Technical Documentation for EduGPT Chatbot

1. Introduction

The project's main objective is to develop an interactive chatbot for students and teachers. This chatbot uses cutting-edge technologies such as Generative AI, OpenAI GPT models, langchain, and vector databases to answer questions based on textbook PDF files, thereby facilitating efficient learning.x

Access the application using the below link :
https://appworkingpy-myyaqbzkhjehxpxslu8cw2.streamlit.app/

2. Key Components and Libraries

- Streamlit: Used for creating a web-based interface for the chatbot.
- LangChain Framework: A suite of tools that allows for the creation of conversational chains, embedding generation, text splitting, and more.
- OpenAI: Utilized for harnessing the power of the OpenAI models.

3. Code Structure and Flow

3.1. Initial Setup:

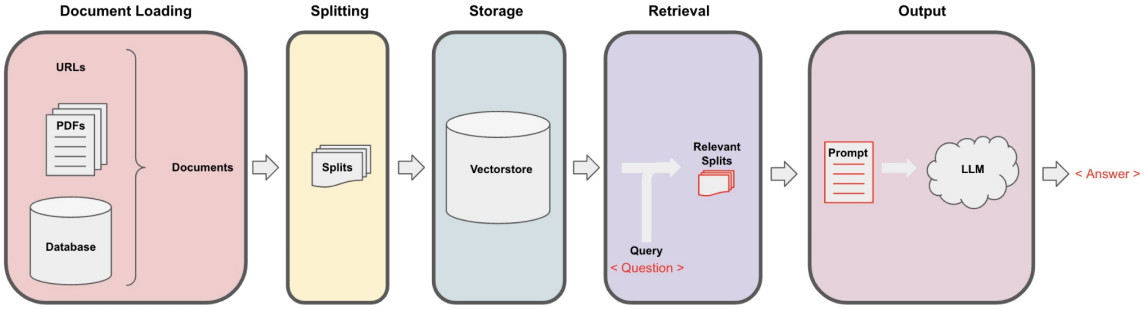
- Setting up the Streamlit page configuration.
- Initializing session states which include past interactions, user input, and other session-specific data.
- Setting up the OpenAI API key.

3.2. Code Structure/ Pipeline :

The pipeline for converting raw unstructured data into a QA chain looks like this:

1. Loading: First we need to load our data. Use the UnstructuredPDFLoader to load the pdf file.
2. Splitting: Text splitters break Documents into splits of specified size
3. Storage: Storage (e.g., often a vectorstore) will house and often embed the splits
4. Retrieval: The app retrieves splits from storage (e.g., often with similar embeddings to the input question)

5. Generation: An LLM produces an answer using a prompt that includes the question and the retrieved data



4. Key Functionalities

4.1. PDF Mapping:

- A mapping structure is provided for different PDF files. This mapping helps users to select a PDF and the corresponding file is used as the source for answering queries.

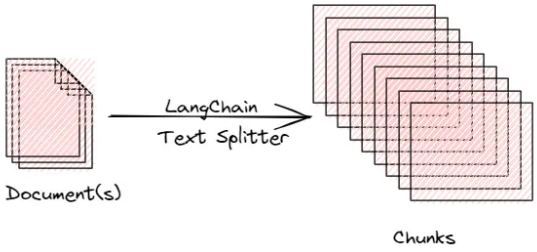
4.2. Chatbot Interface:

- Users can choose the retrieval method they want to use for generating responses.
- Users can input their query, and the chatbot will generate a response using the selected retrieval method and display it.
- The source of the information is also provided in a dedicated 'Sources' section.
- The response from the chatbot is displayed character by character, simulating a typing effect.

5. Technical Stack Used

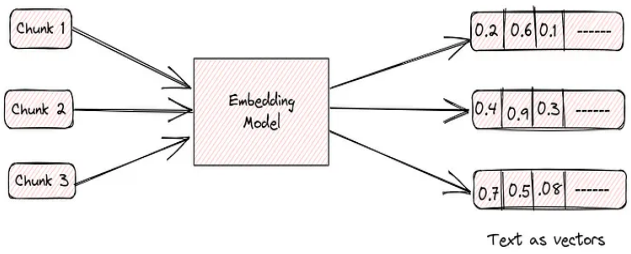
5.1. UnstructuredPDFLoader: This function is responsible for reading the PDF using the UnstructuredPDFLoader class from the LangChain framework. It returns the loaded documents from the provided file path.

