Unstructured to Structured Data

Heat Exchanger Data Extraction:

Documentation: PDF Processing and Interaction Script

Overview

This application module is designed to converse on PDF documents, enabling functionalities such as text extraction, image processing, interaction with document content, and generating JSON data and schemas. It is built with Python and utilises several libraries, including PyPDF2, PyMuPDF (fitz), NumPy, PIL, requests, JSON, Jsonpath_ng and Gradio.

The application aims to serve both programmers and non-programmers by providing an interactive interface for uploading PDFs, extracting data, and querying the document based on a generated schema and information on the PDF document.

Key Features

- **Text Extraction:** Extracts raw text and structural information from PDF documents.
- **Image Processing:** Converts PDF pages to PIL images, optionally cropping whitespace.
- **Base64 Encoding:** Encodes images to base64 format for web transmission.
- **API Integration:** This uses the OpenAI API to generate structured data from images and text. The API will allow you to access the OpenAI selective models, like GPT4.
- **JSON Handling:** This function generates JSON data from text and images, creates JSON schemas for data validation, and reads JSON files. This will help the AI make queries and search faster without reading from the document.
- **Gradio Interface:** A user-friendly interface for uploading PDFs and querying the document content. This is primarily used to demonstrate the MVP of the solution, which allows the user to use the data extraction application.

How It Works

The script is structured into several functions, each handling a specific part of the process:

1. **Environment Setup:** Loads API keys from a **.env** file to secure access to external services.

2. **Text Extraction (get_page_dict):** The function uses the ExtractDict that extracts text and structure information from the PDF using the pymupdf library.
3. **Image Processing (pdf_to_pil_array, pil_to_base64, crop_whitespace):** Converts PDF pages to cropped PIL images and encodes them in base64 format.
4. **Data Generation and Schema Creation (jsonFromLLM, generateSchema):** This function uses the OpenAI API to generate structured JSON data from the extracted text and images and create a JSON schema for the data if it is not yet present/provided.
5. **JSON Handling (readJsonFile):** Reads and validates JSON data from files.
6. **Query Engine (ExtendedJSONQueryEngine):** Extends the llama index JSON to log queries and responses, facilitating interaction with the document content.
7. **Gradio Interface:** Provides an interactive web interface for uploading PDF documents, displaying and highlighting extracted data, and enabling users to query the document based on its content.

Implementation Details

- **API Integration:** The script integrates with the OpenAI API, which requires an API key to access language model services.
- **Security:** Sensitive information, such as the API key, is stored in an environment file (**.env**) and loaded securely.
- **Error Handling:** Includes basic error handling for file operations and API requests, ensuring graceful failure and user feedback.
- **User Interaction:** The Gradio interface allows users to upload PDFs, view extracted data and schemas, and interact with the document content using natural language queries.

Usage

To use the script, users need to:

1. Install the required Python libraries listed at the beginning of the script or in the "requirement.txt" that comes with the cloned repo—using "pip install -r requirement.txt."
2. Obtain an API key from OpenAI and store it in the **.env** file, which should be in the same location as the demo.py file.
3. Run the script in an environment where Python and the necessary libraries are installed. Using "python demo.py."

4. Access the Gradio web interface via the provided URL after launching the script or copy and paste the IP address into the browser.

Conclusion

This script showcases the integration of a document analyzer and interacts with PDF documents in a structured and user-friendly manner. It demonstrates the potential of document specification processing with machine learning models to generate actionable insights from static documents, making it a valuable tool for both technical and non-technical users.