Exploring MIMIC-III Database Through Conversations with an LLM Agent

Abstract:

Large Language Models are capable of generating text to SQL. In this paper I evaluated LangChain's SQLAgent and ChatGPT 3.5 's capabilities to have a conversation with MIMIC-III database. The findings are presented along with future planned tasks.

Introduction:

I am currently doing an internship at Natural Language Processing Laboratory at University of Illinois, Chicago under the guidance of Dr. Barbara Di Eugenio and Rochana Chaturvedi. Here I will document my journey, tasks I worked on, lessons learnt and future work that I will be embarking on.

Orientation:

- I took a course in Health Insurance Portability and Accountability Act (HIPAA) requirements. The reason this course was mandatory is because I will be doing data analysis on MIMIC-III database which is a large, freely-available database comprising de-identified health-related data associated with patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012.
- I signed a data use agreement, which outlines appropriate data usage and security standards, and forbids efforts to identify individual patients.
- With these steps out of the way, I was ready to start on the technical tasks.

Initial Technical Tasks:

- MIMIC-III is provided as a collection of comma separated value (CSV) files, along with scripts to help with importing the data into database systems including PostgreSQL, MySQL, and MonetDB. So my first task was to set up the database on my Macbook.

- Setting up the MIMIC-III database gave me the opportunity to learn the following -
  - Install and familiarize myself with an open source Database tool called DBeaver (https://dbeaver.io/).
  - Understood Entity-Relationship Model of MIMIC-III database from https://pi.cs.oswego.edu/~jmiles3/mimic/Miles-MIMIC-Project_report.pdf.
  - I then analyzed the MIMIC-III database using SQL and familiarized myself with the dataset.

Defining my Research Problem:

Since this is an NLP internship, I wanted to learn the cutting edge innovations happening in Generative AI and Large Language Models and wanted to use that technology to analyze the MIMIC-III dataset.

The increase in text data in the growing digital age has sparked new innovations in information retrieval techniques. For over four decades, Structured Query Language (SQL) has stood as the conventional approach to data querying, originating in 1979. SQL, a versatile language tailored for relational databases, manages structured datasets, and allows for the extraction of information through robust querying capabilities, navigating tables and relationships. Its utility extends to essential tasks like record creation, insertion, updating, deletion, and retrieval—making it an indispensable tool for developers. Given that a substantial portion of business data is conventionally stored in SQL databases, the tools to quicken the process would be highly beneficial for corporations and companies.

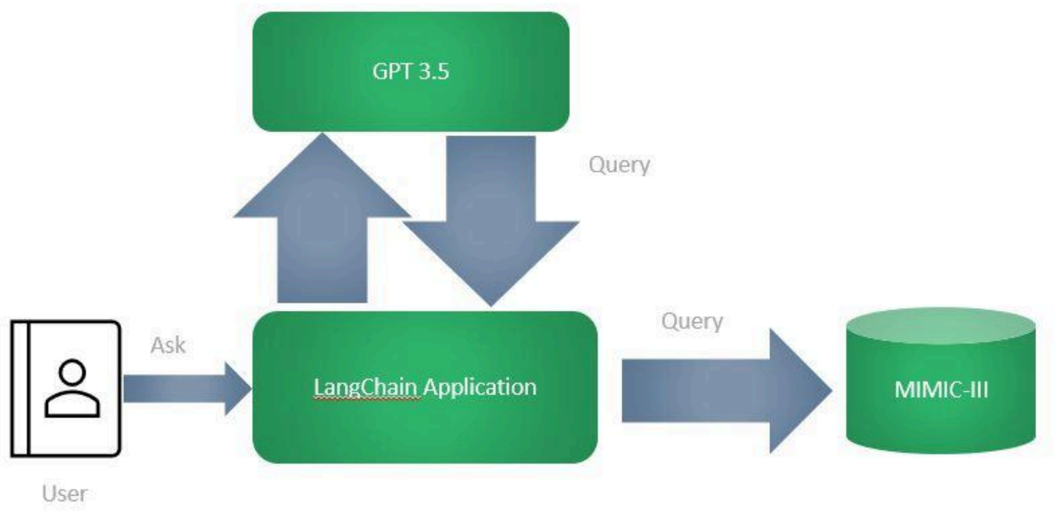
To streamline the querying process and enhance efficiency, the integration of Large Language Models (LLMs) emerges as a viable solution. Noteworthy examples of LLMs include OpenAI’s GPT and Facebook's LLAMA, which have intricate architectures trained on extensive text corpora and leveraging natural language processing for interpreting and generating human-like responses. These models bring forth a novel approach to processing textual information — LLMs, equipped with the ability to decipher intricate word relationships, excel in transforming natural human language into precise SQL queries. By doing so, this simpler conversion promises to save both time and resources in comparison to traditional SQL methodologies, creating a new era for natural language interfaces to databases. This article delves into using an LLM model to generate SQL Queries from human language and evaluating its strengths and limitations as a tool for information retrieval.

