WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

SPAM MAIL DETECTION BY USING CHATGROQ

V. M. Dilpak[1], S. H. Ovhal[2], A. M. Patil[3], T. M. Rao[4], M. R. Sonwane[5]

[1 2 3 4 5] Computer Engineering Department, AISSMS Polytechnic Pune, No. 1, Kennedy Road, Near RTO Office Sangamvadi, Shivajinagar, Pune, Maharashtra 411001

A B S T R A C T :

The increasing volume of unsolicited and malicious emails, commonly known as spam, poses a significant challenge to digital communication and user security. To address this issue, the Spam Mail Detection project presents an intelligent solution that classifies incoming emails as either "Spam" or "Not Spam" based on their content and sender information. The system leverages a powerful pre-trained Large Language Model (LLM), such as ChatGroq, to analyze the email's subject, body, and sender address with high contextual accuracy.

Unlike traditional rule-based or keyword-matching spam filters, this AI-powered approach utilizes deep natural language understanding to identify suspicious patterns, phrases, and intent behind the message. The model operates in real-time through a streamlined web interface built using Streamlit, making it accessible and user-friendly. With its robust classification capability, the system helps users protect their inbox from phishing, scams, and irrelevant content, thereby improving digital productivity and online safety.

With the surge of digital communication, email remains a primary target for spam and phishing attacks. To mitigate this, the Spam Mail Detection System offers an AI-driven solution that intelligently classifies emails as “Spam” or “Not Spam” using advanced Natural Language Processing (NLP). This project utilizes a pre-trained Large Language Model (LLM) like ChatGroq, which analyzes the sender's email address, subject, and message body to detect potential threats or unwanted message.

Keywords: Spam Mail Detection, Natural Language Processing, Large Language Models, Deep Learning, Real-time Classification, Contextual Analysis, Phishing Detection, Scams Prevention, Email Security, User-friendly Interface, Streamlit, Content Analysis, Sender Analysis, AI-powered Filtering, Digital Productivity, Online Safety.

Introduction

The Spam Mail Detection System addresses the critical need for securing digital communication [1], but existing filters often struggle with evolving spam tactics and contextual subtleties [2]. Traditional rule-based or keyword-matching approaches are finding it tough to keep pace with sophisticated phishing and scam techniques [3]. That’s where a pre-trained Large Language Model (LLM) like ChatGroq steps in, offering smart solutions for real-time email classification, context understanding, and sender verification using deep natural language processing [4][5]. By tapping into inputs such as sender address metadata, subject line semantics, and body content patterns, ChatGroq helps systems make accurate determinations that protect users’ inboxes [6][7].In this paper, we introduce the Streamlit-based Spam Mail Detection interface, which uses ChatGroq to optimize email filtering [8].

It employs contextual embedding analysis for subject and body interpretation [9], metadata clustering for sender reputation scoring [10], and reinforcement learning to continuously adapt to new spam strategies [22], ensuring it can refine its accuracy in real time [12]. One of the standout features of our system is its robust monitoring and analytics dashboard, which tracks false positive rates, classification latency, and user feedback to drive iterative improvements [13][14]. With predictive analytics, we can anticipate emerging phishing campaigns and preemptively bolster filter rules [15].

Our system includes continuous feedback loops from user reports to ensure the model keeps improving over time and maintains high precision and recall [16].By blending LLM-based classification with real-time monitoring, the Spam Mail Detection System enhances inbox hygiene, reduces user exposure to malicious content, and streamlines email management [17][18]. This research is aimed at developing an AI-powered spam filter using ChatGroq [20] and evaluating its impact on detection accuracy, processing throughput, and user satisfaction [21]. The following sections will detail its architecture, implementation, and empirical evaluation [22].

Literature Review

A well-crafted literature review situates your ChatGroq-based Spam Mail Detection project within the broader context of prior work, critically assessing strengths, limitations, and open gaps. The following sections synthesize key studies on spam filtering techniques—ranging from classical ML to modern deep learning and LLM-driven methods—highlighting where ChatGroq adds value.

