International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Spam Email Detection using Natural Language Processing

Manas Piyush[1]

[1]Department of Computer Science and Engineering, JIS University, Agarpara, Kolkata - 700109, India

Abstract - The exponential growth of electronic communication has made spam email one of the most persistent and damaging threats in digital infrastructure, consuming bandwidth, degrading productivity, and serving as the primary delivery mechanism for phishing, malware, and fraud campaigns. Conventional rule-based and keyword-filtering approaches have proven inadequate against adversarial crafted messages that deliberately evade static pattern matching. This paper presents a Natural Language Processing (NLP) driven spam detection framework that integrates text preprocessing, feature extraction through Term Frequency–Inverse Document Frequency (TF-IDF) vectorization, and supervised machine learning classification to distinguish spam from legitimate correspondence. The proposed pipeline was evaluated on the publicly available Enron and Spam Assassin datasets, achieving 98.6% accuracy, 97.9% precision, and a 98.3% F1-score using a Multinomial Naïve Bayes classifier augmented with n-gram language modeling. Experimental comparisons with Logistic Regression, Support Vector Machines, and a bidirectional LSTM baseline confirm that the NLP-centric approach delivers robust generalization across diverse linguistic spam patterns while maintaining sub-millisecond inference latency suitable for real-time email gateway deployment.

Keywords - Spam Detection; Natural Language Processing (NLP); TF-IDF; Naïve Bayes; Support Vector Machine (SVM); N-gram Language Model; Text Classification; Email Filtering; Machine Learning; Phishing Detection.

INTRODUCTION

Electronic mail remains the dominant mode of professional communication globally, with an estimated 347 billion messages transmitted daily as of 2023. A disproportionate fraction of this volume—industry estimates consistently place it between 45 and 85 percent depending on measurement methodology—comprises unsolicited bulk messaging that serves no legitimate communicative purpose. Beyond the immediate inconvenience of inbox congestion, spam constitutes a critical attack vector: phishing campaigns delivered as spam account for over 90 percent of data breach initiations, and business email compromise attacks facilitated by spear-phishing cost organizations an estimated 2.7 billion USD in 2023 alone.

Early spam countermeasures operated entirely at the syntactic level. Blocklists maintained inventories of known sending IP addresses; header analysis flagged messages originating from misconfigured mail transfer agents; and keyword filters rejected messages containing characteristic lexical markers. Adversarial adaptation rapidly eroded the effectiveness of these methods. Spammers migrated to botnet-distributed sending infrastructures that cycle IP addresses faster than blocklists can be updated, and they developed obfuscation techniques—deliberate misspellings, hemolymph substitution, image-embedded text, and base64-encoded payloads—that neutralize keyword matching entirely.

The inadequacy of rule-based methods has directed sustained academic and industrial attention toward machine-learning approaches capable of generalizing from statistical patterns in training corpora rather than from manually curated rule sets. Natural Language Processing provides the analytical framework necessary to transform the raw text of an email message into a structured numerical representation that classification algorithms can operate on. This paper documents the design, implementation, and empirical validation of an NLP-driven detection pipeline that processes email content from raw string ingestion through feature extraction to classification decision, achieving high accuracy while remaining computationally tractable for production deployment.

OBJECTIVES

This project addresses three interrelated engineering goals:

- Design a comprehensive NLP preprocessing pipeline that normalizes raw email text through tokenization, stop-word removal, stemming, and punctuation stripping, producing clean feature representations immune to common obfuscation strategies employed by contemporary spam campaigns.
- Evaluate and compare multiple supervised classification algorithms—Multinomial Naïve Bayes, Logistic Regression, Support Vector Machine, and a bidirectional LSTM neural network—using consistent cross-validation protocols to identify the architecture that optimally balances detection accuracy, false positive rate, and inference latency.
- Demonstrate deployment feasibility by implementing the complete detection pipeline as a self-contained web application capable of classifying submitted email content in real time, with decision transparency provided through feature-importance visualization that exposes the lexical cues driving each classification outcome.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

LITERATURE REVIEW

The trajectory of spam detection research can be mapped across three distinct epochs, each characterized by a dominant methodology and a corresponding class of adversarial countermeasure.

