Report: Support Email Classification System

1. Introduction

Customer support teams receive a high volume of emails daily, ranging from billing inquiries to technical issues. Manually categorizing these emails is time-consuming and prone to human error, delaying response times and reducing customer satisfaction. This project aims to automate the classification of incoming support emails into predefined categories (e.g., **Billing Issues**, **Technical Support**, **Account Management**, etc.) while ensuring all **Personally Identifiable Information (PII)** is masked before processing. After classification, masked PII is accurately restored in the final output. The solution is exposed via a **FastAPI**-based RESTful API, deployable on Hugging Face Spaces.

2. Approach

2.1 PII Masking

- **Techniques**: We employed **regular expressions** (regex) for PII detection—covering full names, email addresses, phone numbers, dates of birth, Aadhaar numbers, credit/debit card numbers, CVV, and expiry dates.
- **Masking Logic**: Upon receiving an email text, each PII pattern is located using `re.finditer()`. Matches are replaced in-line with placeholders of the form `[entity_type]`, and original values are stored with their start/end positions.
- **Demasking**: After classification, placeholders are sequentially replaced with the original PII values, preserving document integrity.

2.2 Email Classification

- **Feature Extraction**: We used `TfidfVectorizer` (max 5,000 features, English stopwords) to convert masked email text into numerical feature vectors capturing term importance.
- **Model Selection**: Given the moderate dataset size and clear categories, **Support Vector Machine (SVM)** with probability estimates was chosen for its robustness and

generalization on high-dimensional sparse data.

- **Training Pipeline**:
    1. Load CSV with columns `email` (text) and `type` (label).
    2. Fit TF-IDF on training emails.
    3. Train `sklearn.svm.SVC` on TF-IDF vectors.
    4. Serialize both vectorizer and model as pickled artifacts in `models/`.

3. Model Selection & Training Details

- **Dataset**: `combined_emails_with_natural_pii.csv`, ~[dataset size] entries distributed across categories.
- **Environment**: Python 3.13, `scikit-learn 1.6.1`, `pandas`, `regex`.
- **Training Command**: python train.py data/combined_emails_with_natural_pii.csv
- **Artifacts**:
    - `models/vectorizer.pkl`
    - `models/classifier.pkl`

4. Challenges & Solutions

- **Windows Build Errors**: Installing heavy NLP libraries (e.g., spaCy) on Windows/Python 3.13 led to compilation errors due to outdated GCC.
    - *Solution*: Replaced spaCy NER with pure-regex masking to eliminate C/C++ build dependencies.
- **Column Name Mismatch**: Initial code expected `email_body`/`category`, but dataset used `email`/`type`.
    - *Solution*: Updated `models.py` to reference the correct column names.
- **Placeholder Overlap**: Masking replaced text lengths, shifting subsequent indices.

- *Solution*: Replace matches sequentially from start to end, recalculating offsets accurately for each placeholder.

5. API Implementation & Deployment

5.1 Endpoint Definition

- **URL**: `POST /classify-email`

Request JSON:

```
{ "email_body": "<raw email text>" }
```

- **Response JSON**:

```
{
  "input_email_body": "<original text>",
  "list_of_masked_entities": [
    { "position": [start, end], "classification": "entity_type", "entity": "value" }
  ],
  "masked_email": "<masked text>",
  "category_of_the_email": "<predicted category>"
}
```

5.2 Running Locally

1. Install dependencies: `pip install -r requirements.txt`
2. Train model: `python train.py data/combined_emails_with_natural_pii.csv`
3. Start server: `uvicorn api:app --reload`
4. Test via Swagger UI: `http://127.0.0.1:8000/docs`

5.3 Deployment on Hugging Face Spaces

- Include `app.py`, `api.py`, `models/` artifacts, and `requirements.txt`.
- Expose `uvicorn api:app --host 0.0.0.0 --port $PORT` in `start.sh`.
- API will be accessible at https://shubhamprasad318-email-classifier.hf.space/classify-email

6. Final Output & Testing

After deployment, you can send POST requests to the live Hugging Face endpoint. Here's a sample test with `curl`:

```
curl -X POST https://shubhamprasad318-email-classifier.hf.space/classify-email \
  -H "Content-Type: application/json" \
  -d '{"email_body":"Hello, I'm John Smith, my Aadhaar is 1234 5678 9012. I have a billing question."}'
```

Expected response:

```
{
  "input_email_body": "Hello, I'm John Smith, my Aadhaar is 1234 5678 9012. I have a billing question.",
  "list_of_masked_entities": [
    { "position": [17, 27], "classification": "full_name", "entity": "John Smith" },
    { "position": [42, 56], "classification": "aadhar_num", "entity": "1234 5678 9012" }
  ],
  "masked_email": "Hello, I'm [full_name], my Aadhaar is [aadhar_num]. I have a billing question.",
  "category_of_the_email": "Billing Issues"
}
```

Repository Link: https://github.com/shubhamprasad318/email_classifier

End of Report