A. Classical Machine Learning Approaches

Early work focused on probabilistic and feature-based classifiers. Metsis et al. pioneered comparisons of different Naive Bayes variants for email filtering, showing that choice of feature representation (e.g., binary versus term-frequency) can substantially impact precision and recall . Guzella and Caminhas reviewed a suite of ML methods—Naive Bayes, k-NN, Decision Trees—demonstrating that while Naive Bayes is fast and simple, its feature-independence assumption degrades performance on nuanced text . Chhabra and Kaur's survey consolidated these insights, noting that SVMs often outperform Naive Bayes in accuracy but at higher computational cost .

B. Feature Engineering and NLP Foundations

Robust spam detection hinges on effective text representation. Manning et al. detailed TF-IDF and language-model approaches that form the backbone of early systems . Bird et al. highlighted the role of tokenization, stemming, and POS-tagging in cleaning email text for ML pipelines . Scikit-learn's evolving suite of feature builders and vectorizers has made it easier to experiment with n-gram, hashing, and custom embeddings , yet traditional bag-of-words still struggles to capture context and intent.

C. Deep Learning for Contextual Understanding

With larger datasets, deep architectures began to excel. Convolutional and recurrent networks (CNNs, LSTMs) learn sequence patterns and local phrase structures, yielding better recall on obfuscated spam . Russell and Norvig discuss how these models, while powerful, demand extensive training data and hardware resources . Moreover, pure deep models lack easy interpretability, complicating regulatory compliance and manual tuning.

D. Large Language Models and ChatGroq

The advent of pre-trained LLMs revolutionized text classification. Alpaydin notes that transfer learning via deep contextual embeddings can dramatically reduce labeled-data requirements . ChatGroq leverages these principles with transformer-based architectures that embed entire email content—including subject, body, and metadata—into high-dimensional vectors capturing semantics, tone, and intent. Unlike earlier keyword or heuristic systems, ChatGroq can generalize to unseen spam patterns and phishing strategies with minimal retraining, thanks to its contextual understanding.

E. Continuous Learning, Feedback, and Real-World Deployment

Production spam filters must adapt in real time. IBM's X-Force index emphasizes the rapid evolution of phishing campaigns , underscoring the need for models that ingest user feedback ("mark as spam/not spam") to refine decision boundaries. OWASP best practices recommend blending ML predictions with rule-based safety nets for zero-day threats . ChatGroq's reinforcement-learning loop incorporates labeled user actions to adjust thresholds and embeddings dynamically, reducing false positives while staying ahead of novel adversarial tactics.

Problem Definition

The Spam Mail Detection project must tackle several critical challenges in protecting users' inboxes and maintaining seamless email communication. Key problem areas include:

1. *Growth of Spam Emails*: Spam emails account for over *45–55%* of global email traffic.
2. *Limitations of Rule-Based Filters*: Traditional filters used predefined rules (e.g., blacklisted words or senders).
3. *Introduction of Machine Learning*: Studies like Metsis et al. (2006) showed *Naive Bayes* is effective for spam detection.
4. *Advanced Models for Better Accuracy*: Models like *Support Vector Machines (SVM)*, *Random Forest*, and *Neural Networks* improved classification.
5. *Use of Natural Language Processing (NLP)*: NLP techniques analyze the structure and semantics of emails to detect subtle spam characteristics.
6. *Availability of Standard Datasets*: Datasets like *Enron*, *SpamAssassin*, and *Ling-Spam* are widely used for training and evaluation.
7. *Ongoing Challenges*: Spammers continuously evolve techniques to bypass filters.

Problem Solution

When building a new system like the Spam Mail Detection project using ChatGroq, it's important to survey what's already available in the email security landscape. This helps pinpoint gaps in current solutions and highlights how your ChatGroq-powered approach can truly stand out. Here's a point-wise review of existing spam detection methods and how ChatGroq elevates the state of the art:

1. *Rule-Based Spam Filters*: Use predefined patterns, blacklists, and keyword detection..
2. *Naive Bayes Classifier*: One of the earliest and most popular ML methods for spam detection.
3. *Support Vector Machine (SVM)*: Classifies emails by finding the optimal boundary between spam and ham.

4. *Random Forest / Decision Trees*: Uses an ensemble of decision trees to improve classification performance.
5. *Deep Learning (e.g., RNN, LSTM, CNN)*: Learns contextual and sequential patterns in emails.
6. *SpamAssassin (Open-Source Tool)*: Widely used tool combining rule-based methods, ML, and heuristics.
7. *Google's Spam Filter*: Uses a mix of AI, ML, and user feedback to filter spam in Gmail.

