

# CODE EXPLANATION :

Our application is designed to extract and validate invoice data from PDF files using a combination of PDF processing, Optical Character Recognition (OCR), and the OpenAI GPT-4 API. The application leverages the Streamlit framework to create an interactive web interface, enabling users to upload PDF invoices and obtain structured data in formats like Excel or CSV. Below is a detailed breakdown of the code, section by section:

## 1. Imports and Dependencies

### Explanation:

#### - Standard Libraries:

- `os` : Interacts with the operating system, primarily for environment variables.
- `json` : Handles JSON data parsing and serialization.
- `re` : Utilizes regular expressions for pattern matching and data validation.
- `logging` : Facilitates logging of events, errors, and informational messages.
- `time` : Manages time-related functions, such as delays.

#### - Third-Party Libraries:

- `requests` : Performs HTTP requests to interact with APIs.
- `pandas as pd` : Manages and manipulates structured data using DataFrames.
- `pypdf.PdfReader` : Reads and extracts text from PDF files.
- `pdf2image.convert_from_bytes` : Converts PDF pages to images for OCR processing.
- `pytesseract` : Performs OCR on images to extract text.
- `streamlit as st` : Creates an interactive web application interface.
- `dotenv.load_dotenv` : Loads environment variables from a ``.env`` file.
- `PIL.Image` : Handles image processing.

- `io.BytesIO` : Manages binary data in memory, useful for handling file uploads.

## 2. Configuration and Setup

### Logging Configuration

**Purpose:** Initializes logging to record events, errors, and informational messages to a file named `invoice_extraction.log`. The log level is set to `INFO`, capturing both informational and error messages with timestamps.

### Environment Variables

**Purpose:**

**`load_dotenv()`** : Loads environment variables from a `.env` file, ensuring sensitive information like API keys is not hard-coded.

**`GPT4V_KEY` & `GPT4V_ENDPOINT`** :Retrieves the API key and endpoint URL for the GPT-4 API.

**Error Handling:** If either the API key or endpoint is missing, the application displays an error message using Streamlit and halts execution with `st.stop()`.

### API Headers Configuration

**Purpose:** Sets up the HTTP headers required for making requests to the GPT-4 API, specifying the content type as JSON and including the API key for authentication.

## 3. Helper Functions

a. `get_pdf_text(pdf_doc)`

- **Purpose:** Extracts text from a PDF file, handling both regular text-based PDFs and scanned/image-based PDFs using OCR.

- **Process:**

1. **Initialize Empty Text:** Starts with an empty string to accumulate extracted text.

2. **PDF Reading:**

- Uses `PdfReader` to read the PDF and iterate through each page.
- Attempts to extract text directly using `page.extract_text()`.
- If the extracted text is substantial (more than 50 characters), it's appended to the `text` variable.

3. **OCR Handling:**

- If direct text extraction is insufficient, the function applies OCR:
- Converts the specific PDF page to an image using `convert_from_bytes`.
- Utilizes `pytesseract` to perform OCR on the image.
- Appends OCR-extracted text if it's meaningful (more than 10 characters).

4. **Error Handling:** Logs and displays errors if text extraction fails.

- **Returns:** A single string containing all the extracted text from the PDF.

**b. call\_openai\_api(pages\_data)**

- **Purpose:** Sends the extracted text from the PDF to the GPT-4 API to extract specific invoice-related fields in JSON format.

- **Process:**

1. **Prompt Construction:**

- Defines a template prompt requesting extraction of specific invoice fields.
- Emphasizes the need for the output to be strictly in valid JSON without additional text.

2. **API Request Payload:**

- Includes the system and user messages.

- Sets parameters like ``max_tokens`` for response length and ``temperature`` for response variability.

### **3. API Call:**

- Makes a POST request to the GPT-4 API endpoint with the constructed payload.
- Handles successful responses by extracting the relevant content.
- Implements retry logic with a delay if the rate limit (``429``) is exceeded.

