MID-TERM PROGRESS REPORT ON

DATADIALECT: BRIDGING USERS AND DATABASES

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY:

JASPREET SINGH (2004600)

KESHAV KUMAR ARRI (2004610)

UNDER THE GUIDANCE OF

MANPREET KAUR MAND

(JAN-MAY 2024)

Department of Computer Science and Engineering

GURU NANAK DEV ENGINEERING COLLEGE,

LUDHIANA

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Introduction to project

The rapid advancement of technology has opened up new possibilities in the field of natural language processing. This mid-term report presents the progress of our project, which aims to build an End-to-End Language Model (LLM) capable of interacting with a MySQL database to generate accurate and relevant responses. Our project leverages the power of advanced technologies such as Google Palm, Hugging Face embeddings, Langchain, and Chroma DB to create a robust and efficient question-answer system.

Our project's primary objectives are to create a user-friendly LLM application using Google Palm and Hugging Face embeddings, develop a secure communication interface with MySQL through Langchain's SQL database chain, utilize Chroma DB vector database for optimized storage and retrieval of embeddings, and apply few-shot learning techniques to enhance the system's adaptability and responsiveness.

We have made significant strides towards achieving these objectives. We have successfully experimented with Chroma DB for storage of embeddings, and we have created a notebook for experimentation of using Google Palm with Langchain. We have also developed a secure communication interface with the MySQL database through Langchain. These achievements mark the completion of two out of our four objectives. The MySQL database will serve as the backbone for storing and retrieving data. The interface will be designed to be user-friendly and intuitive, ensuring ease of use for end-users. This project, with its innovative approach and use of cutting-edge technologies, promises to revolutionize the way we interact with databases in the retail domain.

1.2 OBJECTIVES

1. Create a user-friendly Language Model (LLM) application using Google Palm and Hugging Face embeddings for seamless natural language interaction with MySQL databases.
2. Develop a secure communication interface with MySQL through Langchain's SQL database chain, ensuring data retrieval and an intuitive user experience.
3. Utilize Chroma DB vector database for optimized storage and retrieval of embeddings, ensuring scalability.
4. Apply few-shot learning techniques to enhance the system's adaptability and responsiveness in generating accurate answers from the database.

2. SYSTEM REQUIREMENT

The development of this project would require specific software and hardware components to ensure that the application is functional and user-friendly.

Software Requirements:

1. **Google Palm and Hugging Face Embeddings**: The Google Palm LLM model and the Hugging Face embeddings are required for natural language processing tasks. These tools allow the system to understand and generate human-like text, enhancing the user experience.

2. Langchain Framework: The Langchain framework is needed for integrating the various components of the system and providing a seamless interface for interaction with the MySQL database. This ensures that users can interact with the system easily and efficiently.

3. Chroma DB Vector Database: The Chroma DB vector database is required for optimized storage and retrieval of embeddings. This ensures that the system can handle large amounts of data and retrieve information quickly.

4. MySQL: The latest version of MySQL is needed for efficient data management. This ensures that data can be stored, retrieved, and updated efficiently.

5. Software Compatibility: All software components should be compatible with the operating system of the hardware. This ensures that all components of the system can function together without any issues.

6. Security Measures: Appropriate security measures should be in place to protect the data and the system from potential threats. This ensures that the system and its data are secure and reliable.

Hardware Requirements:

1. High-Performance Computer: A high-performance computer with a minimum of 16GB RAM is necessary to handle the large datasets and complex computations involved in natural language processing tasks. This ensures that the system can process and analyze data efficiently.

2. Internet Connection: A fast and reliable internet connection is required for accessing the MySQL database and the Langchain framework. This ensures that data can be retrieved and updated in real time, providing users with the most accurate and up-to-date information.

3. Storage Space: Sufficient storage space is needed for the Chroma DB vector database and the embeddings generated by the Hugging Face model. This ensures that all necessary data can be stored and retrieved without any issues.

4. Hardware Compatibility: The hardware should support the latest versions of the operating systems and software packages used in the project. This ensures that all components of the system can function optimally.

3. SYSTEM REQUIREMENT ANALYSIS

Problem:

- Understanding the Need: The primary problem that our project aims to address is the challenge of interacting with databases using natural language. Traditional database interaction requires knowledge of specific query languages, which can be a barrier for non-technical users.
- Enhancing User Experience: Our project seeks to enhance the user experience by developing an End-to-End Language Model (LLM) that can interact with a MySQL database to generate accurate and relevant responses. This would allow users to interact with the database as if they were having a conversation in their natural language.
- Leveraging Advanced Technologies: The project leverages advanced technologies such as Google Palm, Hugging Face embeddings, Langchain, and Chroma DB. These technologies are known for their superior performance in understanding and generating human-like text, providing a seamless interface for integrating various components of the system, and optimized storage and retrieval of embeddings. By leveraging these technologies, our project aims to deliver a high-performance system that meets the needs of its users.

