Natural Language Text-to-SQL: Evaluating Performance of Large Language Models through Effective Prompt Engineering

Anushka Kulkarni
s3291030
University of Twente
a.h.kulkarni@student.utwente.nl

Manjunath Sakthivel
s3337472
University of Twente
m.sakthivel@student.utwente.nl

Abstract

This paper investigates the performance of large language models (LLMs) in converting natural language questions to Structured Query Language (SQL) queries. It focuses on the role of prompt engineering. The accuracy and efficiency of LLMs in generating correct SQL queries can be highly sensitive to the way prompts are designed. In this study, we benchmark two LLM variants—LLaMA-3-SQLCoder-8B and LLaMA-2-7B-chat-hf—across three types of user prompts: basic, instruction-based, and template-based. We utilize the Yale University's Spider 1.0 NLP dataset for evaluating the LLMs, which includes natural language questions and corresponding SQL queries. The results reveal that prompt engineering impacts the models' accuracy, with LLaMA-3-SQLCoder-8B outperforming LLaMA-2-7B-chat-hf. It provides insights into how various prompt templates can influence query generation accuracy and response time.

1 Introduction

Retrieving structured data from Relational Databases (RDBMS) is essential for businesses but is often challenging due to the complexity of crafting Structured Query Language (SQL) queries, particularly when dealing with multiple schemas, relationships, and complex joins. Writing and executing such queries can be cumbersome and time-consuming. Initial efforts to automate the conversion of natural language questions to SQL have utilized machine learning techniques, including pattern matching, reinforcement learning (Zhong et al., 2017), and deep learning (Kumar et al., 2022). With the advancements in Large Language Models, the accuracy of text-to-SQL conversion has significantly improved, making data querying more accessible and efficient.

To leverage the use of Large Language Models (LLMs) effectively, this study addresses the following research objectives:

Benchmark different LLMs in text-to-SQL conversion by designing an effective user prompt template and analyze how the performance of LLMs is affected by prompt engineering.

The paper is organized as follows: Section 2 reviews relevant work in the field of text-to-SQL conversion and the application of LLMs. Section 3 describes the data utilized for the study. Section 4 outlines the methods employed to evaluate the LLMs. Section 5 presents the experiments conducted and the corresponding results. Finally, Section 6 discusses the outcomes and implications of the findings. The implementation can be found at Github Repository.

2 Related Work

While various LLMs can now efficiently convert text-to-SQL, we need a mechanism to understand which LLM performs the best and under what circumstances, and can the accuracy be improved further by adjusting user inputs. The performance of various LLMs is benchmarked by (Zhang et al., 2024) in their paper 'Benchmarking the Text-to-SQL Capability of Large Language Models: A Comprehensive Evaluation'. In this work, authors focus on three combinations of user prompt template (user inputs): 1) Schema formats (DDL or Simple DDL), 2) Question formats (HTML, MD or, Coded) and 3) Chat format (Complete or Simple). Taking an inspiration by the work done in (Zhang et al., 2024), we focus on how effective user prompting affects the performance of various LLMs.

Comprehensive guidance on effectively prompting Large Language Models (LLMs) for text-to-SQL tasks can be explored in in (Chang and Fosler-Lussier, 2023). t provides insights into how thoughtful prompt design can lead to more accurate

and efficient text-to-SQL

3 Data

The dataset employed in this project is the Yale University's Spider 1.0 NLP Dataset, publicly available on Kaggle. It was primarily designed for developing and testing systems that convert natural language questions into SQL queries. Yale researchers generated the labels by pairing natural language questions to SQL queries ensuring that various representation of queries like basic retrieval, filtering, joins, nested queries, aggregations, and more complex SQL constructs are included in the dataset. The dataset consists of approximately 10,181 questions across 200 databases with schema in various domains. The files train_spider.json and test.json was combined into test_question.json and then converted to csv. This was done because the purpose of this project does not require train and test files seperately. The test_questions.csv file file has 3 columns namely db_id, query (human-labeled SQL) and the corresponding natural language question. Additionally, a database folder contains the schema files for each db_id.

4 Methodology

The models selected for this study are LLaMA-3-SQLCoder-8B and LLAMA-2-7b-chat-hf. Both models are variants of Llama architecture fine-tuned for text-to-SQL tasks and conversations respectively. These models were chosen due to its robust capabilities in understanding both natural language and structured queries. The aim of the project is to evaluate both the model's performance to various prompts compared to the predefined queries in the dataset.

(libraries used)

4.1 Prompt Engineering

The project focuses on the process of designing inputs (prompts) to guide an LLM to generate more accurate and relevant outputs. By crafting good quality and specific prompts, the model can be enhanced to translate natural language into SQL queries precisely.