**4. Error Handling:** Logs and displays errors for failed API calls or exceptions.

**- Returns:** The raw extracted JSON data as a string if successful; otherwise, ``None``.

c. ``validate_data(field, value)``

**- Purpose:** Validates each extracted field using regular expressions to ensure data integrity and assigns confidence levels based on the validation.

### **- Process:**

**1. Pattern Definitions:** Defines regex patterns for specific fields like Invoice Number, Quantity, Date, Email, GSTINs, etc.

#### **2. Validation:**

- If the field has a defined pattern, it checks if the value matches the pattern.

- Assigns "High Confidence" if it matches, "Low Confidence" otherwise.

- For fields without specific patterns, it performs a basic non-empty check, assigning "Medium Confidence" or "Low Confidence" accordingly.

**3. Returns:** A tuple indicating whether the field is valid (``bool``) and the corresponding confidence level (``str``).

d. ``extract_json(raw_text)``

**- Purpose:** Extracts the first JSON object found within a raw text string.

- **Process:**

1. **Regex Pattern:** Uses a regex pattern ``\{.*\}`` to match the first occurrence of a JSON-like structure.

2. **Extraction:**

- Searches the raw text for the pattern.
- If found, returns the matched JSON string.
- Otherwise, returns ``None``.

- **Returns:** The extracted JSON string if found; otherwise, ``None``.

#### 4. Core Function to Process PDF Files

```
create_docs(user_pdf_list)
```

- **Purpose:** Processes a list of uploaded PDF files to extract, validate, and compile invoice data into a structured Pandas DataFrame.

- **Process:**

1. **Initialize DataFrame:**

- Creates an empty DataFrame with columns for all expected invoice fields, along with additional columns for "Confidence" and "Trust".

2. **Metrics Tracking:**

- Initializes a metrics dictionary to track total files processed, successful extractions, and per-field accuracy.

3. **Iterate Over Uploaded PDFs:**

- For each PDF:
  - Increments the total files counter.
  - Displays processing status in Streamlit.
  - Extracts raw text using ``get_pdf_text``.
  - If no text is extracted, logs a warning and skips the file.
  - Displays extracted text in an expandable section for debugging.
  - Calls ``call_openai_api`` to extract invoice data.
  - If API call fails, logs an error and skips the file.
  - Logs and displays raw extracted data for debugging.
  - Extracts JSON from the API response using ``extract_json``.

- If JSON extraction fails, logs an error and skips the file.
- Parses the JSON into a dictionary.
- Validates each field using ``validate_data``, assigns confidence and trust levels.
- Updates metrics based on validation results.
- Appends the validated data to the DataFrame.
- Increments the successful extractions counter.
- Logs and displays success status.

#### **4. Calculate Accuracy Rates:**

- Computes per-field accuracy percentages based on the metrics collected.

#### **5. Save DataFrame to Excel:**

- Attempts to save the DataFrame as ``extracted_invoice_data.xlsx``.
- Logs success or errors accordingly.

#### **6. Display Metrics in Streamlit:**

- Shows total files processed and successful extractions.
- Displays per-field accuracy rates in a styled DataFrame.

#### **7. Provide Download Options:**

- Allows users to download the extracted data as both Excel and CSV files via Streamlit download buttons.

#### **8. Display Trust Assessment Summary:**

- Shows counts of "Trusted" and "Untrusted" data points based on validation.


**- Returns:** The final DataFrame containing all extracted and validated invoice data.

### **5. Streamlit Application Interface**

**- Purpose:** Serves as the entry point for the Streamlit web application, managing the user interface and orchestrating the data extraction process.

**- Process:**

#### **1. Page Configuration:**

- Sets the page title to " Invoice Extraction Bot" and configures the layout to be wide.

## **2. Title and Subheader:**

- Displays the main title and a subheader describing the application's purpose.

## **3. File Uploader:**

- Provides a file uploader widget allowing users to upload multiple PDF files.
- Accepts files with a `.pdf` extension and supports regular, scanned, and mixed PDFs.