In essence, my problem statement is – “Instead of using SQL to analyze MIMIC data, I want to have a conversation with a SQL database via a Large Language Model. This would imply that the LLM would translate plain english to SQL and run the query on the database and provide results. I want to evaluate GPT 3.5’s capabilities in providing answers to queries against MIMIC data”.

LLM Tasks:

- I took a couple of Deep Learning Short Courses from DeepLearning.ai (https://www.deeplearning.ai/short-courses/).
- I learnt the LangChain framework to build applications to interact with Large Language Models.
- I learnt SQLAgent to use it to integrate an LLM with a SQL database.
- I setup a Developer account with Open AI to use GPT-3.5 as my LLM

My initial goal is to build out the following architecture to have the conversation with MIMIC-III database -



I am using Open AI’s GPT-3.5 pre-trained model, which is strong in interpreting and generating new content. Our database is the MIMIC-III repository, a public collection of medical records, and we’ll be exploring our model's ability to retrieve specific tables and information hosted within PostgresSQL, a relational database management platform.

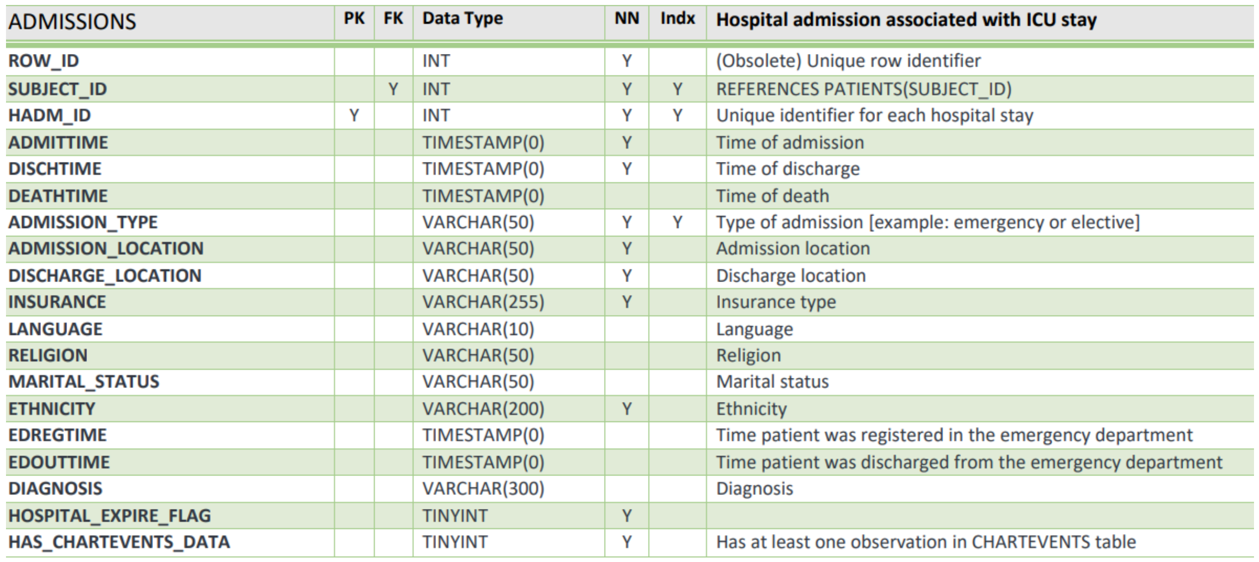
In this example, we’ll be using SQL Agent, a tool that allows us to interact with SQL databases using natural language. This Agent is provided by Lang chain, an open source framework designed to build applications powered by language models. The two features of LangChain are data-awareness and agentic behavior, which enable the model to connect to other sources of data and interact with its environment, allowing its agents to decide which tools to call based on user input. This makes the SQL Agent very powerful in answering questions based on the databases’ schema and databases’ content, recovering from errors with regeneration and traceback, and being compatible with SQL dialect supported by SQLAlchemy and TypeORM.

