논문 25-50-08-03 The Journal of Korean Institute of Communications and Information Sciences '25-08 Vol.50 No.08
https://doi.org/10.7840/kics.2025.50.8.1183

Approaches to Lightweight Text-to-SQL Implementation Based on sLLM

Jae-young Im[◆], Soo-Yeon Yoon[°]

ABSTRACT

This study proposes a lightweight Text-to-SQL implementation based on sLLM (smaller Large Language Model) to solve the high cost and security issues of existing LLM (Large Language Model) based Text-to-SQL models. To this end, we implemented a Text-to-SQL model using DAIL-SQL[1] and Llama3-8B and evaluated its performance using Spider dataset[2]. In this study, we secure the shortcomings of the existing Few-shot Learning method and propose improvement measures such as fine-tuning, knowledge distillation, and selection of similar queries through RAG to improve the performance of sLLM-based Text-to-SQL. By resolving the security vulnerabilities of existing LLM-based Text-to-SQL and presenting an efficient way to implement sLLM-based Text-to-SQL, we expect to expand the utilization of Text-to-SQL in various industries.

Key Words : sLLM, Text-to-SQL

Ⅰ. Introduction

Text-to-SQL is a technology that converts sentences written in natural language into SQL queries that databases can understand. It bridges the gap between users and complex databases, showing high potential in various fields such as business intelligence (BI), data analytics, and chatbot systems. In particular, it simplifies the user-database interaction by retrieving real-time data from the database, reducing the time to make business decisions and improving customer service[3].

Large Language Models (LLMs) have been a hot topic of research in the text-to-SQL space in recent years. By leveraging the language processing power of LLMs, it has become possible to generate sophisticated SQL by better understanding natural language and mapping it to schema information. However, LLM-based text-to-SQL has some fundamental limitations. The main limitation is the inability to use online LLMs for security reasons. Text-to-SQL relies on the internal database schema information of a company or organization, which is sensitive information that should not be leaked to the outside world. In addition, LLM is much more expensive per token than sLLM, which is the main reason for the high cost of LLM-based Text-to-SQL using the Few-Shot Learning method. Finally, domain-specific output is difficult to verify in traditional LLM-based Text-to-SQL. Additional training is required to produce domain-specific results, and LLM requires more expensive and time-consuming training than sLLM.

Previous LLM-based text-to-SQL research has focused on improving the performance of text-to-SQL itself. However, there is a lack of research on security issues, high cost, and difficulty in domain-specific services. In this paper, we propose a lightweight

※ This research was supported by GPU resources provided through the 2025 High-Performance Computing Support Program, funded by the Ministry of Science and ICT and the National IT Industry Promotion Agency (NIPA).(G2025-0572)

◆ First Author : Kookmin University, dlawowo01@kookmin.ac.kr, 정회원
° Corresponding Author : Kookmin University, 1104py@kookmin.ac.kr, 정회원

논문번호 : 202410-261-A-RN, Received October 29, 2024; Revised February 1, 2025; Accepted March 11, 2025

Text-to-SQL model implementation based on sLLM (smaller Large Language Model). To solve the existing problems of the existing LLM-based Text-to-SQL, we designed a sLLM-based Text-to-SQL model. We prove that sLLM is a more suitable language model for Text-to-SQL than LLM, and we also propose specific measures to improve the performance of sLLM-based Text-to-SQL. Through this research, we expect to be able to design a Text-to-SQL model that is suitable for companies and institutions that are sensitive to information leakage and have specialized domains such as national defense and healthcare.

Ⅱ. Related Work

2.1 sLLM

2.1.1 Definition of sLLM

Traditional large language models (LLMs) are highly performant as they contain hundreds of billions of parameters, but they also require vast computing resources and are expensive to learn and infer. The small Large Language Model (sLLM) has emerged to overcome the limitations of LLMs. An sLLM is a small language model with about 1 billion parameters, which maintains the performance of LLMs while increasing efficiency in terms of computing resources and costs.

2.1.2 Background of sLLM

The sLLM emerged to address the shortcomings of traditional LLMs. LLMs provide high performance, but due to their size, they require a lot of resources and time during the learning and inference process. To solve this problem, sLLM is a way to reduce the size of the model and still operate efficiently. It is increasingly recognized that not only the size of the model, but also the quality of the data and specialized training are important factors in improving performance. Instead of simply throwing in large amounts of data, we show that high performance can be achieved with small models using properly processed data and optimized learning strategies.