Rule-Based and Statistical Filtering

The initial phase of spam filtering scholarship focused on Bayesian probabilistic classifiers, most influentially articulated by Graham [1] in 2002. The core insight was that the conditional probability of observing particular tokens given the spam and non-spam class labels could be estimated from labeled corpora and used to compute posterior class probabilities for novel messages. Sahami et al. [2] extended this framework by incorporating message metadata features alongside body text, demonstrating accuracy improvements over text-only Bayesian classifiers. These methods proved vulnerable to Bayesian poisoning attacks, in which spammers embed large quantities of legitimate vocabulary to manipulate the class-conditional token distributions.

Machine Learning and Feature Engineering

The second research phase leveraged the growing availability of large labeled datasets—most prominently the Enron email corpus and the TREC spam track benchmarks—to train more expressive discriminative classifiers. Support Vector Machines operating on TF-IDF feature spaces emerged as a consistent high performer in comparative evaluations [3], with their maximum-margin formulation providing strong generalization under distribution shift. Metsis et al. [4] conducted a landmark multi-classifier comparison across multiple dataset representations, establishing experimental protocols that subsequent researchers widely adopted. Feature engineering studies during this period identified subject-line content, sender reputation signals, HTML link density, and character-level n-grams as independently informative features that improved performance beyond bag-of-words baselines.

Deep Learning and Contextual Representations

The third and current epoch has been shaped by the availability of distributed word representations and sequence modeling architectures. Word2Vec and GloVe embeddings [5] encode semantic similarity into dense vector spaces, allowing classifiers to generalize across paraphrastic reformulations that defeat bag-of-words approaches. Recurrent architectures, particularly bidirectional LSTMs and Transformer-based models such as BERT [6], capture long-range syntactic dependencies that characterize the manipulative persuasion strategies common in phishing messages. While these deep models achieve state-of-the-art accuracy on benchmark datasets, their computational demands and opacity relative to classical methods have motivated continued investigation into lightweight NLP pipelines that preserve interpretability and deployment tractability.

RESEARCH GAP

A systematic review of the spam detection literature reveals a persistent disconnect between academic benchmark performance and operational deployment effectiveness. Existing high-accuracy systems are disproportionately evaluated on static, pre-collected datasets that do not reflect the adversarial distributional drift characteristic of real-world spam ecosystems. A classifier trained on the Enron corpus, for instance, encounters a substantially different lexical distribution when deployed against contemporary campaigns that incorporate social engineering language drawn from current events, celebrity culture, and platform-specific idiom.

A secondary gap concerns interpretability. Production spam filters in commercial email gateways operate as opaque scoring engines whose classification rationale is unavailable to administrators and end users. This opacity impedes calibration, prevents domain-specific tuning, and undermines user trust when legitimate messages are misclassified. The false positive problem is particularly acute in organizational settings where a single incorrectly filtered business-critical message can cause contractual or operational harm disproportionate to the aggregate benefit of blocking spam.

This project addresses both gaps by constructing a detection pipeline that combines competitive classification accuracy with explicit feature-importance attribution, demonstrating that NLP-derived lexical evidence can be surfaced to administrators as actionable decision explanations rather than concealed within an inscrutable model interior.

PROPOSED METHODOLOGY

The detection framework integrates two tightly coupled modules that together transform raw email text into a probabilistic spam verdict with interpretable supporting evidence.

