Custom Chatbot Q&A Documentation

Overview

This application is a document-based chatbot that allows users to upload PDF or TXT files and ask questions related to their content. It leverages LangChain for text processing and ChromaDB for vector storage, enabling retrieval-augmented generation (RAG). The responses are generated using a local LLM (Ollama) without requiring any external API keys.

Features

- **Document Upload**: Supports PDF and TXT files.
- **Context-Aware Q&A**: Answers questions based on the uploaded document content.
- **Retrieval-Augmented Generation (RAG)**: Uses embeddings and vector search to retrieve relevant information before generating responses.
- **Interactive UI**: Streamlit interface with chat-like interaction.
- **Local Embeddings**: Uses HuggingFace embeddings for semantic understanding.
- **Custom Chat Layout**: User messages appear on the right, bot messages on the left.

Architecture

1. **Document Loading**
    - Handles PDF and TXT file formats.
    - Extracts raw text for processing.
2. **Text Splitting**
    - Documents are divided into smaller chunks to maintain context.
    - Overlapping chunks ensure continuity for better understanding.
3. **Vector Store**
    - Each chunk is transformed into vector embeddings.
    - ChromaDB stores vectors for fast semantic search.
4. **Retrieval**
    - User queries are converted into embeddings.
    - Similar document chunks are retrieved using vector similarity.
5. **Answer Generation**

- The LLM (Ollama) generates answers based on the retrieved context.
- Responses are constrained to the document content for accuracy.

6. **Chat Interface**
    - Real-time chat with messages stored in session state.
    - User messages and bot responses are displayed in a visually distinct format.

Workflow

1. User uploads a PDF or TXT document.
2. The system extracts text and splits it into chunks.
3. Chunks are embedded and stored in ChromaDB.
4. User submits a question via the chat interface.
5. The system retrieves relevant chunks using semantic search.
6. The LLM generates a response based solely on retrieved content.
7. The chat interface displays the conversation in real-time.

Usage

- Run the application via Streamlit.
- Upload a document and wait for processing.
- Ask questions in the chat input.
- View responses instantly in the chat interface.

Notes

- The system is fully local and does not require API keys.
- Uploaded files are temporarily stored on the server.
- Each document creates a new vectorstore to isolate context.
- Ideal for small to medium documents; larger datasets may require optimization.

Potential Improvements

- Support for multiple documents and combined context.
- Pagination or history management for long conversations.

- Enhanced UI with avatars, timestamps, or typing indicators.
- Embedding caching to avoid reprocessing repeated documents.