Can LLM Already Serve as A Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs

Anukriti Singh, Bharath Somasundaram, Praneeth Kumar Thummalapalli, Shreya Sudhakaran
as18692@nyu.edu, bs4852@nyu.edu, pt2427@nyu.edu, ss18292@nyu.edu

Introduction

The translation of natural language instructions to SQL commands really is an important contribution to increasing the degree of interaction between computers and human beings—more so with respect to the dynamically advancing field of data management and retrieval. Essentially, the growth in data storage by organizations and businesses has to be proportional to the emerging needs for simple and efficient ways to question such data. This, of course, has thereby generated a lot of excitement around Large Language Models (LLMs) such as the ones being developed by OpenAI: Codex and ChatGPT, to automatically and unerringly generate SQL queries from a natural language description. It is worth noting that even with all these promising improvements, there lie real challenges when handling complex queries, perusing large databases, and interpreting natural language.

The text-to-SQL translation becomes an interface between natural language processing and database management. It stands for the access of data to a broader audience that could not be disclosed to the technical jargon of SQL. That approach could serve a dual purpose: to increase the productivity of data scientists and analysts and to release the value in the database for non-technical users. Recent work thus highlights the possible use of LLM in this field, but finding a model that would account for many databases, understand user intentions to produce the correct SQL, would still remain a very big challenge. The complexity of natural language and database structures demands sophisticated reasoning from these models.

This paper compares LLMs' capability in the generation of SQL queries and discusses how to enhance this skill.

It compares realistic database scenarios against the current approach to identify approaches for LLM in text-to-SQL translation. This study is rigorous in testing and innovative in experimentation, aiming at making attempts to identify better, more effective, and user-friendlier ways of conversion of text to SQL. We will show how the text-to-SQL field should be a big progress after reviewing the latest contributions and pinpointing the current bottlenecks, hence suggesting new ways of doing things. The remaining of this report is structured as follows: elaborating on the methodology, presenting findings and analysis, and finally, we interpret the results and give some future uses for LLMs in SQL query generation. We do hope that this paper will advance scientific understanding not only in the text-to-SQL translation but also help highlight practical applications that use LLMs in today's data-centric world.

Problem Formulation

The text-to-SQL translation process suffers from a number of challenges. The process of translating from natural language (NL) to structured query language (SQL), commonly termed "text-to-SQL translation," suffers from a number of challenges that come to play at the interface of natural language understanding

(NLU) and database query generation. It should, in principle, model the ability to understand semantics from the supplied natural language input, map it to a context structured from the database, and then, in the

1. Understanding the schema of some target database: Being able to form appropriate SQL for a target database is involved in understanding the schema of the database concerning the names of the tables, the relationships between the tables, and constraints on tables.

2. Varigirlity in SQL and complexity: From simple single-table queries to extremely complex ones, SQL queries may have multiple references to tables and might involve nesting along with aggregation and advanced functions.

3. Generalization Across Databases: The main difficulty of Large Language Models (LLMs) lies in the susceptibility of the models to This severely limits the use of LLMs in applications such as text-to-SQL generation that require strong performance. There might be a limitation of the LLMs only to a very particular domain or use case if there is no strong mechanism to enable generalization across databases, reducing their applicability to more general problems in a scenario of data querying.

Related Work

In [1], Yuanzhen Xie et al. propose a new method for text-to-SQL conversion that aims at substantially improving the model handling questions of a greater variety. This method tries to make the model handling questions of a greater variety more effective and efficient.

The information determination contains step filters: it identifies and focuses on the key needed data in building the SQL query, while the rest are filtered out. This is important in one way: to focus the concentration of a model to be deployed on a few relevant things within the question to be asked. The Classification and Hint Module classifies the problems into different categories with respect to their needs: requirements for table joining or requirements for nested query execution and supplies relevant hints to lead the model along its best problem-solving path.

The most important part of our approach is SQL generation. Using templates and hints from previous steps, the given step should accurately generate an optimized SQL query.

The Self-Correction process further identifies and refines these queries to improve accuracy and efficiency, being sensitive to the point that the automation brings errors. Active Learning further improves the model's capability from learning from previous errors.

Finally, it prepares itself not only for the future challenge alterations of query complexity and performance feedback but indeed improves the current performance. This will allow for techniques to be integrated at the submodule level, therefore allowing increased precision in the adjustment and better detection of errors, in particular through the use of advanced algorithms for processing natural language. We do so with active learning, which integrates user feedback to ensure that the model keeps growing well-aligned with real-world application, and it keeps improving in accuracy and efficiency in generating the SQL query. All this offers a fine method for stronger immediate output and adaptability, of relevance towards evolving needs in data querying.

