Chatbot for PDF Documents: A Report

1. Introduction

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

2. Overall Approach

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

1. Preprocessing the PDF: Text extraction using PyPdfLoader and potential cleaning/normalization.
2. Dividing the PDF for Better Retrieval: The document is split into smaller sections, and multiple FAISS vector databases are created to enhance retrieval accuracy.
3. Generating Embeddings: Creating vector representations of text chunks using OllamaEmbeddings, which are stored in FAISS vectorstores for efficient querying.
4. Building the Chat Interface: Constructing a user-friendly interface with Streamlit for user queries and chatbot responses.
5. User Interaction: Capturing user queries through Streamlit's `st.chat_input`.
6. Retrieval and Response Generation: Utilizing Langchain's RetrievalQA pipeline:
   - Retrieving the most relevant passages from multiple FAISS vectorstores based on the user's query.
   - Using Ollama, a large language model (LLM), to generate responses by considering retrieved context, conversation history, and a predefined prompt template.

3. Frameworks/Libraries/Tools Used

- Front-end: Streamlit (web interface)
- Back-end: Langchain (QA pipeline)
- OllamaEmbeddings (text embeddings)
- FAISS (vectorstore for efficient retrieval)
- RetrievalQA (combines retrieval and LLM for response generation)
- Additional Libraries: PyMuPDF (PDF processing), RecursiveCharacterTextSplitter (text chunking)
- Large Language Model (LLM): Ollama (can be replaced with other LLMs)

4. Challenges and Solutions

- Fine-tuning the LLM: Consider fine-tuning Ollama on a domain-specific dataset relevant to your PDF content to enhance its response quality. Libraries like Transformers can be used for this purpose.
- Memory Constraints:

- For large PDFs, dividing the document into smaller chunks and creating multiple FAISS vector databases improves retrieval efficiency.

- Model Memory (Remembrance):

- Implementing conversation memory using Langchain's ConversationBufferMemory` component to retain past interactions and provide contextual responses.

5. Future Scope

- Enhanced User Interface:

- Implement functionalities like answer clarity ratings and navigation within the PDF based on the conversation.

- Advanced Retrieval Techniques:

- Experiment with hybrid retrieval approaches using dense retrieval libraries like Jina.

- Multimodality: Extend capabilities to include image processing and text summarization for handling complex queries.

- Domain-Specific Fine-tuning: Fine-tune the LLM on a dataset related to the specific domain of the PDF for improved performance.

6. Conclusion

This Streamlit chatbot provides an interactive and efficient way to engage with PDF documents. By addressing the mentioned challenges and implementing future enhancements, the chatbot can be further improved to offer a more refined, informative, and user-friendly experience.