**Overview**

This project builds a retrieval-augmented question answering chatbot over PDF documents using:

* LangChain (document loaders, text splitting, chains),
* HuggingFace embeddings (sentence-transformers/all-MiniLM-L6-v2),
* FAISS vector store for fast retrieval,
* HuggingFace-hosted large language model (zephyr-7b-beta),
* Streamlit for the user chat interface.

**PROJECT PHASES**

**Phase 1: Knowledge Base Creation**

* Input raw PDFs.
* Extract text and split into chunks.
* Convert chunks to embeddings.
* Store embeddings in FAISS vector store as the knowledge base.

**Phase 2: Query Processing**

* Input a user question.
* Convert the question to an embedding.
* Perform semantic search to find relevant chunks.
* Rank and pass these chunks to the LLM.

**Phase 3: Answer Generation**

* The LLM generates an answer based on the ranked results.
* Output the answer to the user.

**File Descriptions**

**1. create\_memory\_for\_llm.py**

* **Purpose:**  
  Preprocesses PDF documents by loading, chunking, embedding, and storing vectors in a local FAISS index.
* **Process:**
  + Loads PDFs from data/ directory using DirectoryLoader and PyPDFLoader.
  + Splits PDFs into text chunks with overlap (chunk\_size=500, chunk\_overlap=50).
  + Generates embeddings using sentence-transformers/all-MiniLM-L6-v2.
  + Stores embeddings in FAISS vector store locally (vectorstore/db\_faiss).

**2. connect\_memory\_with\_llm.py**

* **Purpose:**  
  A Streamlit app that connects the saved FAISS vector store and HuggingFace LLM, providing a chat UI for user queries.
* **Main Features:**
  + Loads the FAISS vector store from disk with caching for efficiency.
  + Defines a custom prompt template instructing the LLM to answer only from retrieved context without fabrication.
  + Connects to HuggingFace’s zephyr-7b-beta endpoint with user-provided API token.
  + Implements chat interface to accept user input, run RetrievalQA, and display answers.
* **Usage:**  
  Run with streamlit run connect\_memory\_with\_llm.py.

**3. medibot.py**

* **Purpose:**  
  Your **main chatbot Streamlit app**, integrating FAISS retrieval and HuggingFace LLM with a clean chat UI and session state handling.
* **Features:**
  + Uses st.chat\_input and st.chat\_message to create conversational UX.
  + Caches FAISS vector store loading to avoid repeated disk reads.
  + Handles user messages and chatbot responses stored in Streamlit session state.
  + Defines a strict custom prompt guiding the LLM to answer only from retrieved context, avoiding hallucinations.
  + Retrieves top 3 relevant chunks from FAISS and passes to LLM for response generation.
  + Displays results directly in the chat interface.
* **Environment:**
  + GPU disabled via os.environ["CUDA\_VISIBLE\_DEVICES"] = "-1".

(since streamlit doesn’t support GPU yet )

* + Warnings and telemetry disabled for clean output.
* **To run:**

bash

streamlit run medibot.py

**How It Works — End-to-End Flow**

1. **Document Loading & Vector Store Creation**  
   PDFs in data/ are processed with create\_memory\_for\_llm.py to create a FAISS vector store.
2. **Chat Interface**  
   medibot.py runs a Streamlit app that accepts user questions.
3. **Retrieval**  
   The FAISS index is queried for the top relevant document chunks for the user’s question.
4. **LLM Query**  
   The zephyr-7b-beta model hosted on HuggingFace Hub is queried using the retrieved context with a custom prompt that limits answers to given context only.
5. **Answer Display**  
   The chatbot answer is displayed in a conversational chat UI, preserving chat history.

**Requirements & Setup**

pip install torch==2.7.0+cpu torchvision==0.18.1+cpu --index-url https://download.pytorch.org/whl/cpu

pip install -r requirements.txt

**Requirements.txt:**

faiss-cpu==1.11.0

langchain==0.3.25

langchain-community==0.3.24

langchain-core==0.3.60

langchain-huggingface==0.2.0

langchain-text-splitters==0.3.8

sentence-transformers==4.1.0

streamlit==1.45.1

transformers==4.51.3

**Citations & References**

**Language Model**

* **Zephyr-7B-Beta**  
  Hugging Face H4. *Zephyr: a series of chat-tuned language models*.  
  Available at: https://huggingface.co/HuggingFaceH4/zephyr-7b-beta

**Embedding Model**

* **Sentence-Transformers: all-MiniLM-L6-v2**  
  Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.  
  Available at: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

**Libraries & Frameworks**

* **LangChain**  
  Harrison Chase et al. *LangChain: Building applications with LLMs through composability.*  
  GitHub: <https://github.com/langchain-ai/langchain>
* **FAISS (Facebook AI Similarity Search)**  
  Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs.  
  GitHub: <https://github.com/facebookresearch/faiss>
* **Streamlit**  
  Streamlit Inc. *Streamlit: Turn data scripts into shareable web apps.*  
  Website: <https://streamlit.io>
* **HuggingFace Transformers**  
  Wolf, T., et al. (2020). Transformers: State-of-the-Art Natural Language Processing.  
  Available at: https://huggingface.co/docs/transformers
* **LangChain Community and HuggingFace Integration Libraries**  
  LangChain Community. <https://python.langchain.com/docs/integrations/>

**Documentation**

* OpenAI. (2024). ChatGPT (April 2024 version) [Large language model]. <https://chat.openai.com/>