IE686 Large Language Models and Agents

Project Outline: Text-to-SQL - Team 5

Adrian Hagen
(matriculation number 2115311)

Kilian Ebi
(matriculation number 1961810)

Maximilian Abrams
(matriculation number 2117897)

Patricia Paskuda
(matriculation number 2119717)

Yannik Hahn
(matriculation number 2112929)

October 28, 2024

Submitted to
Ralph Peeters
University of Mannheim

1 Introduction

Nowadays, almost every area in industry, in organizations, or administration etc. tries to leverage data as it can be beneficial in order to, for example, get an advantage over competitors, to more specifically serve customers, or just keep an eye on internal operations and so on. However, for non-technical users obtaining data from, for instance, a relational database (in which corporate, organizational, and administrative data is often still stored) can be quite challenging. Therefore, we want to aid specifically those users in the form that they do not need to learn how to write correct SQL queries themselves in order to get their necessary data. Instead, the questions they have (and for which answers can be found in the data stored in a relational database) can be answered by providing them with the correct SQL queries that they can use to query the respective relational database which are, however, produced by an agent-based system. Our explorative project deals with developing such an agent-based system that transforms natural language text prompts into correct SQL queries. However, the focus here is not only to produce syntactically correct SQL queries, but instead we concentrate on providing correct data results/outputs.

2 Data

Given the vast amount of already existing LLMs fine-tuned on text-to-SQL tasks (Zhu et al. (2024)), we do not think that any fine-tuning of existing LLMs is necessary and given the high amount of computational resources required also not really feasible for this project. It is, however, necessary, to find data to evaluate the approaches we choose on. For this, we already found the following datasets, of which we might use some or all for our evaluation:

1. BIRD (Li et al. (2023)) - BIRD is short for BIg bench forlaRge-scale Database and is a benchmark for large-scale database. Bird claims to contain more complex databases and tables then spider.
2. Spider (Yu et al. (2018)) - Spider is a benchmark for complex SQL queries.
3. SParC (Yu et al. (2019)) - SParC is a large dataset for complex, cross-domain, and context-dependent(multi-turn) semantic parsing and text-to-SQL task (interactive natural language interfaces for relational databases) built upon our Spider task
4. WikiSQL (Zhong et al. (2017)) - WikiSQL is a dataset for text-to-SQL task.

3 Approach

3.1 Use of LLMs

Given the prevalent success of models like GPT, T5, Claude and Gemini in text-to-SQL applications, our approach will also leverage these advanced Large Language Models (LLMs) as the core of our solution (Zhu et al. (2024)) (Pourreza et al. (2024)).

3.2 Methods

One option for prompt engineering involves including database schema information in the context and incorporating examples of correct SQL queries within the prompt. This approach could help the model understand the structure and types of data it interacts with, as well as provide reference points for generating accurate queries.

For implementing a chain of thought, we suggest exploring an approach that breaks down query requirements step-by-step. This would entail reasoning explicitly about table relationships, validating SQL syntax at each stage, and documenting join conditions and filter logic. Such a breakdown might aid in tracking the rationale behind each part of the query, potentially enhancing the accuracy of SQL construction.

To evaluate the utility of zero-shot and few-shot strategies, we plan to test different numbers of examples, such as 0, 1, 3, and 5, to gauge the model's performance with varying levels of prior information. This would involve selecting representative examples spanning a range of query types and complexities to determine the optimal number of examples for each scenario.

A query decomposition framework is another approach under consideration, aimed at managing complex queries. This framework could involve breaking down complex queries into simpler sub-queries, handling each subquery independently, and managing JOIN operations separately. Recombining the results with validation checks could help ensure the integrity and correctness of the final output.

Lastly, feedback loop optimization is proposed as a method for refining the query generation process. This might involve defining stopping criteria based on an optimal amount of refinement runs, as referenced in subsections 3.3 and 4. Additionally, categorizing errors and applying targeted refinements, alongside monitoring performance across iterations, could help achieve continual improvement.

3.3 Multi-Agent Workflow

The envisioned multi-agent system will convert natural language queries into SQL, handling diverse phrasings and intents. It ensures compatibility with database schemas, executes the generated SQL queries, and refines queries based on results or errors through a feedback loop for better accuracy.

This system uses three agents: a SQL agent to generate SQL queries, a database agent to execute them, and a feedback agent to analyze errors and collaborate with the SQL agent for refinements.

For complex queries, we may use nested multi-agent workflows, which could improve accuracy but require more computational resources. Evaluating this trade-off is a key objective of the project.

4 Evaluation methods

4.1 Evaluation of SQL query

Despite our agentic approach using the responses from the database to refine the SQL query, our evaluation metrics will still be based on the accuracy of the SQL query compared to some ground truth SQL query for the given natural language query and not on the quality of the response.

To compare an SQL query predicted by our model to a ground truth SQL query from one of the given benchmarks, one approach is to first split up the SQL query into a set of its individual tokens (e.g. SELECT, FROM, WHERE, etc.) and then calculate the similarity between the two query sets (i.e. using Jaccard similarity). Another approach that could be used is Exact Matching Accuracy (EM), however, this might be too strict and might not allow for any differences in the phrasing of the query. However, we intend to also take it into account in initial evaluations.

4.2 Improvement ability of the Feedback-Agent

To evaluate if and by how much the Feedback-Agent actually improves the accuracy of the SQL query, we could measure accuracy for each refinement run the model performs and then compare it to the accuracy of the SQL query before the refinement run. This could even be used to determine an optimal amount of refinement runs by finding the point of diminishing returns and maybe contrast this to the efficiency (see next subsection).

4.3 Efficiency

One approach to measure this is through the Valid Efficiency Score (VES), an evaluation metric first proposed by the BIRD benchmark (Li et al. 2023). VES considers the validity and efficiency of each generated SQL query, meaning it evaluates whether the query executes correctly, yields accurate results, and performs efficiently compared to the ground truth. Here, efficiency can refer to running time, throughput, memory cost, or merged metrics. If the query equals the gold standard query, the performance will be equal. If it deviates, but still returns the same result, the difference is taken. The overall score is the sum over all queries. Another approach could be to benchmark the time our agentic system needs to generate the query. This could for instance be achieved by comparing the time a human needs to create the query given the same natural language input.

Bibliography

Li, J., B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. C. Chang, F. Huang, R. Cheng, and Y. Li (2023). Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Pourreza, M., H. Li, R. Sun, Y. Chung, S. Talaei, G. T. Kakkar, Y. Gan, A. Saberi, F. Ozcan, and S. O. Arik (2024). Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql.

Yu, T., R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev (2018, October-November). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 3911–3921. Association for Computational Linguistics.

Yu, T., R. Zhang, M. Yasunaga, Y. C. Tan, X. V. Lin, S. Li, H. Er, I. Li, B. Pang, T. Chen, E. Ji, S. Dixit, D. Proctor, S. Shim, J. Kraft, V. Zhang, C. Xiong, R. Socher, and D. Radev (2019). Sparc: Cross-domain semantic parsing in context.

Zhong, V., C. Xiong, and R. Socher (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR abs/1709.00103*.

Zhu, X., Q. Li, L. Cui, and Y. Liu (2024). Large language model enhanced text-to-sql generation: A survey.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
DeepL	Translation	Throughout	+
Perplexity AI	Analysis of Text-to-SQL Models	Use of LLMs	+