2.1.3 Recent Technological Trends in sLLM

Recently, sLLM-related techniques have evolved into a variety of methodologies to achieve model reduction while maintaining performance. First, knowledge distillation techniques have been actively studied to efficiently compress knowledge from large models into small models[4-8]. Second, model lightweighting techniques are also widely used to reduce computing resources by removing unnecessary parameters or optimizing the model structure. In addition, methods such as fine-tuning have been proposed to efficiently train small models tailored to specific domains[9-11].

2.2 Text-to-SQL

2.2.1 Definition of sLLM

In real time, important information is stored in databases. Text-to-SQL is used as a technology to effectively retrieve this important information in real time. Text-to-SQL is a technology that learns the schema structure of a database and converts queries written in natural language into SQL queries that the database can understand. This overcomes the limitation that LLM (Large Language Model) provides results based only on learned information, and allows you to directly retrieve the latest information stored in the database.

2.2.2 Development Process of Text-to-SQL

The early days of Text-to-SQL research were dominated by rule-based approaches. The first rule-based Text-to-SQL can be considered the CHILL system [12] by Zelle and Mooney. It used Inductive Logic Programming (ILP) to implement rule-based Text-to-SQL, and later evolved to generate domain-specific rules [13] to translate natural language questions into SQL. These early efforts focused on generating SQL queries using explicit rules between the database schema and natural language questions.

In the early 2010s, to overcome the limitations of rule-based approaches, statistical approaches began to replace them. In the early 2010s, a statistical approach was proposed[14] to automatically learn the mapping between database schema and natural language using data-driven models, which can be described as an important step forward as a generalized methodology that can be applied to a wide variety of database structures without being specific to a particular domain.

In the late 2010s, advances in deep learning-based natural language processing led to the use of deep learning-based models in text-to-SQL[15-18], with Seq2Seq models gaining significant attention in text-to-SQL research. A representative model that utilizes Seq2Seq models is the Seq2SQL model[19]. This model proposed a method for generating SQL queries from natural language using reinforcement learning, and showed excellent performance on the WikiSQL dataset.

More recently, large language models (LLMs) and transformer-based approaches have significantly improved the performance of Text-to-SQL[20-23]. Models such as RAT-SQL[24] improve Text-to-SQL performance by explicitly modeling schema-query relationships using transformers[25].

2.2.3 LLM-based Text-to-SQL

Recently, the text-to-SQL space has also seen a flurry of research based on Large Language Models (LLMs). According to the Spider[2] leaderboard, four out of the top five (as of April 12, 2024) are Text-to-SQL models based on LLMs. LLM is widely used in Text-to-SQL because it performs well in accurately understanding and processing users' natural language queries.

The basic sequence of Text-to-SQL is systematically performed as shown in Fig. 1 First, there is a step to relate the natural language (NL Query) queried by the user to the schema structure of the database. This step is called Schema Linking, which matches the natural language query with the schema information of the database to identify the database elements that match the intent of the query.

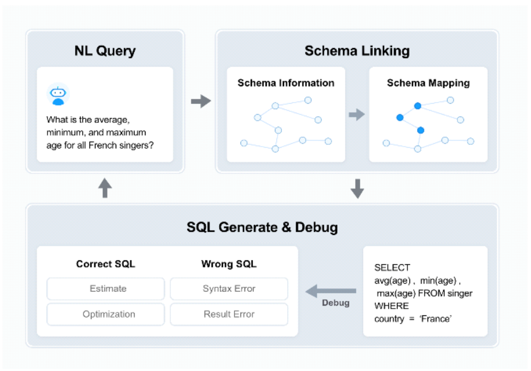
The next step is to generate SQL based on this schema structure. The SQL generation process reflects the meaning of the natural language query and generates a query that conforms to SQL syntax. The complex natural language representation must be properly mapped to the tables, columns, and conditions in the database.

The generated SQL queries are not executed immediately, but are evaluated. The main evaluation factors are whether the query is grammatically correct and produces logically correct results. If grammatical errors or execution errors are found during the evaluation process, the SQL is debugged and fixed [26].

