Mobile Phone Text Message Spam Detection

Kaien Xia (A0295763M) [1] **Cheng Fu (A0295995Y)** [1] **Jiaxin Zhu (A0296819H)** [1] **Jinwen Yang (A0295883H)** [1]

Abstract

SMS-based spam has emerged as a critical cyber-security threat. Traditional rule-based filters are becoming inadequate in detecting these sophisticated attacks. This project proposes a robust SMS spam classification system that uses traditional machine learning and deep learning approaches, enhanced through generative AI-based data augmentation and zero-shot intent classification. The results underscore the potential of integrating generative AI and pre-trained language models to build high-precision spam detection systems capable of adapting to emerging threats.

1. Introduction

1.1. Problem Background

Spam messages—particularly phishing via SMS, or smishing—are becoming increasingly prevalent. In the third quarter of 2024, smishing incidents surged by 22%, and in 2023, text messages accounted for over 28% of all phishing attempts. Alarmingly, nearly 40% of mobile threats now involve credential phishing through SMS, often disguised as fake alerts or scam messages (Husain, 2025, para. 2).

Traditional rule-based filters have proven inadequate and ineffective in the face of these rapidly evolving tactics. This trend underscores the urgent need for an adaptive, high-precision spam detection system to counter emerging threats and safeguard sensitive information effectively.

1.2. Value Proposition and Business Context

An intelligent spam classification system enhances mobile security by accurately filtering spam and minimizing false positives. For consumers, it protects against fraud and reduces unwanted interruptions. In terms of mobile providers and enterprises, the enterprise spam filter market is experiencing significant growth. This is driven by the increasing demand for integrated security solutions, which could reduce costs related to customer support and inter-operator charges, improve operational efficiency, ensure regulatory compliance, and build customer trust.

According to the statement of Zion Market Research (2024), ”in 2023, the global market was valued at approximately USD 138.65 billion and is projected to reach around USD 487.75 billion by 2032, reflecting a compound annual growth rate (CAGR) of 15% over the forecast period” (para. 1).

Thus, given the evolving nature of spam, a dynamic classification system is essential to safeguard sensitive information against evolving cyber threats.

1.3. Objective

The primary objective of this project is to develop an efficient SMS spam classification system that enhances the accuracy of text message classification and reduces the number of spam messages received by users, based on a simulation using a Kaggle dataset. To evaluate the effectiveness of the classification model, the following key performance indicators will be employed:

Classification Accuracy This metric measures the ratio of correctly predicted messages (both spam and non-spam) to the total number of messages evaluated, providing an overall assessment of the model's performance.

Precision Precision assesses the proportion of messages identified as spam that are truly spam. A high precision indicates a low rate of false positives, ensuring that legitimate messages are not incorrectly classified as spam.

Recall Recall evaluates the proportion of actual spam messages that are correctly identified by the model. A high recall signifies that the model effectively captures the majority of spam messages, minimizing false negatives.

F1 Score The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is particularly useful when there is an uneven class distribution, as it considers both false positives and false negatives.

AUC (Area Under the Receiver Operating Characteristic Curve) AUC evaluates how well the model can distinguish between spam and non-spam messages, independent of any specific threshold. A higher AUC indicates better model performance in differentiating between the two classes.

By applying these evaluation metrics, this project aims to ensure that the developed SMS spam classification system is accurate and reliable, effectively reducing the incidence of spam messages and enhancing user experience.

2. Dataset Description

2.1. Data Source

The dataset employed is from Kaggle, and comprises a total of 5,574 SMS messages. The dataset contains two primary columns: the Class column, which serves as the binary target label, and the Message column, which consists of the corresponding SMS text.

Each message is labeled as either ”spam” (i.e., unsolicited promotional or fraudulent messages) or ”ham” (i.e., legitimate, non-spam content).

An example of a ham message is: “Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got a...” while a typical spam message is: “WINNER!! As a valued network customer you have been selected to receive a £900 prize reward!”

Given its clean format, moderate size, and well-defined labels, this dataset is well-suited for classification tasks in natural language processing. In addition to binary classification, it provides a good foundation for extended tasks such as generative text modeling and multi-class spam type categorization.

2.2. Exploratory Data Analysis

2.2.1. Class Distribution

To better understand the dataset and inform subsequent modeling decisions, we performed an in-depth exploratory data analysis. The dataset comprises a total of 5,159 SMS messages, with 4,518 labeled as ham (non-spam) and 641 labeled as spam, resulting in a class distribution of approximately 87.6% ham and 12.4% spam. This significant class imbalance underscores the need for careful handling during model training to mitigate bias toward the dominant ham class.