5.2. CharacterTextSplitter : This splits based on characters (by default "\n\n") and measures chunk length by number of characters. Chunk overlap determines the number of chunks to be overlapped to retain the context of the paragraph.



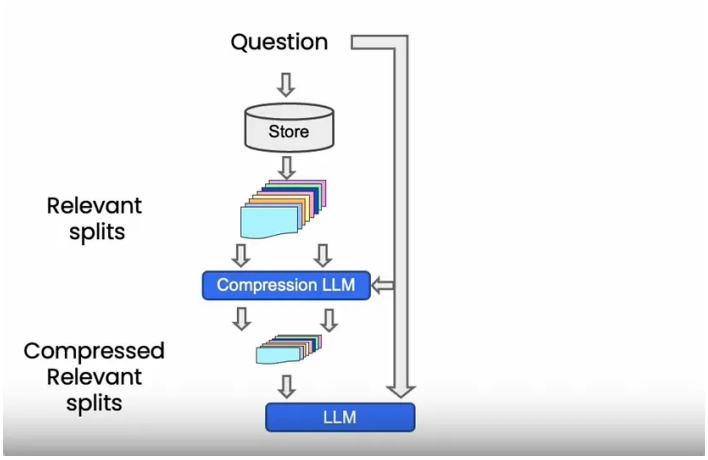
5.3. OpenAIEmbeddings() : OpenAI embedding models.We will be using the embeddings model provided by OpenAI.

5.4. FAISS : Facebook AI Similarity Search has high-dimensional indexing capabilities and fast search performance.
We then use LangChain's abstraction over FAISS and pass it the chunks and the embedding model and it converts it to vectors. These vectors can fit into memory or can also be persisted to local disk.



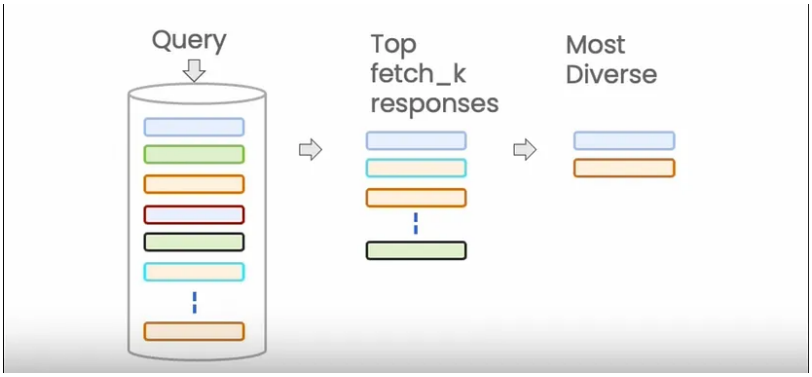
5.5. Contextual Compression : Compression is another approach to improve the quality of retrieved docs. Since passing the full document through the application can lead to more expensive LLM calls and poorer response, it is useful to pull out only the most relevant bits of the retrieved passages.

With compression, we run all our documents through a language model and extract the most relevant segments and then pass only the most relevant segments into a final language model call.This comes at the cost of making more calls to the language model, but it's also good to focus the final answer on only the most important things. And so it's a bit of tradeoff.



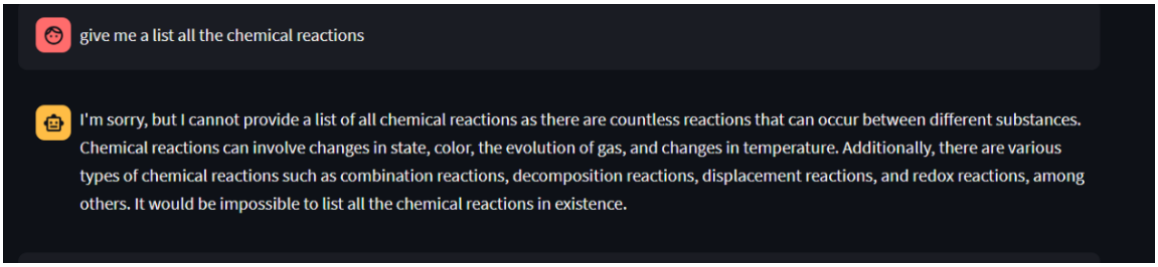
5.6. Maximum Marginal Relevance(MMR) : MMR is an important method to enforce diversity in the search results. In the case of semantic search, we get documents that are most similar to the query in the embedding space and we may miss out on diverse information.

