Project Report: Text-to-SQL with Live User Interaction

1. Introduction

The primary objective of this project is to create an efficient Text-to-SQL system by integrating MySQL, LangChain, and Language Models (LLM). The ultimate goal is to establish a seamless pipeline that enables users to interact with a database effortlessly through natural language queries. The project aims to handle real-time user-input questions, employing advanced Language Models to generate database queries, and subsequently retrieve accurate answers.

2. Rationale for LLM Selection(OpenAI's GPT-3.5-turbo-instruct, Hugging Face's Google Flan-T5-XXL Model)

2.1 Pre-trained Knowledge

The OpenAI model was chosen for the following reasons: OpenAI model is pre-trained on a vast corpus of text, which includes a diverse range of topics and domains. This pre-trained knowledge allows the model to understand and respond to a wide array of user queries without requiring specific training on the product database.

2.2 Text-Based QA Capabilities

The OpenAI model is well-known for its strong performance in text-based question-answering tasks. Its ability to analyze and comprehend textual information makes it suitable for our task of processing live user-input questions and retrieving relevant answers from the product database.

2.3 OpenAI API Integration

Integration with the OpenAI API provides real-time access to powerful language understanding capabilities. This allows our QA system to dynamically interact with the OpenAI model, ensuring that it stays updated with the latest information and optimizations.

2.4 Flexible Text Input Handling

The OpenAI model can handle a variety of text inputs, including natural language queries. This flexibility is crucial when dealing with diverse user queries about product information in different formats.

2.5 No Specific Database Training Required

The model's pre-trained nature eliminates the need for specific training on the product database structure. It can adapt to the database content without prior knowledge, making it suitable for dynamic and evolving databases.

2.6 Continuous Learning

The OpenAI model allows for continuous learning, enabling our QA system to improve over time as it interacts with users and retrieves answers. This aligns with the evolving nature of product databases and user query patterns.

NOTE: I have tried using some model, but code is written for this two model only(OpenAI's GPT-3.5-turbo-instruct, Hugging Face's Google Flan-T5-XXL Model)

3. Langchain

LangChain is utilised in the project to facilitate seamless communication between the OpenAI model, responsible for language understanding, and the MySQL database for data retrieval. It acts as a bridge, ensuring that user queries are processed effectively and that the retrieved information is presented accurately.

SQLDatabase and SQLDatabaseChain

LangChain provides the SQLDatabase and SQLDatabaseChain classes, which are instrumental in establishing connections to the MySQL database. These classes encapsulate the logic for database interactions, including executing queries, fetching results, and managing connections. The SQLDatabaseChain class, in particular, acts as a wrapper around the OpenAI model and the database, facilitating a seamless flow of information.

Dynamic Query Generation

LangChain allows for dynamic query generation based on the user's input and the output from the OpenAI model. The db_agent instance, created with SQLDatabaseChain, leverages LangChain to convert the interpreted user query into a valid SQL query. This dynamic generation ensures that the system can handle a wide range of user queries without requiring predefined SQL templates.

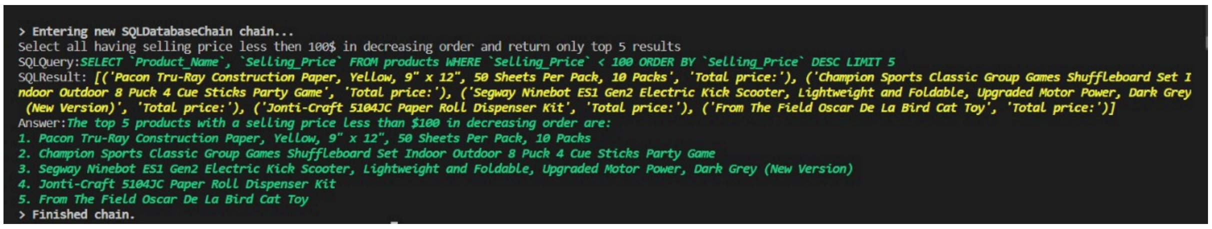
Error Handling and Verbose Mode

LangChain enhances the robustness of the system by providing error handling mechanisms. The verbose mode, enabled in the SQLDatabaseChain initialization, allows for detailed logging and reporting, making it easier to diagnose and resolve issues during the interaction between the LLM and the database.