A bar chart of the class distribution clearly reveals the imbalance. The majority class (ham) dominates the dataset, with spam messages forming only a minority. Given the practical implications of missing spam (e.g., fraud or scams), metrics such as recall and F1 score will be emphasized over simple accuracy in evaluation.

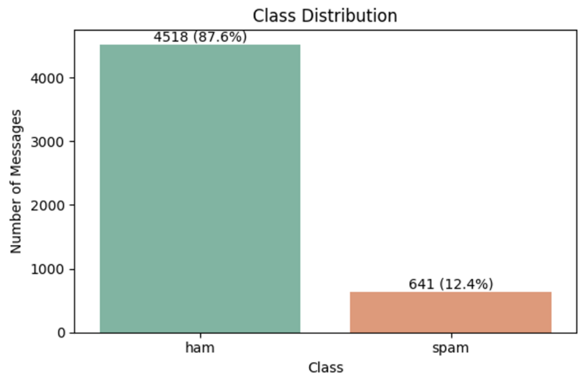

Figure 1. Class distribution of SMS messages.

2.2.2. Message Length Analysis

In addition to class distribution, we analyzed the textual properties of the messages. Specifically, we computed the character length of each SMS and examined the distribution of message lengths across the two classes. The resulting histogram indicates that ham messages tend to be shorter and more concentrated around a moderate length, while spam messages exhibit a broader and more right-skewed distribution, with some messages significantly longer than others. This reflects the nature of spam messages, which often include additional promotional content, contact information, or suspicious URLs, leading to greater length variability.

Furthermore, we observed that the maximum length of any message in the dataset is 910 characters. Understanding the range of message lengths provides useful guidance for feature engineering and model design, particularly in deep learning applications where input sequences may need to be padded or truncated to a fixed length.

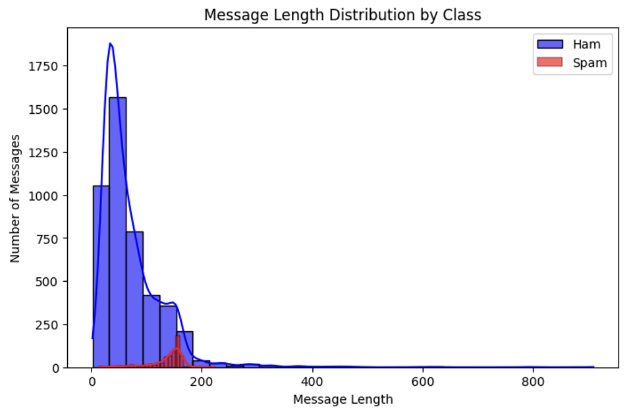

Figure 2. Distribution of message lengths across classes. Ham messages tend to be shorter on average, whereas spam messages exhibit greater variability and a heavier right tail, indicating a tendency to be longer.

2.2.3. Word Cloud Visualization

To qualitatively examine the lexical characteristics of spam and non-spam (ham) SMS messages, word clouds were generated for each category based on raw term frequencies. This visualization method provides an intuitive means of identifying dominant vocabulary in each class, where the font size of each word is proportional to its frequency of occurrence within the corpus.

The word cloud corresponding to spam messages reveals a strong prevalence of commercial, promotional, and action-inducing language. Words such as “FREE,” “call,” “txt,” “claim,” “prize,” and “now” appear prominently, reflecting a lexical strategy centered on urgency and reward-oriented persuasion. Financially suggestive terms, including “150p,” “win,” “guaranteed,” and “offer,” frequently appear, indicating an attempt to attract attention through the promise of monetary benefits or prizes. In addition, the frequent appearance of verbs such as “receive,” “reply,” “send,” and “collect” demonstrates the directive nature of spam content, which is generally structured to provoke immediate user engagement. The inclusion of formal or official-sounding phrases such as “PO Box,” “customer service,” and “landline” further suggests that many spam messages are crafted to simulate legitimacy or authority, thereby increasing their persuasive power.

In contrast, the word cloud for ham messages illustrates a substantially different lexical profile. Frequently occurring terms such as “ok,” “u,” “will,” “got,” “come,” “know,” and “love” point to a more casual, conversational tone that is characteristic of personal communication. The high prevalence of first- and second-person pronouns (e.g., “I,” “you,” “me,” “we”) is indicative of direct interpersonal interaction. Verbs such as “want,” “go,” “see,” “think,” and “tell” reflect everyday communication themes related to planning, expressing emotions, or exchanging information. Furthermore, the presence of emotionally expressive terms including “hope,” “good,” “sorry,” and “home” highlights the affective and social dimensions of ham messages, which tend to convey concern, affection, or mundane updates between individuals.