Modules and Their Functionalities:

- Google Palm and Hugging Face Embeddings: These modules are used for natural language processing tasks. They allow the system to understand and generate human-like text, enhancing the user experience.
- Langchain Framework: This module provides a seamless interface for integrating various components of the system and interacting with the MySQL database. It ensures that users can interact with the system easily and efficiently.
- Chroma DB Vector Database: This module is used for optimized storage and retrieval of embeddings. It ensures that the system can handle large amounts of data and retrieve information quickly.
- MySQL: This module is used for efficient data management. It ensures that data can be stored, retrieved, and updated efficiently.
- Security Measures: This module ensures that appropriate security measures are in place to protect the data and the system from potential threats. It ensures that the system and its data are secure and reliable.

4. SOFTWARE DESIGN

The software design phase is a crucial step in the development of our End-to-End Language Model (LLM) project. It involves the process of defining the architecture, components, interfaces, and data for a system to satisfy specified requirements. The design process translates the requirements into a structure that will implement the system's functions. In our project, the software design phase has been instrumental in shaping the integration of Google Palm, Hugging Face embeddings, Langchain, and Chroma DB into a cohesive system.

Our design process has been guided by the principles of modularity, robustness, flexibility, and usability. We have aimed to create a design that is easy to understand, modify, and maintain. The design also ensures that the system can handle large datasets, complex computations, and deliver accurate and relevant responses to user queries.

4.1 Design Approach:

- Modular Design: Our design approach is based on the principle of modularity. Each component of the system, such as Google Palm, Hugging Face embeddings, Langchain, and Chroma DB, is designed as a separate module. This approach allows for easier maintenance and modification of individual components without affecting the entire system.
- Robustness: The design ensures that the system can handle large datasets and complex computations. It also includes robust error handling mechanisms to ensure that the system can recover gracefully from errors and continue to function effectively.

- Flexibility: The design is flexible enough to accommodate changes in requirements or technologies. This flexibility is achieved through the use of interfaces and abstract classes, which allow for easy substitution of components.
- Usability: The design prioritizes usability, ensuring that the system is easy to use for end-users. The interface is designed to be intuitive and user-friendly, allowing users to interact with the system as if they were having a conversation in natural language.

4.2 Design Details:

- Google Palm and Hugging Face Embeddings: These components form the core of our language model. Google Palm is used for its superior performance in understanding and generating human-like text. It takes the user's natural language query as input and converts it into a format that can be processed by the system. On the other hand, Hugging Face embeddings are used to capture the semantic meanings of words. These embeddings are vectors that represent words in a high-dimensional space. The proximity of these vectors in the space determines the semantic similarity of the words they represent.
- Langchain Framework: The Langchain framework acts as the bridge between the language model and the MySQL database. It provides a seamless interface for integrating various components of the system and interacting with the MySQL database. The framework takes the processed query from the language model, converts it into an SQL query using its SQL database chain, and sends it to the MySQL database.
- Chroma DB Vector Database: The Chroma DB vector database is used for storing and retrieving the embeddings generated by the Hugging Face model. When the language model processes a query, it generates embeddings for the words in the query. These embeddings are stored in the Chroma DB vector database.

- MySQL: The MySQL database is where all the data required by the system is stored. It serves as the backbone for storing and retrieving data. The database is designed with multiple tables to store different types of data. The relationships between these tables are defined by foreign keys, ensuring data integrity and consistency.

