Technical Report: RAG-Based Document Chatbot

Submitted by: Bavishya Sankaranarayanan

Task Deadline: April 7, 2025

Objective:

Build a chatbot that retrieves relevant document content and generates structured, context-aware responses.

Overview:

This project implements a Retrieval-Augmented Generation (RAG) chatbot that enables interactive querying over a single PDF document. It semantically retrieves the most relevant content chunks using vector similarity and generates well-structured responses while avoiding hallucinations.

Tech Stack & Design Choices:

Component	Technology/Library	Rationale
Document Parsing	PyPDF2	Lightweight and accurate PDF extraction
Text Chunking	langchain.text_splitter	Ensures semantic continuity across chunks
Embedding Model	TF-IDF (Scikit-learn)	Fast and local, no API dependency
Vector Store	FAISS (Facebook AI Similarity Search)	Efficient nearest-neighbor retrieval
LLM (Option 1)	OpenAI GPT-3.5 via openai	High-quality structured generation
LLM (Option 2)	Ollama - LLaMA 3 (Local)	Completely offline generation
UI	Streamlit	Clean web interface with file upload & Q/A

System Workflow:

1. User uploads a PDF.
2. Text is extracted and chunked (e.g., 500 tokens with 50 overlap).
3. Chunks are embedded using TF-IDF, and indexed in FAISS.

4. User enters a query ? query is embedded ? Top K (e.g., 3) similar chunks retrieved.

5. Retrieved context and query are passed to an LLM (OpenAI/Ollama).

6. LLM generates a structured response (bullets, tables, or paragraphs).

Response Structuring Approach:

Prompted the LLM with:

"Answer only based on the context. Use bullet points, tables, or short paragraphs. If unsure, say 'This information is not available in the document.'"

Ensures:

- Hallucination reduction
- Answer clarity
- Structured output for easier understanding

Hallucination Control:

- Limited the LLM's scope strictly to retrieved context.
- Fallback clause in prompt: "If unsure, say: 'This information is not available in the document.'"

Challenges Faced & Solutions:

Challenge	Solution
Compatibility issues w/ transformers	Replaced with TF-IDF + FAISS
OpenAI API version breaking	Migrated to new openai>=1.0.0 syntax
Local LLM support	Integrated with Ollama for llama3
Long document latency	Used chunking + top-K filtering

Testing:

Tested using multiple types of PDF documents:

- Reports with summaries and authorship metadata
- Academic-style papers

- Simulated business memos

All tests confirmed:

- Answers were grounded in the uploaded document
- Incorrect or missing answers triggered fallback
- Responses were well-structured

How to Run:

pip install -r requirements.txt

Set OpenAI Key (if using OpenAI):

export OPENAI_API_KEY=your-key-here

Run the App:

streamlit run ui/app.py

Offline Ollama Version:

ollama run llama3

Deliverables:

- Source Code with documentation
- Instructions to run (OpenAI & Ollama)
- Technical Write-up (this report)
- Streamlit UI + Offline Local Support

Conclusion:

This chatbot leverages efficient semantic retrieval and structured natural language generation to create a document-aware assistant. It is modular, flexible, and ready for real-world deployment.