Email Classification System with PII Masking

Technical Report

1. Introduction to the Problem Statement

In the digital age, enterprise support teams receive a high volume of emails addressing diverse issues such as billing inquiries, technical glitches, and account-related concerns. Efficient classification and handling of these emails is essential for optimizing response times and enhancing customer satisfaction. This project aims to develop and deploy an end-to-end email classification system that automatically categorizes support emails into predefined classes—such as *Incident*, *Request*, or *Problem*—leveraging traditional machine learning (e.g., SVM, Random Forest) or advanced deep learning models (e.g., LSTM, Transformers, BERT). A crucial requirement is the secure handling of sensitive data. Prior to classification, all personally identifiable information (PII) and payment card industry (PCI) data—including full_name, email, phone_number, dob, aadhar_num, credit_debit_no, cvv_no, and expiry_no—must be masked using deterministic, non-LLM approaches such as Named Entity Recognition (NER), regular expressions, or rule-based methods. After processing, the original data should be restored in the output. The final system is delivered as a RESTful API (using frameworks like FastAPI or Flask) that accepts raw email input, performs secure PII masking, classifies the content, and returns a structured JSON response. This facilitates seamless, privacy-preserving automation of support workflows in a scalable and production-ready environment.

PII detection system employs a comprehensive architecture with two main detection methods: regex pattern matching for structured data (names, emails, phone numbers, birthdates, Aadhar numbers, credit/debit cards, CVV codes, and expiry dates) and ML-based Named Entity Recognition using spaCy's "en_core_web_sm" model for contextual identification. The workflow begins with text normalization to standardize inputs, followed by sequential application of regex patterns and NER, with results mapped back to the original text through a sophisticated position mapping system that handles preprocessing transformations. The process includes deduplication to resolve overlapping entities, prioritizing regex matches over spaCy detections, and finally applying masking in reverse order to maintain correct indices. The system outputs both masked text and detailed entity information for eight specific PII categories, using exact field names like "full_name" and "credit_debit_no" while focusing on common contextual patterns for consistent detection across English text.

2. Challenges faced and solutions implemented –

During the model development phase, several key challenges were encountered, most notably in terms of computational efficiency, overfitting, and class imbalance. Initially, training the xlm-roberta-base model with 5 epochs and a batch size of 16 proved to be resource-intensive, requiring over two hours on a T4 GPU, thus hindering experimentation. To address this, mixed precision training (FP16) was combined with gradient accumulation, significantly reducing GPU memory usage and enabling training extension to 8 epochs without increasing runtime. However, this surfaced a critical overfitting issue—validation loss began to increase steadily beyond the fifth epoch, despite continued improvement in training performance. After systematic hyperparameter tuning, it was observed that the model reached peak generalization precisely at five epochs. To further mitigate overfitting, weight decay regularization was incorporated, effectively penalizing overly large weights and preventing memorization of noise. Additionally, all emails were masked for personally identifiable information (PII) before training, a strategic decision made to enhance model generalization, protect user privacy, and ensure consistency across training and inference. This allowed the model to learn from contextual patterns rather than overfitting to specific names or identifiers. The dataset's multilingual nature—comprising English, German, Spanish, French, Portuguese, Dutch, and

Italian—posed another layer of complexity, which was effectively managed by selecting the multilingual mdeberta-v3-base model for final deployment. Addressing class imbalance was also essential, given the skewed label distribution. To counter this, class weights were computed and applied through a custom WeightedTrainer, alongside label smoothing (set to 0.1) to reduce prediction bias toward majority classes. Together, these solutions formed a robust training strategy that not only delivered high performance and generalization but also ensured scalability and consistency in real-world deployment scenarios.

3. Approach Taken for PII Masking and Classification

To ensure secure and compliant handling of sensitive user information, the system employs a robust hybrid PII masking strategy combining **rule-based regular expressions** with **machine learning-based Named Entity Recognition (NER)**using spaCy. This dual-layered approach enables precise detection of a wide range of personally identifiable information (PII) embedded in natural language text, including structured and semi-structured formats commonly seen in support emails.

