Streamlit Chatbot for PDF Documents: A Report

Introduction

This report details the development of a chatbot that facilitates user interaction with a provided PDF document. The chatbot leverages Streamlit for the web interface and Langchain libraries to build the question-answering pipeline.

Overall Approach

The core approach utilizes a RetrievalQA architecture powered by Langchain. The process involves:

1. **Preprocessing the PDF:** Text extraction and potential cleaning/normalization.
2. **Generating Embeddings:** Creating vector representations of text chunks using OllamaEmbeddings. These are stored in a Chroma vectorstore for efficient retrieval.
3. **Building the Chat Interface:** Constructing a user-friendly interface with Streamlit for text input and chatbot responses.
4. **User Interaction:** Capturing user queries through Streamlit's `st.chat_input`.
5. **Retrieval and Response Generation:** Employing Langchain's RetrievalQA:
    - Retrieving relevant passages from the vectorstore based on the user's query.
    - Using Ollama, a large language model (LLM), to generate a response considering the retrieved context, conversation history, and a predefined prompt template.

Frameworks/Libraries/Tools Used

- **Front-end:** Streamlit (web interface)
- **Back-end:** Langchain (QA pipeline)
    - OllamaEmbeddings (text embeddings)
    - Chroma (vectorstore)
    - RetrievalQA (combines retrieval and LLM for response generation)
- **Additional Potential Libraries:** PyPDFLoader (PDF processing), text cleaning libraries
- **Large Language Model (LLM):** Ollama (can be replaced with other LLMs)

Challenges and Solutions

- **Fine-tuning the LLM:**
    - Consider fine-tuning Ollama on a domain-specific dataset relevant to your PDF content to enhance its understanding and response generation. Explore libraries like Transformers for this purpose.
- **Memory Constraints:**

  - For large PDFs, process the document in smaller chunks or utilize cloud-based resources with higher memory capacity.
- **Model Memory (Remembrance):**
  - Implement conversation memory using Langchain's ConversationBufferMemory component to store past interactions and provide context for the LLM, leading to more coherent responses.

Future Scope

- **Enhanced User Interface:** Integrate functionalities like:
  - Answer clarity rating for user feedback.
  - Navigation within the PDF based on the conversation.
- **Advanced Retrieval Techniques:** Explore dense retrieval using libraries like Faiss or Jina for improved retrieval accuracy.
- **Multimodality:** Integrate capabilities like image processing or text summarization to handle user queries that might involve these aspects of the PDF content.
- **Domain-Specific Fine-tuning:** Fine-tune the LLM on a dataset related to the specific domain of your PDF to achieve superior performance.

Conclusion

This Streamlit chatbot offers a user-friendly way to interact with PDF documents. By addressing the mentioned challenges and implementing future enhancements, the chatbot can be further refined to provide a more robust, informative, and interactive experience for users.