Architecture : RAG-based Chatbot using LLM and VectorDB

1. **Environment Setup:**
    - Import necessary libraries and modules, such as `os`, `ollama`, `RecursiveCharacterTextSplitter`, `WebBaseLoader`, `Chroma`, and `OllamaEmbeddings`.
    - Set the current working directory (`curr_dir`) which will help save the embeddings to a vector storage database.
    - Storing the embeddings is important as we do not admire re-creating the embeddings again and again whenever program is re-run. Hence, saving it helps as it can be reused and processing power is saved.
2. **Vectorstore (Chroma) Management:**
    - Program checks if the Chroma database file exists.
    - If the file exists, it loads the vectorstore (Chroma) using pre-existing embeddings.
    - If the file doesn't exist, it creates a new vectorstore by loading web documents (Through a web url), splitting them into chunks, generating embeddings, and persisting the vectorstore.
3. **Retrieval Component:**
    - Creates a retriever to retrieve information related to user questions from the vectorstore and then feeds it further into the LLM.
4. **Document Formatting:**
    - Defines a function (`format_docs`) to format documents for input to the model.
    - Formatting is necessary because we want to minimise the average cost per word sent to the model.
5. **Conversational History:**
    - Maintains a conversation history for context-aware responses.
    - Addressing a potential issue:
        - Consider a user question like "Tell me more about that?" It poses a challenge for similarity search in the vector database since it lacks explicit semantic meaning, leading to potentially irrelevant results. The question's ambiguity makes it challenging to identify what "that" refers to.
    - To overcome this challenge, I employed `ollama_generate_question` function. Initially, it sends both the chat history and the user's question to the language model (LLM), requesting a refined question based on the conversation context. The improved question is then used in conjunction with the retrieved information for a more accurate response from the LLM.
6. **Ollama Language Model (LLM):**

- Defines two functions (`ollama_generate_question` and `ollama_llm`) that leverage Ollama for rephrasing user questions and generating responses.
- These functions use the conversational history to enhance conversational awareness.
- The main function `ollama_llm` also gets some of the chat history, to fetch more better results.

7. **RAG Chain (Retrieval-Augmented Generation):**

   - Defined a function (`rag_chain`) that manages the entire conversation flow.
   - It makes use of the Retrieval Component [3] within itself.
   - Appends user questions to the conversation history and generates responses using the Ollama Language Model after retrieving relevant information from the Vectorstore.
   - It also fetched better format of question from ollama llm with conversation history, which helps in conversational awareness of the chatabot.

8. **Chat Loop:**

   - Enters into a chat mode loop, where the user can input questions or commands ("/bye" to exit).
   - Processes user input using the RAG chain and prints the bot's response.

Overall Flow:

- User inputs are processed through a Retrieval-Augmented Generation (RAG) chain, including Ollama for question rephrasing and context-awareness.
- The chatbot leverages a Vectorstore (Chroma) for efficient information retrieval based on the user's queries.
- Conversational history is maintained to enhance the context of the ongoing conversation.

This architecture allows for a dynamic and context-aware chatbot that retrieves information from web documents, generates relevant responses, and adapts its language based on the conversational context.

For more info mail me at divyangmandal03@gmail.com
- call me at +91 86378 84941