In [2] Shuaichen Chang and et.al explore effective ways to use prompt engineering to enhance the performance of Large Language Models (LLMs) in converting natural language questions into SQL queries across different scenarios. This research thoroughly examines various prompt creation techniques, focusing on how databases are represented and how examples are demonstrated in zero-shot, single-domain, and cross-domain environments. It highlights the significance of LLMs' in-context learning capabilities for the text-to-SQL task, evaluating different approaches in prompt design and database depiction, supported by examples in the mentioned settings.

The study reveals that employing different strategies in constructing prompts, particularly in presenting database schemas and in the content and examples shown, introduces variability. This variability complicates the task of comparing approaches due to a lack of uniformity in prompt construction or its impact on LLM performance. It points out that while showcasing examples in various formats helps mitigate LLM sensitivities within specific domains, these measures do not eliminate the need for detailed representation of table contents.

Researchers Chang and Fosler-Lussier utilized the Spider dataset, a cross-domain benchmark for the text-to-SQL task, to evaluate their model's performance in zero-shot, single-domain, and cross-domain settings. They found that the manner in which table relationships and content are presented in the prompts significantly affects LLM performance. The study pinpoints an optimal prompt length that enhances model performance in cross-domain scenarios, suggesting this "sweet spot" might be linked to the inherent characteristics of LLM architecture or its training data.

This research provides insightful observations on developing effective prompts for text-to-SQL tasks, emphasizing the importance of both the database's schema and content, as well as the use of demonstration examples. It illuminates key factors that influence LLM performance in generating SQL from text, laying the groundwork for future studies on how to refine and utilize LLMs for database querying more effectively.

In [3] Zhenwen Li and et.al development the technology of Large Language Models (LLMs) has real impacts on the advancement made in the text-to-SQL technologies, bringing new methodologies toward converting natural language instruction to an executable SQL query. This is an essential translation, so that users without special technical skills can carry out databases querying. Good examples of such invaluable work in this domain are those contributed to by Li and Xie, aimed at finding the most appropriate SQL query from a candidate list created by the text-to-SQL models. They show an automatic generation of test case approach that uses LLM in predicting the expected execution result; hence, they benefit further when candidate SQL required to be queries written for execution by the underlying database are more. That is indeed a great stride in furthering the accuracy and usefulness in the text-to-SQL translations field, using the LLMs ability to understand and produce database queries.

Notably, the method presented by Li and Xie is of a high critic value to the present scope of research in this field of text-to-SQL parsing. It models the database and uses LLMs to find the ground truth or what is expected from the output, resulting from executing the given SQL query over this database. The novelty here lies in the fact that, hence, this approach tackles a relatively less-explored problem of test case generation for Text-to-SQL, which is paramount to the activity of benchmarking and improving the quality of Text-to-SQL systems. Through experiments on ways of generating easily predictable databases for LLMs and designing intuitive prompts, they lay a systematic vb foundation on how the performance of the text-to-SQL model can be improved through reranking based on test cases. The results of their experiments on the Spider validation dataset returned a really big jump in model performance, indicating their approach has great potential to push forward the broader goals of making data querying more intuitive and effective for a wide range of users.

Several studies, including those by Yuanzhen Xie et al., Shuaichen Chang, and Zhenwen Li, explore methods and challenges in improving text-to-SQL conversion.

Methods Architecture and Design

Information Determination

The Information Determination submodule is a critical component in the process of enhancing Large Language Models (LLMs) for text-to-SQL tasks. It serves as the initial step in a workflow designed to refine the model's ability to generate SQL queries from natural language instructions. This submodule's primary function is to filter and focus the model's attention by identifying and isolating the essential data

