Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 08, August-2024

Next-Generation Text-to-SQL: A Survey of Advanced Reasoning Enhancements Techniques

Tejaswini Kumar
Independent Researcher
Columbia Univ Alum

Kishor Yadav Kommanaboina
Independent Researcher
The Ohio State Univ Alum

Bhargava Kumar
Independent Researcher
Columbia Univ Alum

***Abstract*— Text-to-SQL is a crucial task in natural language processing that translates natural language queries into SQL statements, enabling non-technical users to interact with databases seamlessly. The complexity of accurately generating SQL queries from diverse and complex queries has led to the development of various reasoning enhancement techniques. This survey paper reviews the latest advancements in this field, focusing on methods such as Chain of Thought (CoT) prompting, ACT-SQL, QDecomp, Least-to-Most Prompting, Program of Thoughts (PoT), SQL-Craft, and SQL-PaLM. These techniques improve large language models' abilities to handle complex reasoning tasks and generate more accurate SQL queries by incorporating structured thinking processes, automated prompt generation, and iterative refinement. We discuss the methodologies and effectiveness of each approach, identify the challenges they address, and suggest potential future research directions. This survey aims to provide a comprehensive overview of the current state of reasoning enhancement in Text-to-SQL, highlighting the innovations that are driving the field forward.**

***Keywords*— Chain of Thought (CoT), ACT-SQL, Query Decomposition (QDecomp), Least-to-Most Prompting, Program of Thoughts (PoT), SQL-CRAFT, SQL-PaLM, Large Language Models (LLMs), Schema Linking, Pre-trained language models (PLMs)**

I. INTRODUCTION

The task of translating natural language questions into SQL queries, known as Text-to-SQL, is a vital area of research in natural language processing (NLP). By enabling users to interact with databases through natural language, Text-to-SQL systems lower the barrier for data access, making it possible for non-technical users to perform complex data queries without specialized knowledge of SQL.

Despite the progress made, significant challenges persist. These systems must accurately interpret diverse and often complex natural language queries, align them with varying database schemas, and generate precise SQL statements that fulfill the user's intent. Achieving high accuracy in SQL generation is particularly challenging when dealing with ambiguous or multi-step queries that require sophisticated reasoning.

To address these challenges, recent research has focused on enhancing the reasoning capabilities of Text-to-SQL models. Reasoning enhancement techniques improve how models understand, process, and respond to complex queries by incorporating structured thought processes and intermediate reasoning steps, bridging the gap between natural language understanding and SQL query formulation.

This survey explores the latest advancements in reasoning enhancement for Text-to-SQL systems, focusing on several innovative approaches: Chain of Thought (CoT) prompting, ACT-SQL, QDecomp, Least-to-Most Prompting, Program of Thoughts (PoT), SQL-Craft, and Self-Consistency. These methods offer various strategies for guiding models through complex reasoning tasks, improving their ability to generate accurate and effective SQL queries.

In the following sections, we provide a comprehensive overview of each technique, discussing their methodologies, applications, and impact on the field of Text-to-SQL. We also examine the challenges these methods aim to solve and explore potential directions for future research.

II. BACKGROUND

The field of text-to-SQL[1] has evolved significantly over the years, transitioning from rule-based methods to deep learning approaches and, more recently, to the integration of pre-trained language models (PLMs) and large language models (LLMs).

- Rule-based Methods: Early text-to-SQL systems relied on manually crafted rules and heuristics to map natural language questions to SQL queries. These methods, while effective in simple, controlled domains, were heavily dependent on domain-specific knowledge and extensive feature engineering, limiting their flexibility and ability to generalize to diverse and complex queries.
- Deep Learning-based Approaches: The advent of deep learning introduced sequence-to-sequence models and encoder-decoder architectures, such as LSTMs and transformers, into text-to-SQL systems. Techniques like intermediate representations and sketch-based slot filling, as seen in models like RYANSQL, improved the handling of complex queries and cross-domain generalization. Graph neural networks (GNNs) further enhanced these models by capturing relationships between database elements through schema dependency graphs.
- PLM-based Implementation: The adoption of pre-trained language models (PLMs), such as BERT and RoBERTa, marked a significant advancement. Fine-tuning these models on text-to-SQL datasets allowed them to leverage their pre-trained semantic representations and language understanding capabilities, improving the accuracy of SQL generation by incorporating schema information and better understanding database structures and constraints.
- LLM-based Implementation: Recently, large language models (LLMs) like the GPT series have shown promise for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 08, August-2024

text-to-SQL tasks due to their advanced text generation capabilities and extensive knowledge bases. Researchers have explored using prompt engineering and fine-tuning these models on specific datasets to enhance SQL generation. This emerging area of research focuses on leveraging LLMs' reasoning abilities, incorporating domain-specific knowledge, and optimizing fine-tuning strategies to improve performance and generalization.

