Conversational RAG Chatbot using Local LLaMA

Author: AKHIL P M

Date: 19 April 2025

Abstract: This project demonstrates a Conversational Retrieval-Augmented Generation (RAG) system that uses LangChain and local LLaMA (tested via Kaggle) for answering questions based on custom documents. Due to local hardware limitations (4GB RAM), the local LLaMA model was implemented and tested using Kaggle Notebook.

# 1. Introduction & Objective

The objective of this project is to create a conversational AI system that can intelligently answer user queries based on a collection of documents. This system uses a RAG (Retrieval-Augmented Generation) approach to combine document retrieval with generative AI capabilities. The goal is to build an end-to-end pipeline that loads documents, preprocesses them, embeds them, retrieves the most relevant chunks, and uses a language model to generate responses.

# 2. Methodology

The methodology includes several core steps:  
Document Loading : Documents are loaded using LangChain's directory loader.  
Chunking : Documents are split into manageable chunks using token-aware chunking based on tiktoken.  
Embedding : Embeddings are generated using Hugging Face's `nomic-ai/nomic-embed-text-v1 ` model and stored in Chroma DB.  
Retriever : A similarity-based retriever is used to find the top relevant chunks.  
LLM : The conversational generation uses the Grok API. Due to hardware limitations, the local LLaMA model was tested on a Kaggle Notebook local LLM code commented in file.  
Memory : LangChain's `ConversationBufferMemory` is used to maintain chat history.

# 3. System Architecture & Workflow

The architecture consists of the following modules:  
1. Document Loader →  
2. Chunker →  
3. Embedder →  
4. Vector Store (Chroma) →  
5. Retriever →  
6. prediction (Grok API or Local LLaMA via Kaggle) →  
7. app (Gradio)  
  
The system uses a Gradio-based interface for user interaction. A reset button clears the chat context, allowing fresh conversations.

# 4. Challenges & Solutions

Hardware Limitations : The local system only had 4GB RAM, which is insufficient to run LLaMA models locally.  
Solution : Used Kaggle Notebooks to simulate the local LLaMA integration and test its compatibility.  
Context Management : Managing conversation memory dynamically.  
Solution : Used LangChain's memory module and implemented a manual reset button in the Gradio UI.

# 5. Conclusion & References

This project successfully demonstrates a production-ready RAG-based chatbot with modular structure, custom document ingestion, and conversational memory. The use of Kaggle for LLaMA simulation proves the feasibility of future local deployment when hardware allows.

References :  
- LangChain Documentation  
- Hugging Face Transformers  
- Chroma Vector Store  
- Grok API  
- Gradio Library  
- Kaggle Notebooks  
- Chatgpt