The first phase of processing involves **text normalization**—removing excessive whitespaces, unifying Unicode representations, and cleaning special characters—to create a standardized input stream. Following this, **regex-based entity extraction** is applied using carefully crafted patterns for fields such as full_name, email, phone_number, dob, aadhar_num, credit_debit_no, cvv_no, and expiry_no. These patterns are tuned to recognize variations in formatting, such as international phone numbers or diverse date styles.

To supplement regex, the pipeline integrates **spaCy's pre-trained en_core_web_sm model** for contextual PII extraction. This allows the detection of named entities like persons, organizations, dates, and numeric fields that may represent credit cards or identification numbers. Each identified entity is mapped back to its original position in the unprocessed text using a **custom character-position reconciliation algorithm**, ensuring accurate indexing during masking and later demasking.

The system also includes a **deduplication and overlap resolution mechanism**, prioritizing regex matches in cases where both methods detect the same segment. Once finalized, all detected entities are masked using their standardized labels (e.g., [full_name], [email]), and metadata—including classification labels, character positions, and original text—is returned in the required API schema. This approach balances rule precision with machine learning flexibility, delivering high recall and low false positive rates in real-world email streams.

4. Model Selection and Training Details

For the classification component of this system, I conducted a comprehensive benchmarking process to evaluate a variety of machine learning and transformer-based architectures. Initially, I explored traditional models such as **Logistic Regression** and **Multinomial Naive Bayes** due to their lightweight nature and interpretability. However, these models were inadequate for capturing the nuanced, context-dependent semantics inherent in customer support emails, particularly when dealing with noisy, informal, or multilingual data.

To address these limitations, I evaluated several state-of-the-art transformer models, including **xlm-roberta-base**and **mdeberta-v3-base**. After rigorous comparative analysis, I selected **mdeberta-v3-base** as the final classification model due to its superior generalization capabilities, stability during training, and enhanced performance across minority classes. As a multilingual variant of Microsoft's DeBERTa family, mdeberta-v3-base leverages a **disentangled attention mechanism** and **relative position encoding**, enabling it to deliver deeper contextual understanding than conventional BERT-like architectures. While xlm-roberta-base showed competitive inference efficiency, mdeberta-v3-base consistently outperformed it in F1-score and convergence stability, making it the most reliable option for production deployment.

The model training pipeline was meticulously crafted for both effectiveness and efficiency. Input emails were first normalized and masked for PII, followed by tokenization using the DebertaV2Tokenizer, with a maximum sequence length of 128 tokens to balance computational cost and input fidelity. To address class imbalance, **class weights were dynamically computed** from the training labels and integrated into a **custom WeightedTrainer** class, which applied these weights directly within the loss function.

Training was performed using the **AdamW optimizer** with a learning rate of 5e-5, a **cosine learning rate scheduler**, and **120 warm-up steps**. A batch size of 4 was used in conjunction with **gradient accumulation (steps = 4)** to achieve an effective batch size of 16. The model was fine-tuned for **5 epochs**, incorporating **early stopping** with a patience of 2 epochs based on the **weighted F1-score**, which was also used as the key metric for best model selection. Additional performance optimizations included **FP16 mixed precision training** for reduced memory consumption and improved throughput.

Throughout training, model checkpoints were saved after each epoch, with the best-performing model retained for deployment. The final fine-tuned mdeberta-v3-base model achieved outstanding performance:

- **Macro F1-score**: 79.54%
- **Accuracy**: 78.11%
- **Precision**: 78.55%
- **Recall**: 78.11%

These results demonstrate the model's ability to maintain high accuracy while ensuring balanced classification performance across all categories, including underrepresented ones. Its multilingual capacity, efficient runtime, and robustness in handling diverse support queries make mdeberta-v3-base highly suitable for deployment in real-time, latency-sensitive API environments.