Proceedings of CONF-CDS 2025 Symposium: Data Visualization Methods for Evaluatio
DOI: 10.54254/2755-2721/2025.PO23472

Enhancing NL2SQL Conversion: Addressing Schema and Temporal Challenges with a Hierarchical Tree Structure

Yichen Ren

Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China
yrenat@connect.ust.hk

Abstract: This paper dives deeper into the field of Natural Language to Structured Query Language conversion (NL2SQL). Using the widely accepted NL2SQL agent provided with the Spider-2 dataset, it aims to identify basic and common issues present in most NL2SQL agents. Specifically, it evaluates the performance of the original Spider-2 agent and the Spider-2 + DIN-SQL model on the Spider-2 Snow dataset. Out of the 547 results, the thesis manually examines a subset with a statistically significant sample size. The results reveal that current models struggle to understand semi-structured variable names, such as column names in schemas and table names. The performance is abysmal in the absence of relevant illustrative files. Even when such files are available, the agent often fails to correctly interpret the meaning of file names, leading to the selection of incorrect files or tables that hold the data. This study also proposes potential directions for improvement, particularly in cases where file or table names involve temporal elements, such as dates or times. Based on experiments, the thesis believes incorporating a hierarchical tree structure could offer a promising solution.

Keywords: NL2SQL, Spider-2, Natural Language, Hierarchical Tree Structure.

1. Introduction

The conversion from Natural Language to Structured Query Language (NL2SQL) has long been a prominent and evolving topic [1]. It began with traditional rule-based NL2SQL methods, progressed to relatively newer approaches leveraging Long Short-Term Memory (LSTM) for improved schema alignment, and advanced further with models like Bidirectional Encoder Representations from Transformers (BERT), Generative Pre-trained Transformers (GPT), and other Large Language Model (LLM) [2-4].

Meanwhile, the complexity of NL2SQL conversion has significantly increased. What started as a straightforward task involving simple token-level recognition within a single table has evolved into a far more intricate process. Modern approaches require deep semantic understanding, handling multi-table interpretations and interconnections to perform information synthesis and global operations. Furthermore, the inclusion of multi-turn dialogues has added another layer of complexity, making it even more challenging to capture and maintain the core ideas throughout the conversation.

Evolving alongside the advancements in language models and NL2SQL algorithms is the development of benchmark datasets. From the first generation of datasets like WikiSQL and Spider, to the more recent Spider 2 dataset, the benchmarks have grown significantly in complexity [5-7].

DOI: 10.54254/2755-2721/2025.PO23472

The Spider 2 dataset introduces much more intricate inter-table queries, enabling the testing of models on a wider range of data with increasingly complex SQL syntax, including nested joins and intermediate calculations.

This study primarily focuses on the recently published Spider 2 dataset, which emphasizes the integration of multiple tables as a key challenge. For example, some NL queries in Spider involve requests like "the data for the entire month of July," while the corresponding data is stored across multiple tables, with each table representing a single day. In such cases, a LLM must generate an SQL query that combines data from all 31 tables and then performs the required operation on the aggregated data. The Spider 2 dataset also includes an integrated agent that implements some of the most commonly used techniques in NL2SQL tasks. It features a well-designed workflow, which operates as follows: first, the LLM is instructed to list all the files within a directory. Then, the LLM is tasked with identifying files that document the layout and distribution of data within the directory—commonly the "DDL.csv" file in the Spider 2 dataset. After synthesizing the extracted information, the LLM converts NL queries into SQL queries. The prompts used in this process are carefully crafted, employing advanced techniques such as Few-Shot Learning (FSL) and Chain of Thought (CoT) [8, 9].

To generalize further, this study also explores DIN-SQL, one of the most well-known NL2SQL agents [10]. DIN-SQL involves a classification step where NL queries are categorized into three possible types. Based on the classification, it generates SQL queries using specially designed prompts tailored to each query type. Through all those analyses. Thesis identified several common challenges across all NL2SQL agents and, as a result, have formulated and proposed potential improvements to address these issues.

