2023

SQL Copilot Project Report

BRETT
DICKINSON

Table of Contents

Introduction

Problem Statement

How can you make data more accessible to users that don't have the tools or training to query raw data to turn into information?

The foundation of SQL can be easy to learn - select, from, where...
But building onto the foundation to solve more complex problems can be a challenge. Where do I find this data? How do I join it to that? So, SQL can end up being a barrier to non-technical users, users who do not have all the necessary access, or even non-domain experts who do not know where to find what they need in the data models. Generative text models have given us a good solution to tackle these challenges.

These models, like ChatGPT, are very popular - including in the text-to-sql world. It can write great queries for you. And you can level this up by layering a model on top of your own database.

SQL Copilot is a python tool built to do just that - at no cost. Accessing ChatGPT's API charges based on the number of tokens you use, but this tool access's Google's Flan-t5-xxl text generation model through HuggingFace.

At this stage of pre-launch, SQL Copilot is built to run off of a specific training dataset called 'Spider.' It works by taking your natural language input, converting it into an SQL statement, running that on the database, and returning an answer to your question.

The Dataset

This project was built off of the Spider dataset which was built to help test the capabilities of text-to-sql tools. Developed by a group of Yale students, it provides SQLite databases, test questions, corresponding correct SQL statements to help train your models and applications. Currently, the top two performers on the test questions utilize ChatGPT.

10,181 Questions

A wide range of easy questions requiring a single table query to difficult requiring joins and subqueries.

200 Databases

These databases cover a wide range of domains, each with multiple tables and sample data provided.

Currently, this pre-launch version of the product has not been fully tested on all 10,000+ sample questions. Further development is needed to allow for mass-testing underneath the free HuggingFace Inference API.

Processing & Build

Even though this application relies on generative models to do a lot of the heavy lifting, there was a lot of data processing that needed to be done to make this work.

Database Metadata

The Spider dataset comes in a collection of folders containing .sqlite files. Using SQL Alchemy, I pulled out the schema, table, and column names as well as the CREATE statements to create a single source with the metadata from all of the schemas. I then consolidated this info into documents related to each table and loaded them to a Chroma vector database. Chroma has an easy integration with LangChain and also gives the ability to run a similarity search.

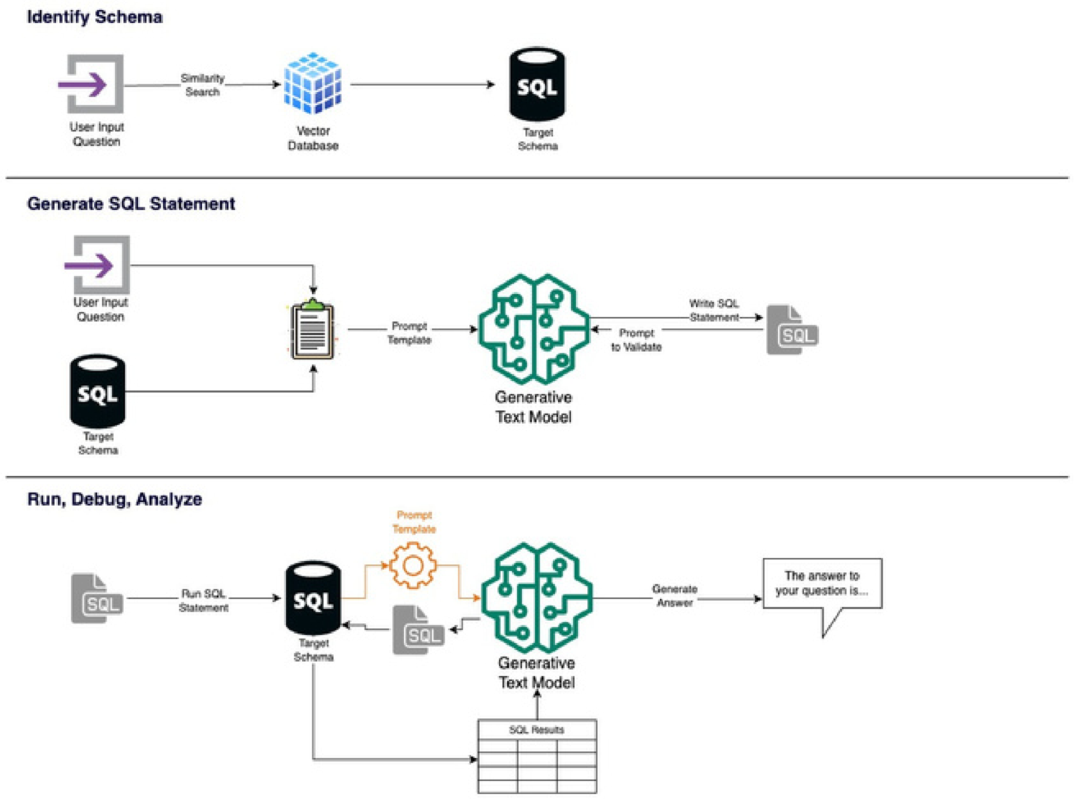
Schema Identification

Something different about this approach vs. some others is that I do not use the provided target schema necessary to answer the question. In my experience, it is not always obvious which schema or table is best to find what you are looking for. Once the user inputs a question, I use the Chroma similarity search to target the schema most likely to contain the data needed to answer it.

Language Model

HuggingFace provides the ability to connect to some great language models in a limited, but no cost way. After some testing, I settled on building this using Google's Flan-t5-xxl model. But after running into token limits testing LangChain's SQL agent, I broke up the flow into segments calling to the LLM multiple times - to write, validation, and if necessary debug a SQL statement, and then analyze the results.

Application Flow



SQL Copilot heavily utilizes LangChain as a framework for interacting with the vector database and language model. The first step is to run the user question through a similarity search on a Chroma vector database that holds the information for each table and column in every database to identify where the relevant data is most likely stored.

It then combines the question with the schema information and prompts the language model to write a SQL query to find the answer to the question. If that query successfully runs on the database, it will prompt the model to also answer the question using the data output. But if there is an error, it will instead prompt the model to debug and write another statement.

This process will run on the top 3 most likely schemas if the 1st or 2nd do not run successfully. Similarly, the debugging step will run a maximum of three times. This allows for inconsistencies in the LLM output but avoids endless loops or hitting the API's maximum runs.

Next Steps

There are many things happening in the world of artificial intelligence and text-to-sql products. So, while I am excited for creating the first iteration of this project, it is the next steps that are the most exciting. To start, there are a few key things that could most improve the application:

1

Handling Token Limits

The larger schemas in the dataset are still running into HuggingFace API token limits. I want to try preserving the no-cost method, and I think I can limit the tokens for some questions by trying to target only relevant tables, and not all within the target schema.

2

Tune Schema Identification

Currently our similarity search identifies the target schema about 50% of the time - 70% within the top 3. I want to improve this by building more fine-tuned training on the dataset.

3

User Customization

I want to give users the ability to bypass the schema targeting and input specific schemas and tables if they know what they are looking for. I also want to provide the ability to stop at running the query (and exporting the data) rather than asking a model to interpret.

I look forward to continually improving and learning from this project as I build upon it.

Acknowledgements

This project would not have been possible without some great tools built by others:

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., ... & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv preprint arXiv:1809.08887.

Chase, H. (2022). LangChain [Computer software]. https://github.com/hwchase17/langchain

https://www.trychroma.com/

Contact

Brett Dickinson

brettcd@icloud.com

https://github.com/BrettlyCD