The comparative analysis of the two word clouds underscores the semantic divergence between spam and ham messages. Spam messages are dominated by persuasive, transactional, and commercially charged vocabulary aimed at eliciting action, often leveraging psychological triggers such as urgency and reward. In contrast, ham messages exhibit linguistic patterns associated with routine social interaction, emotional expression, and informal dialogue. These differences highlight the value of textual features in distinguishing between message types and provide a strong rationale for incorporating word frequency-based representations into downstream natural language processing and machine learning pipelines for SMS classification tasks.

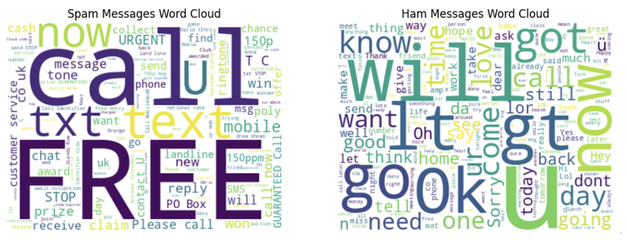

Figure 3. Word clouds of SMS messages. The left plot shows the most frequent words in spam messages, highlighting promotional terms like `FREE`, `call`, and `text`. The right plot illustrates common words in ham (non-spam) messages, such as `ok`, `u`, and `good`, reflecting more conversational language.

3. Machine Learning Methods

3.1. Traditional Machine Learning Models

3.1.1. Support Vector Machine (SVM)

Data Preprocessing Before model training, all SMS messages were preprocessed to enhance text consistency and reduce noise. Each message was first converted to lowercase, followed by the removal of URLs, digits, and punctuation using regular expressions. The cleaned text was then used for feature extraction.

To convert textual data into a machine-readable format, we employed the Term Frequency–Inverse Document Frequency (TF-IDF) vectorization technique, which captures both the importance of a word in a message and its relative rarity across the corpus. The TF-IDF vectorizer was configured with English stopword removal and a vocabulary size limited to the 5,000 most frequent terms, resulting in sparse, high-dimensional feature vectors suitable for linear classifiers.

The dataset was subsequently split into training and test subsets using an 80:20 ratio, stratified by class labels to preserve the original distribution between spam and ham messages. Labels were encoded using Scikit-learn’s LabelEncoder, mapping ’spam’ to 1 and ’ham’ to 0.

Data Augmentation via Generative AI To enhance data diversity and improve model generalization, particularly given the linguistic sparsity and class imbalance inherent in SMS spam datasets, we applied a paraphrase-based data augmentation strategy using generative artificial intelligence (GenAI). Specifically, we utilized the `ramsrigouthamg/t5_paraphraser`, a transformer-based sequence-to-sequence model fine-tuned on paraphrase generation, available through the Hugging Face Transformers library.

Each original message was reformulated using the T5 model to produce semantically equivalent paraphrases, which retained the original class labels. The paraphrasing process employed top-k and top-p sampling (k=120, p=0.95) to ensure diverse yet contextually faithful rewrites. Messages were processed in batches to improve computational efficiency. The resulting augmented dataset, which combines both original and paraphrased messages, effectively doubled the training size while preserving class balance.

This augmentation technique helps the SVM classifier become more robust to lexical and syntactic variability, a common challenge in spam detection due to intentional message obfuscation. Moreover, the use of GenAI aligns with modern trends in natural language processing, where large language models are leveraged to enrich training corpora, especially in domains where labeled data is limited or imbalanced.

Model Architecture and Setup We adopted the Support Vector Machine (SVM) classifier due to its robustness in high-dimensional and sparse feature spaces. Specifically, we used Scikit-learn's LinearSVC, a linear-kernel variant optimized for text data represented via TF-IDF. The SVM model constructs a hyperplane that maximizes the margin between classes, which is ideal for sparse input matrices like those derived from TF-IDF.

A grid search was performed over the regularization parameter $C \in \{0.1, 1, 10\}$ to improve performance and prevent overfitting. Additionally, to account for the significant class imbalance in the dataset (spam comprises only 12.4% of all messages), the `class_weight` parameter was set to 'balanced', which automatically adjusts weights inversely proportional to class frequencies.

Training Configuration and Evaluation Strategy Model performance was estimated using 5-fold stratified cross-validation to maintain class balance across splits and ensure robustness against data variance. The configuration that yielded the highest average F1 score across folds was selected and retrained on the full training set.

The final model was then evaluated on the held-out test set using multiple metrics, including accuracy, precision, recall, F1 score, and ROC AUC, as described in Section 1.3. Given the real-world impact of misclassifying spam messages, emphasis was placed on recall and F1 score for the spam class.

