LLM DATA ENGINEER PRE-ASSIGNMENT

I. Overview

This document provides an in-depth look at a sophisticated data processing system designed to handle and analyze large volumes of textual data as part of the **LLM DATA ENGINEER PRE-ASSIGNMENT**. It details the workflow from initial data ingestion and preprocessing, through vectorization, to querying and generating responses using advanced NLP techniques.

System Setup and Requirements

Dependencies:

- **Python Packages:** pandas, numpy, spacy, annoy, groq
- **Installation:** Requires the installation of Python libraries and setting up environment variables for API access.

Installation Instructions:

- Ensure Python and pip are installed.
- Use the command
  ```
  pip install -r requirements.txt
  ```
  to install the required Python libraries.

II. Data Ingestion and Preprocessing

Purpose & Design Choices:

- **Objective:** To standardize and clean the dataset, ensuring it is suitable for NLP operations and analytics.
- **Rating Normalization:** Ratings outside the standard range (1-5) are adjusted to the median to prevent data skew.
- **Date Standardization:** Ensures consistency in reporting and temporal analysis.
- **Branch Name Standardization:** Reduces variability within the dataset, ensuring that data aggregation and analysis are accurate.

Trade-offs:

- **Data Accuracy vs. Processing Speed:** While replacing outlier ratings with the median improves the dataset's robustness, it may also mask true data variations, potentially impacting analytical outcomes.
- **Detailed Text Cleaning:** Essential for NLP tasks, but requires additional processing time.

Schema Details

1. Fields and Data Types:

- **Review_ID (Integer):** Stores the unique review id.
- **Rating (Integer):** Stores the user rating. Values outside the 1-5 range are adjusted to the median value to maintain consistency and reliability in the dataset.
- **Year_Month (String):** Standardized to a YYYY-MM format to facilitate time-series analysis and ensure uniformity across entries.
- **Branch (String):** Contains standardized names of locations to ensure consistency. Variations like "Paris", "Hongkong", and "California" are standardized to "Disneyland_Paris", "Disneyland_HongKong", and "Disneyland_California", respectively.
- **Review_Text (Text):** The textual content of the review. It undergoes extensive cleaning to remove non-alphanumeric characters and unnecessary whitespace, and is converted to lowercase to standardize for text processing.
- **Reviewer_Location (String):** The geographical location of the reviewer, which could be used for demographic analysis.
-

Column	Data Type	Description
Review_ID	Integer	Unique review id
Rating	Integer	Adjusted ratings between 1 and 5
Year_Month	String	Standardized date format (YYYY-MM)
Branch	String	Standardized branch names
Review_Text	Text	Cleaned and standardized review text
Reviewer_Location	String	Location of the reviewer

How to Run:

- **Command:**

```
python IngestionAndPreprocessing/Data_Ingestion_Pipeline.py
IngestionAndPreprocessing/DisneylandReviewsAll.csv
```

- **Expected Output:** A cleaned and standardized data.db ready for further processing.

III. Vectorization

Purpose & Design Choices:

- **Using spaCy:** Chosen for its balance between performance and accuracy in generating word embeddings.
- **Annoy Indexing:** Utilized for its efficiency in building and querying large-scale high-dimensional data.

Trade-offs:

- **Memory Usage vs. Search Speed:** Annoy trades off increased memory usage for faster retrieval times, beneficial in real-time applications.

How to Run:

- **Command:**
  ```
  python Vectorization/Vector_Embedding.py
  ```
- **Expected Output:** An Annoy index file (embeddings.ann) that stores the vector representations of the preprocessed text data.
-

IV. Query, Retrieval, and RAG

Purpose & Design Choices:

- **Integration with Groq:** Enhances the system's ability to not only retrieve relevant data but also generate context-aware responses using state-of-the-art language models.
- **Sentiment Analysis:** Provides an added layer of data interpretation, crucial for understanding user sentiments and tailoring responses accordingly.

Trade-offs:

- **Complexity vs. User Experience:** The integration of advanced AI models increases system complexity but significantly enhances the user interaction quality.

How to Run:

- **Query and Retrieve Command:**
  ```
  python QueryingRetrievalAndRAG/Query.py
  ```

- **User Interaction:** Users input a query, the system retrieves the most relevant record, performs sentiment analysis, and uses RAG for generating a detailed response.
-

V. Conclusion

This document serves as a comprehensive guide to setting up and operating a robust data processing system capable of handling complex NLP tasks. By following the outlined procedures, users can effectively process, analyze, and generate insights from large text datasets.