2.2.4 Issues with LLM-based Text-to-SQL

LLM-based Text-to-SQL has two main problems. The first is that the LLM is expensive to implement and maintain. Figure. 2 compares the price of GPT-4o and Llama3-8B APIs, you can see that GPT-4o costs 25 times more. From this cost perspective, sLLM is much more economical than LLM.

Second, if schema information is compromised, the database can be exposed to a number of serious security threats[27]. First, SQL Injection attacks can become more sophisticated. If the schema structure is compromised, an attacker can determine the internal structure of the database, including tables, columns,


What is the average, minimum, and maximum age for all French singers?

```
SELECT
avg(age) , min(age) ,
 max(age) FROM singer
WHERE
country = 'France'
```

Fig. 1. Text-to-SQL flowchart

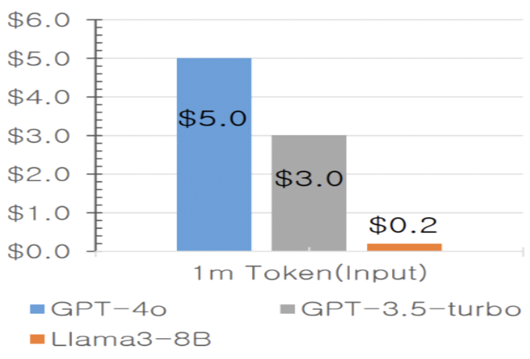


Fig. 2. GPT-4o, GPT-3.5-turbo, Llama3-8B API(input) cost

and so on, which can be used to target specific data. These attacks can enable illegal data manipulation, such as viewing, modifying, or deleting data.

It also increases the risk of data theft. Attackers can utilize schema information to identify the tables where sensitive data is stored, and then target and attempt to steal sensitive information. In particular, such an attack can severely impact the credibility of an organization/enterprise, with the potential for legal and financial damage.

Finally, you may be vulnerable to a database denial of service (DoS) attack. If schema information is leaked, an attacker can design complex, resource-intensive queries based on the structure of the database to deplete the database's resources. This can prevent the database server from functioning properly and severely degrade the availability of your application.

Schema information leakage is more than just a security threat; it is a serious problem that threatens the integrity, confidentiality, and availability of the database. By changing the online-based LLM to an offline-based sLLM that prevents schema leakage and makes it inaccessible from the outside, you can effectively solve the security problem.

Ⅲ. Experimental Setup

3.1 Dataset

For our experiments, we used the Spider dataset, a leading benchmark dataset in the Text-to-SQL field. Spider[2] was annotated by 11 Yale University students to address the problem that existing datasets like WikiSQL contain only simple SQL queries and single tables. The dataset contains complex SQL queries with 10,181 questions and covers a wide range of SQL syntax, including multiple tables, Having, Group By, Limit, and Join. The SQL difficulty level is categorized into Easy, Medium, Hard, and Extra Hard. We collected 200 databases covering 138 domains from various sources, including university databases, DatabaseAnswers, and WikiSQL, from which Yale students wrote 20-50 questions and SQL labels for each database.

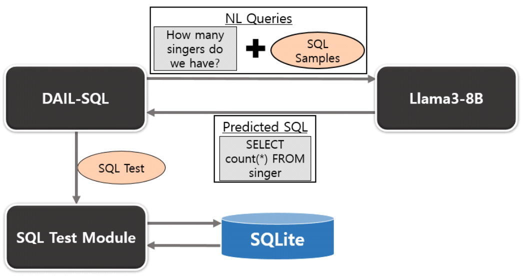


Fig. 3. Lightweight Text-to-SQL Model Architecture with DAIL-SQL and Llama3-8B

3.2 Experimental Model

In this study, we organized the experimental model as shown in Fig. 3, the experimental model was configured as shown in Fig. Predicated SQL was created through DAIL-SQL and Llama3-8B, and actual SQL was executed through SQL Test Module to evaluate the performance.

3.2.1 DAIL-SQL

As a text-to-SQL model, we use DAIL-SQL[1]. DAIL-SQL is a Text-to-SQL model proposed by Dawei Gao, which is a Text-to-SQL model based on Few-shot learning implemented with the DAIL Selection method, which combines Masked Question Similarity Selection (MQS) and Query Similarity Selection (QRS). We chose DAIL-SQL as a text-to-SQL-based model in this study because it has a running accuracy of 86.6\% on the Spider dataset and has shown high adaptability and performance for various domains and complex SQL queries.

