TrendSpotter AI

Project Goals

The primary goal of this project is to build a recommendation bot that suggests fashion products based on user queries. The system leverages semantic search and retrieval-augmented generation (RAG) to provide relevant fashion product recommendations, including detailed information about price, description, and average rating.

Model Used:

Embedding Model

- **Model Name:** paraphrase-MiniLM-L6-v2
- **Library:** Sentence Transformers
- **Purpose:** Generate embeddings for fashion product descriptions to facilitate efficient similarity searches.

Cross-Encoder

- **Model Name:** cross-encoder/ms-marco-MiniLM-L-6-v2
- **Library:** Sentence Transformers
- **Purpose:** Re-rank search results based on the relevance of the product descriptions to the user query.

Language Model

- **Model Name:** GPT-3.5-turbo
- **Library:** OpenAI via LangChain
- **Purpose:** Generate natural language responses based on the search results and user queries.

Data Sources

- **Source File:** FashionDatasetv2.csv
- **Columns:**
  - **p_id:** Unique product identifier
  - **name:** Product name
  - **products:** Product category
  - **price:** Price of the product
  - **colour**: Color of the product
  - **brand:** Brand of the product
  - **img:** Image URL of the product
  - **ratingCount:** Number of ratings
  - **avg_rating:** Average rating
  - **description:** Detailed product description
  - **p_attributes:** Product attributes

Key Design Decisions

Data Preprocessing

- **Cleaning:** HTML tags and special symbols were removed from product descriptions using regex.
- **Metadata Creation:** Each product entry was enriched with metadata containing p_id, products, price, avg_rating, and description.

Embedding and Indexing

- **Persistent Storage:** ChromaDB was used for storing and retrieving embeddings, ensuring efficient query processing.
- **Batch Processing:** Product descriptions and their metadata were batch processed and stored in ChromaDB for similarity searches.

Cache Mechanism

- **Implementation:** A caching mechanism was implemented to store and retrieve frequent queries to reduce response time.
- **Threshold:** A threshold was set to determine whether to use cached results or query the main collection.

Search and Retrieval

- **Similarity Search:** LangChain's Chroma was used for performing similarity searches based on user queries.
- **Cross-Encoder Re-Ranking:** Results from the similarity search were re-ranked using a Cross-Encoder model to improve relevance.

Prompt Engineering

- **Custom Prompt:** A detailed prompt was crafted to guide the language model in generating precise and helpful responses based on search results.
- **Template Usage:** LangChain's PromptTemplate was utilized to structure the prompt dynamically based on user queries and search results.

Challenges and Solutions

Data Cleaning

- **Challenge:** Managing inconsistent or missing data in the dataset.
- **Solution:** Implemented robust preprocessing scripts to clean and format the data consistently, ensuring high-quality input for the models.

Model Integration

- **Challenge:** Ensuring compatibility and seamless integration between different libraries and models.
- **Solution:** Thoroughly reviewed documentation and tested integrations iteratively to resolve any compatibility issues.

Performance Optimization

- **Challenge:** Optimizing the search and retrieval process to reduce latency.
- **Solution:** Implemented a caching mechanism and fine-tuned the threshold for cache usage to balance accuracy and performance.

Response Generation

- **Challenge:** Generating accurate and informative responses to user queries.
- **Solution:** Developed a detailed prompt and used a powerful language model (GPT-3.5-turbo) to generate high-quality responses based on the top-ranked search results.

Detailed Description of the Problem Statement

The Fashion Search AI project aims to address the challenge of finding and recommending fashion products based on user queries. Traditional e-commerce platforms often provide search functionality based on exact keyword matching, which can be limiting. Users may struggle to find products that match their preferences if the search terms they use do not exactly align with product descriptions or attributes.

Key Challenges:

1. **Ambiguity in User Queries:** Users may use varied and ambiguous language to describe their needs, which traditional search engines may struggle to interpret correctly.
2. **Diverse Product Descriptions:** Fashion products have rich and diverse descriptions, making it challenging to match user queries with relevant products effectively.
3. **Data Quality and Inconsistencies:** Fashion datasets often have inconsistencies, missing values, and varied formats in product descriptions, making preprocessing and search difficult.
4. **Performance and Scalability:** Efficiently searching through a large number of product descriptions and providing relevant results in a timely manner is crucial for user satisfaction.

Why LangChain is an Ideal Framework?

LangChain is a powerful framework that integrates various language models and tools, making it ideal for building complex search and retrieval systems. Here's why LangChain is suitable for this project:

1. **Versatility:** LangChain allows for easy integration of different language models and tools, such as semantic search, cross-encoding, and generative responses, which are essential for providing accurate fashion product recommendations.
2. **Customizable Workflows:** With LangChain, you can create customized workflows for processing queries, performing similarity searches, and generating responses. This flexibility is crucial for handling the diverse nature of fashion product descriptions and user queries.
3. **Enhanced Search Capabilities:** LangChain supports integration with various vector databases (e.g., ChromaDB) and allows for advanced search techniques, such as semantic search and re-ranking, which improve the relevance of search results.

Evaluation Parameters

To assess the effectiveness of the Fashion Search AI system, several evaluation parameters are considered:

1. **Accuracy of Search Results:**
   - **Relevance Score:** Evaluate the relevance of the search results based on how well they match the user query. This can be assessed using metrics like precision and recall.
   - **Cross-**Encoder Scores: Use cross-encoder models to re-rank search results and ensure the top results are highly relevant to the query.
2. **Response Quality:**
   - **Clarity and Informativeness:** Assess whether the generated responses are clear, informative, and provide relevant details such as product name, price, and description.
   - **User Satisfaction:** Collect feedback from users regarding the usefulness and accuracy of the responses provided by the AI system.
3. **Performance Metrics:**
   - **Response Time:** Measure the time taken to retrieve and generate responses for user queries. Faster response times contribute to a better user experience.
   - **Scalability:** Evaluate the system's ability to handle increasing volumes of data and user queries efficiently without significant performance degradation.
4. **Data Handling:**
   - **Data Completeness:** Ensure that the preprocessing steps adequately handle missing values and inconsistencies in the dataset.
   - **Data Accuracy:** Verify that the indexed data accurately reflects the original product information and metadata.
5. **Cache Efficiency:**
   - **Cache Hit Rate:** Measure the effectiveness of the caching mechanism by assessing the proportion of queries that are served from the cache versus the main collection.
   - **Cache Response Time:** Evaluate the response time for queries served from the cache compared to those processed directly from the main collection.