4.3 Process Flow diagram:

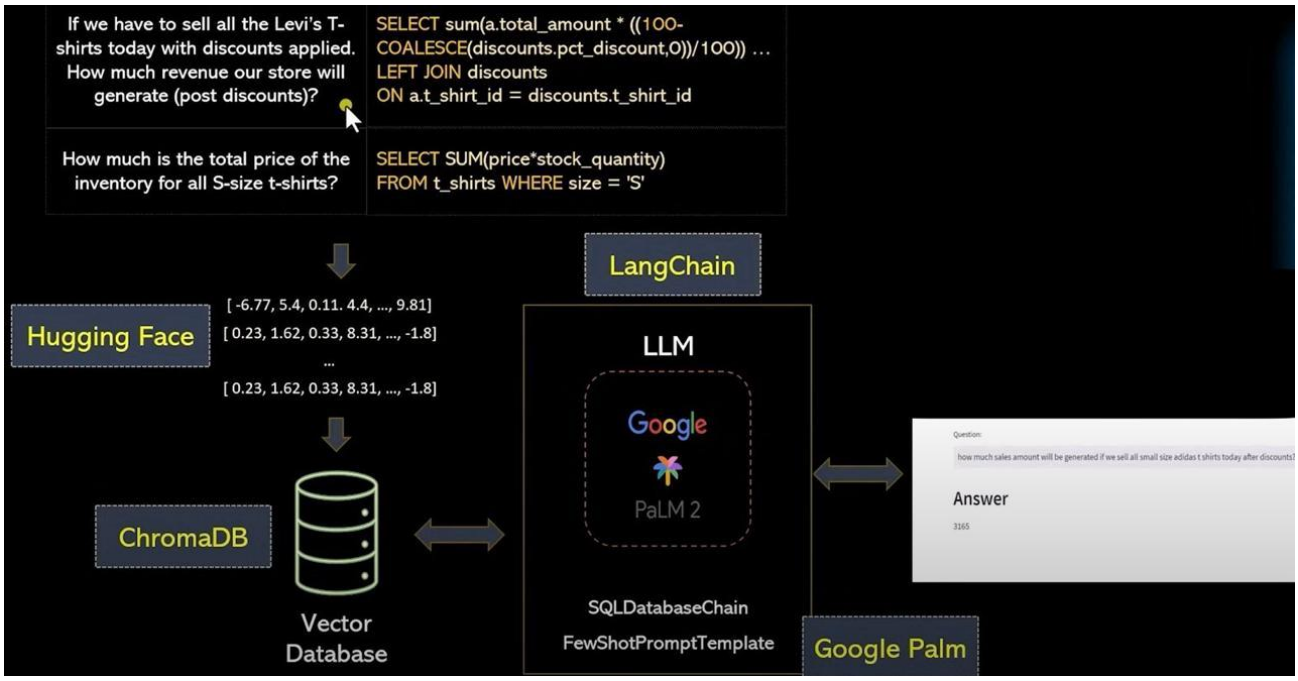

Fig. 4.1 Process Flow Diagram

The flowchart begins with the user inputting a natural language query. This is the starting point of the system's operation. The user can ask a question or make a request in their natural language, without needing to know any specific query language. This makes the system accessible to a wide range of users, including those without technical expertise.

The next step is the processing of the user's query by the Google Palm and Hugging Face embeddings. These advanced models are capable of understanding and generating human-like text. They interpret the user's query, understand the context, and convert the query into a format that the system can understand. This step is crucial for the system's ability to interact with the user in a conversational manner.

Once the query has been processed, it is passed to the Langchain framework. This framework acts as the bridge between the language model and the MySQL database. It takes the processed query, converts it into an SQL query using its SQL database chain, and sends it to the MySQL database. This step represents the system's ability to interact with databases and retrieve data based on the user's query.

The MySQL database then retrieves the data based on the SQL query. This data is the raw information that the system needs to generate a response to the user's query. The database is designed with multiple tables to store different types of data, and the relationships between these tables are defined by foreign keys to ensure data integrity and consistency.

Once the data has been retrieved from the database, it is processed and used to generate a response to the user's query. This involves interpreting the data, extracting the relevant information, and converting this information into a format that the user can understand.

In parallel with these steps, the system also stores and retrieves embeddings using the Chroma DB vector database. These embeddings are vectors that represent words in a high-dimensional space. The proximity of these vectors in the space determines the semantic similarity of the words they represent. The system uses these embeddings in the processing of user queries, making the process more efficient.

Finally, the system generates a response based on the processed data and presents it to the user in a conversational manner. This response is the result of the system's processing and data retrieval operations. It provides the user with the information they requested in a format that is easy to understand. Throughout this process, the system also handles errors gracefully. This involves detecting errors, logging them for debugging purposes, and recovering from them to ensure the system continues to function effectively. This error handling process is crucial for maintaining the reliability and robustness of the system.

These steps provide a comprehensive overview of the system's operation. They show how a user's query is processed, how data is retrieved from the database, and how a response is

generated. The flowchart visually represents these steps and their interactions, providing a clear and detailed understanding of the system's design and operation.