Masked Question Similarity Selection (MQS) replaces table names, column names, values, etc. in questions with mask tokens to minimize the negative impact of cross-domain information. Their embedding similarity is calculated using the k-Nearest Neighbor (kNN) algorithm to increase applicability across different domains. This allows us to effectively select the right SQL samples for few-shot learning.

Query Similarity Selection (QRS) utilizes a prior model to select examples similar to the target SQL to generate initial SQL based on the target question and the database. It uses the generated initial SQL as an approximation and encodes it into a binary discrete phrase vector based on keywords in the SQL.

The best examples are selected by considering their similarity to the approximated SQL and the diversity among the selected examples. In this way, DAIL-SQL can effectively perform text-to-SQL conversion in different domains and select the best SQL queries.

3.2.2 Llama3-8B

In this study, we used the Llama3-8B model released by Meta AI in April 2024 as sLLM. Llama3-8B is a model that realizes high performance with fewer parameters by introducing Grouped Query Attention (GQA) technology, which was previously applied only to large-scale models in Llama2, to small and medium-sized models. In particular, Llama3-8B outperforms Llama2 70B, and also shows superior results in terms of computational efficiency and resource utilization. The study was conducted in May 2024, shortly after the release of Llama3-8B, and these characteristics are consistent with the purpose of this study, which requires a high-performance model in a limited resource environment.

3.2.3 SQL Test Module

To measure the accuracy of Text-to-SQL, we used the method proposed in their study [28]. This method is a test suite-based evaluation method for evaluating the semantic accuracy of Text-to-SQL models, and since traditional string matching or single database comparison methods can lead to errors, we created a small test suite that runs queries across multiple databases to evaluate accuracy. This method is more reliable for semantic evaluation of complex queries and can effectively reduce false negative and false positive errors than traditional metrics.

Ⅳ. Experiment

4.1 Experimental Procedure

4.1.1 Selection of Examples for Few-shot Learning

Masked Question Similarity Selection (MQS) and Query Similarity Selection (QRS) techniques are used to select examples that are appropriate for the questions required for the experiment.In this course, you will learn how to select appropriate examples for Few-shot Learning to improve learning performance.

4.1.2 Querying the Llama3-8B Model

Generate predictive SQL queries by querying the Llama3-8B model with real-world questions along with examples selected for Few-shot Learning.

4.1.3 Evaluation of SQL Queries

Evaluate the generated predictive SQL query to ensure that it works correctly.The SQL queries are executed on the real database and the results match the correct answer (Gold SQL). For example, if you ask Llama3-8B the question “How many singers do we have?” in the Spider dataset, Llama3-8B will return SQL like “SELECT count(*) FROM singer”. This is called Prediction SQL, and we will compare Prediction SQL to Gold SQL.

4.2 Evaluation Metrics

The Spider dataset uses two evaluation metrics: Exact Match Accuracy (EM) and Execution Accuracy (EX). Each metric plays an important role in measuring the performance of the model. First, Exact Match Accuracy (EM) evaluates how well the predicted SQL query (Prediction SQL) matches the gold SQL (Gold SQL) character by character. This metric focuses on determining if the predicted SQL is structurally and syntactically identical to the correct answer. EM will only have a high value if all elements of the SQL query match exactly - keywords, tables, columns, conditional expressions, etc. This is useful for evaluating how accurately the model’s predicted query embodies the correct query, and is especially important for verifying that it has accurately learned the complex syntactic structure of SQL. EM is the percentage of correctly matched queries out of all predictions, expressed as a percentage.

$$\mathrm{EM} = \frac{Number\ of\ Exact\ Matches}{Total\ Number\ of\ Predictions} * 100$$

Execution Accuracy (EX), on the other hand, is a metric that measures how well the results of a predicted SQL query match the results of the correct

SQL when it is executed on the actual database. This metric focuses on evaluating the performance of the model based on the execution results of the SQL query. Because slightly different SQL syntax can often lead to the same result, EX is useful for determining whether the model actually performed the correct database operations. For example, in a SELECT query, there may be multiple ways to return the same query result, and EX evaluates whether these query results match the correct answer. EX is the percentage of SQL executions that matched the correct result, expressed as a percentage.

$$\mathrm{EX} = \frac{Number\ of\ Correct\ Executions}{Total\ Number\ of\ Executions} * 100$$

These two metrics complement each other. EM evaluates how accurately the model generated the same SQL syntax as the correct query, while EX evaluates whether the syntax actually produces the correct result. This allows you to measure both the syntactic correctness and the practical validity of your model. Because EM and EX have their own strengths and weaknesses, using both metrics together provides a more comprehensive assessment of a model's performance.