The evolution from rule-based systems to deep learning models, and now to PLMs and LLMs, illustrates a significant progression in text-to-SQL technology, each stage bringing enhanced flexibility, accuracy, and capability to handle complex queries in real-world applications.

III. REASONING ENHANCEMENT TECHNIQUES

A. Chain of Thought (CoT)

Chain of Thought (CoT)[2], [3], [4], [5] prompting is a reasoning enhancement technique that guides large language models (LLMs) to generate intermediate reasoning steps before producing the final output. This method helps break down complex problems into simpler, manageable steps, mimicking human logical reasoning. In Text-to-SQL tasks, CoT prompting involves instructing the model to "think step by step" when constructing SQL queries from natural language questions. By explicitly generating intermediate steps, the model can systematically address different components of the query, such as table selections, conditions, and joins, which improves its ability to accurately translate complex queries into SQL statements.

Despite its successes, CoT prompting in Text-to-SQL tasks has unique challenges. It often requires extensive human-annotated examples to be effective, which can be time-consuming and labor-intensive. Moreover, its effectiveness heavily relies on the quality and relevance of these examples, which can limit the model's ability to generalize across diverse domains. To overcome these limitations, recent research has focused on refining CoT strategies for SQL generation by incorporating schema-specific knowledge and using automated methods to generate CoT examples, thereby enhancing the model's SQL generation capabilities.

B. QDecomp

QDecomp[5] or Query Decomposition, is a reasoning enhancement technique designed to improve the accuracy of Text-to-SQL models by breaking down complex queries into simpler, more manageable components. This approach focuses on decomposing a natural language query into a series of logical steps, each corresponding to a specific component of the SQL query. By segmenting the query into distinct parts—such as selecting fields, applying filters, and defining table joins—QDecomp allows the model to concentrate on one aspect of the query at a time, thereby reducing cognitive load and minimizing the potential for errors.

The decomposition strategy employed by QDecomp is particularly useful in handling queries that involve multiple operations or require advanced reasoning skills. By structuring the query generation process into a sequence of smaller tasks, QDecomp enhances the model's ability to produce accurate SQL statements for complex queries. Additionally, QDecomp utilizes natural language templates to articulate each logical step of the SQL query, aligning these steps with the execution order of the SQL operations. This method ensures that the generated SQL is both logically coherent and syntactically correct, thereby improving the overall performance of Text-to-SQL models, especially in scenarios involving complex query structures and relationships between database elements.

C. Least-to-Most

Least-to-Most[5] prompting is a reasoning enhancement technique that improves the performance of Text-to-SQL models by decomposing complex queries into simpler sub-queries. This approach starts with the simplest elements of a query and progressively tackles more complex components in a step-by-step manner. By addressing easier parts first, the model reduces overall complexity and provides a foundation for handling more difficult aspects, incrementally building up to a complete SQL statement and reducing the risk of error propagation in multi-step queries.

This technique is particularly effective for queries involving multiple operations, such as joins, aggregations, and nested subqueries, as it helps the model understand the logical flow of the SQL statement. However, while it enhances performance in complex scenarios, it may not be necessary for simpler queries where direct SQL generation is sufficient. Additionally, the iterative process can introduce extra computational overhead due to multiple reasoning steps. Despite these limitations, Least-to-Most prompting remains a valuable strategy for enhancing the reasoning capabilities of Text-to-SQL models, especially when a structured, incremental approach is beneficial.

D. ACT-SQL

ACT-SQL (Automatic Chain-of-Thought SQL)[6] is a reasoning enhancement technique that improves SQL query generation by automatically creating Chain of Thought (CoT) examples for training large language models (LLMs). Unlike traditional CoT prompting, which relies on manually annotated examples, ACT-SQL automates this process, making it more scalable and efficient.

ACT-SQL works by decomposing a natural language query into smaller segments and aligning these with the corresponding columns and tables in the database schema using a similarity function. This automated alignment forms structured CoT prompts, helping the model understand the logical relationships between query components and the schema, thereby enhancing its ability to generate accurate SQL statements.

By systematically generating CoT examples, ACT-SQL improves the model's ability to perform complex reasoning tasks, reducing reliance on manually crafted data and enhancing generalization across different domains. This automation streamlines training and boosts the performance of Text-to-SQL models, especially in scenarios involving diverse and complex database schemas.