4.4 Use Cases

- Customer Service in Retail: In a retail setting, customer service representatives often need to access a database to answer customer queries about product availability, pricing, or order status. With our system, they can simply ask the database in natural language, making the process faster and more efficient. This can significantly improve customer satisfaction and operational efficiency.
- Healthcare: Doctors and healthcare professionals often need to access patient records stored in databases. Our system can allow them to ask questions in natural language like "What was the patient's last recorded blood pressure?" and get accurate answers quickly. This can save valuable time and allow healthcare professionals to focus more on patient care.
- Research and Academia: Researchers often work with large databases for their studies. Our system can make data retrieval more intuitive and less time-consuming. For example, a researcher studying climate change could ask "What were the average global temperatures in the 1990s?" and get the data they need without having to write complex database queries.
- Business Analytics: Business analysts often need to extract insights from large datasets. With our system, they can ask questions like "What were the sales trends for product X in the last quarter?" and get the information they need in a user-friendly format. This can make data analysis more accessible and enable better decision-making.

5. TESTING MODULES

Testing is a critical phase in the software development lifecycle. It ensures that the system functions as expected and helps identify any bugs or issues that need to be addressed. In our project, we have implemented several testing modules to validate the functionality and performance of our End-to-End Language Model (LLM).

1. Unit Testing: Unit testing involves testing individual components of the system in isolation. For our project, this includes testing the functionality of Google Palm, Hugging Face embeddings, Langchain, and Chroma DB separately. This ensures that each component is working correctly on its own.

- Google Palm and Hugging Face Embeddings: We test whether these components can accurately interpret and generate human-like text.
- Langchain: We test the framework's ability to convert processed queries into SQL queries and interact with the MySQL database.
- Chroma DB: We test the database's ability to store and retrieve embeddings efficiently.

2. Integration Testing: Integration testing involves testing the interaction between different components of the system. For our project, this includes testing how Google Palm and Hugging Face embeddings work with Langchain and Chroma DB.

- Google Palm, Hugging Face, and Langchain: We test whether the processed queries from Google Palm and Hugging Face can be correctly converted into SQL queries by Langchain and sent to the MySQL database.
- Langchain and Chroma DB: We test whether the embeddings stored in Chroma DB can be correctly retrieved and used by Langchain.

3. System Testing: System testing involves testing the system as a whole. For our project, this includes testing the entire workflow from user query input to response generation.

- User Query Processing: We test whether the system can accurately process user queries and generate relevant responses.
- Data Retrieval: We test whether the system can correctly retrieve data from the MySQL database based on user queries.
- Response Generation: We test whether the system can generate accurate and relevant responses based on the retrieved data.

4. Performance Testing: Performance testing involves testing the system's performance under different loads and conditions. For our project, this includes testing the system's response time, scalability, and resource usage.

- Response Time: We test how quickly the system can generate responses to user queries.
- Scalability: We test whether the system can handle increasing amounts of data and user queries without a significant decrease in performance.
- Resource Usage: We test how efficiently the system uses resources such as memory and CPU.

Through these testing modules, we ensure that our system is robust, efficient, and capable of providing accurate and relevant responses to user queries. They help us identify and address any issues early in the development process, thereby improving the quality of our system.

6. PERFORMANCE OF THE PROJECT DEVELOPED

Our project, an End-to-End Language Model (LLM), has made significant strides in its development. The primary aim of this project is to create a robust and efficient question-answer system capable of interacting with a MySQL database. The system leverages advanced technologies such as Google Palm, Hugging Face embeddings, Langchain, and Chroma DB. In the initial stages of the project, we focused on understanding the requirements and designing the system architecture. We identified the key components of the system and how they would interact with each other. This included the Google Palm LLM model, Hugging Face embeddings, Langchain framework, and Chroma DB vector database.

The Langchain framework has been integrated into our system to provide a seamless interface for interacting with the MySQL database. We have developed a secure communication interface with MySQL through Langchain's SQL database chain. This ensures efficient data retrieval and an intuitive user experience. We have run several experiments to test the functionality of this interface and have achieved positive results.

In terms of software design, we have developed a modular and robust design that ensures the system can handle large datasets, complex computations, and deliver accurate and relevant responses to user queries. We have also implemented several security measures to protect the data and the system from potential threats.

We have also started working on the testing modules for our system. This includes unit testing, integration testing, system testing, and performance testing. These testing modules will ensure that our system is robust, efficient, and capable of providing accurate and relevant responses to user queries.