LangChain plays a pivotal role in integrating the Language Model with the MySQL database in the Database Table-based QA Mechanism. Its features, including dynamic query generation, error handling, and standardized database access, contribute to the robustness and efficiency of the system. The synergy between LangChain and the OpenAI model ensures a smooth flow of information, enabling the system to provide accurate and relevant answers to user queries about the product database.

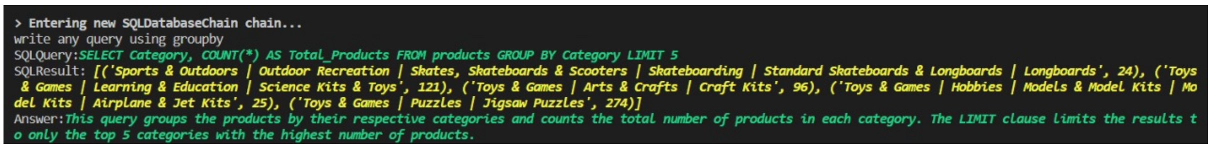
4. Result

OpenAI model(GPT-3.5-turbo-instruct) was giving cool and satisfactory results
And was very quick to give answers
Example 1:
i/p: Select all selling prices having SP less than 100$ in decreasing order return only 5 results.
o/p:



```
> Entering new SQLDatabaseChain chain...
Select all having selling price less then 100$ in decreasing order and return only top 5 results
SQLQuery:SELECT `Product_Name`, `Selling_Price` FROM products WHERE `Selling_Price` < 100 ORDER BY `Selling_Price` DESC LIMIT 5
SQLResult: [('Pacon Tru-Ray Construction Paper, Yellow, 9" x 12", 50 Sheets Per Pack, 10 Packs', 'Total price:'), ('Champion Sports Classic Group Games Shuffleboard Set I
ndoor Outdoor 8 Puck 4 Cue Sticks Party Game', 'Total price:'), ('Segway Ninebot ES1 Gen2 Electric Kick Scooter, Lightweight and Foldable, Upgraded Motor Power, Dark Grey
 (New Version)', 'Total price:'), ('Jonti-Craft 5104JC Paper Roll Dispenser Kit', 'Total price:'), ('From The Field Oscar De La Bird Cat Toy', 'Total price:')]
Answer:The top 5 products with a selling price less than $100 in decreasing order are:
1. Pacon Tru-Ray Construction Paper, Yellow, 9" x 12", 50 Sheets Per Pack, 10 Packs
2. Champion Sports Classic Group Games Shuffleboard Set Indoor Outdoor 8 Puck 4 Cue Sticks Party Game
3. Segway Ninebot ES1 Gen2 Electric Kick Scooter, Lightweight and Foldable, Upgraded Motor Power, Dark Grey (New Version)
4. Jonti-Craft 5104JC Paper Roll Dispenser Kit
5. From The Field Oscar De La Bird Cat Toy
> Finished chain.
```

Example 2:
i/p: write any query using groupby
o/p:



```
> Entering new SQLDatabaseChain chain...
write any query using groupby
SQLQuery:SELECT Category, COUNT(*) AS Total_Products FROM products GROUP BY Category LIMIT 5
SQLResult: [('Sports & Outdoors | Outdoor Recreation | Skates, Skateboards & Scooters | Skateboarding | Standard Skateboards & Longboards | Longboards', 24), ('Toys
 & Games | Learning & Education | Science Kits & Toys', 121), ('Toys & Games | Arts & Crafts | Craft Kits', 96), ('Toys & Games | Hobbies | Models & Model Kits | Mo
del Kits | Airplane & Jet Kits', 25), ('Toys & Games | Puzzles | Jigsaw Puzzles', 274)]
Answer:This query groups the products by their respective categories and counts the total number of products in each category. The LIMIT clause limits the results t
o only the top 5 categories with the highest number of products.
```

Instead Hugging Face Model was giving Poor results and also takes more time compared to OpenAI

5. References

Hugging Face: https://huggingface.co/
Langchain: https://python.langchain.com/docs/use_cases/qa_structured/sql
OpenAI API Documentation: https://beta.openai.com/docs/

6. Conclusion

The project demonstrates the successful integration of the LangChain framework, MySQL database, and OpenAI language model to create a Text-to-SQL system. The implemented agent can understand natural language queries and convert them into SQL commands for database manipulation. The report provides an overview of the setup, implementation, and examples of the Text-to-SQL system in action.