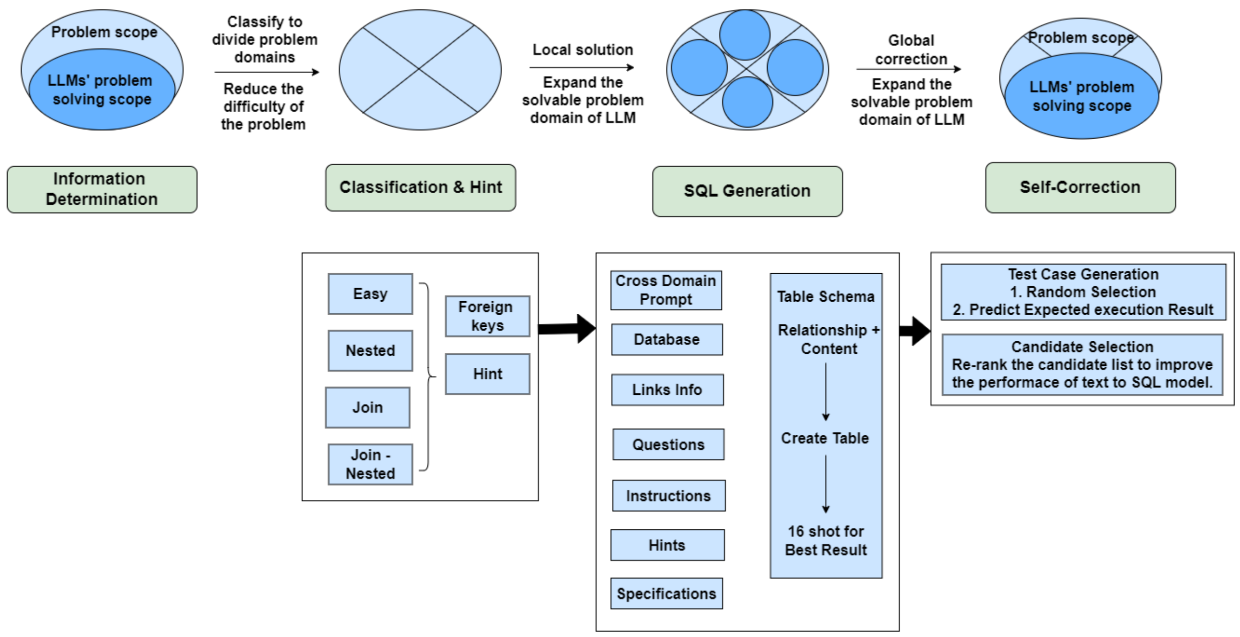

Figure 1: The overall structure of the SQL model

elements needed to accurately formulate the SQL query. Here's a deeper look into its role and importance:

Role and Functionality Data Element Identification: At the outset, Information Determination involves parsing the natural language query to discern the key data elements referenced within. These elements could be specific database fields, tables, or conditions that are crucial for constructing the SQL query. The goal is to understand the query's requirements from a data perspective.

Redundancy Elimination: A significant challenge in interpreting natural language instructions is dealing with redundant or irrelevant information that can clutter the query-making process. This submodule actively filters out such noise, ensuring that only pertinent information is considered in the subsequent stages of SQL generation.

Enhancing Model Focus: By narrowing down the scope of information to be processed, Information Determination helps in concentrating the model's cognitive resources on the task at hand. This focused approach is instrumental in improving the accuracy and relevance of the generated SQL query.

Importance Accuracy Improvement: Precision in identifying the necessary data elements directly impacts the accuracy of the resulting SQL query. A well-executed Information Determination step ensures that the generated query is tightly aligned with the user's intent as expressed in the natural language instruction.

Efficiency Enhancement: Streamlining the information processing at this early stage can significantly enhance the overall efficiency of the SQL generation process. By eliminating unnecessary considerations, the model can generate queries more swiftly and with fewer computational resources.

Adaptability to Complex Queries: In handling complex queries that involve multiple tables, conditions, or aggregated data, the ability to accurately determine the required information becomes even more crucial. This submodule enables the model to tackle such complexities by providing a clear map of essential data elements.

This section discusses the role and functionality of the Information Determination submodule in enhancing LLMs for text-to-SQL tasks.

Classification and Hint

The Classification and Hint submodule is a pivotal component in the workflow paradigm designed to enhance Large Language Models (LLMs) for text-to-SQL tasks. After the initial Information Determination phase, which identifies the essential elements of a query, the Classification and Hint phase further refines the process by categorizing the query based on its characteristics and providing strategic hints to guide the model towards generating an accurate SQL query. This phase is instrumental in tailoring the model's approach to address the specific demands of each query, thereby improving the overall precision and effectiveness of the SQL generation process.

Role and Functionality Problem Categorization: The Classification and Hint submodule analyzes the filtered information from the first phase to categorize the query into predefined classes based on its structural and functional requirements. These categories could include distinctions based on whether the query necessitates joining tables, involves nested queries, requires aggregate functions, or fits into other specific SQL patterns.

Guidance through Hints: Based on the category assigned to a query, this submodule generates targeted hints that act as cues for the LLM. These hints are crafted to steer the model in the right direction, emphasizing key aspects that need to be considered when generating the SQL code. For example, for a query categorized as requiring a table join, the hint might include reminders about ensuring correct join conditions.

