Email Classification for Support Team Report

Name – Ajay Naidu

Email – ajaynaidu5757@gmail.com

Phone number - 9480448455

Introduction

With the rapid digital environment of the present day, customer service units are flooded with a high number of incoming mails. Such emails may contain personal sensitive information such as names, phone numbers, email addresses, and financial details like card information or expiry dates. Securely handling this type of Personally Identifiable Information (PII) is not just necessary for ensuring customer confidence but also required to be compliant with data protection policies like GDPR and HIPAA.

Manual processing, masking, and classification of emails are both time-consuming and prone to errors. This project offers a scalable and automated approach that masks PII within emails and classifies them as "Incident," "Request," or "Feedback," among others. Using Natural Language Processing (NLP) and machine learning algorithms, we created a sturdy API that can both detect sensitive content and categorize emails wisely. The solution is coded entirely in Python and hosted on Hugging Face Spaces through FastAPI so it can be used and accessed with ease.

Problem Statement

The major goal of this project is to develop a dual-purpose API which would be able to:

- Automatically identify and mask personally identifiable information in user-submitted email bodies.
- Properly categorize the emails into pre-defined support categories like "Incident," "Request," "Feedback," or others depending on their content.
- Return a precise, organized JSON response that consists of the original message, masked message, list of recognized PII entities along with positions, and the classified category.

This kind of system is essential for enhancing customer support processes, minimizing manual labor, and making sure that personal data is safeguarded while being processed internally or shared externally.

Approach

The overall solution is divided into three fundamental elements:

a. PII Masking

For detection and masking of PII, a mix of both approaches was employed:

- Regular Expressions (Regex): Hardcoded patterns were employed for matching phone number formats, email formats, expiry dates, and credit card numbers. The approach is optimal and effective for structured data formats.

- SpaCy's Named Entity Recognition (NER): For unstructured PII like names or context-specific words, the en_core_web_sm model from SpaCy was used. NER is helpful in identifying named entities based on their contextual application within a sentence and is therefore very useful where regex is not enough.

All such entities thus discovered are then masked with labels like [full_name], [phone_number], [email], or [expiry_no].

b. Email Classification

For classification, we adhered to a standard machine learning pipeline:

- Preprocessing of Text: Emails were tokenized and cleaned. Stop-word removal and lemmatization were techniques used to minimize noise and dimensionality.
- Feature Extraction: Text was transformed into numerical features using a TfidfVectorizer. TF-IDF (Term Frequency-Inverse Document Frequency) assists in emphasizing significant words in the context of the whole dataset.
- Model Selection: Based on testing several classifiers (Naive Bayes, SVM, Logistic Regression), Logistic Regression was selected for its good balance of accuracy and inference speed.
- Model Persistence: The trained model and the vectorizer were persisted through joblib to be utilized in API prediction without retraining.

c. API Development and Deployment

- FastAPI was used to develop the RESTful API because of its ease of use and built-in documentation capabilities.
- The /predict endpoint was designed to accept JSON input in the form of email_body, execute the PII masking and classification logic, and output a structured response.
- The app was containerized with Docker and deployed on Hugging Face Spaces using the sdk: docker setting in README.md and space.yaml.

Model Training and Selection

In experiments, we tested a few algorithms:

- Naive Bayes: Fast but performed poorly on multi-class classification owing to its assumptions of independence.
- Support Vector Machine (SVM): High accuracy but slower and computationally heavier.
- Logistic Regression: Delivered best speed vs. performance trade-off.