2. Methodology

2.1. Spider dataset

As described above, this paper selects the Spider 2 Snow dataset for bench- marking. There are three datasets in total: Spider 2.0, Spider 2.0-snow, and Spider 2.0-lite, all designed to evaluate the capabilities of language models in handling diverse and complex text-to-SQL tasks in real-world enterprise environments. The Spider 2.0 benchmark includes 632 intricate text-to-SQL tasks derived from industrial database use cases, featuring large-scale schemas with thousands of columns, complex nested structures, and queries often exceeding 100 lines. Tasks require navigating project codebases, consulting SQL dialect documentation, and leveraging external knowledge for data transformations and analytics. Two subsets of Spider 2.0 provide additional focus and flexibility. Spider 2.0-snow is tailored specifically for the Snowflake environment, simplifying tasks by omitting project codebases and restricting outputs to SQL only, while still presenting challenges with diverse SQL dialects and complex queries. Similarly, Spider 2.0-lite is a self-contained subset hosted across databases like Big-Query, Snowflake, and SQLite. It eliminates project codebase interactions and focuses on generating SQL queries from preprocessed schemas and documentation, offering a faster and more streamlined text-to-SQL evaluation. Together, these datasets provide a robust framework for benchmarking and advancing the capabilities of text-to-SQL parsers in enterprise contexts.

2.2. Spider agent

The agent being tested and evaluated are the original Spider 2 agent, which is a framework designed to handle complex, real-world enterprise text-to-SQL tasks by iteratively interacting with SQL workflow environments. It operates by exploring database metadata, schemas, and sample data, writing and refining SQL queries, and performing file operations in Data Build Tool (DBT) projects—structured workflows that allow data teams to transform raw data in a data warehouse into

DOI: 10.54254/2755-2721/2025.PO23472

analytics-ready datasets using SQL models, configuration files, and testing frameworks. The agent debugs and validates results by checking output data against specifications, iterating as needed to correct errors. It supports multi- turn interactions via command-line tools and terminates if tasks are repeated without progress or exceed time limits.

2.3. Evaluation approaches

For convenience and flexibility, this study uses only the Spider 2 snow dataset for evaluation (shown in Figure 1). After collecting the original dataset, the study employs the Spider 2 agent to transform from NL to SQL. Once the results are obtained, the thesis manually verifies them and compares them with the gold answers provided in the Spider 2 dataset. Special attention is given to cases where the generated answers do not precisely match the ground truth, and the reasons for these failures will be discussed further in this study.