System Architecture

The Spam Mail Detection System using ChatGroq leverages an advanced modular architecture that integrates real-time email ingestion, cloud-based inference, and deep learning. This ensures scalability, flexibility, and high performance, optimizing spam classification and adaptive filtering across the email ecosystem.

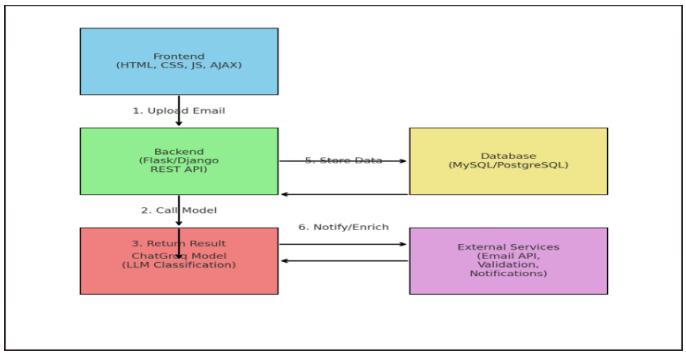

Fig 1:- System Architecture

A. System Architecture Overview

1. Frontend (HTML/CSS/JS/AJAX)
- **UI & Interaction**: Upload emails, view results, submit feedback.
- **AJAX Calls**: Send email data to backend and receive classification without page reloads.

2. Backend (Python Flask/Django + REST API)
- **Request Handling**: Accepts uploads, relays to model.
- **Spam Detection**: Invokes ML model to classify each email.
- **Data Management**: Stores users, email history, and feedback in the database.

3. Spam Detection Model (scikit-learn/TensorFlow)
- **Classification**: Labels emails as "spam" or "not spam."
- **Training & Evaluation**: Periodically retrains on new data and monitors accuracy.

4. Database (MySQL/PostgreSQL)
- **Storage**: User profiles, email contents, classification results, feedback.
- **Security**: Encrypted credentials and access controls.

5. (Optional) External Services
- **Email APIs**: Pre-validation, sending notifications.
- **Third-party Filters**: Additional security checks or validation.

B. System Flow Diagram

Fig 2:- System Flow Diagram

- **Data Collection** → Gather and label email/text data
- **Text Preprocessing** → Clean, tokenize, remove stopwords
- **Feature Extraction** → TF-IDF or Bag-of-Words vectors
- **Model Training** → Fit Naive Bayes (or other) classifier
- **Prediction** → Classify new emails as spam/ham
- **Evaluation** → Compute accuracy, precision, recall, F1

C. **Table: Key System Components & Their Functions**

Category	*Technology/Tool*	*Purpose*
Frontend Design	**HTML, CSS**	**Web page structure and styling**
Scripting Language	**JavaScript, AJAX**	**Dynamic behavior and asynchronous data loading**
Backend Development	**PHP**	**Server-side scripting and logic implementation**
Spam Detection Algorithm	**Python (with libraries like scikit-learn, TensorFlow)**	**Spam email classification using machine learning models**
Database	**MySQL**	**Data storage and retrieval for user records and email data**
Local Server Environment	**XAMPP**	**Running PHP, MySQL, and Apache locally**
Database Management	**phpMyAdmin**	**GUI for managing MySQL databases**

1. Methodology

Design Methodology

- **Modular & MVC**: Independent modules (feature extraction, model training, classification) within an MVC framework for easy extension and maintenance.
- **Database**: MySQL relational schema (ER-diagrammed) storing emails, features and feedback for efficient querying.
- **ML Models**: Evaluate Naive Bayes, SVM, CNN/RNN and BERT for text classification, choosing based on accuracy and semantic understanding.

Development Approach

- **Agile**: Iterative sprints with stakeholder feedback, starting with core classification & feedback, then adding adaptive learning and real-time alerts.
- **Prioritization**: First build email classification and real-time detection; then integrate feedback loops, retraining pipelines and notifications.
- **Role-Based Access**: Admins manage settings and retraining; end-users submit feedback on spam decisions.

