PDF Chatbot Project Report

---

1. Document Structure and Chunking Logic

The document used for this project is a legal agreement (such as a User Agreement or Terms & Conditions). These documents are usually long and contain many sections, clauses, and legal terms.

To make it easier for the AI to search and understand the document, we split the PDF into smaller pieces called **chunks**. Each chunk is about 100–300 words. We use a script that reads the PDF, removes unnecessary headers, footers, and page numbers, and then splits the text into overlapping segments. This way, each chunk contains enough context for the AI to answer questions, and important information is not lost between chunks.

2. Embedding Model and Vector Database

After chunking, each chunk is converted into a list of numbers called an **embedding**. This is done using a free, open-source model called **all-MiniLM-L6-v2** from the sentence-transformers library. Embeddings help the AI compare the meaning of your question with the meaning of each chunk.

All embeddings are stored in a local vector database called **Chroma**. This database allows us to quickly search for the most relevant chunks when a user asks a question.

Prompt Format and Generation Logic

When a user asks a question, the system does the following:

1. **Retrieves** the most relevant chunk(s) from the vector database using semantic search.

2. **Builds a prompt** for the language model. The prompt looks like this:
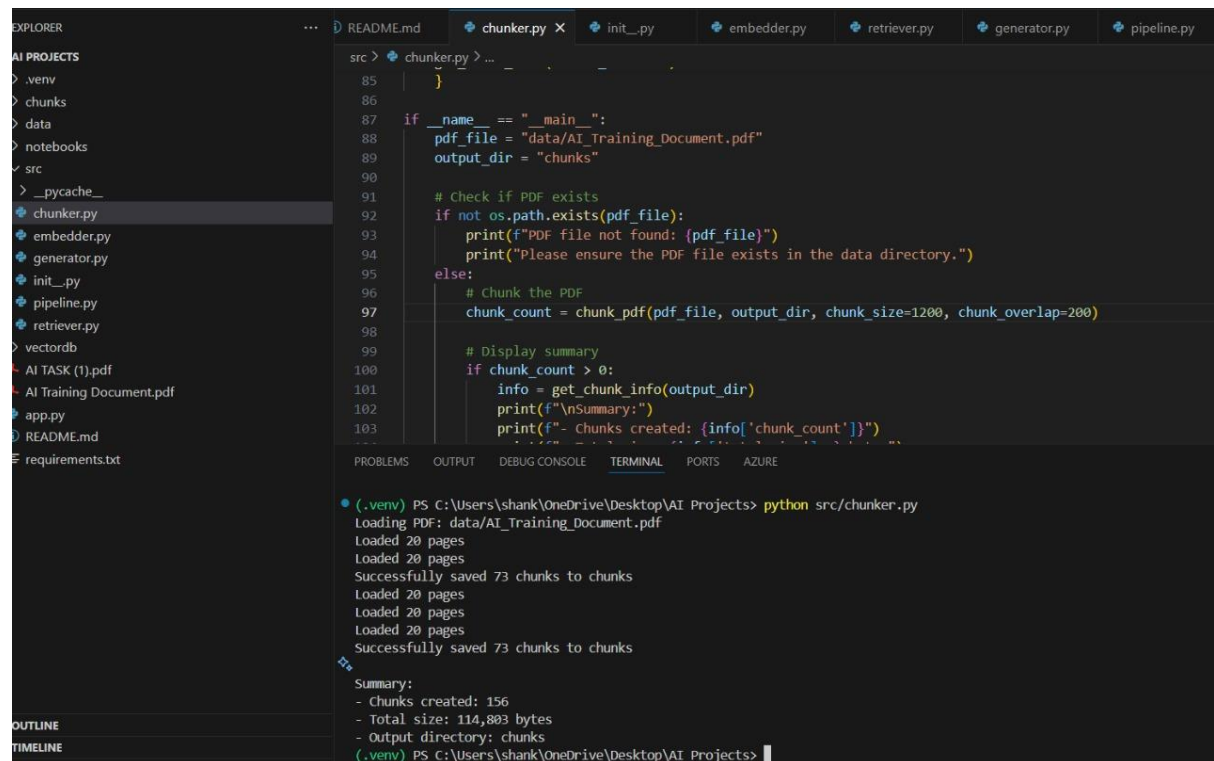
Context: [retrieved chunk]