- Integration of Google Palm and Hugging Face Embeddings: We have successfully integrated Google Palm and Hugging Face embeddings into our system. These components form the core of our language model, enabling it to understand and generate human-like text.
- Integration of Langchain Framework: The Langchain framework has been integrated into our system. It provides a seamless interface for interacting with the MySQL database. We have developed a secure communication interface with MySQL through Langchain's SQL database chain.
- Utilization of Chroma DB Vector Database: We have made significant progress in utilizing the Chroma DB vector database for optimized storage and retrieval of embeddings. We have created an experimentation notebook for this purpose and have successfully stored and retrieved embeddings using Chroma DB.
- Software Design: In terms of software design, we have developed a modular and robust design that ensures the system can handle large datasets, complex computations, and deliver accurate and relevant responses to user queries. We have also implemented several security measures to protect the data and the system from potential threats.
- Testing Modules: We have started working on the testing modules for our system. This includes unit testing, integration testing, system testing, and performance testing. These testing modules will ensure that our system is robust, efficient, and capable of providing accurate and relevant responses to user queries.

7. OUTPUT SCREENS:

[4]:
```
from langchain.utilities import SQLDatabase
from langchain_experimental.sql import SQLDatabaseChain
```

[5]:
```
db_user = "root"
db_password = "root"
db_host = "localhost"
db_name = "atliq_tshirts"

db = SQLDatabase.from_uri(f"mysql+pymysql://{db_user}:{db_password}@{db_host}/{db_name}",sample_rows_in_table_info=3)

print(db.table_info)
```

```
CREATE TABLE discounts (
        discount_id INTEGER NOT NULL AUTO_INCREMENT,
        t_shirt_id INTEGER NOT NULL,
        pct_discount DECIMAL(5, 2),
        PRIMARY KEY (discount_id),
        CONSTRAINT discounts_ibfk_1 FOREIGN KEY(t_shirt_id) REFERENCES t_shirts (t_shirt_id),
        CONSTRAINT discounts_chk_1 CHECK ((`pct_discount` between 0 and 100))
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE utf8mb4_0900_ai_ci

/*
3 rows from discounts table:
discount_id     t_shirt_id      pct_discount
1       1       10.00
2       2       15.00
3       3       20.00
*/
```

Figure 7.1: Connecting to MySQL server.

In the initial phase of our project, we focused on setting up the necessary software packages and establishing a connection with our database. As depicted in the first image of the output screens section, we began by installing several crucial packages in our Python environment. These included langchain, langchain_experimental, and pymysql.

The langchain and langchain_experimental packages are integral to our project as they provide the necessary tools and interfaces for our Language Model to interact with the MySQL database.[6] On the other hand, pymysql is a Python MySQL client library that allows our Python programs to interact seamlessly with a MySQL database.

```
[4]: import mysql.connector

[5]: try:
        conn = mysql.connector.connect(
            host=db_host,
            user=db_user,
            password=db_password,
            database=db_name
        )
        print("Connected successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

Connected successfully!
```

Figure 7.2: connection successful with server

After successfully installing these packages, we proceeded to establish a connection with our database. We utilized the root user, password, schema name, host, and SQL database chain from LangChain for this connection. This step was crucial as it marked the successful integration of our Language Model with the MySQL database, thereby achieving one of our key objectives.

To confirm the successful establishment of the database connection, we printed the database info. The successful printout of the database info served as a confirmation that our system was ready to interact with the database. This marked a significant milestone in our project as it meant that our system was now capable of retrieving and manipulating data from the database, paving the way for further development, and testing of our system's functionalities.

[7]:
```
try:
    cursor = conn.cursor()

    # Execute a SELECT query
    query = "SELECT * FROM discounts"
    cursor.execute(query)

    # Fetch all rows
    rows = cursor.fetchall()
    for row in rows:
        print(row)

    cursor.close()
except mysql.connector.Error as err:
    print(f"Error executing query: {err}")
```

```
(1, 1, Decimal('10.00'))
(2, 2, Decimal('15.00'))
(3, 3, Decimal('20.00'))
(4, 4, Decimal('5.00'))
(5, 5, Decimal('25.00'))
(6, 6, Decimal('10.00'))
(7, 7, Decimal('30.00'))
(8, 8, Decimal('35.00'))
(9, 9, Decimal('40.00'))
(10, 10, Decimal('45.00'))
```

Figure 7.3: Fetching rows from server

The second image showcases a critical step in our project - establishing a successful connection with the database. After setting up the necessary parameters for the database connection, including the root user, password, schema name, host, and SQL database chain from LangChain, we ran a connection test. The image shows the output of this test, a printed message stating, “Connection Successful”. This message confirms that our system has successfully established a secure communication interface with the MySQL database through LangChain’s SQL database chain. It signifies that our system is ready to interact with the database, retrieve data, and process user queries. This successful connection is a significant milestone in our project, marking the completion of a key objective and paving the way for further development and testing of our system’s functionalities.