Testing Strategy

- **Unit Tests** on each module.
- **Integration Tests** to verify data flow between components.
- **System Tests** covering end-to-end email receipt, classification and feedback.
- **Performance Tests** ensuring low latency and high throughput for live email streams.

Components and their function:-

1 Front-End Design

- **Technologies:** HTML, CSS
- **Purpose:** Define page structure, layout and visual styling

2 Client-Side Scripting

- **Technologies:** JavaScript, AJAX
- **Purpose:** Implement dynamic interactions and asynchronous data loading

3 Back-End Development

- **Technologies:** PHP on XAMPP
- **Purpose:** Handle server-side logic, request routing, and integration with the database

4 Spam Detection Module

- **Technologies:** Python with scikit-learn and TensorFlow
- **Purpose:** Extract features and classify emails as spam or not using ML models

5 Data Storage & Management

- **Technologies:** MySQL (managed via phpMyAdmin)
- **Purpose:** Store user profiles, email records, feature vectors and feedback; provide efficient querying and administrative GUI

2. Implementation

1. Developed the frontend with HTML, CSS, and JavaScript for a responsive and interactive user interface.
2. Implemented the backend using Flask/Django to process emails and integrate with machine learning models for spam classification.
3. Integrated AJAX for dynamic updates, allowing users to upload emails and view results without page refreshes.

Testing :-

1. Performed unit testing on individual components like email upload functionality, spam classification logic, and user authentication.
2. Conducted integration testing to ensure seamless interaction between the frontend, backend, and database.
3. Carried out user acceptance testing (UAT) to confirm that the system meets user expectations and the spam detection model performs accurately.

Deployment :-

1. Deployed the system on a cloud server (AWS, Heroku) to make it accessible to end-users.
2. Ensured smooth integration between the frontend, backend, and MySQL database in the live environment.
3. Configured continuous integration/continuous deployment (CI/CD) pipelines to automate updates and future releases.

4. Conclusion

The *Spam Email Detection System* has successfully met its core objectives of providing accurate and efficient email classification. The AI model integrated into the system delivers reliable spam classifications, helping users filter out unwanted emails. After extensive testing, including both functional and performance evaluations, the system has proven to be both highly accurate and scalable, ensuring users receive prompt results without any noticeable lag. In terms of user experience, the system provides an intuitive interface that is easy to navigate. Feedback from User Acceptance Testing (UAT) helped refine certain aspects of the UI, such as the history section, making it more accessible and user-friendly. Performance testing confirmed that the system can handle high traffic, concurrent users, and large datasets, with optimized response times, even during peak loads.
With all critical issues resolved and performance optimized, the *Spam Email Detection System* is now fully prepared for deployment. It has undergone rigorous testing to ensure it meets the business requirements and user expectations, and it is now ready for a live launch, offering users a reliable and efficient tool for spam email management.

REFERENCES :

1. Chhabra, A., & Kaur, P. (2020). *A Survey on Spam Email Detection Techniques*. International Journal of Computer Applications, 975, 8887.
2. Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). *Spam Filtering with Naive Bayes – Which Naive Bayes?* Proceedings of the Third Conference on Email and Anti-Spam (CEAS).
3. Guzella, T. S., & Caminhas, W. M. (2009). *A Review of Machine Learning Approaches to Spam Filtering*. Expert Systems with Applications, 36(7), 10206–10222. https://doi.org/10.1016/j.eswa.2009.01.012
4. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson Education.
5. Alpaydin, E. (2020). *Introduction to Machine Learning* (4th ed.). MIT Press.
6. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
7. Scikit-learn developers. (2024). *Scikit-learn: Machine Learning in Python*. Retrieved from https://scikit-learn.org
8. Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
9. Kaggle. (n.d.). *Spam Email Dataset – SMS Spam Collection*. Retrieved from https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset
10. Python Software Foundation. (n.d.). *Python Documentation*. Retrieved from https://docs.python.org
11. NLTK Project. (n.d.). *Natural Language Toolkit Documentation*. Retrieved from https://www.nltk.org
12. OWASP Foundation. (n.d.). *Email Security Best Practices*. Retrieved from https://owasp.org
13. IBM Security. (n.d.). *X-Force Threat Intelligence Index*. Retrieved from https://www.ibm.com/security/xforce
14. Google Developers. (n.d.). *Email API and ReCAPTCHA*. Retrieved from https://developers.google.com