Build and Deploy Text-2-SQL LLM Using OpenAI and AWS Project Overview

Overview

Current Industry:

In today's data-driven world, organizations across industries rely heavily on data to make informed decisions. Data is often stored in relational databases, which require SQL (Structured Query Language) to retrieve and manipulate information. However, writing SQL queries can be challenging for many professionals, especially those without a technical background, such as business analysts, product managers, or executives.

Despite the widespread availability of data, a significant bottleneck remains: the need for SQL expertise to extract meaningful insights. This creates a dependency on specialized teams, such as data analysts or engineers, to write queries and generate reports. As a result, decision-making can be delayed, and productivity hampered, particularly in situations where quick access to data is critical.

Why a Text-to-SQL Tool is Necessary:

The need for democratizing data access is more pressing than ever. A Text-to-SQL tool bridges this gap by enabling non-technical users to interact with databases using natural language. This reduces the reliance on SQL experts, accelerates decision-making, and promotes a more data-driven culture within organizations.

Moreover, even for experienced coders, writing complex SQL queries can be time-consuming. A tool that can accurately translate natural language into optimized SQL queries can significantly enhance productivity, allowing coders to focus on more strategic tasks rather than spending time on routine query writing.

How LLMs Are Solving Productivity Problems:

Large Language Models (LLMs) like GPT-4 have shown remarkable capabilities in understanding and generating human-like text. When applied to the task of translating natural language to SQL, LLMs can accurately interpret user intentions, generate complex SQL queries, and even optimize them for performance. This reduces the likelihood of errors, enhances query accuracy, and enables faster data retrieval.

LLMs also contribute to productivity by automating routine tasks, reducing the time spent on query formulation, and enabling even non-technical users to retrieve data independently. This empowers a broader range of professionals to engage with data directly, fostering a more agile and responsive organization.

In this project, we will develop a user-friendly Streamlit web application that leverages LLMs to convert natural language queries into SQL commands using Open AI and Databricks.

Prerequisite Project: Kindly ensure the completion of the LLM Project for building and fine-tuning a large language model before proceeding with this project.

Prerequisites: Basics of LLMs, Prompt Engineering, Python and Streamlit

Note: Utilizing Azure and AWS services for this project may result in charges; it is essential to thoroughly review the Azure and AWS documentation to understand the pricing structure and potential costs associated with different resources and usage patterns.

Aim

The aim of the SQLGenPro project is to develop a Streamlit-based web application that utilizes Large Language Models (LLMs) to convert natural language queries into optimized SQL commands, enabling secure database interaction, user authentication, and seamless deployment on AWS.

Data Description

The food delivery application database contains detailed records of users, restaurants, menu items, orders, payments, and reviews, capturing every transaction and interaction within the app. This comprehensive dataset, with tables such as Users, Restaurants, Orders, and Reviews, is designed to support the analysis of customer behaviors, service optimization, and overall business performance.

Tech Stack

- ➔ Language: Python 3.10
- ➔ Libraries: Langchain, Langchain-openai, streamlit, Databricks

➔ Model: GPT-4
➔ Cloud Platform: Microsoft Azure and AWS

Approach

1. Virtual Environment and Dependencies:

- Create a Python virtual environment and install necessary libraries (`Streamlit`, `OpenAI`, `LangChain`, database connectors).
- Configure access to Databricks and Snowflake, including connection setup and testing.
- Set up the development environment in VS Code.

2. LLM Integration and Database Interaction:

- Integrate the LLM via API to convert natural language queries into SQL.
- Develop backend modules to execute SQL queries on Databricks/Snowflake and retrieve results.
- Implement error handling to manage SQL generation and execution issues.

3. Prompt Engineering and SQL Optimization:

- Create and refine prompt templates to accurately generate SQL, including complex queries (CTEs, joins).
- Implement SQL validation and optimization to ensure generated queries are efficient and correct.

4. User Interface and Authentication:

- Build an intuitive Streamlit interface for inputting queries, displaying SQL results, and managing query history/favorites.
- Implement user authentication with secure session management and personalized experiences.

5. Advanced Query Handling and Feature Integration:

- Develop "Quick Analysis," "Deep Dive Analysis," and "Favorites" sections for enhanced user interaction.
- Test the application with complex SQL queries to ensure robustness.

6. Deployment:

- Configure and deploy the application on AWS EC2 with HTTPS for secure access and scalable performance.

Modular code overview:

Once you unzip the modular_code.zip file, you can find the following:

```
├─ .streamlit
│  └─ config.toml
├─ Langchain_Intro.ipynb
├─ README.md
├─ SQLGenPro.png
├─ SQLGenPro.py
├─ SQLGenPro_Live.py
├─ artifacts/
├─ authenticator.yml
├─ data/
├─ helper.ipynb
├─ requirements.txt
├─ src
│  ├─ add_logo.py
│  └─ utils.py
├─ test.sql
└─ utils.ipynb
```

Here is a brief information on the files:

- **Main Application Components:** `SQLGenPro.py` and `SQLGenPro_Live.py` are the core application files, along with Jupyter notebooks like `Langchain_Intro.ipynb` and `helper.ipynb` for development and testing.

- **Supporting Files:** Includes configuration files (`config.toml`, `authenticator.yml`), utility scripts (`src/add_logo.py`, `utils.py`), and project resources like `SQLGenPro.png` for architecture.
- **Project Structure and Dependencies:** Contains `artifacts/` for images, `data/` for datasets, and `requirements.txt` for managing Python dependencies and `readme.md` for setup instructions and dependencies.
  Kindly follow all the instructions for running the code from Readme.md file.

Project Takeaways

1. Learn the basics of LLMs and how they can be leveraged for natural language processing tasks
2. Familiarize yourself with the technologies used in the project: Streamlit, OpenAI, Databricks, and Snowflake
3. Learn how to create and manage a Python virtual environment and set up the development environment in VS Code
4. Understand how to securely connect and interact with databases on Databricks
5. Learn how to integrate an Open AI API to convert natural language inputs into SQL queries
6. Understand the principles of prompt design and chaining to generate accurate and complex SQL queries
7. Learn how to implement robust error handling to manage SQL generation and execution issues
8. Discover techniques for validating and optimizing SQL queries to enhance performance and reliability
9. Learn how to implement secure user authentication and manage user sessions within a web application
10. Understand how to test and validate complex SQL queries within the application to ensure accuracy and performance
11. Learn how to dynamically generate and render ERD diagrams using LLMs and integrate them into the application
12. Learn how to implement a self-correction feedback loop that allows the application to refine and improve the accuracy of generated SQL queries over time
13. Acquire knowledge of how to configure and deploy web applications securely on AWS EC2, including HTTPS setup

14. Learn best practices for ensuring the security and scalability of deployed web applications