Project Report: Document Reader Chatbot with Chroma Database

Approach Taken

Script 1: Data Preparation and Database Integration

The first script focuses on preparing data and integrating it into a Chroma database using LangChain and OpenAI technologies. It follows these steps:

1) Data Loading: Markdown documents are loaded from a specified directory (data/books) using LangChain's DirectoryLoader.
2) Text Chunking: Documents are split into smaller chunks for efficient processing using LangChain's RecursiveCharacterTextSplitter.
3) Database Integration: Chunks are then stored in a Chroma database using Chroma.from_documents with OpenAIEmbeddings for contextual indexing.

Script 2: Streamlit Application Development

The second script develops a Streamlit application for user interaction with the integrated database and chatbot:

1) Environment Setup: Environment variables are loaded using dotenv for security, including the OpenAI API key.
2) User Interface: A Streamlit interface allows users to input questions.
3) Data Retrieval: Upon user query, the application retrieves relevant information from the Chroma database using db.similarity_search_with_relevance_scores.
4) Response Generation: The retrieved context is used as a template for generating responses through OpenAI's chatbot model (ChatOpenAI).

Challenges Faced

1) Data Volume: Handling large volumes of text data efficiently required optimizing text chunking and database storage strategies.
2) Metadata Handling: Ensuring accurate metadata extraction and storage alongside text chunks posed initial challenges.
3) Integration Complexity: Integrating multiple libraries (LangChain, OpenAI) and ensuring compatibility posed initial development hurdles.
4) API Key Management: Securely managing and integrating the OpenAI API key for deployment was critical.

Solutions Implemented

1) Optimized Chunking: Adjusted chunk sizes and overlap parameters in RecursiveCharacterTextSplitter for better performance.
2) Enhanced Metadata Handling: Improved metadata extraction methods to ensure accurate document indexing and retrieval.
3) Library Compatibility: Resolved integration issues by updating dependencies and ensuring version compatibility.
4) Secure API Key Handling: Implemented secure environment variable management with dotenv for seamless deployment.

Conclusion

The integration of LangChain and OpenAI in developing a Streamlit application with Chroma database integration offers a robust solution for efficient information retrieval and chatbot interaction. Overcoming initial challenges through optimized data handling and secure deployment practices ensures a reliable user experience.