Approach

I have used RAG (Retrieval Augmented Generation) technique to make LLM answer questions from the document provided using Langchain. Which is described into following steps:

Step 0: Import all LLMs and embedding models using API keys.

Step1: Create chunks of document

Step 2: Create vector embeddings using an embedding model

Step 3: Store the embeddings either locally or in a vector DB(ASTRADB, Pinecone)

Step 4: Creating a retrieval chain-

We build a chain which will have a sub chain for processing previous message history and adding it to the current context.

The chain we have built uses the input query directly to retrieve relevant context. But in a conversational setting, the user query might require conversational context to be understood. For example, consider this exchange:

Human: "What is Task Decomposition?"

AI: "Task decomposition involves breaking down complex tasks into smaller and simpler steps to make them more manageable for an agent or model."

Human: "What are common ways of doing it?"

In order to answer the second question, our system needs to understand that "it" refers to "Task Decomposition."

We'll need to update two things about our existing app:

1. **Prompt**: Update our prompt to support historical messages as an input.
2. **Contextualizing questions**: Add a sub-chain that takes the latest user question and reformulates it in the context of the chat history. This can be thought of simply as building a new "history aware" retriever. Whereas before we had:
    - query -> retriever
      Now we will have:
    - (query, conversation history) -> LLM -> rephrased query -> retriever

Contextualizing the question

First we'll need to define a sub-chain that takes historical messages and the latest user question, and reformulates the question if it makes reference to any information in the historical information.

Step 5: Stateful management of chat history

Automatically inserting and updating chat history

Step 6: Calling the chatbot

Frameworks/libraries/tools

- Langchain- A library in python to work with all kinds of LLM, like OpenAI, Google Gemini, GROQ, etc.
- HuggingFace – For importing embedding model English language using Hugging Face token embedding model used-BAAI/bge-large-en-v1.5
- Groq Inferencing Engine- For using open-source LLM models as inference API since running on local requires a lot of computing power.
  LLM model- Llama3-70b-8192
- FAISS vector store to store vector embeddings
- Python
- Streamlit

Problems/Solutions

1. Choosing an LLM: Since we want to minimise the cost hence using LLM models like OpenAI, Google would be expensive, so I went for open source models but they can't be used on local because extensive memory they require. Hence I used Groq inferencing engine. Another was to check which model gave the best results and tweaking the parameters tc.
2. Choosing an embedding model which free and open-source yet very effective was a tedious task. I used Hugging Face and its MTEB leaderboard for choosing the right embedding model.
3. We can also use hugging face inference APIs of LLM models to use LLMs.
4. Prompt engineering – Developing the right prompt so that the LLM understands its and returns right answer was very hard. And I used little bit of ChatGPT and my own English capabilities to devise a prompt which gives relevant output.

Future Scope

1. We can implement a complaint registering feature using the chatbot which could be like taking to an employee/ call centre executive which takes our complaints.
2. Book appointment to visit tasting centre of Jessup Cellars just by conversing with the chatbot.
3. Order, Cancel and Track order from the chatbot though ordering can be difficult and may give rise problems.
4. We can also implement an AI Voice assistant on phone calls which perform all the work described earlier.
5. If changes in corpus are made, we can easily create new vector embeddings and replace with older ones.
6. If size of corpus increase we can use an external vector database like AstraDB, PineCone to store our vector embedding.

Name- Piyush Kumar Srivastav
Mail- mesrivastava.piyush@gmail.com