Problem Description Chosen:

Create a rudimentary Question Answering system that can answer questions related to the Israel Hamas war.

1. You can use any open-source LLM (LLAMA, T5, BERT), library (langchain (https://www.langchain.com/), LLamaIndex (https://www.llamaindex.ai/)) or algorithm (BM25, Maximal Marginal Relevance) that you want.
2. Also, you can augment the existing articles by extracting additional information from Wikipedia or DBpedia

Initial Thinking:

When I came across the problem statement, it was quite similar to the ones I have worked in my internship works for different client projects. So, I went ahead and created a basic flow of working that will suffice for solving the problem statement.

My Initial Idea was to first preprocess the data received in JSON file and convert into a csv and read into a dataframe for better understanding. The next step I though was to index this data using Pinecone and then retrieve and pass it to a LLM for the final response generation. Now I have previously worked with OpenAI Embeddings and used OpenAI as the LLM for the entire RAG use case. But this time I had to use open source models and hence it was a new learning experience for me in that part.

PREPROCESSING

I analysed and checked the JSON file and saw it is well structured into key value pairs and the content can be extracted quite easily. Apart from that I used Regular Expressions to perform the basic preprocessing and removal of unnecessary stopwords and punctuations. Also one thing I did for filtering the unrelated articles in the database, I used a list of relevant words and discarded the articles not having those words. Although this is not the most efficient way and maybe some useful data was lost in this step. Further to improve, we can use some more efficient method to filter out articles and keep the ones relevant to our use case.

At the end of this step I had the processed csv file ready with the columns: title, content, source and date.

STORING AND INDEXING

I thought of using ChromaDB as my vector database. The reason for choosing ChromaDB was first it was open source and saves the data locally. And secondly the integration with LangChain is seamless and efficient for building the application. For Embeddings I used “sentence-transformers/all-mpnet-base-v2 “ from HuggingFace. I faced some issues initially to understand the model args and running it without error. But eventually I read some documentation and was able to make it work properly.

The next part was chunking the data and storing and indexing the chunks. This was the lengthiest process till now for me. It took around 2.5 hours to index all the data to the vector database. I had 100K chunks from the 8966 relevant articles in my processed.csv.

After this step I had my vectordb setup locally with all my data indexed.

Question Answering System

The last step was building the QA system. I utilized the Mistral 7B open Source model as the LLM and created a pretty simple chain that takes the question, retrieves the context and passed to LLM for the output. But the output was highly unstructured with question, context as well as Answer merged together.

So I thought of creating a chain to streamline the process and finally process the output and return only the answer at the end. I used LECL for this and created the chain.

Also according to the problem description, it was mentioned we can augment our data with data online as well. So I though of doing so and used the DBPedia to fetch relevant data based on user query and return it as context along with the retrieved context from the vector db. Both of these sources were now being used as context and passed to LLM for the final output generation.

Apart from the standalone Jupyter notebook, I created a simple user interface using Streamlit to enter the question and select the max_length and allows user to see the answer in the interface.

Future Improvements

The interface can be made more intuitive and also include the sources of the answer along with the answer to ensure credibility to user while referring an answer. Apart from that the answer generation can also be improved further with functionalities like fetching the chunk metadata which is the source link, and scraping that link and augmenting those information as well for better contextual understanding of the LLM and more accurate response.