RAG Chatbot Report – Amlgo Labs Assignment

1. Document Structure & Chunking Logic

The provided AI Training Document (~10,500+ words) was processed using LangChain’s PyPDFLoader. Since large documents cannot be passed directly to LLMs, the text was split into smaller segments using RecursiveCharacterTextSplitter with a chunk size of 1200 characters and an overlap of 100 characters. This ensures semantic continuity and maximizes retrievability. The resulting chunks were stored in a /chunks directory for reuse.

2. Embedding Model & Vector Database

Embeddings were generated using sentence-transformers/all-MiniLM-L6-v2, a lightweight model optimized for semantic similarity tasks. The embeddings were stored in FAISS, a high-performance similarity search library. The top-3 most relevant chunks are retrieved for every query. The FAISS index is persisted in the /vectordb folder for faster reuse in subsequent runs.

3. Prompt Format & Generation Pipeline

The chatbot implements a Retrieval-Augmented Generation (RAG) pipeline. The retriever fetches the most relevant chunks and injects them into a structured prompt along with the user’s question. This augmented prompt is passed to the generator (LLM) to produce a factual, grounded answer.

Prompt Template:

Answer ONLY using the provided pdf context. If the context is insufficient, tell the user politely that this is not covered in the training document. Always behave professionally.

4. Model Selection

The generator model used is meta-llama/Llama-3.1-8B-Instruct hosted on HuggingFace Inference API. It is an instruction-tuned LLM capable of producing coherent and grounded answers when provided with context. It was integrated with LangChain’s ChatHuggingFace wrapper.

5. Streamlit User Interface

The chatbot was deployed with Streamlit. Key features include:

- First greeting message from assistant
- Streaming (token-by-token) responses for natural chat experience
- Modern chat bubbles UI (assistant left, user right)
- Expandable panel displaying retrieved source chunks
- Sidebar showing model info and clear chat option

6. Example Queries

- Q: tell me the capital of India – ■ Answered: 'This information is not covered in the provided training document.'

- Q: so what can you tell me? – ■ Provided a summary of the document context.
- Q: What is the purpose of this training document? – ■ Retrieved and explained training objectives.
- Q: Who created this chatbot? – ■ Answered: 'This information is not covered in the provided training document.'
- Q: what are various disputes – ■ Listed disputes such as user vs eBay claims, access to services, agent actions, product sales, arbitration scope, pre-existing disputes, and intellectual property infringements.

7. Limitations & Improvements

- May occasionally hallucinate if context is weak. - HuggingFace inference API introduces response latency (3–6 seconds). - Streaming is simulated word-by-word since HuggingFace API does not natively support live token streaming. - Currently limited to one document; scaling to multiple requires re-embedding.

Future Improvements:

- Implement true token-level streaming
- Support multiple documents with metadata-based filtering
- Deploy on Streamlit Cloud or HuggingFace Spaces
- Add optional user authentication

8. Conclusion

This project successfully implements an end-to-end RAG chatbot pipeline. It demonstrates skills in data preprocessing, embeddings, retriever-generator architecture, prompt engineering, and deployment via a user-friendly Streamlit UI. The chatbot is capable of providing grounded and factual responses from the training document while offering a professional chat experience.

Prepared by: Pulkit Chauhan