High Level coding steps include -

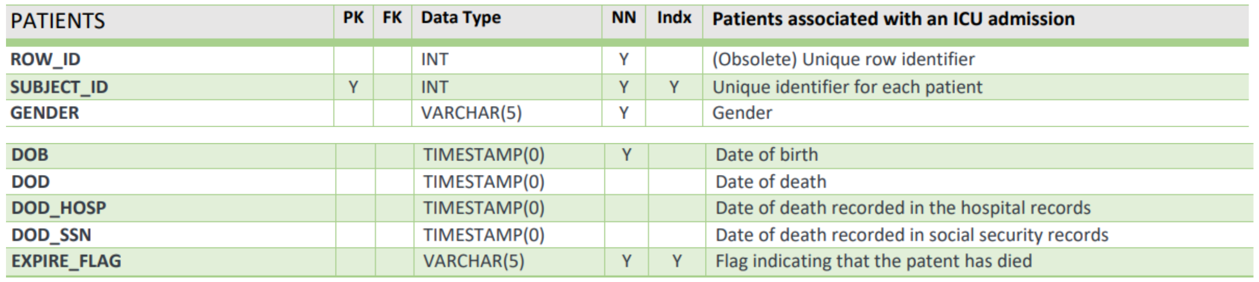
1. Installing necessary packages
2. Import necessary libraries
3. Define Prompt Template
4. Setup Database Connection but limit tables that you want to have a conversation against. MIMIC-III had 26 tables. In this iteration, I focused conversation on only Admissions, Patients and ICUStays.
5. Create SQL Agent to take zero shot prompts
6. Issue Prompt and watch the Model perform Chain of Thought based reasoning of (Action, Observation, Thought)

Code is available at
https://github.com/zurich123/MIMIC-III-LLM/blob/main/langchain-openai.ipynb

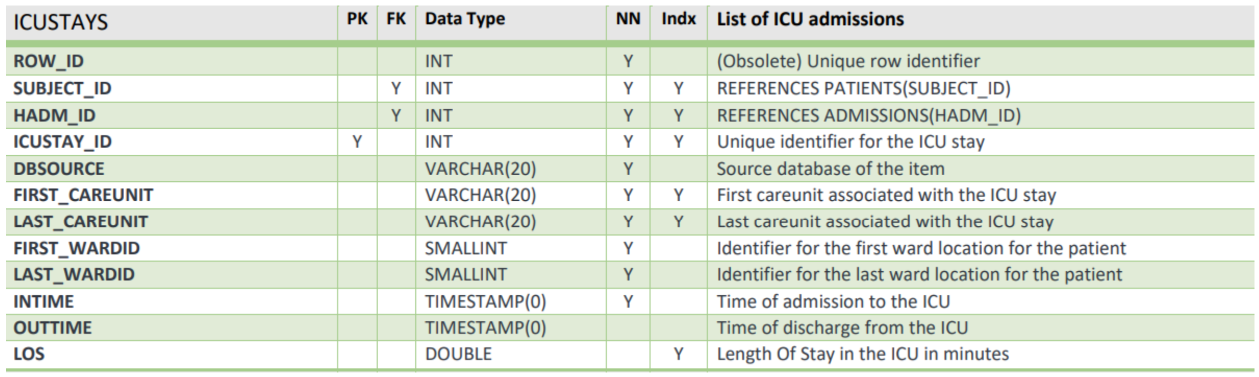
Schema for the 3 Tables -



ADMISSIONS	PK	FK	Data Type	NN	Indx	Hospital admission associated with ICU stay
ROW_ID			INT	Y		(Obsolete) Unique row identifier
SUBJECT_ID		Y	INT	Y	Y	REFERENCES PATIENTS(SUBJECT_ID)
HADM_ID	Y		INT	Y	Y	Unique identifier for each hospital stay
ADMITTIME			TIMESTAMP(0)	Y		Time of admission
DISCHTIME			TIMESTAMP(0)	Y		Time of discharge
DEATHTIME			TIMESTAMP(0)			Time of death
ADMISSION_TYPE			VARCHAR(50)	Y	Y	Type of admission [example: emergency or elective]
ADMISSION_LOCATION			VARCHAR(50)	Y		Admission location
DISCHARGE_LOCATION			VARCHAR(50)	Y		Discharge location
INSURANCE			VARCHAR(255)	Y		Insurance type
LANGUAGE			VARCHAR(10)			Language
RELIGION			VARCHAR(50)			Religion
MARITAL_STATUS			VARCHAR(50)			Marital status
ETHNICITY			VARCHAR(200)	Y		Ethnicity
EDREGTIME			TIMESTAMP(0)			Time patient was registered in the emergency department
EDOUTTIME			TIMESTAMP(0)			Time patient was discharged from the emergency department
DIAGNOSIS			VARCHAR(300)			Diagnosis
HOSPITAL_EXPIRE_FLAG			TINYINT	Y		
HAS_CHARTEVENTS_DATA			TINYINT	Y		Has at least one observation in CHARTEVENTS table



