Documentation: PDF Question Answering App using LangChain, Hugging Face, FAISS, and Streamlit

# Overview

This application is a PDF-based Question Answering system that allows users to upload a PDF and ask questions about its content. It uses LangChain, Hugging Face Embeddings, FAISS Vector Store, and Streamlit for UI. The backend uses Conversational Retrieval QA to return context-aware answers from the uploaded document.

# 1. Key Definitions

- LangChain: A framework that simplifies building LLM applications by chaining together components like LLMs, memory, retrievers, etc.

- FAISS: A fast and scalable vector store library for similarity search.

- Hugging Face Embeddings: Embedding models that convert text into vector representations used for similarity search.

- ConversationalRetrievalChain: A LangChain utility that handles both retrieval and memory for question-answering.

- Streamlit: A Python-based UI library for building interactive web apps.

# 2. Why These Tools Were Used

Hugging Face vs OpenAI:

- Hugging Face: Free and local model support. No API keys required. Used here to reduce cost and avoid rate limits.

- OpenAI: More powerful LLMs but requires API keys and can be expensive.

FAISS vs Pinecone or Chroma:

- FAISS: Lightweight, fast, and suitable for local applications.

- Pinecone: Cloud-based, scalable, but not free for all features.

- Chroma: Also good, but FAISS is better supported in LangChain and simpler for deployment.

LangChain vs LlamaIndex:

- LangChain: Better abstraction for chaining LLM + Retriever + Memory. Great documentation and active ecosystem.

- LlamaIndex: More focused on indexing and retrieval. Less flexible for complete workflows like Conversational QA.

Streamlit vs Gradio or Flask:

- Streamlit: Simple, fast to build interactive data apps.

- Gradio: More focused on model demos.

- Flask: Lower-level, more control, but slower to prototype.

# 3. Core Features

- Upload and parse PDFs

- Chunk documents intelligently

- Generate vector embeddings

- Store and retrieve chunks using FAISS

- Use Hugging Face FLAN-T5 for answering

- Maintain conversation context

- Fully interactive UI with Streamlit

# 4. Where It Can Be Used

- Academic Research Assistants

- Legal or Policy Document Summarization

- Customer Support Automation

- Internal Company Docs Q&A

- Personal Note Q&A Engine

# 5. Program Flow (API-Specific Walkthrough)

Entry Point:

- main() is the entry point via `if \_\_name\_\_ == "\_\_main\_\_"`

Step-by-Step Flow:

File Upload:

- uploaded\_file = st.file\_uploader("Upload a PDF file", type="pdf")

- Triggers PDF reading and parsing if a file is uploaded.

PDF Loading:

- loader = PyPDFLoader(temp\_file\_path)

- pages = loader.load\_and\_split()

Text Splitting:

- splitter = RecursiveCharacterTextSplitter(...)

- docs = splitter.split\_documents(pages)

Embedding Generation:

- embedding = HuggingFaceEmbeddings(model\_name="sentence-transformers/all-MiniLM-L6-v2")

- vectordb = FAISS.from\_documents(docs, embedding)

Retrieval:

- retriever = vectordb.as\_retriever(search\_kwargs={"k": 3})

LLM Setup:

- llm\_pipeline = pipeline("text2text-generation", model="google/flan-t5-base", ...)

- llm = HuggingFacePipeline(pipeline=llm\_pipeline)

Chain Creation:

- qa\_chain = ConversationalRetrievalChain.from\_llm(llm=llm, retriever=retriever, memory=memory)

User Interaction:

- response = qa\_chain.invoke({"question": user\_question})

# 6. Alternatives Considered But Not Used

| Component | Used | Alternatives Not Used | Reason |

|------------------|------------------------------|------------------------|-------------------------------------------|

| Embedding Model | Hugging Face MiniLM | OpenAI Embeddings | OpenAI costs money; Hugging Face is free |

| Vector DB | FAISS | Pinecone, Chroma | FAISS is faster, simpler for local use |

| LLM | FLAN-T5 (Hugging Face) | GPT-3.5, Claude | FLAN-T5 is free and sufficient |

| LangChain | ✅ | LlamaIndex | LangChain has more flexibility |

| UI | Streamlit | Gradio, Flask | Streamlit is easier for layout and UX |

# 7. Conclusion

This app provides a practical, cost-efficient solution for querying PDF content using open-source tools. By combining LangChain's conversational framework, free Hugging Face models, and Streamlit's intuitive UI, it delivers an end-to-end AI solution without incurring API costs or complex deployments.

LangChain’s powerful chaining and memory management, paired with FAISS and Hugging Face, makes this setup ideal for developers looking to build localized intelligent Q&A systems from documents.