Auto-Regressive Generative Large Language Models for Text-to-SQL Generation

Nan Jiang
jiang719@purdue.edu

Yi Wu
wu1827@purdue.edu

Introduction

Text-to-SQL generation is crucial due to data explosion and the demand for intuitive database querying interface.

Task Definition: Given a database schema and a natural language question, generate the corresponding SQL query.

Challenges:

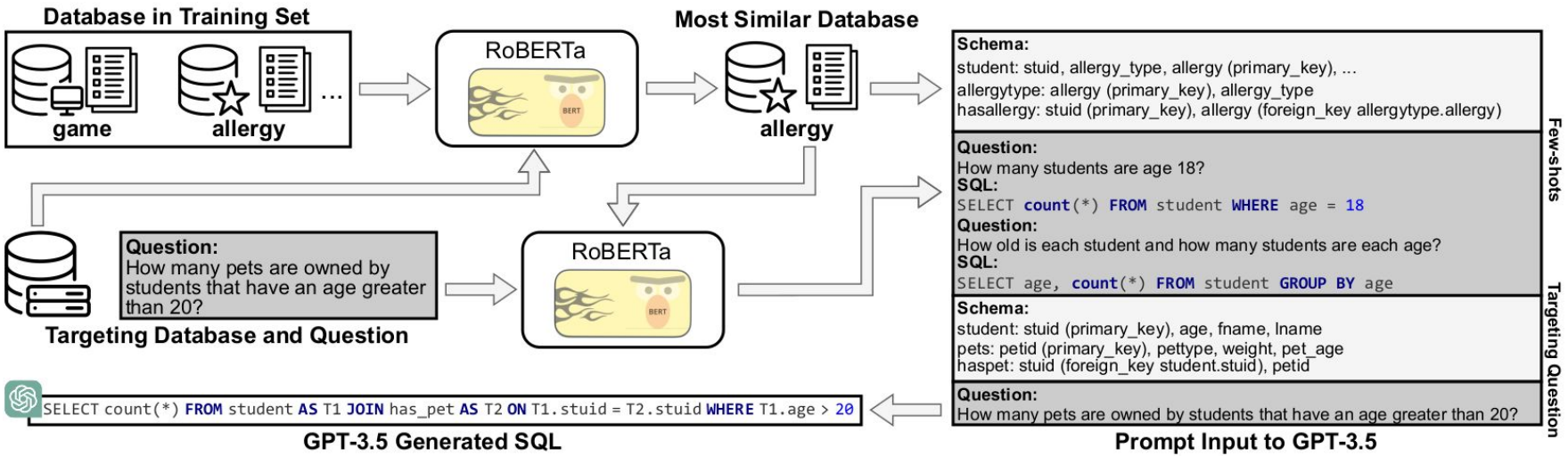
1. SQL has strict syntax, which is different from natural language.
2. Representing a database is hard (e.g., the inner linking among tables)

Methods

- Use LLM such as GPT-3.5 to generate SQL
  - Few-shot learning is important to adopt GPT-3.5 to this task
  - Selection of similar few-shot is also helpful
- Fine-tune open-sourced LLM with training data
  - Fine-tuning is more efficient to suite downstream task than few-shot learning
- Predicting which columns will be used in the SQL query
  - Using graph convolutional network
  - This helps to refine the search space by removing the non-related columns from the database