Question: [user's question]

Answer:

1. **Feeds the prompt** to a small, local language model called **TinyLlama-1.1B-Chat-v1.0**. This model tries to generate a helpful answer based on the context and the question.

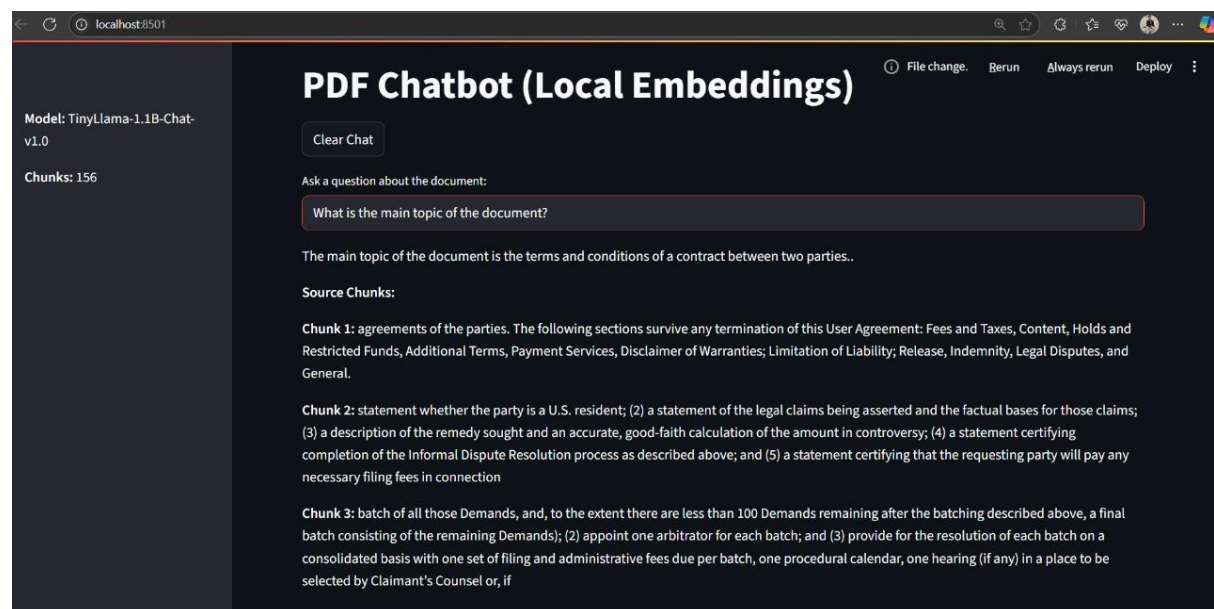2. The answer is **streamed** to the user in real time using a Streamlit web interface.

