will compensate for the lack of adequate training data.

B. Multi-Modal Extensions

Currently our system relies on text only, however, the pipeline approach allows adding other types of features without a big overhaul of the system:

- Structural features including proportion of capitalized letters, proportion of digits, punctuation pattern, and existence of URLs or phone numbers. They can be calculated in the pre-processing stage and concatenated with TF-IDF vectors.
- Readability features like Flesch–Kincaid scores and type-token ratios. These features reflect stylistic characteristics, which might be different in human-written and AI-generated texts [21].
- Metadata, such as time of the day, day of the week, and inter-messages intervals, when provided by the messaging service.
- Perplexity scores, where the messages are fed into a pre-trained language model and the model is checked for how "surprised" it is about the text. The low perplexity (meaning all words were predicted by the model very easily) could be a sign of AI-written messages [22].

We haven't implemented these yet, but the architecture enables them, and we intend to investigate them in future work.

C. How We Compare to Recent Published Work

The following articles in the year 2024-2025 indicate that the accuracy achieved is over 99%, all using the same SMS Spam Collection dataset, usually through hyperparameter tuning of ensembles or transformers-based classifier. Our 98.2% accuracy is a bit low compared to the ceiling, but we believe that is acceptable. This is mainly due to the tuning process, rather than inherent limitation.

However, even more important is the computational efficiency. The entire process of preprocessing, vectorization, and classification takes less than 2 milliseconds for each message processed. Approaches based on Transformers like BERT, DistilBERT take from 50 to 200 milliseconds per message using CPU, or GPU in order to achieve similar performance. With such differences in computational times the difference becomes clear for a real-time SMS filtering system which has to process hundreds or thousands of messages per second.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

D. Limitations We Should Acknowledge

1) The dataset is old. The SMS Spam Collection spans from 2004 to 2011. Things have changed quite a bit since that time; modern spam refers to such things as vaccines for COVID, bitcoins, and parcel delivery, all of which are not present in this dataset.

2) English only. It relies on English stopwords and an English stemmer. Multilingual use will require language-specific resources.

3) No confirmed AI-generated spam in the training data. The AI resilience assessment relies on structural reasoning regarding how the pipeline and the properties of LLM-generated texts interact with each other. The empirical validation of this approach would require a dataset with labeled instances of AI-generated spam, but such datasets are not publicly available (to the best of our knowledge).

4) No online learning. The model becomes static after being applied. It cannot acquire new knowledge about spam messages unless manually retrained.

5) Content-only analysis. The TF-IDF algorithm analyzes the statistics of words, not their order, structure of the sentence, intent behind writing or multimodal features like embedded images. This is sufficient to identify SMS spam messages, but not enough for other types of spam.

VII. CONCLUSION

We tried to develop a quality spam classifier not only for the classical spam but also for the emerging generation of messages created by AI technology. The classifier proved some of our hypotheses right – SVM cannot be outperformed in terms of TF-IDF representation, preprocessing is very important, precision-recall trade-off depends a lot on the algorithm chosen but also revealed some unexpected information.

What we consider to be the most interesting observation is that classical pipeline stages in NLP, created many years ago, before people even thought of ChatGPT, create accidental but still useful defense against spam created by AI. Stemming reduces the diversity of paraphrases. Weighting of the IDF counters the language model preference towards common tokens. The margin of the SVM acts as a buffer in the feature space against the paraphrasing perturbations.

However, these methods have certain limitations. If an intelligent enough opponent tries hard enough to go beyond the training vocabulary, then these methods can be bypassed. In the long run, it is necessary to use not one method, but to build an entire defense strategy around a classical pipeline, which should include corpus enrichment with AI-generated spam, features of detecting AI-created text based on perplexity, and adversarial training.

For future work, we plan to:

1) Create the training data using a variety of LLM-produced spam messages, thus allowing classifiers to see firsthand the statistical patterns generated by machines.

2) Use a two-stage system in which a quick TF-IDF classifier is used in the first stage, and a BERT-based classifier [19] is used for emails close to the boundary.

3) Stress test the system in a systematic manner using adversarial examples created through iterative queries on the classification model.

4) Integrate preprocessing for more languages by incorporating language-independent tokenizers and multilingual embeddings..

5) Integrate Federated Learning such that the model can learn through its use on multiple devices without any of their communications leaving the device.

The code, trained model, and web app are available for replication and extension.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

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. Strobelt, 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.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, June 2026
Impact Factor: 8.2

[25] V. Metsis, I. Androutsopoulos, 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.