PATIENTS	PK	FK	Data Type	NN	Indx	Patients associated with an ICU admission
ROW_ID			INT	Y		(Obsolete) Unique row identifier
SUBJECT_ID	Y		INT	Y	Y	Unique identifier for each patient
GENDER			VARCHAR(5)	Y		Gender
DOB			TIMESTAMP(0)	Y		Date of birth
DOD			TIMESTAMP(0)			Date of death
DOD_HOSP			TIMESTAMP(0)			Date of death recorded in the hospital records
DOD_SSN			TIMESTAMP(0)			Date of death recorded in social security records
EXPIRE_FLAG			VARCHAR(5)	Y	Y	Flag indicating that the patent has died



ICUSTAYS	PK	FK	Data Type	NN	Indx	List of ICU admissions
ROW_ID			INT	Y		(Obsolete) Unique row identifier
SUBJECT_ID		Y	INT	Y	Y	REFERENCES PATIENTS(SUBJECT_ID)
HADM_ID		Y	INT	Y	Y	REFERENCES ADMISSIONS(HADM_ID)
ICUSTAY_ID	Y		INT	Y	Y	Unique identifier for the ICU stay
DBSOURCE			VARCHAR(20)	Y		Source database of the item
FIRST_CAREUNIT			VARCHAR(20)	Y	Y	First careunit associated with the ICU stay
LAST_CAREUNIT			VARCHAR(20)	Y	Y	Last careunit associated with the ICU stay
FIRST_WARDID			SMALLINT	Y		Identifier for the first ward location for the patient
LAST_WARDID			SMALLINT	Y		Identifier for the last ward location for the patient
INTIME			TIMESTAMP(0)	Y		Time of admission to the ICU
OUTTIME			TIMESTAMP(0)			Time of discharge from the ICU
LOS			DOUBLE		Y	Length Of Stay in the ICU in minutes

Results:

I asked the following queries -

Query	Result	Comments
How many patients have insurance on Medicaid?	Failed	Model queried the wrong table (Supposed to check Admissions table, but checked Patients table) and said did not know the answer.
How many admitted patients have insurance of Medicaid	Success	Adding the “admitted” patients helped the Model pick the right table of Admissions to get the answer.
How many admitted patients stayed in ICU	Success*	Model only queried the ICUStays table and did not do a join with the Admissions table. Only reason, answer is correct is because all admitted patients actually stayed in an ICU.
How many admitted patients did not stay in ICU	Success	Model did a join of Admissions and ICUStays and gave the right answer.
What is the average ICU Stay?	Success	Model figured out LOS column is length of stay and calculated the average correctly.
What is the average Admission Stay?	Failed	Model calculated average admitted time but realized it is a timestamp and tried to CAST it and got a OutputParserException so we need to add more robust error handling to Agent outputs
Admission Stay is the	Success	By adding additional

difference between DISCHTIME and ADMITTIME. What is the average Admission Stay?		details in the Prompt, the Model used the details and provided the right answer.
What is the average time a patient is staying in the hospital	Success	By rephrasing the question, Model gave the right answer. The field name “ADMITTIME” confused it earlier.
How many patients died in the hospital	Success	Model gave right answer and figured out that admissions table has the deathtime field
What percentage of patients are married?	Failed	Model hallucinated that a column of marital status exists on the Patients table.
What percentage of admitted patients are married	Success	Adding “Admitted” helped Model do a join between Admissions and Patients table to provide the right answer.

Findings and Analysis:

- We got around 60% accuracy on our sample queries. By just updating prompts, we were able to make the Model provide the right answer.
- For large databases, this approach could get expensive as the Model is loading the schema of all tables to figure out which tables to use to generate the query.
- Model has hallucinated columns on a table.
- For large databases, if we load all the tables, we could run into context window size limitations.
- All the SQL generated is syntactically correct even for wrong answers, so Text to SQL generation capabilities seem pretty robust.

Conclusion

In conclusion, while the world of generative AI offers a new approach to database information retrieval, there's still much that needs to be researched before it can ever truly match a data analyst writing SQL. However, the concept offers clear benefits in pursuing this approach to bring insights closer to non technical users.

Future Work:

- I want to run the same queries against other models (open source such as Llama) and other Text-to-SQL fine tuned models (CodeLlama) and provide a comparison.
- I want to leverage LangSmith's evaluation model to compare and also track all my requests and responses, tokens used etc.

References

1. MIMIC-III Clinical Database – Alistair Johnson, Tom Pollard, Roger Mark https://physionet.org/content/mimiciii/1.4/
2. Reconstruction of MIMIC-III Database for Data Analytics - Joseph Miles https://pi.cs.oswego.edu/~jmiles3/mimic/Miles-MIMIC-Project__report.pdf