Queries are generated by passing the prompt and the question as input for the model. Different types of queries were applied in this study. The basic prompt directly asks the model to generate an SQL query for a given question. The schema related to the question is also provided in the prompt. The instruction-based prompt provides instructions on the format and constraints of the query. Finally, the template-based prompt uses a more structured approach where requirements like accuracy, efficiency, clarity and limitations are also provided. Furthermore, database normalization is ensured to improve query performance.

- **Basic prompt:**
  Generate a SQL query to answer this question: '{question}'.
  {schema}

- **Instruction-based prompt:**
  Generate a SQL query to answer this question: '{question}'
  – Important: You must use the INTERSECT clause instead of the IN clause.
  – For any comparison involving lists of values, use INTERSECT between SELECT statements.
  – Example:
  – Instead of:
  *– SELECT * FROM table1 WHERE column1 IN (SELECT column2 FROM table2);*
  – Use:
  *– SELECT * FROM table1 WHERE column1 = ANY (SELECT column2 FROM table1 INTERSECT SELECT column2 FROM table2);*

  DDL statements:
  {schema}

- **Template-based prompt:**
  Generate a SQL query to answer this question: '{question}'
  – Important: You must use the INTERSECT clause instead of the IN clause.
  – For any comparison involving lists of values, use INTERSECT between SELECT statements.
  – Example:
  – Instead of:
  *– SELECT * FROM table1 WHERE column1 IN (SELECT column2 FROM table2);*
  – Use:
  *– SELECT * FROM table1 WHERE column1 = ANY (SELECT column2 FROM table1 INTERSECT SELECT column2 FROM table2);*

  Requirements:

– Correctness: Ensure all joins, conditions and aggregations are correct based on the schema and avoid 'NULL' issues where applicable.
– Efficiency: Avoid unnecessary columns, tables and joins to enhance efficiency.
– Clarity: Use table aliases to clarify complex logic.
– Limitation: Ensure only one column is selected per query.

Normalization Consideration:
– Ensure that the schema adheres to at least 3NF (Third Normal Form), which involves eliminating redundant data and ensuring that all non-key attributes are fully dependent on the primary key.
– If applicable, normalize tables to reduce data anomalies, ensuring that each table represents a single entity or concept.
– Avoid repeating groups of data and ensure that relationships between tables are appropriately represented through foreign keys.

DDL statements:
{schema}

4.2 Evaluation

Since the same query can be written in multiple ways it is often the case that exact matches are zero but the results match because they are logically same. Hence, we primarily use two evaluation metrics:

- **Exact Match** - The query generated by the model and the query in golden truth is an exact match.
- **Results** - The result obtained on executing the query generated by the model is same as the result obtained on executing the query in golden truth.

Additionally, the response time is also used for evaluation because it is an important measure for usability. This is the time taken for an LLM to generate the SQL query.

5 Experiment and Results

Describe your experiment setup, evaluation process and report your project's outcomes, such as classification accuracy or any pertinent evaluation metrics. Include any noteworthy findings that arose from your analysis.

5.1 Experiment setup

The experiment was conducted using Google Colab Pro equipped with T4 GPU. Due to limitations in computational resources, the dataset was restricted to only 450 questions for this study. The study focuses on evaluation on two large language models Llama-3-sqlcoder-8b and Llama-2-7b-chat-hf.

- **Llama-3-sqlcoder-8b** is an open source model available to download directly from Hugging Face.
- **Llama-2-7b-chat-hf** requires authorization and can be accessed only using an API key. This model was integrated using the Hugging Face API.

The experiment involved testing each model with three different prompts. These prompts were designed to test how the model's performance can be enhanced to generate accurate SQL queries.

5.2 Evaluation Process

The queries generated by the model are written to a file *generated_queries.txt*. The file *test_questions.csv* contains the golden truth which is used for comparison.
Firstly, an exact match is checked between the two queries. The results of this comparison is written into a file *comparison_results.txt*. The generated query is written followed by 'Y' or 'N', if they are an exact match or not, respectively. All the queries that have 'N' are then checked for the results. Both the generated query and gold truth query are executed in the database and the results are compared. Similar to the previous comparison, the generated query is written to a file *final_comparison_results.txt* followed by 'Y' or 'N', if the results match or not. The accuracy can be defined as:

$$\frac{\text{No. of exact match or same result queries}}{\text{Total number of queries}} \quad (1)$$

The response time for each model is measured by a progress bar implemented using the tqdm library in Python.

5.3 Results

The model accuracies are shown in Table 1 and response time in Table 2.

