Training Day 16 Report

Overview

This report demonstrates how a Retrieval-Augmented Generation (RAG) pipeline works in practice through a simple use case: answering user questions from a collection of uploaded documents using retrieval and generation.

Use Case: Chatbot with Custom PDF

Objective: Build a chatbot that answers questions about a PDF document (e.g., an academic paper or user manual) using a RAG pipeline.

Step-by-Step Flow

1. **Upload Document:** The user uploads a PDF file to the system.
2. **Text Extraction:** The document is parsed and split into smaller text chunks.
3. **Embedding Generation:** Each chunk is converted into a vector using a pre-trained embedding model (e.g., Sentence Transformers).
4. **Indexing:** The vectors are stored in a vector database (e.g., FAISS, ChromaDB).
5. **User Query:** The user types a natural language question.
6. **Retrieval:** The system finds top-k relevant chunks using vector similarity.
7. **Generation:** The retrieved chunks and user query are sent to an LLM, which produces a grounded response.

Sample Interaction

- **User:** ”What are the key findings of the report?”
- **Retrieved:** Paragraphs from the conclusion and results section.
- **LLM Output:** ”The report highlights significant growth in renewable energy adoption and outlines policy recommendations for emerging economies.”

Tools Used

- **Text Splitter:** LangChain or custom chunker.
- **Embeddings:** Sentence Transformers, OpenAI embeddings, or Hugging Face.
- **Vector Store:** FAISS or Chroma.
- **LLM:** OpenAI, Anthropic, or similar APIs.

Conclusion

This example demonstrates how RAG enables question answering over private or domain-specific documents. It extends the power of LLMs by grounding answers in actual data, making AI more reliable and useful in custom applications like chatbots, assistants, and enterprise search.