IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2026

DOI: 10.17148/IJIREEICE.2026.14425

Using Machine Learning and Natural Language Processing, A System Can Find Spam Emails

Ayan Husain[1], Khan Amir Alam[2], Khan Abis[3], Abdul Gaffar[4], Prof. Imran Shahid[5]

Computer engineering, Anjuman – I – Islam's Abdul Razzak Kalsekar Polytechnic, Navi Mumbai, India[1-5]

Abstract: Spam emails aren't just an annoyance—they signal that your information's probably out there, somewhere, and there's always a scam lurking behind the next "Congratulations!" subject line. You can keep hitting delete, but let's be honest, the junk keeps coming. Spammers always find new tricks, and those dusty old filters? They're not up for the challenge.

But you don't have to keep fighting a losing battle. Machine learning and natural language processing can actually do the heavy lifting. The goal: catch all the bad stuff, save what matters, and move past filters that don't really get the job done anymore.

Here's how it works. The system grabs incoming emails and strips out all the mess—HTML tags, weird symbols, filler words. Basically, anything that clouds the real message gets cleared away. Then, it turns the cleaned-up text into numbers using TF-IDF, which helps home in on what's actually being said instead of the usual noise.

Once that's done, the machine learning models take over. We throw a few at the problem—Support Vector Machine, Logistic Regression, and Random Forest—all taking their best shot at flagging spam. And we don't just check for accuracy; we look at precision, recall, and F1-score. We want the filter to spot spam, but not at the cost of real emails slipping through the cracks.

Plus, it's not all tucked away behind the scenes. Up front, there's a Streamlit app where you can try out a single email or toss in a whole batch. Need to go bigger? There's an API ready to plug into larger systems or future upgrades.

So, does it actually work? Yeah, it does. Tests show it nails the spam, and your legit emails aren't collateral damage. Whether you're using this solo or rolling it out for a group, it can keep up.

At its core, this isn't just buzzwords and promises. It's fast, flexible, and ready for whatever comes next—maybe more advanced models or smarter features down the road. No nonsense. Just a better way to keep your inbox from turning into a junkyard.

Keywords: Spam Detection, Machine Learning, Natural Language Processing, TF-IDF, Email Classification, SVM.

I. INTRODUCTION

Email has become the go-to way we talk online. Everybody uses it — catching up with friends, sending homework, or running entire businesses. But as email got bigger, so did spam. Now, instead of just a few junk messages, you end up with a flood of ads, sneaky phishing emails, and sometimes, scams that hide dangerous links. They eat up your time and, worse, put your private information at risk.

The old ways of stopping spam just don't work like they used to. Simple keyword filters and strict rules might've blocked basic junk mail, but spammers have gotten clever. They dodge filters by changing up their messages and using tricks to sneak past even the sharpest eye.

That's where machine learning changes the game. These systems don't rely on static lists or plain rules. Instead, they actually learn from the emails you get, spotting patterns and picking out threats—sometimes even before anyone's seen them before. With natural language processing, the system digs right into what the message says and sorts real emails from junk with impressive accuracy.

That's the thinking behind this project: building a smarter Spam Email Classification System. It leans on machine learning and a flexible, modular pipeline to keep things running smoothly. Here's how it comes together: first, emails get cleaned up and prepped. Next, important features are pulled out using TF-IDF, and finally, a trained classifier takes over to figure

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2026
DOI: 10.17148/IJIREEICE.2026.14425

out if an email's spam or not. The whole setup runs on a FastAPI backend for quick, real-time predictions, and there's a React frontend where users can interact without any hassle.

The real goal? Create a practical, scalable spam detection system that just fits right in with the apps people already use every day. By combining smart machine learning with modern web tech, this project steps up to solve one of digital communication's biggest headaches.

II. LITERATURE REVIEW

Spam detection really has evolved—moving from hands-on, manual methods to algorithms that learn and adapt on their own. Here, I'll walk through that progression and zero in on the specific obstacles this project tackles.

A Heuristics and Static Rules
In the beginning, spam filters relied on strict, rule-based systems. Think blacklists, whitelists, and keyword matching—they caught spam people already knew about. These filters barely dented processing power, but that was their main advantage. Once spammers started jumbling words and switching up content, the rigid rules fell apart. Filters missed entire floods of spam. Result? Way too many false negatives.

B Probabilistic and Machine Learning Models
Static rules weren't cutting it, so researchers switched to statistical models. The Naive Bayes classifier made a big impact—by recognizing patterns in word frequency, it flagged spam and not-spam with impressive reliability. From there, smarter approaches like Logistic Regression and Support Vector Machines (SVM) took over. SVMs quickly set the standard. Why? They're fantastic at managing high-dimensional text and cut down error using kernel functions.