SQL Generation - How to Prompt LLMs for Text-to-SQL: A Study in Zero-shot, Single-domain, and Cross-domain Settings

The paper "How to Prompt LLMs for Text-to-SQL: A Study in Zero-shot, Single-domain, and Cross-domain Settings" by Shuaichen Chang and Eric Fosler-Lussier investigates the effective use of prompts to improve the performance of Large Language Models (LLMs) like GPT-3 Codex and ChatGPT in generating SQL queries from natural language questions (NLQs). This research is significant because it addresses the challenge of converting NLQs to SQL queries, enabling users to interact with databases without needing to know SQL. The study is set against the backdrop of varying methodologies in prompt construction that have emerged in previous research, leading to inconsistencies in evaluating the impact of these strategies on LLM performance.

Core Contributions The paper makes several key contributions to the field of text-to-SQL translation with LLMs:

Comparative Evaluation of Prompt Strategies: It provides a systematic evaluation of different strategies for constructing prompts, particularly focusing on how databases and demonstration examples are represented. This is crucial for understanding how the details included in prompts influence LLM performance across three settings: zero-shot, single-domain, and cross-domain.

Insight into Prompt Construction: The study reveals that the representation of table relationships and content within prompts is vital for effectively prompting LLMs. Interestingly, while in-domain demonstration examples can mitigate the models' sensitivity to various representations of database knowledge, they cannot entirely substitute for direct representation of table content.

Identification of Optimal Prompt Length: Through experimentation, the research identifies an optimal length for prompts that enhances LLM performance, particularly in the cross-domain setting. This finding suggests that there is a ”sweet spot” for prompt length, likely influenced by the models’ architecture and training data specifics.

Comprehensive Methodology: The paper employs a rigorous experimental setup using the Spider dataset, a benchmark for cross-domain text-to-SQL tasks. This allows for a thorough evaluation of GPT-3 Codex and ChatGPT models under varied conditions, providing a broad view of their capabilities and limitations.

Guidance for Future Research: By detailing the impact of prompt construction on LLM performance, the paper offers valuable insights for future research in text-to-SQL translation. It underscores the importance of careful prompt design and highlights areas where further improvements can be made.

Impact and Implications This research is pivotal for advancing the use of LLMs in database querying applications, making data access more intuitive and efficient for a wide range of users. By elucidating the critical aspects of prompt construction that influence model performance, the study lays the groundwork for developing more robust and effective text-to-SQL translation methodologies. Furthermore, the identification of an optimal prompt length opens new avenues for optimizing LLMs for specific tasks, balancing the need for comprehensive input information against the constraints of model architecture.

Overall, ”How to Prompt LLMs for Text-to-SQL” significantly contributes to our understanding of leveraging LLMs for SQL query generation. It not only provides a clear evaluation of current methodologies but also charts a path forward for enhancing the capabilities of these powerful models in practical applications.

Self-Correction - Testing and selecting the best SQL results from the candidates - Using LLM to select the right SQL Query from candidates

The Self-Correction module is a sophisticated phase in the workflow meant to guarantee more accuracy and effectiveness with which to handle ambiguities in natural language instruction-derived SQL queries, using Large Language Models (LLMs). This module becomes very important in the sharpening of the text-to-SQL translation task, because it defines how the best SQL query among a set of candidate queries is picked. First, we explain how the Self-Correction module works, focusing on test cases and the best possible determination of the SQL results from candidate one; all this is done by leveraging LLMs to choose the best among the best.

Functionality and Process

Candidate SQL Query Generation: The text-to-SQL model takes a question in natural language as input and produces a list of candidate SQL queries for that question. It contains a model’s best attempt to make translations of given instructions into SQL, taking into account the language’s complexity and the nuances of the database schema. Test Case Generation and Evaluation: The Self-Correction module generates test cases in order to arrive at which candidate query is most likely to be correct. All these test cases, in fact, are nothing else but scenarios or examples where every candidate SQL query is exercised. Generation of test cases is a sensitive job, wherein it becomes necessary for the module to generate or predict the possible inputs and expected outputs (execution results) for the SQL queries.

In other words, given the test cases for a collection and the context of the database, it would be able to make use of LLM powers with GPT-4 or Codex and, thereby, automatically prepare a sufficiently filled collection with respect to that context and the expected output from executing an SQL query over that database. Execution and comparison: The test cases are run for each candidate SQL query, and the module compares the actual outputs per query with the output it had earlier generated during the test case generation.

