Email Classification System for Support Team

Introduction

- **Objective**: Automatically classify incoming support emails into categories like *Billing Issues*, *Technical Support*, etc.
- Ensure PII (Personally Identifiable Information) and PCI (Payment Card Information) is masked before processing.
- Restore original content after classification for final output.

Approach Overview

1. PII Masking (No LLMs)

A **rule-based masking mechanism** using **regular expressions (Regex)** was implemented to detect and replace sensitive personal information in email texts.

- Example:
  "Hello, my name is John Doe" → "Hello, my name is [full_name]"
- Masked data is stored securely for demasking after processing.

2. Email Classification (LSTM on GPU)

- Cleaned and tokenized text using nltk and Tokenizer.
- Used **Bidirectional LSTM model** with:
  - Embedding → LSTM → Dense → Dropout → Softmax.
- Trained with **Adam optimizer**, **early stopping**, and **LR reduction**.
- Achieved good accuracy and generated a detailed classification report.

3. Demasking & API Output

1) Demasking Process:

- After classification, the system performs a demasking step.
- All previously masked placeholders (like [full_name], [email], [phone_number], etc.) are replaced with their original values.
- These original values were securely stored before classification.

2)Purpose:

- Ensures classification is done with PII securely hidden.
- Restores full and accurate email content after classification is complete.

3)Final API Output Includes:

When a user sends an email to the API, the response includes:

- input_email_body: The original email content submitted.
- list_of_masked_entities: A list of detected sensitive entities like names, emails, phone numbers, etc. Each entity includes:
    - position: Start and end index of the entity in the text.
    - classification: The type of entity (e.g., PERSON, EMAIL).
    - entity: The actual value found before masking.
- masked_email: The email content with sensitive entities replaced/masked.
- category_of_the_email: The predicted class/category of the email

4)Importance:

- Balances data privacy and practical usability.
- Helps support teams act on classified emails **without losing important user details**.
- Test the API on Swagger: https://khushi2488-email-classifier-with-pii-masking.hf.space/docs
- Hugging Face Spaces : https://khushi2488-email-classifier-with-pii-masking.hf.space

Model Selection and Training

- **Goal**: To classify support emails into categories like Billing Issues, Technical Support, Account Management, etc.
- **Chosen Model**:
    - **Bidirectional LSTM (BiLSTM)** – This deep learning model reads the email from both directions (forward and backward), helping it understand the full context of the sentence better than a standard LSTM.
- **Libraries Used**:
    - TensorFlow and Keras for deep learning model building.
    - NLTK for text cleaning (stopwords).
    - Scikit-learn for encoding labels and evaluation metrics.

- **Text Preprocessing**:
  - Lowercased all text to maintain uniformity.
  - Removed punctuation using regular expressions.
  - Removed common English stopwords to reduce noise in the text.
- **Tokenization and Padding**:
  - Used Tokenizer to convert cleaned text into sequences of numbers.
  - Applied pad_sequences() to make all sequences the same length (128 tokens).
- **Target Label Encoding**:
  - Used LabelEncoder to convert category labels into numeric values.
  - Applied to_categorical() for one-hot encoding since we are doing multi-class classification.
- **Data Split**:
  - 80% for training and 20% for testing.
  - Used stratify=y to keep label distribution balanced in train/test sets.
- **Model Architecture**:
  - Embedding Layer: Maps each word to a dense vector of 128 dimensions.
  - Bidirectional LSTM: 128 units, learns context from both directions.
  - Dense Layer: Fully connected layer with ReLU activation.
  - Dropout Layer: 50% dropout to reduce overfitting.
  - Output Layer: Softmax activation to classify into multiple categories.
- **Model Compilation**:
  - Optimizer: Adam (learning rate = 0.0005).
  - Loss Function: categorical_crossentropy (used for multi-class classification).
  - Metrics: Accuracy.
- **Callbacks Used**:
  - EarlyStopping: Stops training if the model doesn't improve after some epochs.
  - ReduceLROnPlateau: Reduces learning rate when validation loss stagnates.
- **Model Training**:
  - Trained for up to 100 epochs with batch size 32.

- Used 10% of the training data for validation.

- **Model Evaluation**:
  - Accuracy on test data printed.
  - Used classification_report to get precision, recall, F1-score.
  - Confusion_matrix showed category-wise predictions.
- **Model Saving**:
  - Final trained model saved as optimized_email-fc-gru-model.h5 for deployment in the API.

Challenges Faced and Solutions Implemented

1. **Masking the Data**
   - **Challenge**: Masking PII like name, email, phone, card numbers, etc., was tricky due to varying formats in emails.
   - **Solution**:
     - Used **custom regular expressions** for each entity (e.g., email, dob, aadhar_num, etc.).
     - Replaced them with standardized tags like [email], [phone_number], etc.
     - Ensured no personal info was used during training for privacy and compliance.
2. **Choosing the Right Model (Class Imbalance)**
   - **Challenge**: The dataset had **imbalanced classes** – some categories had more emails than others, affecting model performance.
   - **Solution**:
     - Initially tried traditional ML models, but accuracy was low due to imbalance.
     - Shifted to **Bidirectional LSTM**, which handled context better.
     - Focused on improving performance using **categorical cross-entropy** and **label encoding** to fairly learn all classes.
3. **Overfitting**
   - **Challenge**: The model performed well on training but poorly on validation data (overfitting).

- **Solution**:
    - Used **Dropout layers** to prevent the model from memorizing training data.
    - Applied **EarlyStopping** to stop training when validation loss stopped improving.
    - Used **ReduceLROnPlateau** to lower the learning rate when stuck, improving generalization.

Conclusion

- This project achieved an effective **email classification system** for support teams.
- **PII data** was masked using **Regex-based methods** to ensure privacy.
- A **Bidirectional LSTM** model was trained, offering better accuracy over traditional ML models.
- To avoid overfitting, techniques like **dropout**, **early stopping**, and **learning rate reduction** were used.
- The solution was wrapped into an **API**, ready for real-world deployment.