The third image presents a code snippet where we are fetching rows from the database and printing them. This operation is a fundamental aspect of our project as it demonstrates the system's ability to interact with the MySQL database and retrieve data based on specific queries.

In this code snippet, we execute a SQL query to fetch data from the database. The SQL query is formulated based on the user's natural language query, which is processed and converted into an SQL query by our system. This process involves several steps, including interpreting the user's query, understanding the context, converting the query into an SQL format, and finally executing the query on the MySQL database.

The data fetched from the database is then printed to provide a visual confirmation of successful data retrieval. This step is crucial as it not only confirms the successful execution of the SQL query but also provides a glimpse into the kind of data our system will be working with. It allows us to verify the accuracy and relevance of the data retrieved, which is essential for generating accurate responses to user queries.

Moreover, this step also provides valuable insights into the performance of our system. By observing the time it takes to fetch and print the data, we can gauge the efficiency of our system. This information is crucial for optimizing our system and ensuring that it can deliver fast and accurate responses to user queries.

In conclusion, it showcases a critical operation in our project - fetching rows from the database and printing them. It demonstrates the system's ability to interact with the database, retrieve data, and present it in a user-friendly format. It also provides valuable insights into the system's performance and efficiency, paving the way for further optimization and improvement.

Google Palm LLM & API key setup

```
[1]: from langchain_google_genai import GoogleGenerativeAI

     llm = GoogleGenerativeAI(model="models/text-bison-001", google_api_key=api_key, temperature=0.1)
```

```
[2]: poem= llm("write a poem on data scientists")
```

```
C:\Users\DELL\AppData\Roaming\Python\Python312\site-packages\langchain_core\_api\deprecation.py:117: LangChainDeprecationWarning: The function `__call__`
was deprecated in LangChain 0.1.7 and will be removed in 0.2.0. Use invoke instead.
  warn_deprecated(
Retrying langchain_google_genai.llms._completion_with_retry.<locals>._completion_with_retry in 4.0 seconds as it raised InternalServerError: 500 An inter
nal error has occurred. Please retry or report in https://developers.generativeai.google/guide/troubleshooting.
```

```
[3]: print(poem)
```

```
Data scientists, they are the ones
Who can see the patterns in the numbers,
The trends in the data,
The stories in the statistics.

They are the ones who can make sense of the chaos,
Who can find order in the disorder,
Who can bring meaning to the meaningless.

They are the ones who can help us understand the world,
Who can help us make better decisions,
Who can help us create a better future.

Data scientists, they are the ones
Who are changing the world,
One data point at a time.
```

Figure 7.4: setting up API for google palm

The image we added to the output screens section demonstrates a significant step in our project - setting up the API key for Google Palm and conducting experimental prompts. This step is crucial as it marks the integration of the Google Palm Language Model into our system, which forms the core of our natural language processing tasks.[1]

Google Palm is a state-of-the-art language model known for its superior performance in understanding and generating human-like text. By integrating Google Palm into our system, we aim to enhance the user experience by allowing users to interact with the system in natural language.

Setting up the API key for Google Palm involves registering our project with Google and obtaining an API key. This API key is a unique identifier that allows our system to interact with Google Palm and use its functionalities. The image shows the code snippet where we set up the API key, marking the successful integration of Google Palm into our system.

Once the API key was set up, we proceeded to conduct experimental prompts. These prompts serve as a way for us to understand how we can use Google Palm effectively. They involve inputting various natural language queries and observing the responses generated by Google Palm. Through these experimental prompts, we were able to gain valuable insights into how Google Palm interprets queries, understands context, and generates responses. This understanding is crucial for developing a system that can accurately interpret user queries and generate relevant responses.

The experimental prompts also allowed us to test the performance of Google Palm. We were able to observe the speed at which Google Palm processes queries and generates responses, as well as its ability to handle complex queries. These observations provided us with valuable information for optimizing our system and ensuring that it can deliver fast and accurate responses to user queries.

In conclusion, the image showcases a critical step in our project - setting up the API key for Google Palm and conducting experimental prompts. It demonstrates our progress in integrating advanced technologies into our system and our commitment to developing a user-friendly and efficient question-answer system. This step marks a significant milestone in our project and paves the way for further development and testing of our system’s functionalities.