## **4. Extract Data Button:**

- When clicked, triggers the data extraction process if files are uploaded.
- Displays a loading spinner while processing.
- Calls `create_docs` with the uploaded PDF files.
- If extraction is successful and data is present:
  - Displays the extracted data in a Streamlit DataFrame.
  - Shows a summary of trust assessments.
  - Indicates completion with a success message.
- Handles cases where no data is extracted or no files are uploaded by displaying appropriate warnings or errors.


## **6. Run the Application**

- **Purpose:** Ensures that the `main()` function runs only when the script is executed directly, not when imported as a module. This is standard practice in Python to prevent unintended execution of code when modules are imported elsewhere.

## **Overall Workflow**

### **1. User Interaction:**

- Users access the Streamlit web interface and upload one or multiple PDF invoice files.

- Upon clicking the " Extract Data" button, the application begins processing.

## **2. PDF Processing:**

- For each uploaded PDF:
  - The application attempts to extract text using `PdfReader`.
  - If direct text extraction is insufficient, OCR is applied to extract text from images.

## **3. Data Extraction via GPT-4 API:**

- The extracted text is sent to the GPT-4 API with a prompt requesting specific invoice fields.
- The API returns a JSON-formatted response containing the extracted data.

## **4. Data Validation and Confidence Assessment:**

- Each extracted field is validated using predefined regex patterns.
- Confidence levels are assigned based on the validation results.
- Trust assessments determine whether the data points are considered reliable.

## **5. Compilation and Output:**

- Validated data is compiled into a Pandas DataFrame.
- Metrics on extraction performance and field accuracy are calculated and displayed.
- Users can download the extracted data in Excel or CSV formats.

## **6. Logging and Error Handling:**

- All significant events, successes, and errors are logged to `invoice\_extraction.log`.
- Users are informed of any issues via the Streamlit interface, ensuring transparency and ease of troubleshooting.

## **Key Features and Considerations**

### **- Robust Text Extraction:**

- Combines direct text extraction with OCR to handle both digital and scanned PDFs, ensuring comprehensive data capture.



**- API Interaction:**

- Utilizes the GPT-4 API to intelligently extract structured data from unstructured text, leveraging advanced natural language processing capabilities.

**- Data Validation:**

- Employs regex-based validation to ensure the accuracy and integrity of extracted fields, enhancing reliability.

**- User-Friendly Interface:**

- Streamlit provides an intuitive and interactive web interface, making the application accessible to users without technical expertise.

**- Error Handling and Logging:**

- Comprehensive error handling ensures that users are informed of issues, while logging facilitates debugging and monitoring.

**- Performance Metrics:**

- Tracks and displays extraction performance and field accuracy, providing insights into the system's effectiveness.

**- Download Options:**

- Offers flexibility by allowing users to download extracted data in multiple formats, catering to different workflow needs.

## **Potential Enhancements**

While the application is already feature-rich, here are some suggestions for further improvements:

**1. Multi-Language Support:**

- Extend OCR and data extraction capabilities to handle invoices in multiple languages.

**2. Enhanced Validation:**

- Incorporate more sophisticated validation techniques, such as cross-referencing with known databases or using machine learning models for anomaly detection.

### **3. User Authentication:**

- Implement user authentication to secure access to the application and manage user-specific data.

### **4. Scalability:**

- Optimize the application for handling large volumes of PDFs, possibly by integrating with cloud services or leveraging asynchronous processing.

### **5. Feedback Mechanism:**

- Allow users to provide feedback on extraction accuracy, facilitating continuous improvement of the system.

### **6. Visualization:**

- Add data visualization features to provide graphical insights into the extracted data, such as charts or dashboards.

## **Conclusion**

This Python script represents a robust solution for automating the extraction and validation of invoice data from PDFs. By integrating PDF processing, OCR, advanced language models, and a user-friendly interface, it streamlines the workflow for businesses and individuals dealing with large volumes of invoices. The structured approach to configuration, modular helper functions, and comprehensive error handling make the application both effective and maintainable.