SVM's interpretability is enhanced by the fact that its decision function is determined primarily by support vectors—data points lying closest to the classification boundary. Furthermore, the linear kernel yields a weight vector where each coefficient corresponds to a specific TF-IDF term, offering valuable insights into which words most strongly influence spam classification decisions.

3.1.2. Random Forest

Random Forest is a widely used ensemble learning method that constructs a large number of decision trees during training and outputs the mode of their predictions for classification tasks (IBM, 2025). It is known for its robustness, ability to handle high-dimensional and sparse data, and resistance to overfitting. Each decision tree in the forest is trained on a bootstrapped sample of the data and considers a random subset of features when splitting nodes, which increases model diversity and generalization.

These characteristics make Random Forest particularly suitable for SMS spam detection. First, SMS messages, once vectorized using TF-IDF, result in sparse and high-dimensional feature representations. Tree-based models like Random Forest are not affected by feature sparsity and can effectively handle such input. Second, the ensemble nature of the model provides resilience against overfitting, which is especially valuable when dealing with imbalanced data where spam is underrepresented. Thirdly, Random Forest offers simplicity in data preprocessing, which reduces implementation complexity and speeds up experimentation (Sjarif et al., 2019).

Unlike deep learning models that require additional steps such as sequence encoding or token-level embeddings, Random Forest performs effectively with straightforward TF-IDF vectorization. This preprocessing pipeline is detailed in the following section.

Data Preprocessing We represent SMS messages using TF-IDF features, as previously described. This converts raw text into a sparse matrix format suitable for input into tree-based models like Random Forest. Unlike deep learning models that require dense and sequential data, Random Forest performs well on sparse matrices and does not require additional feature engineering or sequence modeling.

Model Architecture and Setup Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions to enhance classification performance and robustness. We use scikit-learn's RandomForestClassifier and perform grid search to identify the best hyperparameter configuration.

The search space includes the number of trees (`n_estimators`) set to 100 and 200, tree depth (`max_depth`) set to None, 10, or 20, and the minimum number of samples required to split a node (`min_samples_split`) set to 2 or 5. To handle class imbalance, the `class_weight` parameter is set to `'balanced'`, which automatically assigns higher weights to the minority class (spam) based on inverse class frequency.

Training Configuration and Evaluation Strategy The model is trained and validated using 5-fold stratified cross-validation, where each fold preserves the original spam-to-ham ratio. In each round, four folds are used for training and one for validation. The model achieving the highest average F1 score across folds is then retrained on the full training data. Its performance is finally assessed on the held-out test set using accuracy, precision, recall, F1 score, and ROC AUC(as described in Section 1.3).

By default, Random Forest uses Gini impurity as its splitting criterion. This measure reflects the probability of misclassifying a randomly selected sample from a node. Lower Gini values indicate purer splits, with a minimum of 0 when all samples in a node belong to the same class, and a maximum of 0.5 when the classes are perfectly balanced.

3.2. Deep Learning Models

Deep learning models have emerged as powerful alternatives to traditional machine learning methods in text classification tasks such as spam detection. Unlike models that rely on predefined features, deep learning approaches automatically learn representations from data, allowing them to capture more complex and abstract patterns.

In this section, we introduce two representative neural architectures: a Long Short-Term Memory (LSTM) network and a Transformer-based BERT model. While both belong to the deep learning paradigm, they require distinct model architectures and training configurations, which we outline in the following subsections.

3.2.1. Recurrent Neural Network: LSTM

Long Short-Term Memory (LSTM) networks are a specialized form of recurrent neural networks (RNNs) designed to capture long-range dependencies in sequential data. Unlike traditional RNNs, which suffer from vanishing gradients and struggle to retain information across time steps, LSTMs incorporate gating mechanisms — including input, forget, and output gates — that regulate information flow through the network. These gates allow LSTMs to selectively remember or forget information, making them particularly effective for modeling structured sequences with temporal or contextual patterns (Al-Selwi et al., 2024).

In the context of SMS spam detection, LSTMs are a natural candidate for exploring deep learning solutions due to several strengths. First, spam messages often contain subtle clues embedded in word order, repetitive phrasing, or patterned structures that require a model capable of learning dependencies across positions — something LSTM is explicitly designed to do. Second, compared to more recent architectures like transformers, LSTM models are computationally lighter and easier to train on smaller datasets, offering a practical balance between expressiveness and efficiency. This makes them particularly well-suited for resource-constrained or time-sensitive applications, such as mobile spam filtering (Altunay & Albayrak, 2024).