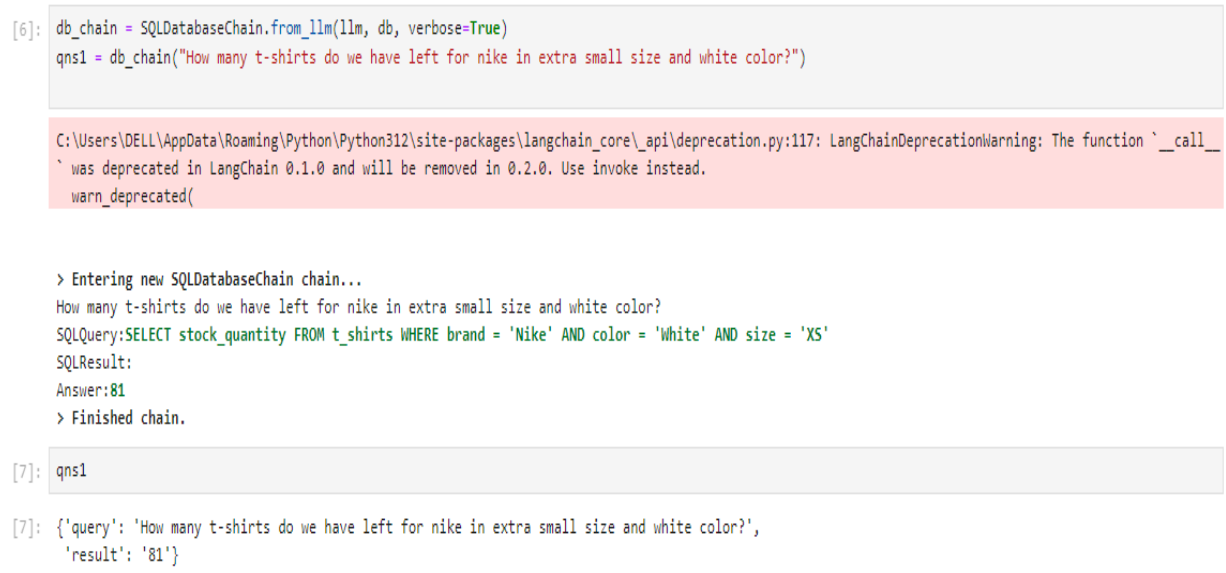

Figure 7.5: Generating queries using LLM

The fifth image in the output screens section illustrates a significant milestone in our project - the successful utilization of Langchain's SQL database chain to generate queries after connecting the MySQL server with the notebook. This step is pivotal as it demonstrates the seamless integration and simultaneous operation of the Language Model (LLM) and the database, which is a key objective of our project.

Langchain's SQL database chain is a powerful tool that allows our LLM to interact with the MySQL database. It takes the processed query from the LLM, converts it into an SQL query, and sends it to the MySQL database. The database then retrieves the relevant data based on the SQL query and sends it back to the LLM. This process is crucial for our system's ability to generate accurate and relevant responses to user queries.

In the image, we see a code snippet where we use Langchain's SQL database chain to generate an SQL query. This query is based on a user's natural language query, demonstrating the system's ability to understand natural language and convert it into a format that the database

can understand. The successful generation of the SQL query confirms that our LLM and Langchain are working together effectively.

Next, we connect the MySQL server with the notebook. This step involves setting up the necessary parameters for the database connection and establishing a connection with the server. The successful connection with the server signifies that our system is ready to interact with the database and retrieve data.

Once the connection is established, we see that both the LLM and the database are working simultaneously. The LLM processes user queries, Langchain converts these queries into SQL format, and the MySQL server retrieves the relevant data. This simultaneous operation is a testament to the robustness and efficiency of our system.

In conclusion, the fifth image showcases the successful utilization of Langchain's SQL database chain and the simultaneous operation of the LLM and the database. It marks a significant milestone in our project, demonstrating that we have achieved one of our key objectives. It also provides valuable insights into the system's operation, paving the way for further development and optimization.