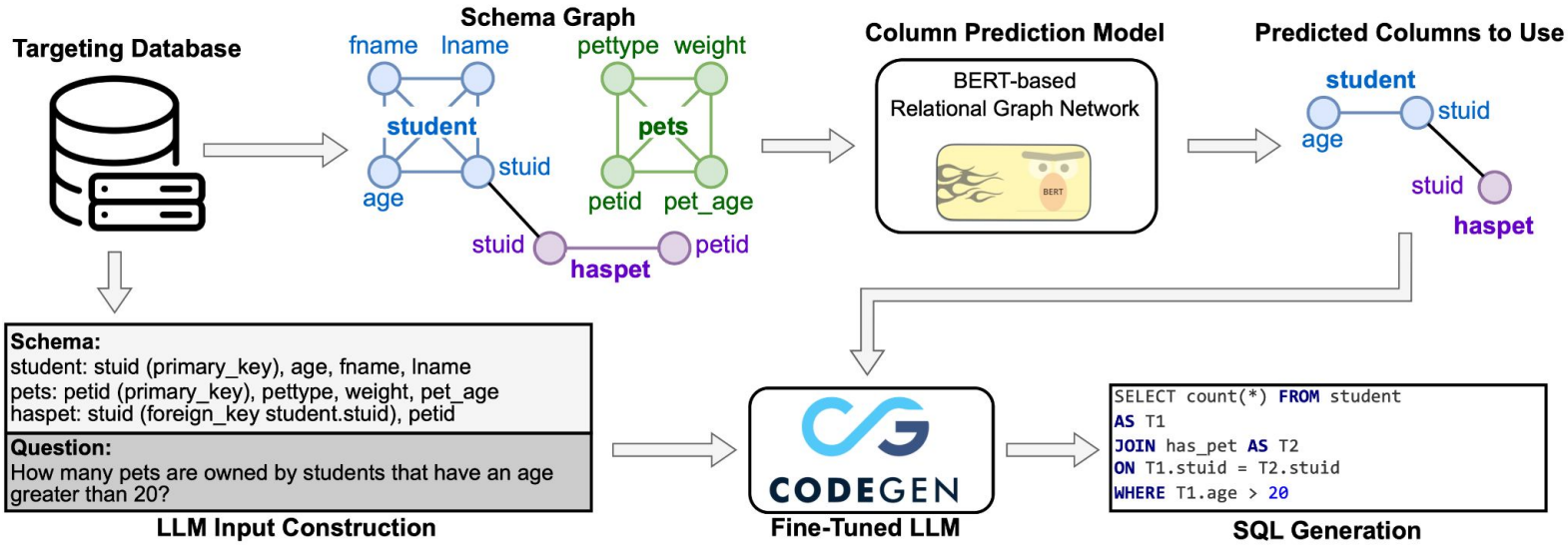
Few-shot Selection

- Given the targeting database schema and the user question
  - Select the most similar database schema from the training set
  - Select the top-10 similar user questions
- Use RoBERTa to obtain the text embeddings => compute cosine similarity



Predicting columns and fine-tuning LLM

- Fine-tune a pre-trained Codegen model to generate SQL query, given the database schema and question as input.
- Fine-tune a BERT-based graph convolutional network to predict the columns to be used.



Evaluation

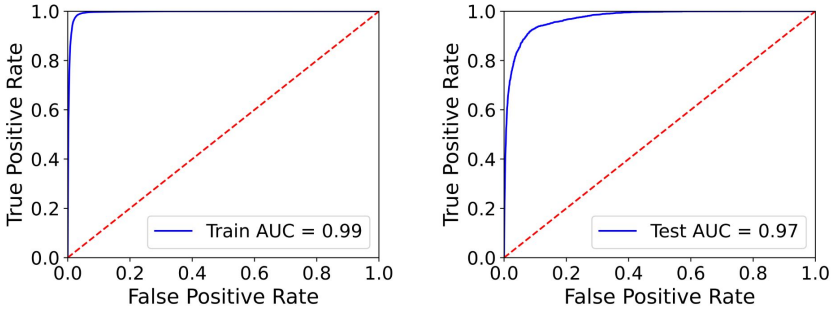
- Fine-tuned LLM
  - Execution match (QEX and IEX) is low, but exact match is relatively high
- The improvement of adding column prediction is minor
- GPT-3.5 performs poor with few-shot learning

Techniques	Spider		CoSQL				SParC			
	QEM	QEX	QEM	QEX	IEM	IEX	QEM	QEX	IEM	IEX
PICARD	75.5	79.3	56.9		24.2					
STAR			59.7		30.0		66.9		46.9	
CQR-SQL			58.4		29.4		67.8		48.1	
RASAT	75.3	80.5	58.8	67.0	27.0	39.6	67.7	73.3	49.1	54.0
GPT-3.5	58.3	69.4								
Fine-Tuned LLM	70.0	75.8	54.8	56.9	28.7	26.3	65.3	66.9	48.3	49.1
+ Column Prediction	70.6	76.2	54.7	56.7	30.0	26.6	65.9	67.6	48.6	49.3

Table 1: Comparison of our approach with the baselines on three datasets.

Evaluation

- Column prediction
  - Since the labels are very unbalanced (1: 10), AUC is a better metric.
  - Column prediction can be pretty accurate



(a) Training set (b) Testing set

Figure 4: AUC of predicting columns on the training set and testing set.

Discussion

- GPT-3.5 does not work well as we thought for this Text-to-SQL task
  - Limited tokens for few-shot examples
  - Low execution rate of SQL generated
- Adding domain knowledge (e.g., SQL syntax) to fine-tuning is promising.
- Future work
  - Use reinforcement learning to improve the quality (e.g., execution rate) of SQL
  - Build a larger and better dataset – We find wrong data in both training and test set
  - Fine-tune LLaMa