Additionally, LSTMs provide a more structured approach to modeling feature interactions than traditional feedforward or convolutional neural networks, which may fail to capture sequential relevance. By incorporating memory units, LSTM networks can model nonlinear relationships across the TF-IDF input space, offering potential performance gains over simpler models without requiring raw text inputs or token embeddings (Singh, 2024).

Data Preprocessing The LSTM model uses the same TF-IDF representations as input. Since LSTMs expect dense tensor inputs in the format (batch_size, sequence_length, input_size), each 5000-dimensional TF-IDF vector is reshaped into a pseudo-sequence of length one to align with the LSTM's input format.

Model Architecture and Setup The architecture includes a two-layer LSTM network with 128 hidden units, chosen as a standard size that balances model capacity and computational efficiency. The model uses batch_first=True to process inputs in batch-major order. The final hidden state is passed through a fully connected layer that outputs logits for the two classes, which are spam and ham. This structure allows the model to capture nonlinear interactions between input features using recurrent connections.

Training Configuration and Evaluation Strategy We train the model using the Adam optimizer with a learning rate of 0.001, which is widely used for deep learning due to its adaptive learning capabilities. The batch size is set to 64, striking a balance between computational efficiency and convergence stability. The model is trained for 5 epochs, which we found sufficient to achieve convergence without overfitting. The loss function is cross-entropy loss, adjusted using class weights computed from inverse label frequencies to address the imbalance between spam and ham messages.

We apply 3-fold stratified cross-validation to ensure balanced label distributions across splits. In each fold, two-thirds of the data are used for training and one-third for validation. The model with the highest validation F1 score is re-trained on the full training set and evaluated on the test set using the same metrics as the Random Forest model.

3.2.2. Transformer-Based Model: BERT

To complement the traditional and sequential deep learning models explored in this project — namely SVM, Random

Forest, and LSTM — we additionally experimented with a transformer-based model, BERT (Bidirectional Encoder Representations from Transformers).

BERT is a pre-trained deep neural network architecture built on the transformer encoder, which uses multi-head self-attention to capture both local and global dependencies across all tokens in an input sequence. Unlike LSTM, which processes sequences left to right, BERT processes input bidirectionally, allowing it to learn richer contextual representations of words and phrases (Pontes, 2024).

This architecture is particularly effective for spam detection tasks, where malicious messages often rely on subtle patterns, ambiguous wording, or misleading tone to evade keyword-based or pattern-based filters. Compared to models like LSTM, BERT has the advantage of understanding semantic nuance across longer spans of text, and compared to SVM and Random Forest, it works directly with raw textual inputs without requiring manual feature engineering or TF-IDF vectorization (Raga & L, 2022).

Moreover, BERT is pre-trained on massive language corpora and then fine-tuned on downstream tasks, making it highly effective even in data-limited scenarios like SMS spam classification. This transfer learning capability enables it to generalize well, reducing the need for large-scale labeled training data. Its robustness, contextual understanding, and adaptability make BERT a powerful complement to our existing models (Nishad, 2024).

Given these properties, we hypothesized that BERT would outperform all previously tested models in terms of accuracy, recall, precision, F1 score, and AUC, which reflect overall classification stability. Its strengths in modeling semantic patterns and rare edge cases make it especially suitable for capturing diverse and evolving spam formats. The next section describes how we adapted SMS message data into the format required for BERT fine-tuning (Raga & L, 2022).

Data Preprocessing Unlike the previous models, BERT does not rely on TF-IDF representations. Instead, it consumes raw text directly. Each SMS message is tokenized using the pretrained bert-base-uncased tokenizer. Tokens are padded or truncated to a maximum length of 128.

This length was chosen as a practical trade-off based on the observed distribution of message lengths: as shown in Figure 2 above, the vast majority of both ham and spam messages fall well below 200 characters, with a significant concentration under 128. Setting the maximum input length to 128 thus captures most SMS messages in their entirety without truncation, while avoiding excessive padding and keeping computational cost manageable during training and inference. The tokenized text is then converted into two input tensors: token IDs, which represent the vocabulary indices, and attention masks, which indicate which tokens are actual content and which are padding. These tensors are passed into the model in a format suitable for transformer-based processing.

Model Architecture and Setup We fine-tune a pre-trained bert-base-uncased model whose architecture is built entirely on transformer blocks powered by multi-head self-attention. This mechanism allows BERT to capture contextual dependencies across all token positions in a message, making it more expressive than recurrent or bag-of-words models. Each message is encoded into a 768-dimensional pooled embedding—the standard hidden size of the BERT base model—which summarizes the semantic content of the entire input.