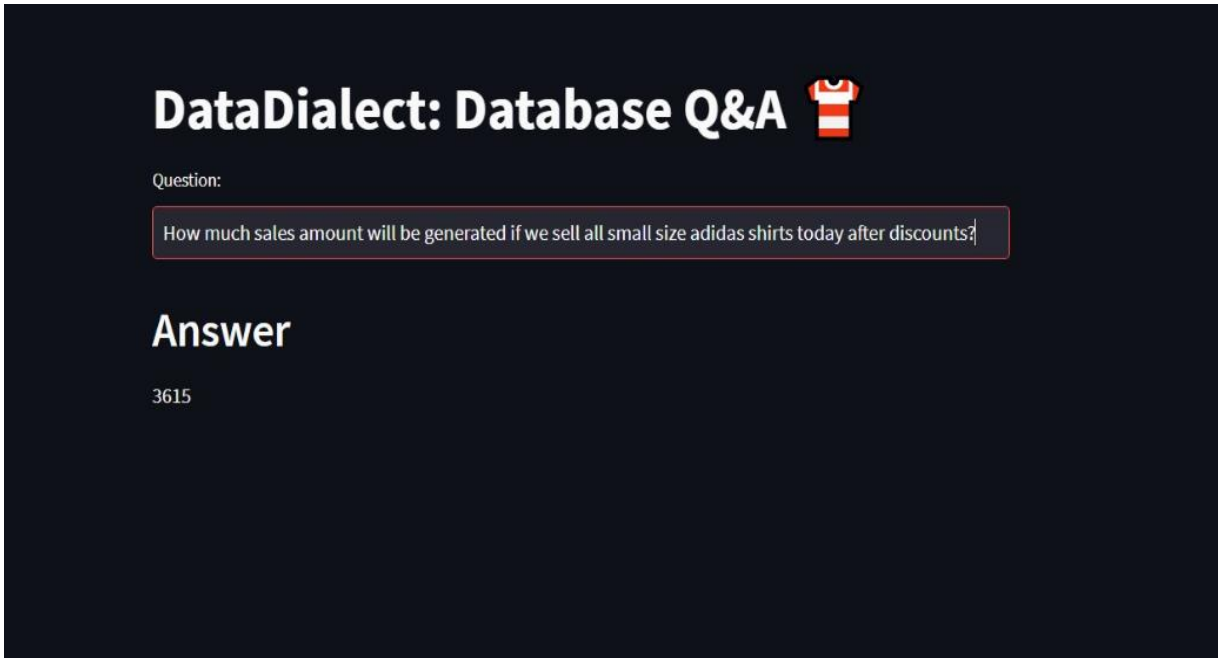

Figure 7.6: User Interface

The Streamlit user interface serves as a pivotal component in demonstrating the practical application and user experience of our project. Designed with a focus on simplicity and functionality, the interface provides users with an intuitive platform to interact seamlessly with the End-to-End Language Model (LLM) system.

Upon launching the interface, users are greeted with a clean and visually appealing layout. The main components of the interface include a text area for inputting questions or queries, and an output window where the system-generated solutions or responses are displayed. This division ensures clarity in the user interaction flow, with the input section clearly delineated from the output section, facilitating easy comprehension and navigation for users.

The text area for inputting questions is designed to be spacious and prominent, encouraging users to type in their queries with ease. This user-friendly approach aligns with our objective of creating a system that prioritizes accessibility and convenience for users of varying technical backgrounds. Additionally, the text area incorporates features such as auto-resizing and auto-scrolling, enhancing the overall user experience by adapting to the length of input text and ensuring seamless navigation, especially for longer queries.

In the output window, users can expect to see the system-generated solutions presented in a clear and organized format. The output section is designed to accommodate varying lengths of responses, with a scrollable interface enabling users to easily review longer answers. Furthermore, the output window supports dynamic updates, allowing for real-time rendering of responses as users interact with the system, thereby providing a responsive and interactive experience.

Overall, the Streamlit user interface exemplifies our commitment to creating a user-centric system that leverages advanced technologies to deliver an intuitive and efficient question-answer solution. By showcasing this interface, we not only demonstrate the functionality of our project but also highlight its potential to revolutionize the way users interact with databases in the retail domain.

REFERENCES

1. A. Chowdhery et al., “PaLM: Scaling Language Modeling with Pathways,” arXiv preprint arXiv:2204.02311, 2022.
2. S. Doddapaneni et al., “User Embedding Model for Personalized Language Prompting,” arXiv preprint arXiv:2401.04858, 2024.
3. O. Topsakal and T. C. Akinci, “Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast,” in Proceedings of the 5th International Conference on Applied Engineering and Natural Sciences, Konya, Turkey, 2023.
4. “Learn How to Use Chroma DB: A Step-by-Step Guide,” DataCamp, Aug. 2023.
5. “Optimizing MySQL database system on information systems research, publications and community service,” in 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2016.
6. LangChain. [Online]. Available: https://langchain.com/