E. SQL-PaLM (Self Consistency)

SQL-PaLM (Pathways Language Model)[7] enhances the reasoning capabilities of Text-to-SQL models through the concept of self-consistency, which is based on approaching complex reasoning problems in multiple ways, each potentially leading to a correct solution. For Text-to-SQL, this involves generating several different SQL queries for a natural language

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 08, August-2024

query and selecting the most consistent one based on execution feedback or voting mechanisms. This approach increases the robustness and reliability of SQL generation by considering multiple perspectives and pathways to find the best solution.

SQL-PaLM applies self-consistency by using diverse prompts or variations in query generation to explore different SQL statements. The model evaluates these queries based on their execution results or alignment with expected outputs, selecting the one with the highest consistency score. This method refines SQL generation accuracy and helps identify and correct errors that could occur with a single-path approach. By incorporating multiple reasoning pathways and executing them in parallel, SQL-PaLM effectively handles complex queries and adapts to various database schemas, enhancing overall Text-to-SQL system performance.

F. Program of Thoughts(PoT)

Program of Thoughts (PoT)[8] is a reasoning enhancement technique that extends Chain of Thought (CoT) prompting by incorporating intermediate programming steps, such as generating Python code, to aid SQL query generation. PoT leverages Python's computational and logical structuring capabilities to enhance the model's reasoning process. In Text-to-SQL tasks, PoT requires the model to generate Python code snippets as intermediate steps, breaking down complex queries into simpler operations that can be systematically translated into SQL. This approach allows the model to perform arithmetic computations, logical reasoning, and data manipulation tasks before converting them into SQL syntax.

By generating and executing Python code, PoT improves the model's ability to handle complex queries involving calculations, aggregations, and conditional logic, verifying each step's correctness and reducing errors in the final SQL output. This method also enhances interpretability, as the Python code provides a transparent view of the model's thought process. PoT is particularly effective for datasets requiring advanced arithmetic reasoning or multi-step computations, utilizing Python's robust capabilities before SQL generation. Overall, PoT enriches Text-to-SQL models, enabling more accurate and reliable SQL queries by combining natural language understanding with programmatic reasoning.

G. SQL-CRAFT

SQL-CRAFT (SQL through Interactive Refinement and Enhanced Reasoning)[8] is a reasoning enhancement technique designed to improve Text-to-SQL models by integrating interactive correction loops and Python-enhanced reasoning. Central to SQL-CRAFT is the Interactive Correction Loop (IC-Loop), which iteratively refines SQL queries based on feedback from the database management system (DBMS). If errors or inefficiencies are detected in the initial SQL query, the IC-Loop prompts the model to revise the query, mimicking human trial-and-error refinement to produce more accurate and executable SQL statements.

SQL-CRAFT also incorporates Python-enhanced reasoning to handle complex queries better. By generating intermediate Python code, the model can perform logical operations, calculations, and data manipulations that are challenging to express directly in SQL. This dual approach leverages Python's flexibility and SQL's querying capabilities, ensuring intermediate steps are logically sound before translating them into SQL. By combining interactive refinement with Python-enhanced reasoning, SQL-CRAFT enhances the model's ability to generate precise and executable SQL queries, especially for complex natural language inputs.

IV. CHALLENGES AND LIMITATIONS

Despite recent advancements, Text-to-SQL systems still face several challenges that limit their real-world effectiveness and usability:

- Limited and Diverse Training Data: Many Text-to-SQL models are trained on small, homogeneous datasets, which restricts their ability to generalize across different domains. Creating diverse, high-quality datasets requires extensive resources and domain expertise.
- Handling Complex and Ambiguous Queries: Models often struggle with complex queries involving multiple tables, nested subqueries, or advanced reasoning. Ambiguities in natural language can also lead to incorrect SQL generation, even with techniques like Chain of Thought (CoT) prompting and Program of Thoughts (PoT).
- Adaptability to Different Database Schemas: Adapting models to diverse and evolving database schemas remains challenging due to the unique structures and constraints of each database. Although schema-aware models and graph-based neural networks offer improvements, they are still limited by the static nature of training schemas.
- Error Propagation and Execution Failures: Errors in one step of multi-step query processes can propagate, affecting the entire SQL generation. Variations in SQL dialects across different database management systems (DBMS) further complicate the accurate execution of generated queries.
- Evaluation Metrics and Protocols: Current evaluation methods focus mainly on exact match accuracy, which may not fully capture model performance in practical scenarios, especially for complex or ambiguous queries. More comprehensive metrics are needed to assess robustness and correctness.
- Computational Costs: Text-to-SQL models, particularly those based on large language models (LLMs), require substantial computational resources for training and inference, limiting their scalability in resource-constrained and real-time environments.