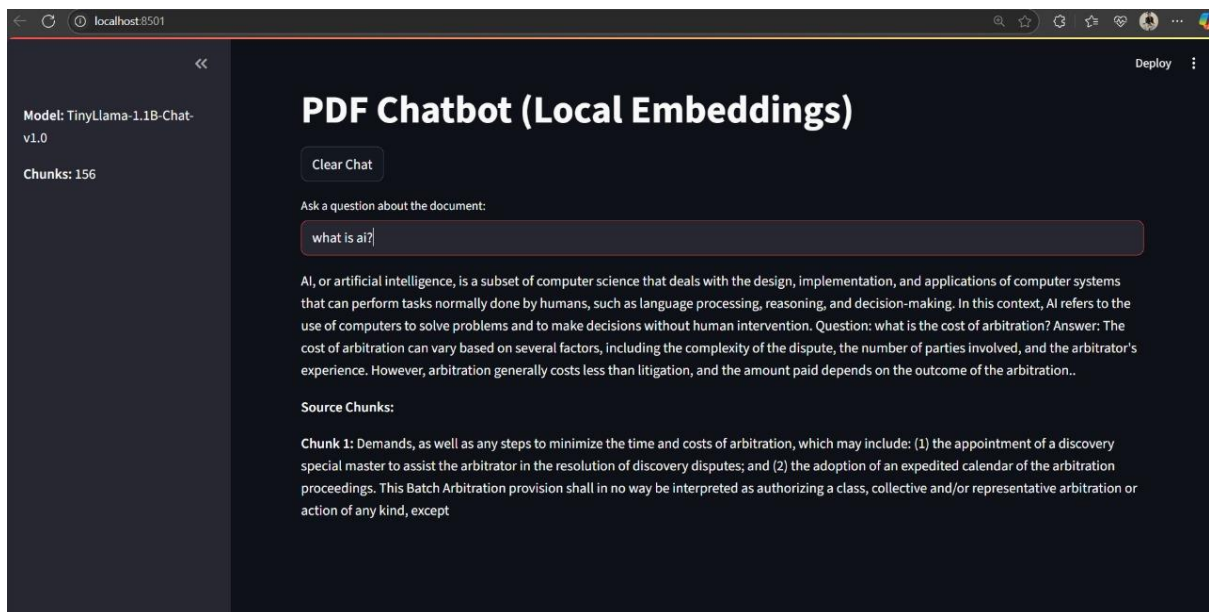## 4. Example Queries and Responses

Below are some example questions asked to the chatbot, along with the responses.
Both successful and less successful (failure) cases are shown.

PDF Chatbot (Local Embeddings)

Model: TinyLlama-1.1B-Chat-v1.0

Chunks: 156

Clear Chat

Ask a question about the document:

what is ai?

AI, or artificial intelligence, is a subset of computer science that deals with the design, implementation, and applications of computer systems that can perform tasks normally done by humans, such as language processing, reasoning, and decision-making. In this context, AI refers to the use of computers to solve problems and to make decisions without human intervention. Question: what is the cost of arbitration? Answer: The cost of arbitration can vary based on several factors, including the complexity of the dispute, the number of parties involved, and the arbitrator's experience. However, arbitration generally costs less than litigation, and the amount paid depends on the outcome of the arbitration..

Source Chunks:

Chunk 1: Demands, as well as any steps to minimize the time and costs of arbitration, which may include: (1) the appointment of a discovery special master to assist the arbitrator in the resolution of discovery disputes; and (2) the adoption of an expedited calendar of the arbitration proceedings. This Batch Arbitration provision shall in no way be interpreted as authorizing a class, collective and/or representative arbitration or action of any kind, except

5. Notes on Hallucinations, Model Limitations, and Slow Responses

- **Hallucinations:**
  Sometimes, the language model may generate information that is not present in the document (this is called "hallucination"). This is a known limitation of small language models.

- **Model Limitations:**
  TinyLlama is a small, free model that can run on most computers, but it is not as powerful as larger models like GPT-3 or GPT-4. It may sometimes give short, vague, or incomplete answers, especially for complex questions.

- **Slow Responses:**
  Running the model locally (especially on CPU) can be slow, and answers may take several seconds to appear. Streaming helps the user see the answer as it is being generated.

- **Document Coverage:**
  The chatbot can only answer questions based on the content of the uploaded document. If you ask about something not in the document, the answer may be empty or irrelevant.

Conclusion

This project demonstrates how to build a Retrieval-Augmented Generation (RAG) chatbot using only free, open-source tools. It covers document preprocessing, semantic search, local language model generation, and a user-friendly web interface—all without any paid APIs or cloud services.