4.3 Experimental Results

According to Table.1, which compares the performance with the existing LLM-based Text-to-SQL model, the DAIL-SQL +Llama3-8B model has a performance of EX 69.2, which is 82.9\% of the performance of the comparison model, DAIL-SQL + GPT-4 model. This result proves that the sLLM-based lightweight text-to-SQL design approach can solve the problems of the existing LLM-based text-to-SQL model and maintain high performance. In particular, this result is meaningful because it shows competitive performance despite adopting a lightweight structure with sLLM.

Table 1. DAIL-SQL + Llama3-8B experimental results and the accuracy of existing Text-to-SQL models

Model	EM	EX
DAIL-SQL + Llama3-8B	46.8	69.2
DAIL-SQL + GPT-4[9 shot]	72.8	83.4
DIN-SQL + GPT-4 [29]	60	85.3
RESDSQL-3B + NatSQL[30]	72.0	79.9

4.4 Results Analysis

In Table 1, the DAIL-SQL + Llama3-8B model performs 46.8 in Exact Match Accuracy (EM) and 69.2 in Execution Accuracy (EX). EM evaluates how well a Predicated SQL query matches Gold SQL character by character, and is a measure of the structural correctness of SQL. The result of 46.8 on EM may seem low, but this is because the nature of SQL is such that there are many different syntaxes that can produce the same result, and EM only evaluates structural matching. On the other hand, EX scored 69.2, which is about 82.9\% better than the 83.4 of EX for the comparison model, DAIL-SQL + GPT-4. This shows that the lightweight sLLM model can provide competitive performance while maintaining high cost-effectiveness and security.

Also, as shown in Table 2, the accuracy of DAIL-SQL + Llama3-8B model by SQL difficulty tends to decrease gradually as the difficulty increases. At the EASY level, both EX and EM recorded a high accuracy of 85.1\%, but as the difficulty level increased to MEDIUM, HARD, and EXTRA, the performance dropped sharply to 68.2 for EX and 46.8 for EM. These results indicate that the model is struggling to handle difficult SQL queries. This is likely due to the fact that the general performance of Llama3-8B is worse than that of LLM. We found that Llama3-8B is less accurate on complex SQL compared to LLM because it uses Few-shot Learning, which is highly influenced by the general performance of the model.

From the failure cases presented in Table 3, we can see that the proposed model tends to fail in cases of high SQL complexity, especially for JOIN operations involving multiple tables. For example, the

Table 2. Results of DAIL-SQL + Llma3-8B by SQL Difficulty Level

	EASY	MEDIUM	HARD	EXTRA	ALL
EX	85.1	74.4	61.5	39.8	68.2
EM	85.1	46	33.3	14.5	46.8

Table 3. Failure Case of DAIL-SQL + Llma3-8B

	Prediction	Gold
EASY	SELECT AVG(LifeExpectancy) FROM country WHERE Continent = 'Africa' AND Region = 'Central'	SELECT avg(LifeExpectancy) FROM country WHERE Region = "Central Africa"
MEDIUM	SELECT COUNT(DISTINCT continent) FROM countrylanguage WHERE language = 'Chinese'	SELECT COUNT(DISTINCT Continent) FROM country AS T1 JOIN countrylanguage AS T2 ON T1.Code = T2.CountryCode WHERE T2.Language = "Chinese"
HARD	SELECT p.birth_date FROM poker_player p ORDER BY p.earnings ASC LIMIT 1	SELECT T1.Birth_Date FROM people AS T1 JOIN poker_player AS T2 ON T1.People_ID = T2.People_ID ORDER BY T2.Earnings ASC LIMIT 1
EXTRA	SELECT state FROM VOTES GROUP BY state ORDER BY COUNT(*) DESC LIMIT 1	SELECT T1.area_code FROM area_code_state AS T1 JOIN votes AS T2 ON T1.state = T2.state GROUP BY T1.area_code ORDER BY count(*) DESC LIMIT 1

SQL queries generated by the model on HARD and EXTRA difficulty levels did not accurately capture the relationships between tables compared to the target SQL query, leading to incorrect results. This shows that the model's performance degrades significantly as the complexity of the SQL query increases.