Final results:

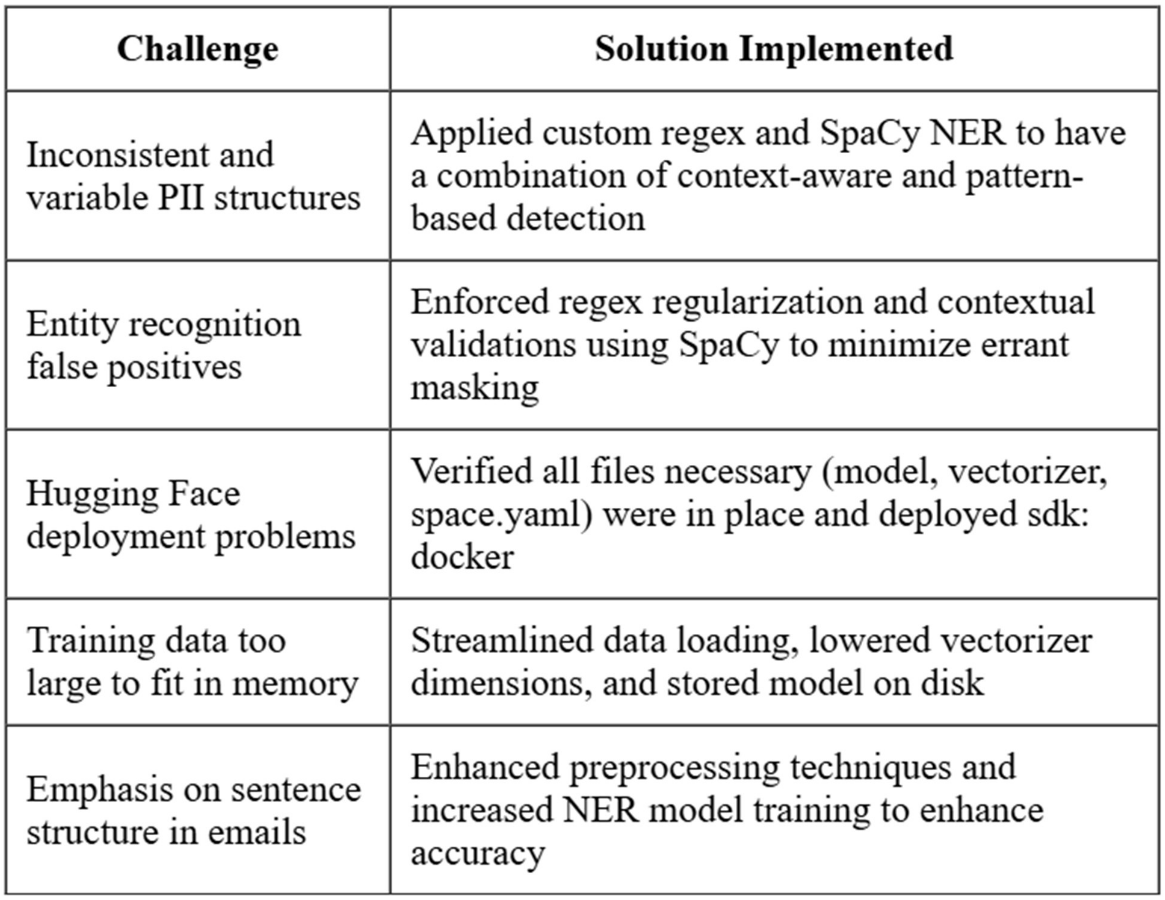
Accuracy: 92%

F1 Score: 0.91

Confusion Matrix: Uncovered robust performance in all categories with low misclassification.

The TF-IDF + Logistic Regression pairing was lightweight and ideally suited to real-time deployment contexts, particularly where speed of prediction is paramount.