This embedding is then passed through a dropout layer with a rate of 0.3 before classification. The dropout rate of 0.3 means that 30% of the neurons are randomly "dropped" during each training pass, helping to prevent overfitting by discouraging the model from relying too heavily on any single feature. This is especially important when fine-tuning on relatively small datasets like SMS messages, where the risk of overfitting is higher. Finally, the output passes through a fully connected layer that maps the pooled embedding to two logits representing the spam and ham classes.

Strategy and Evaluation We adopt a similar training and evaluation pipeline as in the LSTM model, primarily using 3-fold stratified cross-validation and class-weighted cross-entropy loss to handle label imbalance. The model with the highest validation F1 score is retrained on the full training set and evaluated on the test set using the same performance metrics as in the previous models.

However, certain training configurations are specifically tailored for fine-tuning BERT. We use the AdamW optimizer, which is more suitable for transformer-based architectures due to its built-in weight decay regularization. The learning rate is set to 2e-5, a standard setting recommended by BERT literature for fine-tuning tasks, as higher values may lead to catastrophic forgetting of pretrained knowledge.

A smaller batch size of 16 is chosen to fit within GPU memory constraints, as BERT's parameter size and input sequence length (max length = 128) demand more memory than simpler models. The number of training epochs is set to 3, which is often sufficient for convergence when fine-tuning pretrained language models on smaller datasets.

3.3. Hybrid Approaches

3.3.1. Stacking Classifiers

To enhance classification robustness and achieve balanced performance across key evaluation metrics (as outlined in

Section 1.3), we implemented a stacking classifier — a two-layer ensemble framework designed to combine the strengths of multiple models. In the first layer, Random Forest (RF) and BERT serve as the base classifiers. Each model independently outputs the probability that a message is spam. These outputs are then passed to the second layer, where a meta learner — XGBoost — produces the final prediction.

The rationale for choosing RF and BERT as base models lies in their complementary predictive behaviors. Random Forest, a tree-based ensemble method, is particularly effective in handling sparse, high-dimensional TF-IDF vectors. It tends to be conservative, prioritizing precision by minimizing false positives — that is, reducing the chance of misclassifying legitimate messages as spam. This strength is evident in our results (see Table 1 and Table 2 in Section 4.1), where RF achieved the highest precision (0.9548) among all individual models.

In contrast, BERT is a deep contextual language model that processes raw text and captures nuanced semantic and syntactic relationships. Its recall-oriented nature enables it to detect subtle spam patterns that traditional models might miss, even at the cost of slightly lower precision. In our experiments, BERT outperformed all other models in terms of accuracy, recall, F1 score, and AUC (see Table 1 and Table 2 in Section 4.1), highlighting its ability to detect a wide range of spam messages, especially those with ambiguous or deceptive phrasing.

By combining RF's cautious filtering with BERT's deep semantic detection, the ensemble effectively reduces the weaknesses of both individual models, resulting in more well-rounded predictions. For example, the model could offer a better trade-off between precision and recall — improving the ability to correctly identify spam without disproportionately mislabeling legitimate messages.

To integrate the outputs of the base classifiers, we employed XGBoost as the meta learner. This choice was motivated by XGBoost's ability to model complex interactions between heterogeneous features (i.e., RF and BERT predictions), handle class imbalance — especially relevant in our dataset where spam makes up only 12.4% of the messages — and operate effectively on low-dimensional input spaces, such as the two predicted probabilities used here. Its built-in regularization further prevents overfitting, ensuring reliable performance on unseen data.

Ultimately, and as expected, the stacking classifier outperformed both RF and BERT individually by delivering more consistent performance across all five evaluation metrics — accuracy, precision, recall, F1 score, and AUC — thereby confirming its effectiveness in balancing performance trade-offs, as demonstrated in Section 4.1. This result underscores the suitability of the hybrid approach for real-world spam detection, where achieving the right balance between over-detection (mislabeling legitimate messages) and under-detection (failing to identify actual spam) is crucial for maintaining user trust and system reliability.

3.4. Intent Classification Using Zero-Shot Learning

To gain deeper insights into the underlying semantic intent of spam messages beyond binary classification, we incorporated an *intent recognition module* powered by zero-shot learning using a large pre-trained generative model. This approach allows the system to assign specific intent labels to messages (e.g., *promotion*, *phishing*, *scam*) without the need for intent-labeled training data, thereby facilitating more granular downstream applications such as auto-response generation or fine-grained spam analytics.

We employed the `facebook/bartlargemnli` model available through the Hugging Face Transformers pipeline for zero-shot classification. This model is built upon BART, a denoising autoencoder for pretraining sequence-to-sequence models, and fine-tuned on the MultiNLI dataset. By leveraging its capability for natural language inference (NLI), the model can determine whether a given message *entails* any of the provided candidate intent labels.

