Exercise 5

Introduction

The system is designed to process PDF files and respond to user queries using the content of the uploaded file as its context. At any given time, only one file is considered active until a new file is uploaded. The file's embeddings are stored in ChromaDB, and both the embedding model and the large language model (LLM) are sourced from OpenAI. The implementation leverages LangChain to construct the overall workflow and manage the query-response process efficiently.

Code Steps.

Step 1: Environment Variables

We configure the OpenAI API key as an environment variable stored in a .env file. This ensures the key remains secure and is not exposed when sharing the code.

Step 2: Initializing Variables

In this step, we initialize key components:

- Embeddings: We use OpenAiEmbeddings to generate vector representations of the text.
- Vector Store: The embeddings are stored in ChromaDB for efficient retrieval.
- Text Splitter: This component is explained in the next section.
- LLM: The large language model utilized is gpt-3.5-turbo, wrapped inside LangChain's ChatOpenAI. A key setting here is configuring the temperature to 0 to ensure deterministic responses, minimizing randomness.

Step 3: Document Processing

Storing documents as embedded vectors ensures efficient real-time querying. The process involves:

1. Loading Documents: PDFs are loaded using PyPDF.
2. Splitting Text: Documents are divided into smaller chunks using LangChain's recursive text splitter, which splits the content hierarchically until the chunks are of manageable size. This splitter requires two parameters:
   - Chunk Size: Determines the size of each split.
   - Chunk Overlap: Defines how much text overlaps between the end of the nth chunk and the start of the $(n+1)^{th}$ chunk.

Step 4: Querying

After processing, queries can be performed using the embeddings stored in the database. The LangChain RetrievalQA module handles the querying process with the following configurations:

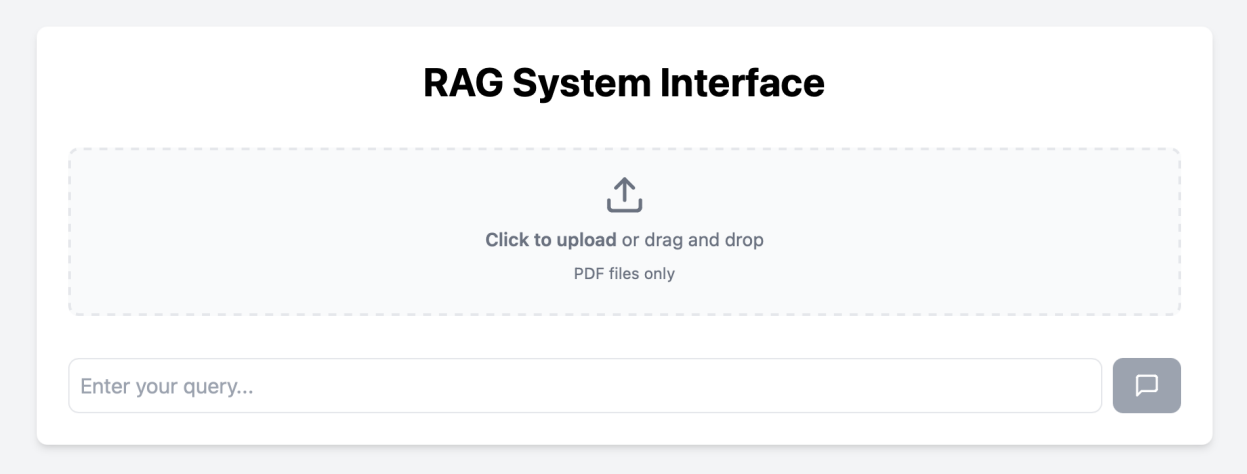
- LLM: The selected large language model processes the query.
- Chain Type: Set to "stuff", meaning retrieved documents are concatenated into a single input string for the LLM. This approach works well for simple documents as the model processes all relevant information simultaneously.

- Document Retrieval: The retriever fetches three documents, although only the final answer is displayed to the user. The retrieved documents are retained for logging purposes.

