Email Classification for Support Team Report :

Introduction to the Problem Statement

This project addresses the need for an automated system to classify customer support emails into predefined categories while ensuring privacy through PII (Personally Identifiable Information) masking. The system is designed to:

1. Mask sensitive information in incoming emails
2. Classify the emails into appropriate categories
3. Return the classification results with properly formatted output
4. Provide these services through an API endpoint

The solution is particularly valuable for support teams that handle a high volume of emails and need to route them efficiently while maintaining customer privacy.

Approach

PII Masking Approach

For PII masking, we implemented a rule-based approach using regular expressions. This approach was chosen for several reasons:

1. **Efficiency**: Regex patterns are computationally efficient
2. **Precision**: Custom patterns can be tailored for specific PII types
3. **No LLM dependency**: As required, this approach doesn't rely on large language models

The PII masking component identifies and masks eight types of sensitive information:

- Full names
- Email addresses
- Phone numbers
- Dates of birth
- Aadhar card numbers
- Credit/debit card numbers
- CVV numbers
- Card expiry dates

Each entity is replaced with a tag indicating the type of information that was masked (e.g., [full_name], [email]). The original text and position information are stored for output formatting.

Email Classification Approach

For email classification, we used a machine learning approach with the following components:

1. **Text preprocessing**: Converting emails to numerical features using TF-IDF vectorization
2. **Classification model**: Random Forest classifier

This approach was selected because:

- It handles text data effectively
- It's robust against overfitting
- It provides good performance with relatively small datasets
- It can handle the multi-class nature of support categories

System Integration

The system integrates the PII masking and classification components in a pipeline:

1. Receive email text
2. Mask PII using regex patterns
3. Classify the masked email using the trained model
4. Format the output with masked text and detected entities

API Implementation

The API was implemented using FastAPI, which provides:

- Modern, fast framework with automatic documentation
- Type checking and validation
- Asynchronous request handling
- Easy deployment options

Model Selection and Training Details

Feature Extraction

We used TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert the email text into numerical features. This approach:

- Captures the importance of words in the context of the entire corpus
- Reduces the impact of common but uninformative words
- Creates a sparse matrix suitable for machine learning algorithms

Configuration:

- Maximum features: 10,000
- N-gram range: (1, 2) to capture both individual words and common phrases

Classification Model

We selected Random Forest as our classification algorithm due to its:

- Robustness to overfitting
- Ability to handle high-dimensional data
- Good performance on text classification tasks
- Interpretability of feature importance

Configuration:

- Number of estimators: 100
- Random state: 42 (for reproducibility)

Training Process

The training process follows these steps:

1. Preprocess the dataset (handle missing values, extract class labels)
2. Split data into training (80%) and testing (20%) sets
3. Train the TF-IDF vectorizer and Random Forest classifier
4. Evaluate model performance using accuracy and classification report
5. Save the trained model for later use

Challenges and Solutions

Challenge 1: Effective PII Detection

Challenge: Creating regex patterns that can catch different variations of PII without excessive false positives.

Solution: We developed comprehensive patterns for each PII type and tested them against various inputs. The patterns were refined through iteration to balance recall (catching all PII) and precision (minimizing false positives).

Challenge 2: Handling Different Email Formats

Challenge: Support emails can come in various formats, lengths, and styles.

Solution: The TF-IDF vectorizer helps normalize these differences by focusing on important terms rather than structural elements. We also included n-grams to capture meaningful phrases.

Challenge 3: API Output Formatting

Challenge: The API needs to return results in a specific JSON format with proper entity positions.

Solution: We created a dedicated function to format the output according to the requirements, ensuring that detected entities maintain their correct positions even after masking operations.

Challenge 4: Deployment Considerations

Challenge: Ensuring the solution works smoothly in a deployed environment.

Solution: We developed a flexible application structure that can work both locally and in cloud environments, with clear configuration options and error handling.

Future Improvements

1. **Enhanced PII Detection**: Incorporate more sophisticated pattern recognition or hybrid approaches
2. **Model Optimization**: Explore different classification algorithms or deep learning approaches
3. **User Feedback Integration**: Add mechanisms to learn from user corrections
4. **Performance Monitoring**: Implement logging and monitoring to track system performance over time

Conclusion

The Email Classification System effectively addresses the requirements by providing a robust solution for classifying support emails while maintaining privacy through PII masking. The system's modular design allows for easy extensions and improvements, making it adaptable to evolving business needs.