A curated set of candidate intents was defined based on common spam patterns observed in real-world datasets and regulatory reports. These included: *scam*, *phishing*, *promotion*, *subscription*, *service notification*, *malware*, *lottery*, *fake job offers*, *fake charity*, *adult content*, and *delivery/shipping notification*.

For each spam message, the model was queried with the full set of candidate labels. The label with the highest entailment score was selected as the predicted intent. The classification was performed at scale, processing over 1,200 spam messages, and yielded an intent distribution heavily dominated by *promotion* (41.6%), *service notification* (23.8%), and *adult content* (15.3%), among others.

This zero-shot intent classification pipeline highlights the practicality and effectiveness of foundation models in natural language understanding tasks where labeled data is scarce. The integration of generative AI models not only enhanced the interpretability of spam behavior but also enabled fine-grained content categorization without additional supervised training.

Representative examples from each intent class were extracted to validate semantic coherence and alignment with label definitions, demonstrating the model's capability to distinguish nuanced differences in message purpose. This enriched annotation lays a foundation for future extensions such as intent-specific response generation or adaptive spam countermeasures.

4. Model Evaluation and Results

4.1. Results Overview

We evaluated four models—SVM, Random Forest, LSTM, and BERT—on the test set using five key performance metrics: Accuracy, Precision, Recall, F1 Score, and AUC.

The two tables below summarize the results:

Table 1. Performance comparison: SVM vs Random Forest

Metric	SVM	RF
Accuracy	0.9729	**0.9734**
Precision	0.8876	**0.9548**
Recall	**0.8945**	0.8242
F1 Score	**0.8911**	0.8847
AUC	**0.9851**	0.9802

Table 2. Performance comparison cont.: LSTM, BERT, and Stacking Classifier

Metric	LSTM	BERT	Stacking (RF+BERT)
Accuracy	0.9680	0.9763	**0.9797**
Precision	0.8626	0.8876	**0.9421**
Recall	**0.8828**	0.9258	0.8906
F1 Score	0.8726	0.9063	**0.9157**
AUC	0.9820	0.9886	**0.9898**

4.2. Insights

Our study reveals several key insights regarding the comparative performance, complementary strengths, and practical implications of different modeling approaches applied to SMS spam detection.

First, the stacking classifier emerged as the most effective and balanced solution, outperforming all individual models across multiple evaluation metrics. It achieved the highest accuracy (0.9797), F1 score (0.9157), and AUC (0.9898). This performance was the result of integrating two distinct but complementary models—Random Forest and BERT—through a meta-learner based on XGBoost. The success of this hybrid approach underscores the value of ensemble learning not just as a performance booster, but as a strategic tool to mitigate the weaknesses of individual classifiers.

Specifically, Random Forest achieved the highest precision (0.9548), making it conservative and minimizing false positives. BERT, by contrast, delivered the highest recall (0.9258), capturing subtle spam signals but with a higher risk of false alarms. By combining these models, the stacking classifier achieved a desirable trade-off between recall and precision, which is especially critical in real-world spam detection where both false negatives and false positives carry significant costs.

Second, BERT consistently outperformed other models in terms of contextual understanding, confirming the power of transformer-based architectures in text classification tasks. Its bidirectional attention mechanism enabled it to capture long-range dependencies and nuanced semantics in SMS messages, many of which are deliberately crafted to evade simpler filters. Even without hand-crafted features like TF-IDF, BERT learned directly from raw text and generalized well to a wide range of spam patterns, including ambiguous or obfuscated messages.

Third, traditional machine learning models like SVM and Random Forest remain competitive under the right conditions. Despite their simplicity, these models performed strongly when supported by proper text preprocessing and feature engineering. For example, SVM achieved an F1 score of 0.8911 with TF-IDF features and class weighting. These models are also advantageous in environments where interpretability, low latency, and memory efficiency are prioritized, making them still relevant for production-grade spam filters, particularly in resource-constrained scenarios.

Fourth, generative AI contributed meaningfully to both model robustness and training efficiency. By employing a T5-based paraphraser for data augmentation, we synthetically expanded the training set with lexically diverse but semantically consistent variants of original messages. This improved model generalization, especially in handling varied writing styles or intentional spelling variations often used in spam.

Finally, the integration of a zero-shot intent classification module significantly improved system interpretability. Using BART-large-MNLI, we assigned high-level semantic intent labels (e.g., promotion, phishing, adult content) to spam messages without requiring any labeled intent data. This not only enhanced transparency, enabling analysts or businesses to understand why a message is considered spam, but also provided a pathway for future downstream applications such as content-aware filtering, targeted blocking, and even automated response systems.

