Brance <Position>Task

Name: Ayush Vatsal

Linkedin Profile: https://www.linkedin.com/in/ayushvatsal/

Date Challenge Received: 17-07-23.

Date Solution Delivered: 19-07-23.



1. Problem Statement

What was the task and how you understood it.

The task was to build a RAG(Retrieval Augmented Generation) chatbot. For user question RAG module would retrieve context from knowledge document and generation phase llm would personalize answer using retrieval knowledge.

Retrieval Augmented Generation combines information retrieval from a knowledge base and in-context learning of an LLM. I listened to Patrick Lewis from “**Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**” paper talk about training models with information retrieval. I also studied the Langchain documentation for retrieval question answering chains.

2. Approach

Your approach to the problem. Mention any assumptions made.

Approach

The approach for building the baseline chatbot (version of the chatbot having only the required deliverables, text chat without hallucination from the knowledge base, no voice, single language) was as follows:

The Multi-Lingual Voice-based RAG Chatbot is designed with a step-by-step approach to deliver a comprehensive conversational experience. It starts by loading the document, provided in a txt file, using Langchain's TextLoader. The document is then split into manageable chunks using Langchain's markdown header splitter, ensuring better organization with header metadata. These chunks are embedded using OpenAI's powerful LLM, based on the GPT-3.5 model. Retrieval is facilitated by employing a similarity search with cosine distance to identify the top three most relevant chunks. As the knowledge base is relatively small, a chain-type “stuff” is adopted for enhanced performance. To enable in-context learning, a Retrieval QA chain is utilized, providing specific instructions to avoid responses beyond the knowledge base. The integration of Whisper API enables language detection and transcription of voice inputs, expanding the chatbot's capabilities to understand multiple languages. Additionally, another chain is incorporated to translate detected language queries into English. For English queries, the chatbot directly processes the input. Finally, the chatbot translates the English responses back to the detected language using the translation chain. The Gradio interface enhances user interactions by offering a convenient microphone recording option for seamless voice-based communication with the chatbot. This comprehensive approach ensures that the Multi-Lingual Voice-based RAG Chatbot delivers accurate and contextually relevant responses, offering an inclusive and user-friendly conversational experience across different languages.

Assumptions: Knowledge base is in English and follows the md template. (Both of these assumptions, if violated can be accommodated by simple changes in the code)

Bonus Features

The following bonus features can be added to the chatbot:

* Evaluation of the answers: Evaluation of the chatbot’s answers can be done using many techniques, some of them are perplexity, Bilingual Evaluation Understudy BLEU score or human evaluation.
* Supporting multi-linguality: This can be done by using an LLM that supports multiple languages.
* Adding speech capabilities: This can be done by using a speech-to-text engine and a text-to-speech engine.

