Email PII Masker and Classifier

By Spurthi Srivatsa

Introduction

The **Email PII Masker and Classifier** is a tool I developed to address two key challenges in handling support emails: safeguarding sensitive personal information and automating the categorization of incoming messages. In a typical support environment where teams receive a high volume of emails daily, this tool ensures that personally identifiable information (PII) such as names, email addresses, and phone numbers is masked to maintain user privacy. Simultaneously, it classifies the emails into predefined categories such as *Request*, *Incident Problem*, and more, to streamline the response process.

Tech Stack

Technology	Purpose
Regex	PII detection and masking
TF-IDF	Feature extraction from text
Naive Bayes	Email classification
FastAPI	API development
Docker	Deployment containerization
Hugging Face	Hosting and live demo
Python 3.9	Primary programming language

How It Works

My project has two main steps: masking personal info and classifying emails, all tied together in a FastAPI pipeline that works in real-time. It starts when a user sends an email through the API first, it masks any sensitive info like names or phone numbers, then it processes the email to figure out its category, and finally, it send back the results in a neat JSON format. This whole process happens quickly, so support teams can use it on the fly!

System Flow Diagram

PII Masking: Implementation Approach

To ensure the protection of personally identifiable information (PII), I developed a robust masking mechanism using regular expressions (regex). This module scans incoming emails and replaces sensitive data with anonymized placeholders—ensuring privacy without losing context.

How It Works

The mask_pii function in utils.py identifies and masks a wide range of PII types using specialized regex patterns:

- **Credit/Debit Card Numbers**: Detects 16-digit sequences, often grouped in fours, with optional separators (hyphens or spaces).
- **Aadhar Numbers**: Recognizes India's 12-digit unique ID format (xxxx-xxxx-xxxx).
- **Date of Birth (DOB)**: Supports multiple formats like 12-Jan-1995, 12/05/1995, etc.
- **Card Expiry Date**: Matches MM/YY or MM/YYYY formats.
- **Emails**: Captures both standard and internationalized email fomats.
- **Full Names**: Uses context-aware logic to minimize false positives while masking names intelligently.
- **Phone Numbers**: Covers global number formats with or without country codes.
- **CVV Codes**: Accurately identifies 3-digit numbers placed between non-digit characters to avoid misclassification.

The mask_pii function in utils.py executes the masking process, ensuring accurate detection and replacement.

- **Example:** "John Doe at john.doe@example.com" → "[full_name] at [email]".
- **Example:** "Contact +91 1234567890" → Contact [phone_number]
- **Example:** "My card number is 4111 1111 1111 1111" → "My card number is [credit_debit_no]"

Email Classification: Methodology and Implementation

For email categorization, the Multinomial Naive Bayes algorithm was selected due to its proven efficacy in text classification tasks.

- **Methodology**:
    - **Preprocessing:** Email content is cleaned by removing HTML tags, stop words, and tokenizing the text.
    - **Feature Extraction:** TF-IDF vectorization transforms the text into numerical features.
    - **Model Training:** The model is trained on labeled email data to predict categories.
- **Implementation Details**:
    - Data was sourced from combined_emails_with_natural_pii.xlsx.
    - The scikit-learn library was utilized for TF-IDF vectorization and model training.
    - The trained model was serialized using joblib for integration with the API.
    - The system outputs category labels (e.g., "Incident") along with confidence scores.

API Development and Deployment

The solution is integrated into a FastAPI application, facilitating real-time email processing and classification.

- **Workflow**:
    - **Input:** Emails are submitted via a POST request to the / endpoint.
    - **Processing:** PII masking, TF-IDF vectorization, and classification are performed sequentially.
    - **Output**: A JSON response is returned, containing the masked email, list of masked entities, and the email's category.
- **Technical Details**:
    - The pipeline is implemented in api.py using FastAPI.
    - The application was deployed on Hugging Face Spaces using Docker, configured to operate on port 7860.

Conclusion

The Email PII Masker and Classifier secures emails and enhances support workflows by masking sensitive information and automating categorization. It leverages machine learning and API development to deliver a robust solution, improving data privacy and operational efficiency for support teams in real-world scenarios.