Email Classification API - Technical Documentation

# 1. Overview

The Email Classification API is a scalable solution designed to classify financial emails into predefined request types and subtypes. It uses OpenAI's GPT model with fine-tuning capabilities, supports batch processing, and performs duplicate detection with MongoDB. The API handles various file formats, including email (EML), PDF, and DOCX, and returns structured JSON responses with request types, confidence scores, priority flags, and duplicate status.

# 2. Architecture

Components:  
- Flask API: Handles incoming requests, file uploads, and batch processing.  
- MongoDB: Detects duplicate emails by storing email hashes.  
- OpenAI GPT Model: Classifies emails based on fine-tuned financial request types.  
- Preprocessing Module: Cleans and extracts text from emails, PDFs, and DOCX files.  
- Batch Processing: Enables concurrent classification of multiple files.  
- Docker: Containerizes the application for easy deployment and scalability.

# 3. Key Features

- File Support: Accepts EML, PDF, and DOCX formats.  
- Preprocessing: Cleans and normalizes text.  
- Classification: Uses a fine-tuned OpenAI GPT model for classification.  
- Duplicate Detection: Uses MongoDB to flag duplicate emails.  
- Batch Processing: Handles multiple files simultaneously using multi-threading.  
- Scalability: Dockerized for horizontal scaling.

# 4. API Endpoints

## 4.1 Batch Classification Endpoint

Endpoint: /batch\_classify  
- Method: POST  
- Input: Multiple files (email, PDF, DOCX)  
- Output: JSON with classified request types, confidence scores, and duplicates.

## 4.2 Fine-Tuning Endpoint

Endpoint: /fine\_tune  
- Method: POST  
- Input: CSV dataset file  
- Output: Fine-tuning job ID

## 4.3 Fine-Tuning Status Endpoint

Endpoint: /fine\_tune\_status  
- Method: GET  
- Input: Job ID (query parameter)  
- Output: Fine-tuning status

# 5. Preprocessing Steps

- Text Extraction:  
 - For emails: Extracts plain text from the message body.  
 - For PDFs: Extracts text using pdfminer.  
 - For DOCX: Extracts paragraphs using python-docx.  
- Text Cleaning:  
 - Removes HTML tags and excess whitespace.  
 - Converts text to lowercase for normalization.  
- Entity Extraction:  
 - Uses spaCy for named entity recognition (NER).

# 6. Duplicate Detection

- MongoDB stores the hash of each email for fast lookup.  
- On email classification, the hash is checked against the database.  
- Duplicate emails are flagged with is\_duplicate: Yes.

# 7. Output Format

The API returns structured JSON with the following fields:  
- is\_duplicate: Flags if the email is a duplicate.  
- requests: Detected request types with confidence scores and priorities.  
- entities: Extracted named entities.

# 8. Scalability & Deployment

- Dockerized Application:  
 - Easily deployable with Docker Compose.  
 - Supports scaling by running multiple containers.  
- Batch Processing:  
 - Uses ThreadPoolExecutor for concurrent processing.

## 8.1 Additional Scalability Options

- Kubernetes (K8s): Deploying the API on Kubernetes clusters allows auto-scaling based on workload.  
- Load Balancing: Use NGINX or AWS Elastic Load Balancer (ELB) to distribute incoming requests evenly.  
- Horizontal Scaling: Spin up multiple instances of the application to handle larger email volumes concurrently.  
- Caching: Implement Redis or Memcached to cache frequent classification results, reducing the load on the model.  
- Queue System: Use RabbitMQ or Kafka for asynchronous batch processing of large volumes of emails.  
- Cloud Storage: Integrate with Amazon S3 or Google Cloud Storage to handle large file uploads efficiently.

# 9. Environment Variables

- OPENAI\_API\_KEY: OpenAI API key for classification and fine-tuning.  
- MONGO\_URI: MongoDB connection string.

# 10. Future Enhancements

- Support for more file formats (e.g., CSV, TXT).  
- Integration with cloud storage services.  
- Improved fine-tuning dataset generation pipeline.

# 11. Conclusion

The Email Classification API offers a robust and scalable solution for financial email classification with batch processing, duplicate detection, and fine-tuning capabilities. It is optimized for deployment with Docker and MongoDB, ensuring high performance and efficiency. The API can be further scaled using Kubernetes, load balancing, and caching strategies to handle larger workloads efficiently.