**Project Report: Local Retrieval-Augmented Generation (RAG) Chatbot**

**1. Introduction**

This project involves the design and implementation of a **Retrieval-Augmented Generation (RAG)** chatbot that can answer user queries based on the content of uploaded documents, specifically PDFs. The chatbot operates **entirely offline and free of cost** by leveraging **local Large Language Models (LLMs)**, document embedding techniques, and an intuitive chat interface.

The primary objective is to provide accurate, context-aware, and natural responses to user queries by combining the reasoning ability of LLMs with external domain-specific knowledge provided by the user.

**2. Problem Statement**

Modern LLMs like ChatGPT or Bard are powerful but require internet access and often need expensive API keys. This creates a barrier for:

* Users in low-connectivity or offline environments.
* Institutions with privacy-sensitive data.
* Developers and students looking for cost-free solutions.

This project eliminates those barriers by running all components locally using free, open-source tools.

**3. Objectives**

* Build a chatbot that uses local LLMs to generate responses.
* Integrate document retrieval using vector similarity search.
* Support uploading user documents (PDFs) as knowledge sources.
* Ensure the solution is completely offline and free to use.
* Provide a simple and interactive user interface.

**4. System Architecture**

**Components**

1. **User Interface (Streamlit)**: A web-based UI for chat and document upload.
2. **Document Loader (PyMuPDF)**: Extracts content from PDF files.
3. **Text Splitter**: Divides documents into chunks suitable for embeddings.
4. **Embedding Generator (SentenceTransformers)**: Converts text into vectors.
5. **Vector Store (ChromaDB)**: Stores document chunks in vector format.
6. **Retriever (LangChain)**: Finds relevant chunks for a given query.
7. **LLM (Ollama - Mistral/LLaMA)**: Processes context and generates answers.

**Data Flow**

**[User Input] → [Retriever] → [Relevant Chunks] → [LLM via LangChain] → [Answer Output]**

**5. Tools and Technologies**

|  |  |
| --- | --- |
| **Component** | **Tool/Library** |
| LLM | Ollama (e.g., Mistral) |
| Embeddings | sentence-transformers |
| Document Parsing | PyMuPDF |
| Vector DB | ChromaDB |
| LLM Framework | LangChain |
| UI | Streamlit |

**6. Implementation Details**

**File: requirement to be download**

1. **streamlit**
2. **langchain-community**
3. **chromadb**
4. **sentence-transformers**
5. **pymupdf**
6. **Ollama**

**File: app.py**

* Handles Streamlit UI for uploading files and chatting.
* Stores user messages in session state.
* Calls chat.py for response generation.

**File: chat.py**

* Loads the vector database and initializes a LangChain QA chain.
* Uses the Ollama local LLM (e.g., Mistral) to generate responses.

**File: retriever.py**

* Loads and splits PDF content using PyMuPDF and RecursiveCharacterTextSplitter.
* Generates vector embeddings using SentenceTransformers.
* Stores embeddings in a ChromaDB persistent database.

## 7. How to Use

* Install all packages in requirement.text
* Launch the app:
* Upload a PDF document.
* Type a question related to the document content.
* Receive context-based answers from the local LLM.

**8. Advantages**

* **Offline capability**: No internet required.
* **Free to use**: No subscription or API cost.
* **Private and secure**: Your data never leaves your machine.
* **Custom knowledge base**: Upload and use your own documents.

**9. Limitations and Future Work**

|  |  |
| --- | --- |
| Limitation | Suggested Improvement |
| No chat memory/history | Integrate conversation memory logic |
| Only supports PDFs | Add support for .docx, .txt, etc. |
| No authentication or role-based access | Add user management and login |

**10. Conclusion**

This project demonstrates a powerful yet accessible RAG-based chatbot that utilizes offline, local LLMs to answer domain-specific queries. It is ideal for learning, secure environments, and private applications without incurring cloud service costs.

The approach aligns with the growing demand for edge AI, privacy, and cost-efficient LLM deployment.

**11. References**

* LangChain: https://www.langchain.com
* Ollama: https://www.ollama.com
* ChromaDB: https://www.trychroma.com
* Streamlit: https://streamlit.io
* Sentence Transformers: https://www.sbert.net