Project Report: Email Spam Detector

Author: Zain ul Abdin Ghani

Intern ID: ARCH-2508-0376

1. Introduction

This project is an Email Spam Detector application. It uses machine learning to analyze email content and determine if the email is spam or not. The project has a web interface where users can paste email text and get an instant spam detection result.

The main goal is to help users identify unwanted or harmful emails quickly and easily.

2. Technologies Used

- **Frontend:** React.js and Tailwind CSS
  For building the user interface. React provides a dynamic experience while Tailwind CSS helps in designing a responsive and clean UI.
- **Backend:** Flask
  Flask runs the API server. It loads the machine learning model and handles prediction requests.
- **Machine Learning:** Python, scikit-learn
  The ML model is based on Multinomial Naive Bayes. It uses TF-IDF vectorization to convert email text into numbers the model can understand.
- **Text Processing:** NLTK
  This library helps clean the email text, removing stopwords, URLs, and applying stemming.
- **Model Storage:** joblib
  The trained model and vectorizer are saved using joblib to load them quickly for predictions.

3. Project Structure

- **frontend/**
  Contains React code including the main component SpamDetector.jsx. It handles input, calls the backend API, and displays results.
- **backend/**
  Contains Flask API code, model training script, prediction code, and saved ML models.
- **data/**
  Includes the dataset used for training the spam detection model.

4. How It Works

Frontend

Users enter the full email content in a text box. The app checks that the text has at least 100 characters and 15 words for accurate analysis. After validation, the content is sent to the backend API.

Backend

The backend receives the email content and processes it. It applies text cleaning and TF-IDF vectorization, then passes the processed data to the trained Naive Bayes model. The model returns a prediction: spam or not spam.

The backend sends this result back to the frontend in JSON format.

Frontend Display

The frontend shows the result to the user with a color-coded message: red for spam and green for legitimate emails. It also shows a preview of the analyzed text.

5. Features

- Real-time spam detection with quick response.
- Input validation to ensure minimum email content size.
- Clear and simple user interface with responsive design.
- Loading indicators during analysis.
- Detailed error messages if anything goes wrong.
- No data storage: user input is only used temporarily for prediction.

6. Model Training

The model is trained using a dataset of emails labeled spam or not spam. The training steps are:

- Load and clean the data.
- Combine the subject and message text.
- Convert text into numbers using TF-IDF vectorizer.
- Train a Multinomial Naive Bayes classifier on the vectorized data.
- Save the model and vectorizer for later use in the backend.

The model achieves good accuracy on both training and testing data.

7. How to Run the Project

Backend Setup

1. Go to the backend directory.
2. Install Python dependencies using pip install -r requirements.txt.
3. Train the model by running python train_model.py (only once or when retraining).
4. Start the Flask server with python app.py. It runs on localhost port 5000.

Frontend Setup

1. Go to the frontend directory.
2. Install node packages using npm install.
3. Start the React development server with npm run dev.
4. Open the browser at http://localhost:3000 to access the app.

8. Limitations and Future Improvements

- The model works best with English emails.
- The input length restrictions may block very short emails. This can be improved.
- Adding support for attachments and HTML emails can make detection more robust.
- Hosting the backend on a cloud server will allow remote access.
- Implementing user accounts to save analysis history is a possible extension.

9. Conclusion

This project shows how machine learning can be applied to email spam detection. It combines a friendly React interface with a Python backend running a trained model. The application is easy to use and provides fast, accurate results.

It is a helpful tool for anyone who wants to check suspicious emails before opening them.

10. Author

Zain ul Abdin Ghani

Full-Stack Developer