AI - BASED CHATBOT USING RAG

By: Neha Sara Sam

**OBJECTIVE:**

This project task aims to build a chatbot that can answer questions based on the content of a pdf uploaded by a user. The key technique used here is RAG (Retrieval-Augmented Generation), which enhances LLMs by grounding their answers in external documents.

**CONCEPT:**

**1.User Interface:**

Allows the user to upload a PDF and ask questions. Gradio is used as it is quick to set up, user-friendly and ideal for prototypes.

**2.PDF Processing:**

Extracts and cleans the text from the uploaded PDF. PyPDF2 library is used for PDF parsing. This step ensures clean and structured text is parsed for embedding.

**3.Text Embedding:**

Converts text into numerical vectors (embeddings) for semantic search. It splits the long PDF text into smaller chunks using a text splitter. It uses an embedding model called Hugging Face Sentence Transformers, which converts each chunk into a vector.

**4.Vector Store:**

Stores embeddings in a database that allows similarity search. Uses a vector data called FAISS to store and index embeddings. This allows quick retrieval of the most relevant chunks when a user asks a question.

**5.RAG Pipeline:**

Retrieval-Augmented Generation combines:

* **Retrieval:** Finds the top matching chunks from the vector store using the user’s query.
* **Generation**: Feeds the retrieved chunks to LLM (Large Language Model) like Gemini/GPT, to generate a context-based answer.

This is required to get more accurate answers and to reduce hallucinations by grounding responses in real documents.

**LIBRARIES USED:**

1. os
2. gradio
3. PyPDF2.PdfReader
4. langchain.text\_splitter.RecursiveCharacterTextSplitter
5. langchain\_community.embeddings.HuggingFaceEmbeddings
6. langchain\_community.vectorstores.FAISS
7. langchain\_google\_genai.ChatGoogleGenerativeAI
8. langchain\_core.prompts.ChatPromptTemplate
9. langchain.chains.combine\_documents.create\_stuff\_documents\_chain
10. langchain.chains.create\_retrieval\_chain

**1.os**

This is used for setting environment variables like google API key. It helps to securely manage credentials (by using .env).

**2.gradio**

It builds the chatbot’s user interface. It uploads pdf file, triggers processing, takes input, shows output and is a layout container for UI.

**3.PyPDF2.PdfReader**

This library is used to extract text from each page of a pdf. It is important as it converts documents into raw text for further processing.

**4. langchain.text\_splitter.RecursiveCharacterTextSplitter**

It breaks long text into “manageable overlapping chunks”. It ensures that important context near chunk boundaries is not lost. It is used mainly because it is recursive and can handle large documents efficiently.

**5. langchain\_community.embeddings.HuggingFaceEmbeddings**

It transforms text chunks into “dense numerical vectors”. The model “all-MiniLM-L6-v2” is used, which a small but powerful model for sentence embeddings.

**6. langchain\_community.vectorstores.FAISS**

FAISS (Facebook AI Similarity Search) stores the vectors (from embeddings) and enables fast nearest-neighbour search. It allows retrieving the most relevant chunks when a question is asked.

**7. langchain\_google\_genai.ChatGoogleGenerativeAI**

It interfaces with Google’s Gemini LLMs (like gemini-1.5-flash). It accepts prompt and context. Gemini is used as its’s fast, accurate and integrates well with LangChain.

**8. langchain\_core.prompts.ChatPromptTemplate**

It structures the input to the LLM in a controlled, template-based format. It keeps the chatbot’s behaviour predictable and grounded in the document.

**9. langchain.chains.combine\_documents.create\_stuff\_documents\_chain**

It is a chain that “stuffs” all retrieved documents into one prompt and sends it to the LLM. It is useful when we want to include all relevant information in the same context window.

**10.langchain.chains.create\_retrieval\_chain**

It creates a full RAG pipeline that:

* Retrieves relevant chunks (via FAISS)
* Feeds them to the LLM with a structured prompt
* Returns a grounded answer

**CONCLUSION:**

This project demonstrates how to build a functional AI chatbot using RAG using:

* Open-source tolls for PDF reading,
* State-of-the-art LLM (Gemini) for answer generation.
* A clean user interface with gradio