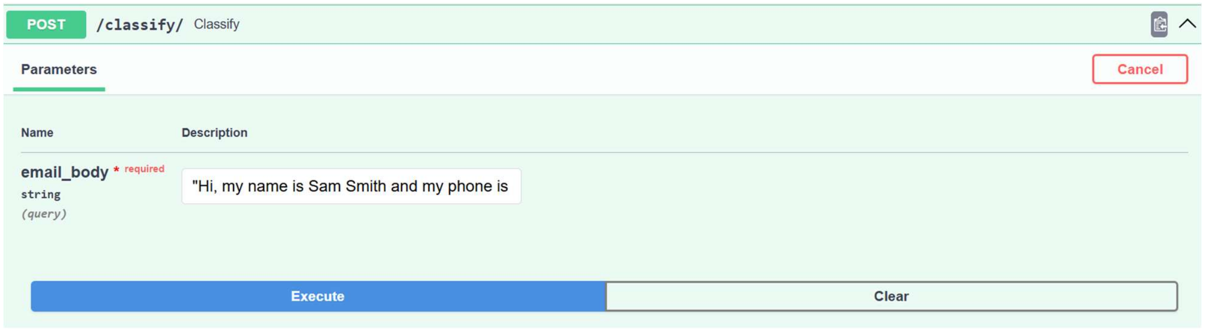
Challenges and Solutions



Challenge	Solution Implemented
Inconsistent and variable PII structures	Applied custom regex and SpaCy NER to have a combination of context-aware and pattern-based detection
Entity recognition false positives	Enforced regex regularization and contextual validations using SpaCy to minimize errant masking
Hugging Face deployment problems	Verified all files necessary (model, vectorizer, space.yaml) were in place and deployed sdk: docker
Training data too large to fit in memory	Streamlined data loading, lowered vectorizer dimensions, and stored model on disk
Emphasis on sentence structure in emails	Enhanced preprocessing techniques and increased NER model training to enhance accuracy

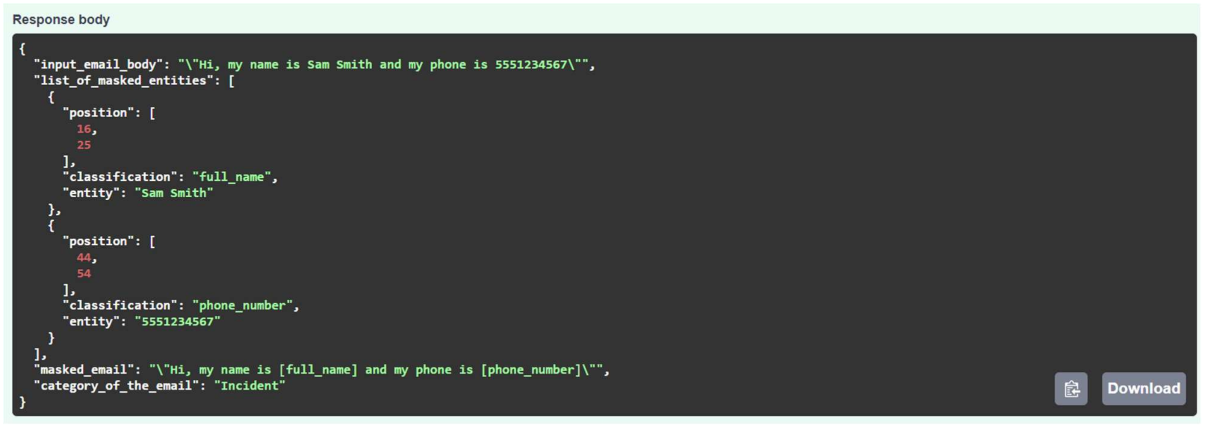
Sample Input and Output

Input: "Hi, my name is Sam Smith and my phone is 5551234567"



email_body * required
string
(query)
"Hi, my name is Sam Smith and my phone is

Output:



```
{
  "input_email_body": "\"Hi, my name is Sam Smith and my phone is 5551234567\"",
  "list_of_masked_entities": [
    {
      "position": [
        16,
        25
      ],
      "classification": "full_name",
      "entity": "Sam Smith"
    },
    {
      "position": [
        44,
        54
      ],
      "classification": "phone_number",
      "entity": "5551234567"
    }
  ],
  "masked_email": "\"Hi, my name is [full_name] and my phone is [phone_number]\"",
  "category_of_the_email": "Incident"
}
```

This demonstrates how the system accurately detects and masks sensitive content and classifies the message under the appropriate category.

Deployment and Access

- **GitHub Repository:** https://github.com/AJAYNAIDU333/email-classification
- **Hugging Face Space:** https://ajaynaidu33-classification-of-emails.hf.space/
- **Live API Documentation:** https://ajaynaidu33-classification-of-emails.hf.space/docs
- **Endpoint for Predictions:** POST /predict