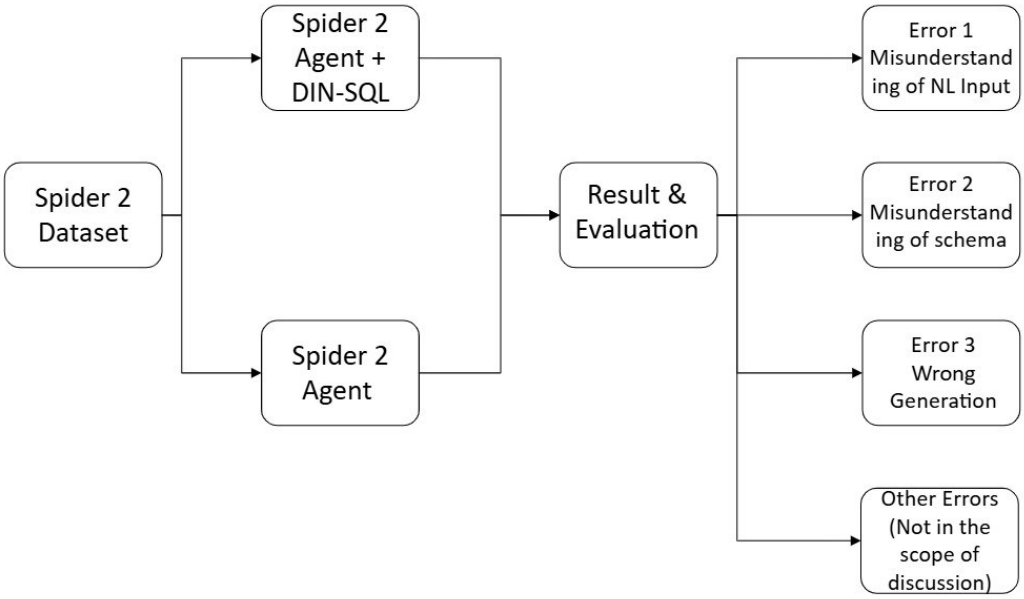

Figure 1: Workflow of the study (picture credit: original)

3. Results and discussions

The errors can generally be categorized into three types. The first type occurs when the agent fails to understand the intent of the NL input. The second type of error arises when the agent correctly interprets the intent of the NL query but fails to comprehend the schema or the inter-relationships between tables. The third type of error occurs when the agent fully understands both the NL query and the dataset schema but generates SQL queries with syntax errors. A detailed illustration and analysis of these three types of errors are provided below, all using the same NL input “Find the number of customers in total throughout the whole July”.

3.1. Error 1: misunderstanding of NL input

In this case, the agent fails to understand the intention of retrieving data for all days in July—specifically, from July 1st to July 31st. Instead, in most instances of this error, thesis observe that both Spider 2 agents, with or without DIN-SQL, typically focus only on the first day, July 1st, and

DOI: 10.54254/2755-2721/2025.PO23472

query solely for that date. This highlights the difficulty current agents face in accurately interpreting the semantic meaning of NL queries or identifying critical keywords.

3.2. Error 2: misunderstanding of files and schemas

In the Spider 2 dataset, data for different dates is stored in separate JSON files, while a DDL.csv file provides comprehensive information about the database structures and corresponding schemas for each file. In this case, the agent already understand that it needs to retrieve data from July 1st to July 31st. However, it fails to recognize that files such as 20220701.json and 20220702.json represent data for July 1, 2022, and July 2, 2022, respectively. As a result, instead of querying all files in the format 202207xx.json, the agent queries only a single file. Within that file, it generates SQL queries to select dates from July 1st to July 31st, which is incorrect.

3.3. Error 3: syntax error in generation

This mistake often arises from variations in SQL dialects, where minor differences between SQL variants are overlooked, leading to queries that cannot be executed. One example of such an error in the Spider 2 dataset is that some data keys are stored using double quotes (e.g., “time”), while the generated SQL queries often use single quotes (e.g., ‘time’) or no quotes at all (e.g., time). Snowflake, one of the data storage systems, does not tolerate such discrepancies and will frequently return errors. In contrast, other SQL dialects may be more forgiving of these minor mismatches.

3.4. Other errors

Other types of errors are also identified throughout the experiments. However, as they are not directly related to the focus of this study, this study will only present one typical example below without further discussion. Specifically, current agents still struggle to justify the generated answers. DIN-SQL does not incorporate this workflow, so this study focuses solely on the Spider 2 agent. Spider 2 includes an automated process that, after generating an output result, evaluates and verifies whether the results align with the intent of the NL query. If the results are incorrect, the agent is prompted to generate a new SQL query and query the database again. This process repeats until the agent determines that the results are correct. However, in most cases, the agents fail to recognize these errors properly. The process often works as follows: if any output is generated—regardless of its accuracy—it is deemed acceptable by the agent, even if it does not truly satisfy the query’s intent.

4. Possible solutions

Based on the previous observations and inspired by the tree-like Frequent Pattern (FP)-growth algorithm, this study proposes a hierarchical tree-structured approach to address these issues [11]. The thesis focuses on one specific topic: time domain recognition. There are two main reasons for this focus. First, time-domain naming methods are among the most frequently encountered when naming databases, tables, or schemas. Second, time-domain names often consist of various numerical formats, which, on their own, do not convey much semantic information. In contrast, verbal names composed of one or more words inherently carry meaning that can be inferred from the words themselves and their sequence. Even state-of-the-art methods like those used in IRNet do not perform well in this context. The thesis can only uncover the true meanings of these numerical representations by aggregating all the names and analyzing them from a global perspective. This unique characteristic of time-domain names makes it essential to develop a system that can effectively interpret both the temporal information provided in schemas or files and the time-related references embedded in NL

DOI: 10.54254/2755-2721/2025.PO23472

queries. To address this problem, the thesis's hierarchical tree-structured approach operates as follows in Figure 2.