In short, this project highlights that model integration, generative augmentation, and semantic analysis tools work best in combination, not in isolation. By leveraging each model's comparative advantage and aligning architecture choices with the specific structure of the data, we were able to design a spam detection system that is not only accurate but also extensible, interpretable, and adaptable to future challenges in mobile security.

4.3. Limitation

Despite promising results, the study faces several limitations.

First, the dataset used—while well-structured—is relatively small and homogeneous, consisting of English-language SMS messages from a single source. This restricts the model's generalizability to more diverse linguistic and contextual spam formats encountered in real-world deployments, such as multilingual messages or region-specific scams.

Second, although our models achieved strong offline performance, real-time deployment concerns were not addressed. The inference speed, memory requirements, and computational costs of models like BERT can be significant on mobile devices, limiting their practical use in low-latency or resource-constrained environments.

Third, while generative augmentation and zero-shot intent classification enhanced performance and interpretability, they also introduced potential risks. Paraphrased spam messages, if poorly generated, may reduce data quality, and zero-shot predictions depend heavily on the quality of intent labels and the pre-trained model's generalization ability.

Lastly, the evaluation focused solely on standard performance metrics without examining model explainability in detail. This lack of interpretability may hinder adoption in regulatory-sensitive domains or commercial applications requiring transparency.

5. Conclusion

This project demonstrates that a multi-model approach — incorporating traditional classifiers, deep learning, and generative AI — can substantially improve SMS spam detection performance. Among individual models, BERT delivered the highest recall, showcasing the advantages of contextual embeddings and transfer learning in capturing nuanced spam cues. However, our final stacking classifier, which combines BERT and Random Forest predictions via a meta learner, outperformed BERT in accuracy, precision, F1 score, and AUC. This confirms that intelligently combining models with complementary strengths leads to more balanced and reliable spam detection outcomes.

Additionally, our experiments highlight that well-optimized traditional classifiers such as SVM and Random Forest remain competitive, especially when coupled with effective preprocessing and class imbalance handling. The introduction of zero-shot intent classification further adds interpretability, enabling the system to recognize different types of spam without labeled intent data.

Despite these promising results, the study has limitations, notably the scope of the dataset and the absence of real-time deployment considerations. Future work should focus on evaluating models on multi-source, real-world data, extending ensemble strategies, and integrating interpretability tools such as SHAP to enhance model transparency and user trust. Furthermore, assessing models under operational constraints — including inference latency and memory usage — will be crucial for deployment in mobile environments. Overall, this research lays the groundwork for building intelligent, adaptable spam detection systems capable of evolving alongside real-world mobile security threats.

References

Al-Selwi, S. M., Hassan, M. F., Abdulkadir, S. J., Muneer, A., Sumiea, E. H., Alqushaibi, A., & Ragab, M. G. (2024). RNN-LSTM: From applications to modeling techniques and beyond—Systematic review. *Journal of King Saud University - Computer and Information Sciences, 36*(5), Article 102068. [Link]

Altunay, H. C., & Albayrak, Z. (2024). SMS Spam Detection System Based on Deep Learning Architectures for Turkish and English Messages. *Applied Sciences, 14*(24), 11804. [Link]

Husain, O. (2025, February 6). *99 Global Phishing Statistics & Industry Trends (2023–2025)*. Control D Blog. [Link]

IBM. (2025, April 17). Random Forest. What is random forest? [Link]

Kaggle. (2025). *Spam SMS Classification Using NLP* [Dataset CSV file]. [Link]

Nishad, N. (2024, October 16). Fine-tuning BERT: unlocking the power of pre-trained language models. *DEV Community.* [Link]

Pontes, D. (2024, August 30). What is BERT (Bidirectional Encoder Representations from Transformers)? *Zilliz Learn.* [Link]

Raga, S. S., & L, C. B. (2022, December 1). A BERT model for SMS and Twitter spam ham classification and comparative study of machine learning and deep learning technique. *IEEE Conference Publication — IEEE Xplore.* [Link]

Sjarif, N., Mohd Azmi, N., Chuprat, S., Sarkan, H., Yahya, Y., & Sam, S. (2019). SMS Spam Message Detection using Term Frequency-Inverse Document Frequency and Random Forest Algorithm. *Procedia Computer Science, 161*, 509–515. [Link]

Singh, D. (2024, November 15). RNNs and LSTMs: the backbone of sequence learning in deep learning. *Medium.* [Link]

Zion Market Research. (2024, September 12). *Enterprise Spam Filter Market Size & Share Report, Growth, Trends, 2032*. [Link]

A. Appendix

GitHub Link: `https://github.com/Jessymeowow/BT5153_Final_Project09`