The idea behind MMR is we first query the vector store and choose the “fetch_k” most similar responses. Now, we work on this smaller set of “fetch_k” documents and optimize to achieve both relevance to the query and diversity among the results. Finally, we choose the “k” most diverse response within these “fetch_k” responses.

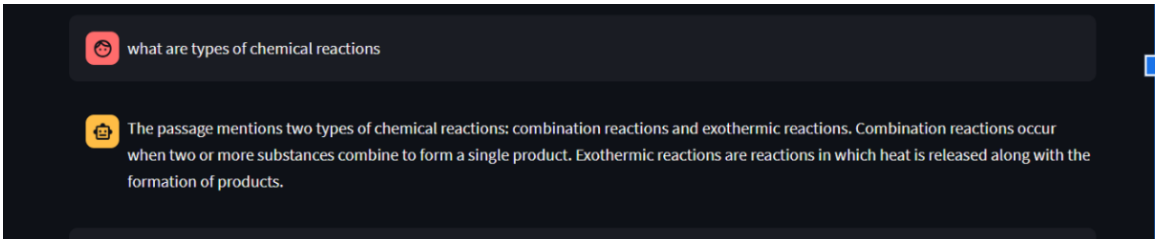


6. Comparison Report

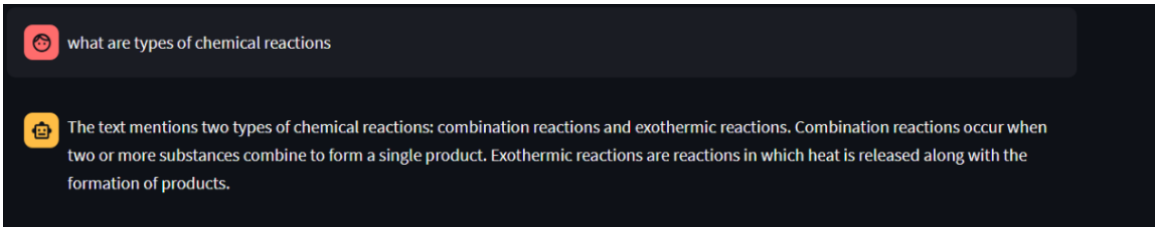
PDF : Science Book
Model : GPT-3.5-Turbo
Retrieval Method : Conversational Retrieval
Temperature: 0.5



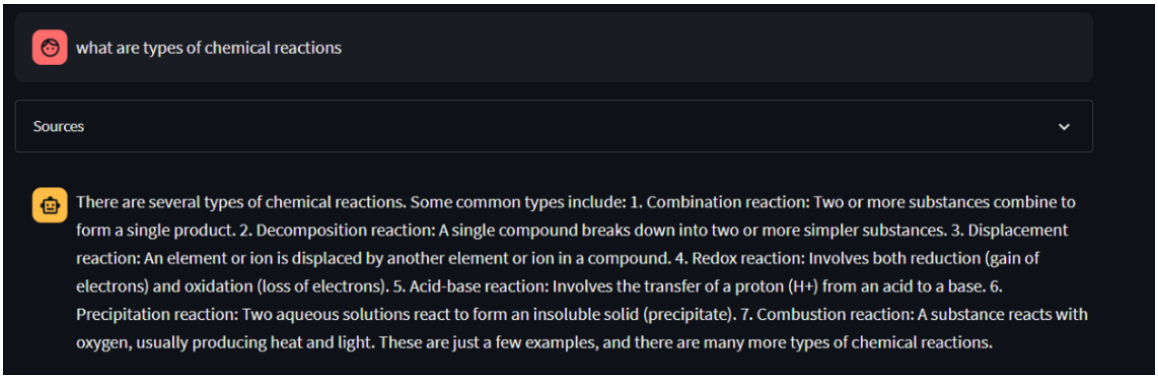
PDF : Science Book
Model : GPT-3.5-Turbo
Retrieval Method : Retrieval QA
Temperature: 0.5



PDF : Science Book
Model : GPT-3.5-Turbo
Retrieval Method : Multi Query
Temperature: 0.5



PDF : Science Book
Model : GPT-3.5-Turbo
Retrieval Method : Compression Retrieval
Temperature: 0.5



7. Conclusion

The EduGPT chatbot provides an interactive platform for users to ask questions related to content from specific PDF files. Leveraging the OpenAI API and the LangChain framework, it ensures efficient and contextually relevant responses. Additionally, the use of Streamlit makes the user interface intuitive and user-friendly.