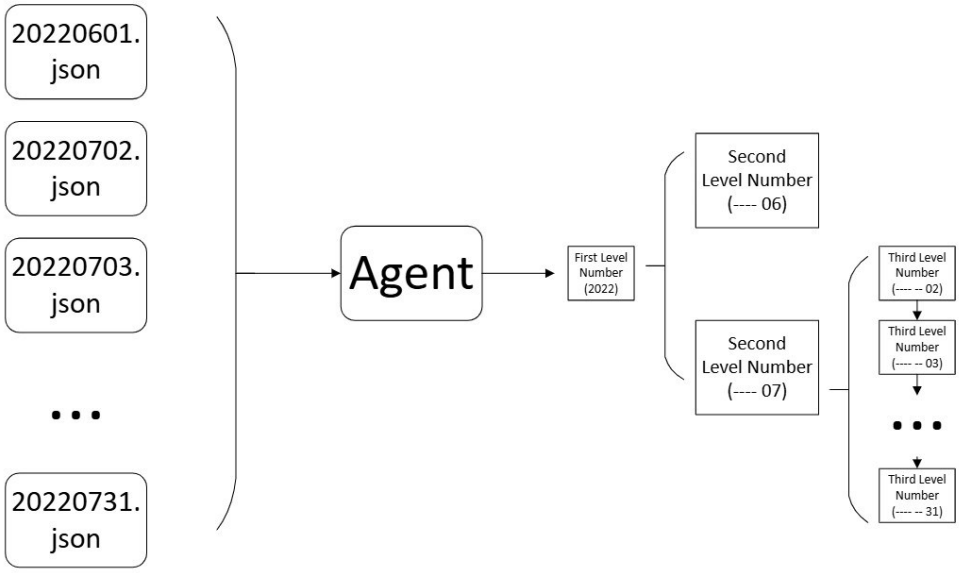

Figure 2: Tree structure agent (picture credit: original)

Here thesis will first discuss the design on the dataset side. The entire content of a directory is initially read into the agent, which then determines the type of filenames present. It is assumed that most files within a directory follow a consistent naming convention, as this is common practice. The agent is programmed to detect three types of file naming systems, includes Whole-Domain: In this system, filenames include all standard time- domain components, such as Year, Month, Day, Hour, Minute, and Second; Large-Domain: This system includes only Year, Month, and Day in the filenames; Small-Domain: This system includes only Hour, Minute, and Second in the filenames.

While the Whole-Domain naming system offers the same functionality as the Large-Domain and Small-Domain systems, the latter two are designed to use less memory, making them more efficient to implement. The agent determines the type of time-domain data provided by checking for the presence of specific components, such as Year or Hour. In edge cases where the data includes elements from multiple systems (e.g., filenames containing Year, Month, Day, and Hour), it is defaulted to the Whole-Domain system. For entries where certain components are missing, the agent assigns a null value. For each type of data, a prompt is also designed using few-shot learning to instruct the LLM to perform hierarchical clustering and construct a tree. Since thesis's focus is on the time domain, all values must also be valid in numerical form. For entries represented as characters, they are converted into numerical values. The leaf nodes of the tree represent the least significant time-domain components. Additionally, the agent construct interconnections between nodes, where each link carries a value representing the difference between the connected nodes. These trees are stored as arrays, with each level of the tree containing pointers that reference the corresponding locations in the directory.