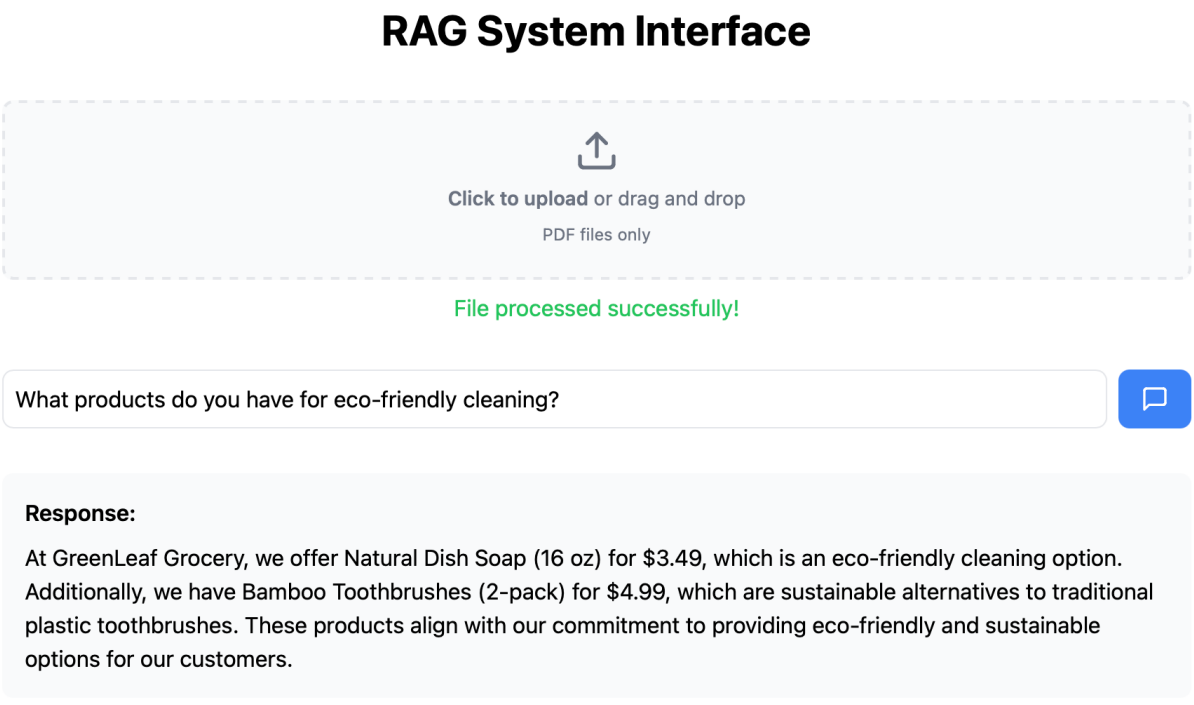
The project also includes Flask and React code for the web interface, which is beyond the scope of this explanation.

Demonstration.

To run the system, execute the app.py file. This file integrates the functionality defined in the rag_system.py file, which contains the code described earlier. Once the application is running, navigate to localhost:8080 in your web browser. You will be greeted with a user interface that allows you to upload a PDF file. For demonstration purposes, I will upload an AI-generated PDF about a grocery store, which is available in the data directory. Upon successful upload, the system will display a “File uploaded successfully” message. At this point, the system is ready to process user queries based on the uploaded document.



Query 1



Query 2

Do you guys provide any discounts?

Response:

Yes, we provide discounts on various products. Some of the current discounts include 15% off all organic coffee beans, 10% off all dairy products when you buy three or more items, and seasonal specials like 20% off all pumpkins and squash during the Fall Pumpkin Festival.

Query 3

I am in the mood to eat 5 pounds of peanut butter, how much will it cost me?

Response:

If 1 lb of Organic Peanut Butter costs $3.49, then 5 lbs of Organic Peanut Butter would cost $17.45.

Query 4

What are your store hours and where are you located

Response:

Our store hours are as follows:
- Monday - Friday: 8:00 AM to 8:00 PM
- Saturday: 9:00 AM to 6:00 PM
- Sunday: 10:00 AM to 4:00 PM

We are conveniently located in downtown Oakridge at 123 GreenLeaf Street, Oakridge, CA 95001.

Query 5

Explain theory of relativity

Response:

I'm here to help with information related to the products and deals offered at our store. I don't have the specific details on the theory of relativity. If you have any questions about our products or promotions, feel free to ask!

Observation on Results

The system's performance could be improved by increasing the chunk size, allowing the model to process more context within a single chunk. Below is an analysis of the responses to user queries:

1. First Response: The system provided an accurate and relevant answer.
2. Second Response: While the response was correct, it lacked key details such as the discounts on baking products and an explanation of BOGO (Buy One, Get One) deals or the 5% discount offered for using reusable bags. Including these details would have made the response more comprehensive.
3. Third Response: The system's answer was precise and met the query's expectations.
4. Fourth Response: Similarly, this response was clear and effective, addressing the query fully.
5. Fifth Response: This query was designed to test the system's boundaries by asking it to provide insights unrelated to the grocery store context. The model appropriately declined the request, maintaining the integrity of its intended functionality.

These results demonstrate the system's capability to handle user queries effectively while highlighting areas where adjustments, such as modifying chunk size, could enhance performance.