AI Gift Recommendation with RAG and Vector Database - Report

Objective

The goal of this project is to develop a domain-specific application that combines the strengths of a Large Language Model (LLM) for understanding and processing natural language queries with the efficiency of a vector database for data storage and retrieval. The application focuses on providing personalized recommendations based on user inputs using Retrieval-Augmented Generation (RAG).

Approach

1. **Domain Selection:**
   The chosen domain is gift recommendations based on a predefined dataset.

2. Data Collection and Preprocessing:
   A CSV file containing relevant data for gift recommendations was provided. The dataset includes columns such as age, gender, relationship, occasion, budget, max budget, gift, rating, link, image link, and interest.
   Mock data was manually generated based on the structure and content of the dataset.

3. Vector Database Implementation:
   FAISS (Facebook AI Similarity Search) was chosen as the vector database due to its efficiency in handling large-scale similarity search.
   The mock data was split into text chunks using the RecursiveCharacterTextSplitter.
   These chunks were then converted into embeddings using Google Generative AI and stored in the FAISS vector database.

4. Application Development:
   A Streamlit application was developed to provide a user interface for inputting natural language queries.
   The backend logic was implemented to use LangChain for query processing and the FAISS vector database for data retrieval.
   The application includes functionalities for generating and storing mock data as well as querying the stored data.

5. Evaluation and Testing:
   The application was tested with various queries to evaluate its performance and accuracy.
   Example queries were provided to ensure the chatbot's responses were sensible and adhered to the constraints provided by the user.

Challenges Faced and Solutions

1. Data Preprocessing:
   Challenge: Ensuring the dataset was clean and suitable for indexing.
   Solution: Filled missing values with empty strings and combined relevant columns into a single text field for each entry.

2. Vector Database Storage:
   Challenge: Efficiently storing and retrieving text chunks using embeddings.
   Solution: Used FAISS for its efficient handling of large-scale similarity search and integrated it with Google Generative AI for generating embeddings.

3. Prompt Engineering:
   Challenge: Creating prompts that would generate detailed and accurate mock data.
   Solution: Manually generated mock data based on the dataset structure to ensure it accurately represented real-world scenarios.

4. User Interaction:
   Challenge: Developing an intuitive user interface for inputting queries and receiving responses.
   Solution: Used Streamlit to create a simple and interactive user interface, allowing users to input questions and view responses seamlessly.

Implementation Details

1. Environment Setup:
   Loaded environment variables using `dotenv` and configured Google Generative AI with the API key.

2. Mock Data Generation:
   Manually created mock data entries based on the dataset structure.

3. Text Splitting and Storage:
   Split mock data into chunks using RecursiveCharacterTextSplitter.
   Stored these chunks in a FAISS vector store after converting them into embeddings.

4. Conversational Chain:
   Defined a prompt template and created a conversational chain using LangChain and Google Generative AI.

5. User Interaction:
   Set up Streamlit to allow users to input queries and generate/store mock data.
   Processed user queries by retrieving relevant documents from the vector store and generating responses using the conversational chain.

Conclusion

The project successfully developed a domain-specific recommendation system that combines LLM capabilities with the efficiency of a vector database. The chatbot can handle user queries, generate sensible and accurate responses, and provide relevant recommendations based on a predefined dataset. The approach taken ensures that the system meets the specified requirements and provides a robust solution for personalized recommendations.