This comparison helps to bring out which of the two candidate SQL queries closely yield the results as per the expected outcomes, hence indicating the accuracy and effectiveness in the translation of the natural language original instruction. Re-Ranking and selection:

The candidate SQL queries are re-ranked, and the results are selected for the candidates whose performance, in terms of accuracy on test cases, was best by the Self-Correction module. Based on the above criteria, the module weighs the best-ranked SQL query in order to be selected as the best possible translation of the natural language instruction into SQL. This is important as it will give direct impact to the effectiveness and reliability of the text-to-SQL translation system.

Importance

Improved Accuracy: After repeated testing on the generated test cases, the Self-Correction module facilitates users to have an improved accuracy in SQL query formulation from the natural language input. It ensures a higher probability for the executed database queries to return desired results.

Error Reduction: The module will consider identifying and correcting the errors contained in the SQL queries provided by the candidate before their final selection, with the aim of reducing possibilities of running wrong or inefficient queries over the database. Better model performance: the process of evaluating the test cases helps in self-correction and refining, benefiting in better performance from continuous improvement with respect to the capability of the model to translate the instructions in natural language accurately to SQL queries. It further helps iterate the model to make it be more accurate, thus improving the performance and reliability of the overall text-to-SQL system. In summary, Self-Correction can be taken as a module crucial in the whole cycle of text-to-SQL translation, which leverages the powers of LLMs to produce test cases, assess candidate SQL queries, and finally, the selection of the most accurate query. This module showcases state-of-the-art AI and ML methodologies that can be harnessed to improve accuracy and efficiency in automated programming tasks, hence leading to friendlier user access to data querying.

Result

Based on the literature review we have come up with an architecture that generates SQL queries using LLM. The architecture initially takes the relevant information necessary from the database and recognises the relevant data necessary. Then the proposed architecture uses the result specified in [2] for constructing the best prompt for the job. Using the prompt the LLM will generate multiple queries, these go through the next stage of the architecture which deals with choosing the right query. This is done through the adaptation of the algorithm specified in [3]. The resulting architecture will be an integrated implementation of the all proposed novel solution of the papers [1][2] and [3]. It is expected that on adapting the above mentioned architecture LLM can be easily used as a Database inference engine.

References

1. Chang S. & Fosler-Lussier E. (2023). *How to Prompt LLMs for Text-to-SQL: A Study in Zero-shot Single-domain and Cross-domain Settings.* `https://arxiv.org/abs/2306.12345`

2. Li Y. & Xie S. (2024). *Self-Correction - Testing and selecting the best SQL results from the candidates - Using LLM to select the right SQL Query from candidates.* `https://arxiv.org/pdf/2305.11853.pdf`

3. Li Y. & Xie S. (2024). *SQL Generation - How to Prompt LLMs for Text-to-SQL: A Study in Zero-shot Single-domain and Cross-domain Settings.* `https://arxiv.org/pdf/2401.02115.pdf`

4. Samuel Arcadinho, David Aparício, Hugo Veiga & António Alegria *T5QL: Taming language models for SQL generation.* `https://arxiv.org/pdf/2209.10254.pdf`

5. Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding & Jingren Zhou *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation.* `https://arxiv.org/pdf/2308.15363.pdf`

6. Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding & Jingren Zhou *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation.* `https://arxiv.org/pdf/2308.15363.pdf`

7. Han Wang, Archiki Prasad, Elias Stengel-Eskin, Xiuyu Sun, Yichen Qian & Mohit Bansal *Soft Self-Consistency Improves Language Model Agents.* `https://arxiv.org/pdf/2402.13212v1.pdf`

8. Xiang Lisa Li & Percy Liang *Prefix-Tuning: Optimizing Continuous Prompts for Generation.* `https://arxiv.org/pdf/2101.00190.pdf`

9. Yuanzhen Xie, Xinzhou Jin, Tao Xie, Mingxiong Lin, Liang Chen, Chenyun Yu, Lei Cheng, Chengxiang Zhuo, Bo Hu & Zang Li *Decomposition for Enhancing Attention: Improving LLM-based Text-to-SQL through Workflow Paradigm.* `https://arxiv.org/pdf/2402.10671v1.pdf`

10. Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer & Christopher Ré1 *LANGUAGE MODELS ENABLE SIMPLE SYSTEMS FOR GENERATING STRUCTURED VIEWS OF HETEROGENEOUS DATA LAKES.* `https://arxiv.org/pdf/2304.09433.pdf`

11. Zhenwen Li & Tao Xie *Using LLM to select the right SQL Query from candidates.* `https://arxiv.org/pdf/2401.02115.pdf`