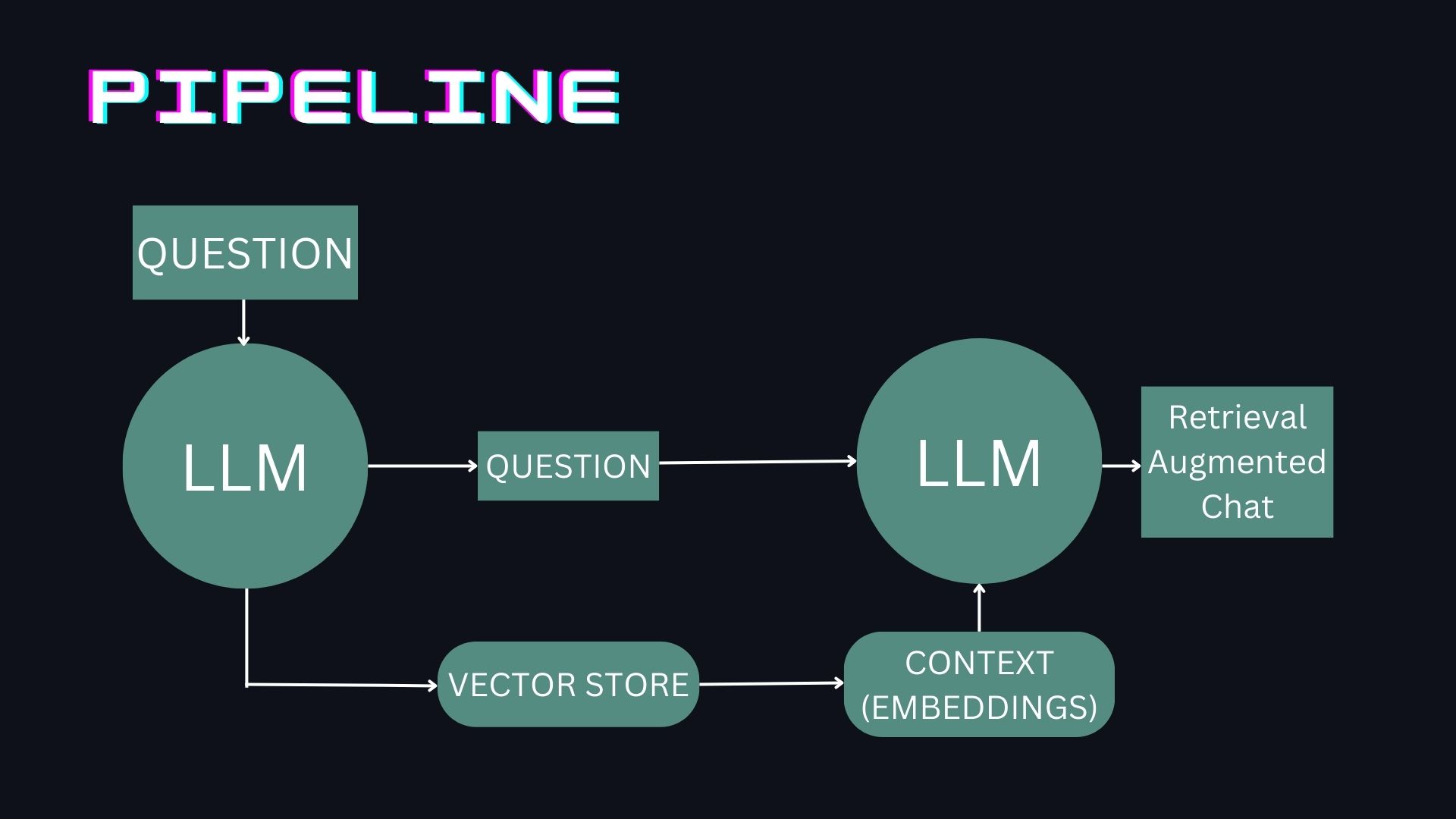
Deliverables

The deliverables for the chatbot include:

* Working solution: This is the code for the chatbot that can be run on a local machine or a cloud platform.
* A clean, efficient, explanatory, and maintainable code: This is the code for the chatbot that is well-organized, easy to read, and easy to maintain.
* A small writeup on approach, assumptions, and future scope: This is a document that describes the approach to building the chatbot, the assumptions that were made, and the future scope of the project.

3. Solution

Details about your solution. Illustrate performance and design with diagrams.



A diagram of the process used to create baseline chatbot.

A screenshot of a computer

Description automatically generated

Screenshot of the baseline chatbot.

Link to the baseline chatbot: <https://ayush-vatsal-rag-chatbot.hf.space/>

A screenshot of a chatbot

Description automatically generated

Screenshot of the multilingual model for audio in English

A screenshot of a computer

Description automatically generatedScreenshot of the multilingual model for audio in another language, Hindi

The steps taken are as follows:

1. Loading the document: Since the document is a txt file using Langchain’s Text file loader TextLoader to load the file.
2. Splitting the document into chunks: The document was formatted like a markdown, using the markdown header splitter from Langchain to get better splits with headers in the metadata. The file was split on header 1, then header 2 and finally header 3.
3. Embedding the chunks: While the instructor embedding models have better performance on the benchmarks, its performance is a bit debatable as it uses task specific headers in its embeddings. Using OpenAI’s embeddings as the LLM is also GPT 3.5 (from OpenAI).
4. Retrieval: Using the similarity search (Cosine distance) to retrieve top k chunks (k is set to 3) from the embeddings.
5. Different retrieval techniques: Here the chain type stuff is used as the knowledge base is relatively small. Refine, map reduce or map rerank could be used for larger knowledge bases.
6. In-context learning: Using the Retrieval QA chain, with a custom prompt template, the context and questions are passed into the model. The prompt has specific instructions to not answer questions outside of knowledge base.
7. Gradio Interface: A gradio chatbot interface is made for better user interactions.

Multi-Lingual Voice based RAG Chatbot:

In addition to the features of the baseline chatbot, these features are added to the Multi-Lingual Voice based RAG Chatbot:

1. Whisper API to detect language and transcribe the text.
2. Another chain to translate text from detected language to English.
3. Running the chatbot for the English query.
4. Translating answer back to the detected language using the chain.
5. Gradio Interface is made to interact with the models, using the microphone recording option.

4. Future Scope

Thoughts on how you could have improved the solution.

While Gradio is good for initial prototyping, adding a JS based chat interface could help in taking both voice and text input simultaneously. Further, LLMs are static in nature and thus do not have any memory. A chat history could be added to the prompt to simulate a conversation memory. Due to time constraints, evaluation methods were also left unexplored. Finally, a TTS engine could be added to make the chat-bot completely handsfree.

While the OpenAI models perform very well, we could fine-tune a smaller model with the context from the knowledge base to get better results, faster responses by using a smaller model and complete privacy.