V. Conclusion and Implications

In this study, we propose an implementation of sLLM-based text-to-SQL model to compensate for the limitations of LLM, rather than focusing on performance improvement of existing LLM-based text-to-SQL research. As a result of the study, the sLLM-based model showed stable results in terms of performance compared to the existing LLM model, and the possibility of significantly reducing the cost of deployment and maintenance was confirmed. In particular, sLLM is different from LLM in that it is easy to deploy offline and can provide domain-specific services. These features are expected to be a major factor in selecting sLLM-based text-to-SQL models over LLM-based text-to-SQL models in companies and institutions with specialized domains and network environments that are isolated from the outside world, such as defense, healthcare, and public institutions.

In this study, we found that sLLM-based Text-to-SQL models still have limitations in handling complex SQL queries. To overcome this, we propose additional performance improvement measures as follows. First, we introduce fine-tuning techniques such as LoRA(Low-Rank Adaptation)[31] and QLoRA(Quantized LoRA)[32] to enhance the performance of sLLM. We expect these techniques to be useful in improving both efficiency and accuracy through domain-specific learning while maintaining model size. In addition, we utilize Knowledge Distillation techniques to transfer LLM's high SQL generation performance to sLLM by using LLM as a teacher model, which improves the ability to handle complex SQL while maintaining lightweight. Finally, by employing a vector database to retrieve schema elements semantically aligned with natural language queries and integrating them using the Retrieval-Augmented Generation (RAG) [33] framework, the system can better capture complex schema structures and inter-table relationships. This approach enhances query accuracy by enabling context-aware identification and mapping of relevant tables and attributes, while also improving efficiency in large-scale relational databases through automated schema retrieval.

In future work, we will focus on training sLLM on additional datasets, such as BIRD[34], in addition to the Spider dataset, using the Fine-Tuning technique to precisely learn the relationships between database tables and optimize the generation of complex SQL queries. Fine-tuning is expected to further enhance the performance of sLLM-based Text-to-SQL models and greatly expand their applicability in various domains. Ultimately, the results of this research will enable domain-specific optimized SQL query generation, enabling efficient data processing in various industries.

References

[1] D. Gao, H. Wang, Y. Li, et al., "Textto-sql empowered by large language models: A benchmark evaluation," *arXiv preprint arXiv: 2308.15363*, 2023.

[2] T. Yu, R. Zhang, K. Yang, et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and textto-sql task," *arXiv preprint arXiv:1809. 08887*, 2018.

[3] H. Yoon, J. Heo, J. Kim, and P. Kang, "Text-to-SQL for Korean language based on multilingual BERT," *J. Korean Inst. Ind. Eng.*, vol. 48, no. 1, pp. 91-104, Mar. 2022.

[4] N. Lee, T. Wattanawong, S. Kim, et al., "Llm2llm: Boosting llms with novel iterative data enhancement," *arXiv preprint arXiv: 2403.15042*, 2024.

[5] Y. Gu, L. Dong, F. Wei, and M. Huang, "Minillm: Knowledge distillation of large language models," in *The Twelfth Int. Conf. Learn. Representations*, 2024.

[6] G. Hinton, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503. 02531*, 2015.

[7] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *Int. Conf. Machine Learn., PMLR*, pp. 1607-1616, 2018.

[8] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," *arXiv preprint arXiv:1908. 09355*, 2019.

[9] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also fewshot learners," *arXiv preprint arXiv:2009. 07118*, 2020.

[10] H. Li, Q. Ai, J. Chen, et al., "Blade: Enhancing black-box large language models with small domain-specific models," *arXiv preprint arXiv:2403.18365*, 2024.

[11] W. Shen, C. Li, H. Chen, et al., "Small llms are weak tool learners: A multi-llm agent," *arXiv preprint arXiv:2401.07324*, 2024.

[12] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," in *Proc. National Conf. Artificial Intell.*, pp. 1050-1055, 1996.