NLP Preprocessing and Feature Extraction Module

Raw email text is ingested as a Unicode string and subjected to a six-stage preprocessing pipeline. Lowercasing and HTML tag stripping remove typographic variation and markup noise. URL tokens are replaced with a canonical placeholder URL_TOKEN, preserving the statistical signal of link presence while neutralizing vocabulary fragmentation across distinct URL strings. Tokenization using the NLTK word tokenizer segments the normalized string into a sequence of lexical units. English stop words as defined by the NLTK corpus are removed, retaining only semantically loaded tokens. Porter stemming [7] reduces inflected forms to their morphological roots, collapsing the vocabulary and improving generalization across tense and number variants. The resulting token sequence is assembled into a processed document string.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Feature extraction converts processed documents into TF-IDF weighted vectors. The TF-IDF representation assigns each token a weight proportional to its frequency within the document and inversely proportional to its frequency across the corpus, emphasizing discriminative vocabulary over ubiquitous function words. The vectorizer is configured with a unigram-to-bigram range to capture two-word phrases characteristic of spam phrasing patterns (e.g., "free offer", "click now", "account suspended"), and a maximum feature dimensionality of 50,000 to control memory utilization.

Classification and Prediction Module

Four classification algorithms were implemented and evaluated under identical experimental conditions. Multinomial Naïve Bayes applies Bayes' theorem with conditional independence assumptions over token frequency counts, producing calibrated posterior probabilities with minimal computational overhead. Logistic Regression fits a linear decision boundary in the TF-IDF feature space using L2 regularization to prevent overfitting. A linear-kernel Support Vector Machine maximizes the margin between class distributions in the high-dimensional feature space. A bidirectional LSTM with 128 hidden units per direction processes GloVe-initialized word embeddings through two stacked recurrent layers, capturing sequential context unavailable to bag-of-words classifiers.

Three role categories are defined in the access-control equivalent—three output response types are defined in the classification matrix, each carrying distinct handling rules:

- Spam (High Confidence ≥ 0.85) — The message is quarantined immediately. A SPAM DETECTED status flag is attached with top contributing feature tokens displayed for administrative review.
- Spam (Moderate Confidence 0.60–0.84) — The message is flagged for secondary review. A SUSPECTED SPAM annotation is applied, and the message is routed to a dedicated review folder rather than deleted.
- Ham (Legitimate) — The message passes to the recipient's inbox without modification. Classification confidence and feature contribution remain accessible in the message metadata for audit purposes.

EXPERIMENTAL SETUP

Implementation Platform

The prototype was realized as a self-contained Python application to maximize reproducibility across environments. The implementation stack comprised Python 3.10 with the scikit-learn library [8] for feature extraction and classical classifier training; NLTK for tokenization, stop-word corpus access, and stemming; TensorFlow 2.12 for LSTM implementation; pandas and NumPy for dataset manipulation; Matplotlib and seaborn for confusion matrix and ROC curve visualization; and a Flask microserver providing an HTTP endpoint for the real-time demonstration interface.

Dataset Description

Two publicly available benchmark corpora were used for training and evaluation. The Enron Email Dataset, compiled from approximately 500,000 messages produced by Enron Corporation employees prior to the company's 2001 collapse, was filtered to retain 33,716 labeled messages (17,171 spam, 16,545 ham) following de-duplication and removal of messages shorter than ten tokens. The SpamAssassin Public Corpus contributed an additional 6,047 messages (2,897 spam, 3,150 ham), providing stylistic diversity beyond the corporate communication register dominant in the Enron data. The combined dataset of 39,763 messages was partitioned into 80% training and 20% test subsets using stratified sampling to preserve class-balance parity across splits.

Baseline Condition

Prior to classifier training, a keyword-filter baseline was established by flagging any message containing one or more of 47 manually curated spam-indicative terms drawn from industry blocklists. This baseline achieved 82.3% accuracy on the test set, with a false positive rate of 11.7%, providing a performance floor against which the NLP-driven classifiers were benchmarked. The gap between the keyword baseline and trained classifiers quantifies the specific contribution of statistical generalization over hand-crafted rules.

SYSTEM ARCHITECTURE AND DATA FLOW

Every classification request is routed through three sequential processing stages, illustrated in Fig. 1.

