Book Recommendation System

Softnerve Technology Private Limited

Approach-

Initially, the solution was implemented by using LangGraphs but the generated result was very disappointing.

Then, Open-source LLMs from HuggingFace were used. Most of the popular highly efficient models like Mistral, Llama and Google Gemma are Gated models and getting license for it would take some time. Non-gated ones like GPT2 and Falcon could be implemented successfully but my system could only render models around 7B parameters. These 7B models gave better but below average text generated results not to mention the extremely long runtime.

Finally, Google Gemini API was used with gemini-pro as the base model. It delivered results above expectation with good speed and quality. Hence, this approach was considered most efficient.

Final Description-

- **Langchain and Google Generative AI Integration**: The code uses Langchain, a framework for developing applications powered by language models, in combination with Google's Generative AI (specifically, the 'gemini-pro' model). It leverages the ChatGoogleGenerativeAI class from langchain_google_genai to interact with the AI model. The code sets up prompt templates and LLM chains using Langchain's PromptTemplate and LLMChain classes, allowing for structured interactions with the AI model for generating book recommendations and details.
- **Streamlit for User Interface**: The application's frontend is built using Streamlit, a Python library for creating web apps with minimal code. Streamlit components like st.title(), st.selectbox(), st.button(), and st.write() are used to create an interactive interface. The app allows users to select book genres, request top 100 and top 10 book lists, and get details about specific books. Streamlit's session state (st.session_state) is utilized to maintain data between user interactions.
- **Python-based Backend Logic**: The backend logic is implemented in Python, orchestrating the flow between user inputs, AI model interactions, and result

displays. It uses conditional statements to control the flow of the application (e.g., ensuring the top 100 books are fetched before the top 10). The code also employs Python's string formatting and list manipulation to process the AI-generated responses and present them in a user-friendly format.