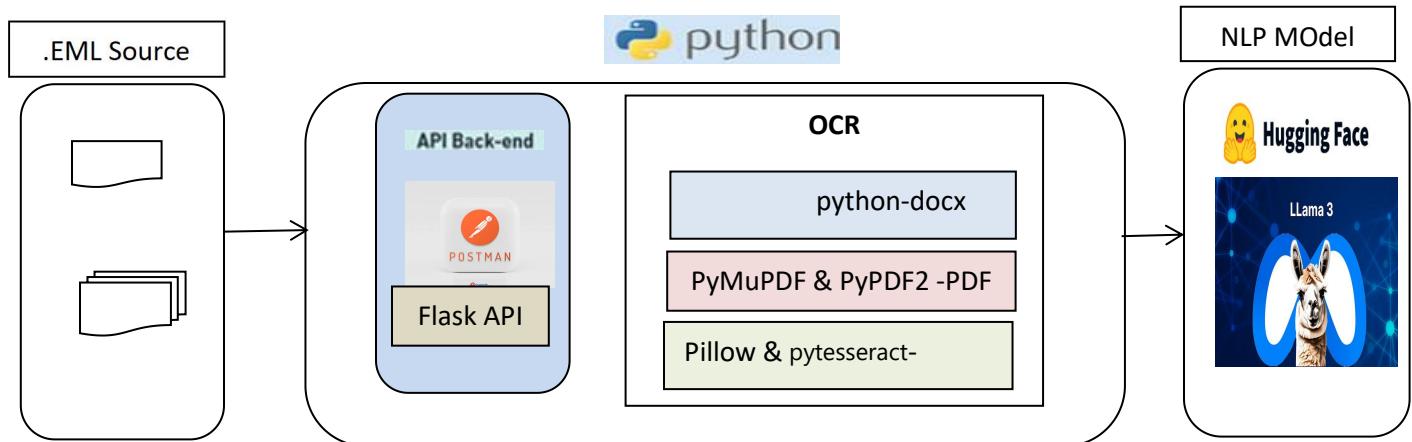


Technology Architecture



Flask API : Used to Processing .eml Files and Batch File Processing

This Flask API provides two endpoints:

- 1 Single .eml File Processing: Accepts an .eml file as input, extracts attachments, and processes them based on file type.

End Point : /process/file - Single .eml File Processing

- 2 Batch Processing from a Folder: Allows users to select a folder path and process multiple files at once.

End Point 2: /process/path - Batch Processing

Method: POST

Description:

Accepts a single .eml file as input or .eml files from folder path

Extracts attachments from the email if any .docx, .pdf, jpeg, and .gif files.

Returns extracted text from the attachments as well email body.



The screenshot shows a POST request in Postman. The URL is `http://127.0.0.1:5000/process/file?`. The 'Body' tab is selected, showing a single file named 'AU Transfer Request.eml' attached. The file type is listed as 'File'. The 'Send' button is visible at the top right.

Why Use OCR(Optical Character Recognition) Approach?

1. Handles Multiple File Formats (.docx, .pdf, .jpeg, .gif)
2. Uses Optimal Libraries (PyMuPDF for PDFs, pytesseract for images)
3. Supports OCR for Image-based Text Extraction
4. Stores Data in a Structured Array for Further Processing to LLM Model

This script **parses a .eml email file**, extracts **attachments**, and reads their content based on their file type: The extracted content is stored in an **array of dictionaries** for further processing.

Use Cases :

1. **Automated Data Entry**
2. **Extracting Text from Files & Handwritten Notes**
3. **Processing PDF , JPEG , Docx Documents (OCR for text extraction)**

Limitations:

1. **Handwriting Recognition** → Lower accuracy for messy handwriting
2. **Complex Backgrounds** → May cause OCR errors
3. **Low-Resolution Images** → Blurry images reduce accuracy
4. **Languages & Fonts** → Some languages require additional training

LLaMA 3 Model :

Meta is working on **next-gen Llama models**

Expected to be **larger, more powerful, and efficient**

Focused on **better reasoning, coding, and multimodal capabilities**

Key Features of LLaMA Models

- ✓ **Optimized for Efficiency:** Smaller models with performance comparable to larger AI models
- ✓ **Open-Source:** LLaMA 2 is freely available for commercial and research use
- ✓ **Fine-Tuning Support:** Customization for chatbots, coding, and other NLP applications
- ✓ **Transformer-Based Architecture:** Uses an improved version of **decoder-only transformers**

OutPut From .Eml File:

```
{  
  "request_type": "***Summary:** The purpose of this document is to request an AU Transfer for account"  
}
```

Challenges :

Achievements:

1. Successfully implemented reading of **single and batch .eml files** as input.
 2. Extracted and processed **attachments** from .eml files, including .jpeg, .docx, and other formats.
 3. Developed a **model that analyzes request types** and provides structured output.
 4. Implemented **content extraction** from email bodies, images (.jpeg), and documents .docx).
-

Challenges & Next Steps:

1. **Integrating an Offline LLM Model** – Downloading and running an **offline LLM model** from **Hugging Face** efficiently.
2. **Generating Quality Test Data** – Creating **diverse and high-quality datasets** for model training and validation
3. **Expanding Use Case Coverage** – Enhancing functionality to support **broader scenarios** and meet all requirements.