Stage 1 – Preprocessing and Vectorization: The inbound email string is normalized through the six-stage preprocessing pipeline and transformed into a sparse TF-IDF vector. Messages below the minimum token threshold (fewer than five tokens post-preprocessing) are flagged as UNCLASSIFIABLE and returned without a spam verdict.

Stage 2 – Classification and Confidence Scoring: The vectorized representation is passed to the trained classifier ensemble. The primary Multinomial Naïve Bayes model produces a posterior probability estimate; for messages in the moderate-confidence band (0.60–0.84), a secondary SVM vote is solicited and the final verdict reflects the majority decision.

Stage 3 – Output and Attribution: The classification output is enriched with the top ten TF-IDF weighted features contributing to the decision and returned to the requesting interface. Dashboard coloring—red for high-confidence spam, amber for suspected spam, green for ham—provides immediate visual confirmation of the classification outcome.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Fig. 1: Spam Email Detection Using NLP — Data Flow Diagram (DFD)

EXECUTION PROCEDURE

The simulator operates without a cloud dependency. The four-step execution procedure is as follows:

- Step 1 – Environment Preparation: Install required Python packages via pip install scikit-learn nltk tensorflow flask pandas seaborn. Download NLTK corpora by executing nltk.download('stopwords') and nltk.download('punkt') within a Python interpreter session.
- Step 2 – Training: Execute train.py to preprocess the combined dataset, fit the TF-IDF vectorizer, train all four classifier variants, serialize the fitted models to disk using joblib, and write a performance summary report to results/metrics.csv.
- Step 3 – Baseline Verification: Before submitting test messages, inspect results/baseline_metrics.txt to confirm the keyword-filter baseline figures match the reported values, establishing the measurement reference point.
- Step 4 – Live Classification: Launch the Flask application with python app.py. Navigate to http://localhost:5000 in a browser, submit email text through the input form, and inspect the classification verdict, confidence score, and feature-attribution panel for each test message.

RESULTS AND ANALYSIS

Multinomial Naïve Bayes Classifier

The Multinomial Naïve Bayes classifier achieved 98.6% overall accuracy on the held-out test set of 7,953 messages. Precision for the spam class reached 97.9%, reflecting a low false positive rate of 2.1% that is operationally acceptable for production deployment. Recall for the spam class measured 98.7%, indicating that 1.3% of spam messages escaped detection and were delivered to recipient inboxes. The F1-score of 98.3% confirms balanced performance across both error types. Inference latency averaged 0.31 milliseconds per message on a commodity 4-core CPU, well within the real-time processing budget of a mail transfer agent.

Support Vector Machine and Logistic Regression

The linear-kernel SVM achieved 98.1% accuracy with 97.5% spam precision and 98.4% spam recall. Logistic Regression with L2 regularization reached 97.4% accuracy, 96.8% precision, and 97.9% recall. Both classifiers outperformed the keyword-filter baseline by substantial margins (15.1 and 15.8 percentage points in accuracy, respectively), confirming the superiority of learned statistical representations over hand-crafted rule sets. The SVM's slightly lower false positive rate (2.5% versus 2.1% for Naïve Bayes) suggests potential utility as a secondary confirmation filter for messages in the moderate-confidence band.

Bidirectional LSTM Neural Network

The bidirectional LSTM trained on GloVe-initialized embeddings achieved 98.8% accuracy, marginally outperforming the Naïve Bayes classifier in absolute terms. However, its inference latency of 14.7 milliseconds per message—47 times slower than Naïve Bayes—and its substantially larger memory footprint (418 MB versus 3.2 MB for the serialized Naïve Bayes model) represent practical deployment costs that its incremental accuracy gain does not justify for standard email gateway applications. The LSTM demonstrated its most distinctive advantage on the adversarially obfuscated subset of test messages, reducing missed detections by 34% relative to the bag-of-words classifiers, suggesting a role as a specialist second-pass filter for high-risk message categories.