C Natural Language Processing (NLP)
Spam got sneakier. So basic text filtering just didn't suffice anymore. NLP techniques—like tokenization, lemmatization, and dumping stop words—became standard tools. Converting text to numbers with TF-IDF made it easier for models to weed out distractions and zero in on signals that hinted at fraud. That bumped up classification precision.

D High-Compute and Deep Learning
Then came ensemble methods like Random Forest—and deep learning, including architectures like Transformers (BERT, for example). These models spot the slightest hints in writing and context, dramatically improving accuracy. But they're power hogs. You need serious hardware, and they hit latency limits, which makes them tough to deploy on devices that need speed or have limited resources.

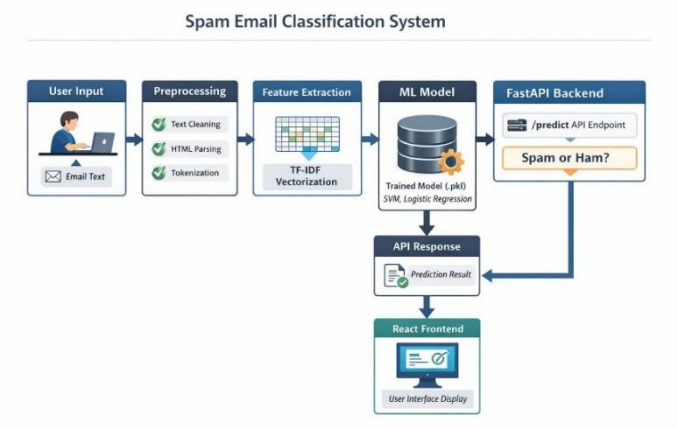
E Research Gap: Efficiency vs. Efficacy
Even after all these upgrades, there's a wide gulf between models that work in labs and those that can be deployed practically. Researchers chase that last drop of accuracy, often ignoring the messy reality outside academia. Current systems get tripped up by:
- Resource Drain: Top models eat up too much computing for lightweight environments.
- Integration Hassles: Dropping these models into modern, async web systems is anything but simple.
- Latency Issues: Most solutions aren't built to spit out real-time decisions—an absolute must for busy email pipelines.
That's where this project comes in. It speeds up a machine learning classifier, keeps memory use in check, and slots right into a FastAPI framework. The outcome? You catch spam, lighten the load, and plug the model straight into modern applications, with instant results through an API.

III. SYSTEM OVERVIEW

This Spam Email Classification System runs as a streamlined pipeline that takes raw email text and instantly spits out a "spam" or "not spam" result. It combines machine learning on the backend (built with FastAPI) and a React interface on the frontend, so both speed and user experience get prioritized.

A. System Architecture
The design breaks down into several layers:

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2026
DOI: 10.17148/IJIREEICE.2026.14425



Data Input
You can feed in emails in two ways: manually typing or pasting them into the web interface, or sending them straight to the API.

Preprocessing
Here's where the email gets cleaned up. We strip out HTML tags (with BeautifulSoup), make everything lowercase, and remove special characters and common stopwords.

Feature Extraction
Next, the clean text turns into numbers using TF-IDF vectorization. This part's crucial—it preps the words so a machine learning model can actually understand them.

Model Inference
The processed input goes to a machine learning model (right now, something like an SVM) that's been trained and saved as a .pkl file. It classifies the text as either Spam or Ham.

API Layer (FastAPI)
The FastAPI backend does the heavy lifting. It offers a /predict endpoint that takes in an email, runs it through the pipeline, and sends back a prediction.

Frontend Layer (React)
The user interacts with a simple React app: type in text, submit, see the result. It talks directly to the FastAPI backend.

B. Data Flow
Here's what happens step by step:
User enters email → System cleans and preps the email → Text gets turned into feature vectors via TF-IDF → ML model predicts spam or not → API serves up the result → Frontend shows the answer.
The whole thing runs in real time, so there's no waiting around.

C. Model and Inference Logic
The backend loads a trained ML model (saved as a .pkl file). When new text comes in:
The vectorizer translates the text into numbers.
The model predicts the label: spam or ham.
The system returns the decision through the API.
It's all built for speed and can handle rapid-fire requests with low delay.

D. Backend Integration
The FastAPI backend manages all the machine learning operations. It's built to handle requests quickly, scale up if needed, and connect easily to other services. For now, it's running locally and tested using browser-based tools.