Model	Prompt 1	Prompt 2	Prompt 3
sqlcoder	63.33%	65.78%	66.89%
chat-hf	51.33%	53.24%	55.45%

Table 1: Accuracy results

Model	Prompt 1	Prompt 2	Prompt 3
sqlcoder	63	97	82
chat-hf	130	172	155

Table 2: Response Time (in minutes)

6 Discussion

6.1 Reflection

Llama-3-sqlcoder-8b is a Llama variant fine-tuned for text-to-SQL tasks whereas Llama-2-7b-chat-hf is a Llama variant fine-tuned for conversational tasks. Therefore, sqlcoder is expected to perform better than the chat-hf and this is confirmed by the results in terms of both accuracy and response time.
Prompt engineering has proven to be an effective approach in improving the LLM's ability to generate accurate SQL queries. By refining the prompts, we were able to enhance the models' performance in producing more relevant results. However, implementing contextual prompts could have generated better queries.
However, there were some challenges related to database creation. Some schema files were found to be incorrect, leading to failures in queries involving those tables. These errors caused certain queries to always return a result of 'N' for exact match, impacting the overall accuracy.
Another major issue arose from the presence of the same table name across multiple schemas. This resulted in conflicts when generating queries. In an attempt to resolve this, we renamed the tables using the schema name as a prefix (e.g., schemaName_tableName). However, this solution added complexity to the process, requiring additional measures to compare the queries with the golden truth correctly.
Additionally, the exact match criterion proved to be problematic. This is largely due to the "black box" nature of large language models, where the generation process is not fully transparent. As a result, the model may generate queries that are logically equivalent but semantically different from the queries in the golden truth, causing them to fail the exact match evaluation despite producing the correct results.

6.2 Limitations

A significant limitation of this study is computational resources. Despite using Colab Pro having a GPU RAM of 15 GB, these models give best results only for RAM greater than 20 GB. Both models used in this study — Llama-3-sqlcoder-8b (8 billion parameters) and Llama-2-7b-chat-hf (7 billion parameters) — are relatively small in comparison to other state-of-the-art models.
The execution time is another concern. Even with the reduced dataset of 450 questions used in this study, the execution time was approximately one or two hours. The total dataset comprising of over 10,000 questions would not execute in a reasonable amount of time. Exploring techniques such as model pruning or quantization could reduce the computational overhead.
An assumption taken for the scope of this study is the diversity of the dataset. Although it has been mentioned as having wide range of queries in the dataset description, this is not verified independently by us. To mitigate this, a manual review or more systematic analysis could be performed on the dataset to ensure that it contains different types of SQL queries, in turn leading to a more accurate translation of text to SQL queries.

6.3 Ethical impacts

Deploying text-to-SQL solutions in the real-world setting raises several ethical and societal implications. One primary concern is data privacy and security. When integrating LLM systems to databases, personal or sensitive information could be queried and retrieved. Vulnerabilities in the system could lead to data breaches or unauthorized access. This would violate privacy rights.
LLMs often inherit bias through their training data. This is a concern to us because in our study we used pretrained LLMs. For example, models might struggle to generate accurate SQL queries for underrepresented user demographics or rare database schemas, thus violating the principle of justice.
Transparency of LLM is also an ethical concern. Black box models lack explanation on how they arrive to a solution and can lead of misinterpretation of generated queries. This can be mitigated to an extent by employing explainable AI techniques.
Finally, there many be concerns for job displacement. By automating the task of generating SQL queries, there may be an impact on employment where roles relying on manual data analysis and

query crafting would become deprecated. However, this may also open opportunities for management roles for such systems.

6.4 Conclusion

In this study, we evaluated two Llama variants, Llama-3-sqlcoder-8b and Llama-2-7b-chat-hf, for their ability to convert natural language queries into SQL statements. The results confirm that Llama-3-sqlcoder-8b, fine-tuned specifically for text-to-SQL tasks, outperformed Llama-2-7b-chat-hf, which is optimized for conversational tasks. Through prompt engineering, we were able to enhance the performance of both models, achieving better accuracy and response times. However, challenges related to schema inconsistencies, table name conflicts, and the limitations of exact match criteria were encountered. Despite these challenges, the models demonstrated promising capabilities in generating accurate SQL queries, showing the potential of LLMs for automating database querying.

References

S. Chang and E. Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *Preprint*, arXiv:2305.11853.

Ayush Kumar, Parth Nagarkar, Prabhav Nalhe, and Sanjeev Vijayakumar. 2022. Deep learning driven natural languages text to sql query conversion: A survey. *Preprint*, arXiv:2208.04415.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. 2024. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *Preprint*, arXiv:2403.02951.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *Preprint*, arXiv:1709.00103.