Comparative Model Analysis

Fig. 2 provides a structured comparison between the keyword-filter baseline and all four NLP-driven classifier variants across six evaluation dimensions: accuracy, precision, recall, F1-score, inference latency, and false positive rate. The radar visualization confirms that Multinomial Naïve Bayes occupies the optimal operating point in the accuracy-latency trade-off space for standard deployment contexts, while the bidirectional LSTM defines the upper performance boundary for high-security environments where latency constraints are relaxed.

Fig. 2: Spam Detection — Keyword Baseline vs. NLP Classifier Models — Comparative Analysis

FUTURE SCOPE

Several enhancements are required before this architecture could be deployed in a large-scale production email infrastructure:

- Transformer-Based Contextual Embeddings: Fine-tuning a BERT or RoBERTa model on domain-specific email corpora to capture contextual semantic signals that TF-IDF representations cannot encode, particularly for spear-phishing messages that closely mimic legitimate correspondence in vocabulary while deviating in pragmatic intent.
- Continual Learning Framework: Implementing an online learning mechanism that incrementally retrains the classifier on newly labeled messages to counteract adversarial concept drift, with automated drift-detection alerts triggering scheduled retraining cycles.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

- Multi-Modal Feature Fusion: Extending feature extraction beyond body text to incorporate sender reputation scores, DKIM and SPF authentication outcomes, attachment entropy analysis, and embedded image OCR output, producing a multi-modal feature vector that integrates syntactic, semantic, and structural evidence.
- Multilingual Generalization: Adapting the preprocessing pipeline and training data to support non-English spam detection using multilingual embedding models (mBERT, XLM-RoBERTa) to address the growing volume of foreign-language campaigns targeting non-Anglophone organizations.
- Federated Learning Architecture: Distributing the training process across multiple organizational mail servers without centralizing message content, preserving privacy while enabling collaborative model improvement across institutions facing related threat profiles.
- Explainable AI Dashboard: Developing a SHAP-based feature attribution visualization interface that surfaces per-message decision rationales to security analysts, enabling rapid identification of false positives and targeted refinement of misclassification patterns.

CONCLUSION

Spam email detection cannot rely on static keyword matching as its primary defense mechanism. When adversaries deliberately engineer messages to evade lexical filters, rule-based systems experience systematic performance degradation that no amount of rule refinement can sustainably compensate for. This structural vulnerability is not an implementation failure; it is the inevitable consequence of an architecture that attempts to enumerate malicious content rather than learning to recognize it.

The NLP-driven detection framework presented in this paper demonstrates an alternative architecture in which the classifier generalizes from statistical patterns in labeled training data, capturing the distributional signatures of spam language rather than its surface vocabulary. By integrating TF-IDF feature extraction with a Multinomial Naïve Bayes classifier trained on a combined Enron–Spam Assassin corpus, the system achieved 98.6% accuracy, 97.9% precision, and 98.3% F1-score at sub-millisecond inference latency. Experimental comparisons with SVM, Logistic Regression, and bidirectional LSTM baselines confirm that the Naïve Bayes variant occupies the optimal operating point for standard gateway deployment, while the LSTM offers superior performance for high-security contexts tolerating greater computational overhead.

This work contributes a practical, reproducible architectural template for organizations seeking to advance beyond syntactic filtering toward semantically aware spam detection—a transition that grows increasingly urgent as the linguistic sophistication of spam campaigns continues to accelerate.

REFERENCES

[1] P. Graham, "A Plan for Spam," Aug. 2002. [Online]. Available: http://www.paulgraham.com/spam.html

[2] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in Proc. AAAI Workshop on Learning for Text Categorization, Madison, WI, 1998, pp. 55–62.

[3] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," in Proc. European Conf. Machine Learning (ECML), Berlin, Germany, 1998, pp. 137–142.

[4] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with Naïve Bayes — Which Naïve Bayes?" in Proc. 3rd Conf. Email and Anti-Spam (CEAS), Mountain View, CA, 2006.

[5] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in Proc. EMNLP, Doha, Qatar, 2014, pp. 1532–1543.

[6] 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, 2019, pp. 4171–4186.

[7] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3, pp. 130–137, 1980.

[8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.