DOI: 10.54254/2755-2721/2025.PO23472

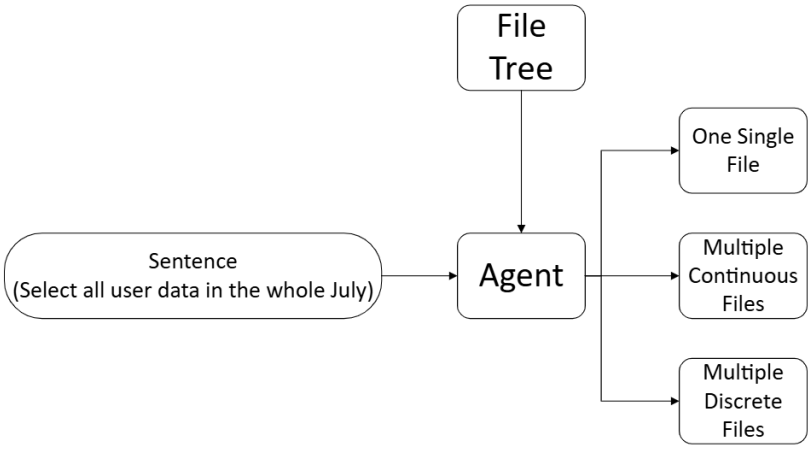

Figure 3: Tree structure NL (picture credit: original)

The Figure 3 above illustrates the second part of the proposed design. In this stage, the agent first extracts the numerical values from the NL input query. It then classifies the query into one of three categories: a single file, multiple continuous files, or multiple discrete files, based on the structure of the file tree. For each input type, a corresponding prompt is designed to handle it effectively.

5. Conclusion

This study explored the challenges and limitations of existing NL2SQL agents in converting natural language queries into SQL commands, explicitly focusing on the Spider 2 Snow dataset. Through manual evaluation of the Spider 2 agent and the Spider 2 + DIN-SQL model, there are three primary error types: misinterpretation of NL intent, miscomprehension of schema or file structures, and syntax errors due to SQL dialect variations. The thesis proposed a hierarchical tree-structured approach to address these issues, particularly for improving time-domain recognition in file and schema naming. By leveraging the tree structure, the agent can more effectively interpret temporal references in NL queries and align them with the underlying dataset organization. The thesis is still actively refining this approach to enhance its effectiveness and adaptability for complex real-world NL2SQL tasks.

References

[1] *Liu, X., Shen, S., Li, B., Ma, P., Jiang, R., Luo, Y., Zhang, Y., Fan, J., Li, G., & Tang, N. (2024). A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? ArXiv print, 5109.*

[2] *Katsogiannis-Meimarakis, G., & Koutrika, G. (2021). A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems, 2846-2851.*

[3] *Jha, D., Ward, L., Yang, Z., et al. (2019). IRNet: A general purpose deep residual regression framework for materials discovery. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2385-2393.*

[4] *Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2023). Text-to-SQL Empowered by Large Language Models: A Bench- mark Evaluation. ArXiv print, 15363.*

[5] *Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. ArXiv print, 103.*

[6] *Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2019). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. ArXiv print, 8887.*

[7] *Lei, F., Chen, J., Ye, Y., Cao, R., Shin, D., Su, H., Suo, Z., Gao, H., Hu, W., Yin, P., Zhong, V., Xiong, C., Sun, R., Liu, Q., Wang, S., & Yu, T. (2024). Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. ArXiv print, 7763.*

DOI: 10.54254/2755-2721/2025.PO23472

[8] *Archit, P., & Lee, M. (2022). Learning from Few Examples: A Sum-mary of Approaches to Few-Shot Learning. ArXiv print, 4291.*

[9] *Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain of Thought Prompting Elicits Reasoning in Large Language Models. ArXiv print, 11903.*

[10] *Pourreza, M., & Rafiei, D. (2023). DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. Advances in Neural Information Processing Systems, 36, 36339-36348*

[11] *Shawkat, M., Badawi, M., Elghamrawy, S. M., Reham Arnous, & El-desoky, A. I. (2021). An optimized FP-growth algorithm for discovery of association rules. The Journal of Supercomputing, 78(4), 5479–5506.*