Email Classification for Support Team

Problem Statement:

The aim of this project is to develop an intelligent email classification system tailored for a company's customer support team. The system is expected to perform two primary functions:

1. **Identify and mask any Personally Identifiable Information (PII) and Payment Card Industry (PCI) details** found in incoming email content.
2. Importantly, the masking process must be implemented without using Large Language Models (LLMs). You are allowed to use regular expressions, traditional machine learning, deep learning techniques, or Named Entity Recognition (NER) to achieve this.
3. **Categorize each email**—once it has been masked—into one of several predefined support categories such as Billing Issues, Technical Support, Account Management as specified in the dataset.

This approach ensures that sensitive user data is protected during processing and that customer queries are directed to the appropriate team for resolution.

Approach Overview

PII Masking:

To mask sensitive information without relying on Large Language Models (LLMs), the following techniques were used:

- spaCy Named Entity Recognition (NER): Uses the lightweight en_core_web_sm model to detect PERSON, ORG, and DATE entities, masking them as [full_name], [org], or [dob].
- Regular Expressions (Regex): Custom patterns detect:
  - Email Address (email)
  - Phone Number (phone number)
  - Credit/Debit Card Number (credit_debit_no)
  - Aadhar Number (Aadhar Num)
  - CVV (cvv_no)
  - Expiry Date (expiry no)
  - Date of Birth (dob)

- Multi-Layered Detection: Combines spaCy NER, regex matching, and custom rule-based name detection to maximize recall.

• Reversible Masking: Original text and positions are stored for demasking after classification.

Email Classification

We used a hybrid and modular approach to classify emails into 4 categories:

- Incident
- Request
- Change
- Problem

The classification pipeline includes:

• Keyword-Based Classification for high-confidence cases, bypassing ML when possible.

• Machine Learning fallback using TF-IDF vectorization (1-2 grams, top 10,000 features) and Logistic Regression with balanced class weights for ambiguous cases.

• Enhanced preprocessing with text cleaning, tokenization, and lemmatization.

• Fallback ML model training from sample data to ensure API robustness if the primary dataset is missing or corrupted.

MODEL TRAINING AND TESTING DETAILS

- Dataset: Provided email classification dataset.
- Split: 80% training / 20% testing with stratified sampling.
- Preprocessing: Text cleaned, lowercased, whitespace normalized, and subject prefixes removed.
- Vectorization: TF-IDF with max_features=10,000, ngram_range=(1,2).
- Model: Logistic Regression with balanced class weights (max_iter=1000, random_state=42).
- Accuracy: Training accuracy varies based on dataset quality, with fallback synthetic training for missing data

The trained model is automatically loaded into the FastAPI application for real-time inference, with graceful fallback handling and comprehensive logging throughout the prediction pipeline.

Challenges Faced and Solutions Implemented

Challenge: Missing Training Data

Solution: Created fallback model with synthetic training data based on keyword patterns

Challenge: Class Imbalance

Solution: Used class weight ='balanced' in Logistic Regression

Challenge: PII Detection Accuracy

Solution: Multi-layered approach combining spaCy NER with regex patterns and overlap removal

Challenge: Model Reliability

Solution: Hybrid classification with keyword-based confidence scoring before ML prediction

Challenge: Database Dependency

Solution: Graceful fallback to SQLite when PostgreSQL unavailable

Challenge: spaCy Model Dependencies

Solution: Automatic model downloading and fallback handling.

FINAL OUTPUT:

• API Endpoint (POST): https://rayvanthk-email-classifier.hf.space/classify

• GitHub Repository: https://github.com/krayvanth09/Email_Classification

The API accepts a POST request with an input email body, masks PII, classifies the email, and returns the masked version, classification result, and list of masked entities in a structured format.