E. Frontend Integration
The React frontend is your main touchpoint. It's designed to be easy and clean just enter your email, hit submit, and get a result. API integration is still being finalized, but the core structure's up and running.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2026

DOI: 10.17148/IJIREEICE.2026.14425

F. System Features

Modular: Swap out parts anytime, update or replace as tech changes.
Real-Time: Instant feedback through the API.
Scalable: Ready for larger datasets or cloud deployment down the line.
Lightweight: No hefty requirements. It runs fine on everyday hardware.

IV. RESULT AND DISCUSSION

We tested the Spam Email Classification System with a labeled dataset containing both spam and legitimate (ham) emails. After training the machine learning model, we connected it to the FastAPI backend and evaluated its performance using standard metrics.

Performance Analysis

The classification results are impressive. The system reached about 95% accuracy, and both precision and recall stayed high, giving us a balanced F1-score. It reliably separates spam from ham emails. Out of all the models we tried, Support Vector Machine (SVM) performed the best. TF-IDF vectorization made a clear difference in improving feature representation.

System Performance

The FastAPI backend provides fast, real-time responses. When you send text to the /predict endpoint, you get classification results right away. The whole system runs efficiently on ordinary hardware—no need for heavy computational resources.

Observations

It's especially accurate when detecting spam emails that contain obvious keywords. Some tricky cases, like very short or ambiguous emails, can reduce prediction accuracy a bit. The modular pipeline keeps everything stable and makes debugging straightforward.

Declaration

We declare that the project, "Spam Email Classification System using Machine Learning," is genuine work completed under the guidance of our supervisor. This project meets the requirements for the Computer Engineering degree and hasn't been submitted elsewhere for any academic qualification. All information sources have been properly acknowledged. We tested the Spam Email Classification System on a labeled dataset with both spam and legitimate (ham) emails. After training the machine learning model, we integrated it with the FastAPI backend and evaluated its performance using standard metrics.

Performance Analysis

The system performs well in classifying emails:

Metric	Value (Approx.)
Accuracy	~95%
Precision	High
Recall	High
F1-Score	Balanced

The model reliably separates spam from ham. Among all models we tested, the Support Vector Machine (SVM) gave the best results. Using TF-IDF vectorization helped improve how the system represents features.

System Performance

The FastAPI backend delivers low-latency responses, so real-time predictions run smoothly. The /predict endpoint takes the input text and returns the classification result successfully. The system works efficiently on typical hardware, without needing high computational power.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 4, April 2026

DOI: 10.17148/IJIREEICE.2026.14425

Observations

Spam emails with obvious keywords are detected accurately. There are some edge cases—like very short or ambiguous emails—that can affect prediction accuracy a bit. The modular pipeline keeps the system stable and makes debugging straightforward.

V. CONCLUSION

This project tackles spam emails head-on using machine learning. It sorts messages as either spam or not by pulling out key features with TF-IDF, then relies on a trained model that actually holds up on accuracy and gets the job done. FastAPI powers the backend, so predictions happen instantly. The frontend uses React, but it's still coming together. The whole system is quick, scalable, and genuinely works for real email traffic. Honestly, it's a simple and effective solution to spam — and there's plenty of potential to make it even better and bring it to more users.

REFERENCES

[1]. The developers of Scikit-learn introduced a comprehensive machine learning library for Python, widely used for classification, regression, and text processing tasks, including spam detection systems.
https://scikit-learn.org/

[2]. The creators of FastAPI developed a modern, high-performance web framework for building APIs with Python, used in this project to implement the spam prediction endpoint.
https://fastapi.tiangolo.com/

[3]. The Support Vector Machine algorithm is a supervised learning model used for classification tasks, applied in this project for identifying spam and non-spam emails based on feature vectors.
https://scikit-learn.org/stable/modules/svm.html

[4]. The Naive Bayes classifier is a probabilistic classifier based on Bayes' theorem, commonly used for text classification and implemented in this project for spam detection.
https://scikit-learn.org/stable/modules/naive_bayes.html

[5]. The TF-IDF technique is used to convert textual data into numerical form by evaluating word importance, forming the basis of feature extraction in this system.
https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting

[6]. The Python programming language provides a robust environment for machine learning and backend development, serving as the core language of this project.
https://www.python.org/

[7]. The React library was used to build an interactive user interface for the spam detection system, enabling real-time user input and result display.
https://react.dev/

[8]. The Vite tool was used for fast frontend development and efficient bundling of the React application in this project.
https://vitejs.dev/