[13] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability," in *COLING 2004: Proc. 20th Int. Conf. Computational Linguistics*, pp. 141-147, 2004.

[14] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," in *Proc. VLDB Endowment*, vol. 8, no. 1, pp. 73-84, 2014.

[15] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on wikisql with table-aware word contextualization(2019)," *arXiv preprint arXiv:1902.01069*, 2019.

[16] A. Neelakantan, Q. V. Le, M. Abadi, A. McCallum, and D. Amodei, "Learning a natural language interface with neural programmer," *arXiv preprint arXiv:1611. 08945*, 2016.

[17] J. Guo, Z. Zhan, Y. Gao, et al., "Towards complex text-to-sql in cross-domain database with intermediate representation," *arXiv preprint arXiv:1905.08205*, 2019.

[18] P. Wang, T. Shi, and C. K. Reddy, "Text-tosql generation for question answering on electronic medical records," in *Proc. The Web Conf. 2020*, pp. 350-361, 2020.

[19] V. Zhong, C. Xiong, and R. Socher, "Seq2sql generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[20] I. Trummer, "Demonstrating gpt-db: Generating query-specific and customizable code for sql processing with gpt-4," in *Proc. VLDB Endowment*, vol. 16, no. 12, pp. 4098-

4101, 2023.
[21] V. Câmara, R. Mendonca-Neto, A. Silva, and L. Cordovil-Jr, "Dbvinci-towards the usage of gpt engine for processing sql queries," in *Proc. 29th Brazilian Symp. Multimedia and the Web*, pp. 91-95, 2023.
[22] Y. Wu and H. Wang, "Schema-aware multi-task learning for complex text-to-sql," *arXiv preprint arXiv:2403.09706*, 2024.
[23] I. Trummer, "Codexdb: Generating code for processing sql queries using gpt-3 codex," *arXiv preprint arXiv:2204.08941*, 2022.
[24] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," *arXiv preprint arXiv:1911.04942*, 2019.
[25] X. Dong, C. Zhang, Y. Ge, et al., "C3: Zeroshot text-to-sql with chatgpt," *arXiv preprint arXiv:2307.07306*, 2023.
[26] B. Zhang, Y. Ye, G. Du, et al., "Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation," *arXiv preprint arXiv:2403.02951*, 2024.
[27] Đ. Klisura and A. Rios, "Unmasking database vulnerabilities: Zero-knowledge schema inference attacks in text-to-sql systems," *arXiv preprint arXiv:2406.14545*, 2024.
[28] R. Zhong, T. Yu, and D. Klein, "Semantic evaluation for text-to-sql with distilled test suites," *arXiv preprint arXiv:2010.02840*, 2020.
[29] M. Pourreza and D. Rafiei, "Din-sql: Decomposed in-context learning of text-to-sql with self-correction," *Advances in NIPS*, vol. 36, 2024.
[30] H. Li, J. Zhang, C. Li, and H. Chen, "Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql," in *Proc. AAAI Conf. Artificial Intell.*, vol. 37, pp. 13067-13075, 2023.
[31] E. J. Hu, Y. Shen, P. Wallis, et al., "Lora: Lowrank daptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.
[32] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in NIPS*, vol. 36, 2024.
[33] P. Lewis, E. Perez, A. Piktus, et al., "Retrievalaugmented generation for knowledge-intensive NLP tasks," *Advances in NIPS*, vol. 33, pp. 9459- 9474, 2020.
[34] J. Li, B. Hui, G. Qu, et al., "Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls," *Advances in NIPS*, vol. 36, 2024.

Jae-young Lim

2023~Current : Kookmin University, Graduate School of Software Technology, Major in Artificial Intelligence
2017 : The Academic Credit Bank System, Major in Information and Communication Engineering
<Research Interests> sLLM/LLM, Text-to-SQL, LMM, Data Architeture
[ORCID:0009-0009-0579-1669]

Soo-Yeon Yoon

2017 : Ph.D. degree in IT Policy Engineering, Soongsil University
2020~Current : Assistant Professor, School of Software, Kookmin University
2017~2020 : Adjunct Professor, Duksung Women's University
2009~2020 : Adjunct Professor, Soongsil University
<Research Interests> Large-scale AI Models, NLP, LLM/MLLM, Big Data Analysis
[ORCID:0000-0002-4148-5433]