GenAI Email Classifier with OCR -Developer Guide

Version: 1.0

Date: March 27, 2025

Prepared by: [**Elite Eagles]**

Purpose: To demonstrate the functionality of the Email Classifier with OCR and its new web-based UI for classifying email files and generating service intake requests and routing to the appropriate teams and individuals based on skills and roles

---

Table of Contents

---

1. Introduction

The **Email Classifier with OCR** is a Python-based tool designed to process email files (.eml, .pdf, .docx) by extracting text, classifying content, and generating service intake requests for financial institutions. The latest enhancement includes Optical Character Recognition (OCR) for image-based documents and a Flask-based web UI for a professional, user-friendly experience.

Key Features

- Supports .eml, .pdf, and .docx files.
- Extracts text and metadata using pdfminer and python-docx.
- Applies OCR with pytesseract for image-based PDFs and embedded images.
- Classifies emails using Google Gemini API.
- Generates structured service intake requests.
- Web UI with modern, responsive design.

---

2. System Requirements

Hardware

- **OS:** Windows 11 (or compatible OS)
- **RAM:** 8 GB minimum (16 GB recommended)
- **Storage:** 2 GB free space

Software

- **Python:** 3.12.6 or higher
- **Dependencies:**
    - google-generativeai
    - pdfminer.six
    - python-docx
    - pytesseract
    - pdf2image
    - Pillow
    - flask
- **External Tools:**
    - **Tesseract OCR:** Installed at C:\Program Files\Tesseract-OCR
    - **poppler:** Installed at C:\poppler with bin added to PATH

---

3. Setup Instructions

Step 1: Clone or Create Project Directory

- Create a directory:
  C:\Syed\Workspace\Hackathon_2025\Email_Classification_Project

Step 2: Set Up Virtual Environment

Cmd prompt

```
cd C:\Syed\Workspace\Hackathon_2025\Email_Classification_Project
python -m venv my_env
my_env\Scripts\activate
```

Step 3: Install Dependencies

Cmd prompt

```
pip install google-generativeai
pip install requests
pip install pandas
pip install matplotlib
pip install requests
pip install transformers
pip install torch PyPDF2 pytesseract pdf2image
pip install openai
pip install spacy
```

Step 4: Install Tesseract OCR

- Download from Tesseract GitHub.
- Install to C:\Program Files\Tesseract-OCR.
- Add to PATH: set PATH=%PATH%;C:\Program Files\Tesseract-OCR.

Step 5: Install poppler

- Download from poppler binaries.
- Extract to C:\poppler.
- Add to PATH: set PATH=%PATH%;C:\poppler\bin.

---

4. Demo Preparation

Sample Files

Prepare the following test files:

1. **Text-based EMl FILE :** main_email.eml (e.g., a loan repayment email).
2. **Text-based PDF:** scanned_email.pdf (screenshot of an email saved as PDF).
3. **Text-based DOCX:** scanned_email.pdf (screenshot of an email saved as PDF).
4. **DOCX/PDF with Embedded Image:** email_with_image.docx (Word doc or PDF with an email screenshot).

Directory Structure

```
Email_Classification_Project/
|___backend.py
├──genai_email_classifaction_engine.py
├── genai_email_classifier.py
├── email_classification.json (output)
└── service_intake_request.json (output)
```

---

5. Running the Demo

Step 1: Launch the Application

cmd

```
cd C:\Syed\Workspace\Hackathon_2025\Email_Classification_Project
my_env\Scripts\activate
python email_classifier_with_ocr_flask.py
```

- Output: Running on http://0.0.0.0:5000 (Press CTRL+C to quit)

Step 2: Access the Web UI

- Open a browser and navigate to http://localhost:5000.
- You'll see the **Email Classifier** UI with an upload form.

Step 3: Upload a File

- Click **Choose File**, select a sample file (e.g., main_email.eml), and click **Classify**.
- Wait for the “Processing…” message to disappear, then view the results.

---

6. Sample Scenarios

Scenario 1: Text-Based .eml

- **File:** main_email.eml
- **Content:** Loan repayment notification from Bank of America.
- **Action:** Upload and classify.

Scenario 2: Image-Based PDF

- **File:** scanned_email.pdf
- **Content:** Scanned image of the same email.
- **Action:** Upload to test OCR extraction.

Scenario 3: DOCX with Embedded Image

- **File:** email_with_image.docx
- **Content:** Word document with an email screenshot.
- **Action:** Upload to test OCR on embedded images.

---

7. Expected Outputs

Scenario 1: Text-Based .eml

Web UI Display

- **Email Classification:**
  - **Request Type:** Money Movement-Inbound
  - **Sub-Request Type:** Principal
  - **Confidence:** 0.95
  - **Metadata:** From: Bank of America notifications@bofa.com (mailto:notifications@bofa.com), To: Wells Fargo…
  - **Signature:** Regards, John Doe

    - **Attributes:** Loan Repayment Date: 10-Nov-2023, Repayment Amount: USD 10,000,000.00...
- **Service Intake Request:**
    - **Request Type:** Money Movement-Inbound
    - **Sub-Request Type:** Principal
    - **Assigned To:** Loan Processing Team
    - **Summary:** Loan repayment of USD 10,000,000.00 for deal CUSIP 123456789 due 10-Nov-2023
    - **Priority:** High
    - **Description:** Request to process a loan repayment...
    - **Details:** Sender: Bank of America...

Saved Files

- `email_classification.json` and `service_intake_request.json` with the same data.

Scenario 2: Text-Based PDF/Docx

- Similar output, but metadata limited to the text resides in the PCF orf docx document is extracted

Scenario 3: DOCX with Embedded Image

- Similar output, with OCR-extracted text from the embedded image.

---

8. Troubleshooting

Issue: “No module named X”

- **Solution:** Ensure all dependencies are installed (`pip install X`).

Issue: “Tesseract is not installed”

- **Solution:** Verify Tesseract path and PATH variable.

Issue: Blank Results

- **Solution:** Check file format compatibility and image quality for OCR.

Issue: Server Not Starting

- **Solution:** Ensure port 8000 is free; stop other running Flask apps.

---

9. Conclusion

This demo showcases the **Email Classifier with OCR**’s ability to process diverse file types, extract text (via text extraction or OCR), classify emails, and generate service requests through a modern web UI. The tool is robust, user-friendly, and ready for financial institution workflows.

Next Steps:

- Test with additional file types and edge cases.
- Enhance UI with download buttons for JSON files.
- Deploy to a production server if needed.