Addressing these challenges is crucial for enhancing the accuracy, robustness, and usability of Text-to-SQL systems in diverse applications.

V. FUTURE DIRECTIONS

To further enhance Text-to-SQL systems and address current challenges, several future research directions are proposed:

- Expanding and Diversifying Training Data: Creating larger, more diverse datasets across various query types and database schemas is essential for improving model generalization and handling of complex queries. Collaborative efforts to develop multilingual and multi-domain datasets could significantly advance the field.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 08, August-2024

- Improving Model Adaptability and Robustness: Research should focus on enhancing model adaptability to diverse and evolving schemas through dynamic schema representation and schema-agnostic models, allowing better handling of changes in database structures.
- Enhancing Reasoning and Understanding: Developing advanced reasoning techniques that manage complex logic and ambiguity in natural language queries is crucial. Combining symbolic and neural reasoning or leveraging external knowledge bases can improve SQL query accuracy.
- Optimizing Computational Efficiency: As models become more complex, optimizing computational efficiency through efficient algorithms and model compression is important for real-time deployment and use in resource-constrained environments.
- Advancing Evaluation Protocols: Developing comprehensive evaluation metrics and benchmarks that reflect real-world performance, considering accuracy, robustness, efficiency, and adaptability, is necessary for better assessing model effectiveness.

By pursuing these directions, the NLP community can enhance the performance, flexibility, and usability of Text-to-SQL systems, making them more reliable for various real-world applications.

VI. CONCLUSION

Text-to-SQL systems have evolved significantly from early rule-based methods to sophisticated models that leverage deep learning, pre-trained language models (PLMs), and large language models (LLMs). These advancements have greatly improved the accuracy and flexibility of SQL generation, enabling models to handle more complex queries and adapt to diverse database schemas. Techniques like Chain of Thought (CoT) prompting, ACT-SQL, and schema-aware modeling have enhanced the reasoning capabilities of these systems, making them more robust in interpreting and generating SQL queries from natural language inputs.

Despite these advancements, several challenges remain. Current models still struggle with handling highly complex or ambiguous queries, adapting to evolving database schemas, and operating efficiently in resource-constrained environments. Additionally, the lack of diverse training data and comprehensive evaluation metrics limits the generalization and practical deployment of these systems. Future research should focus on developing more adaptable and efficient models, expanding the diversity of training datasets, and creating better evaluation protocols that capture real-world performance. By addressing these challenges, the field can continue to advance towards building more accurate and user-friendly Text-to-SQL systems that empower users to interact with databases seamlessly across various domains.

ACKNOWLEDGMENT

I would like to thank Editor Aakash for their valuable assistance in editing this paper. His attention to detail has helped improve the manuscript.

REFERENCES

[1] Z. Hong et al., ‘Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL’, arXiv preprint arXiv:2406.08426, 2024.
[2] D. Gao et al., ‘Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation’, Proceedings of the VLDB Endowment, vol. 17, no. 5, pp. 1132–1145, Jan. 2024, doi: 10.14778/3641204.3641221.
[3] Q. Zhang, J. Dong, H. Chen, W. Li, F. Huang, and X. Huang, ‘Structure guided large language model for sql generation’, arXiv preprint arXiv:2402.13284, 2024.
[4] J. Li et al., ‘Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls’, Adv Neural Inf Process Syst, vol. 36, 2024.
[5] C. Y. Tai, Z. Chen, T. Zhang, X. Deng, and H. Sun, ‘Exploring Chain of Thought Style Prompting for Text-to-SQL’, in EMNLP 2023 - 2023 Conference on Empirical Methods in Natural Language Processing, Proceedings, 2023. doi: 10.18653/v1/2023.emnlp-main.327.
[6] H. Zhang, R. Cao, L. Chen, H. Xu, and K. Yu, ‘ACT-SQL: In-Context Learning for Text-to-SQL with Automatically-Generated Chain-of-Thought’, in Findings of the Association for Computational Linguistics: EMNLP 2023, 2023. doi: 10.18653/v1/2023.findings-emnlp.227.
[7] R. Sun et al., ‘SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL (extended)’, arXiv preprint arXiv:2306.00739, 2023.
[8] H. Xia et al., ‘SQL-CRAFT: Text-to-SQL through Interactive Refinement and Enhanced Reasoning’, arXiv preprint arXiv:2402.14851., 2024.