RB-SQL: A Retrieval-based LLM Framework for Text-to-SQL

Anonymous ACL submission

Abstract

Large language models (LLMs) with in-context learning have significantly improved the performance of text-to-SQL task. Previous works generally focus on using exclusive SQL generation prompt to improve the LLMs' reasoning ability. However, they are mostly hard to handle large databases with numerous tables and columns, and usually ignore the significance of pre-processing database and extracting valuable information for more efficient prompt engineering. Based on above analysis, we propose RB-SQL, a novel retrieval-based LLM framework for in-context prompt engineering, which consists of three modules that retrieve concise tables and columns as scheme, and targeted examples for in-context learning. Experiment results demonstrate that our model achieves better performance than several competitive baselines on public datasets BIRD and Spider [1].

1 Introduction

Text-to-SQL is a task of converting natural language questions into SQL queries that are used to obtain the answers from the database. It has attracted widespread research attention and application in database querying.(Qin et al., 2022; Sun et al., 2023). Early methods utilize pre-trained models to encode the input sequence. Some researchers decode queries by abstract syntax trees (Wu et al., 2023; Guo et al., 2019; Wang et al., 2020), while others use predefined sketches (He et al., 2019). Recent works focus on extracting the question-to-SQL patterns generalized by training an encoder-decoder model with text-to-SQL corpus (Hui et al., 2022; Li et al., 2023a,b; Zheng et al., 2022; Gao et al., 2024). More recently, there has been growing interest in using Large Language Models (LLMs) to explore novel approaches for guiding SQL generation, and some remarkable progress has been significantly made in prompt and chain of thought. Fig 1 (a) shows an example of utilizing LLM to solve text-to-SQL task.

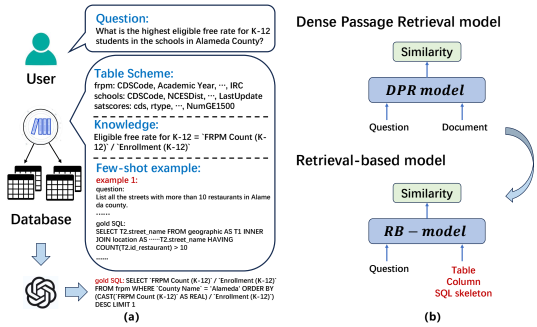

Figure 1: (a) An example of utilizing LLM to solve text-to-SQL task. (b) The diagrams of DPR model and our proposed RB-model. Compared with DPR model, RB-model expands the input from *document* to other data types *(i.e., table, column, SQL skeleton)*.

Different from prior studies, the fundamental solution in LLM-based text-to-SQL has primarily focused on using exclusive SQL generation prompt approaches to obtain a fully correct SQL query (Gao et al., 2024). Existing approaches tend to maintain the whole tables and its corresponding columns as the table schema in databases. Thus, it will possibly introduce a large amount of redundant information in the prompt that is irrelevant to the original question, especially for the complex multi-table queries (e.g., nested or joined queries) and extremely large single tables. The excessive redundancy can significantly introduce negative noise and exceed the LLMs context window length limitations. In addition, previous works tend to ignore the significance of both pre-processing database and valuable information extraction, thus limiting the interpretability and prompt engineering efficiency. Therefore, efficient information retrieval for tables and columns could significantly improve the performance of text-to-SQL. Moreover, the hallucination in text-to-SQL is also a notorious prob-

[1] https://anonymous.4open.science/r/Anonymize-A5E7

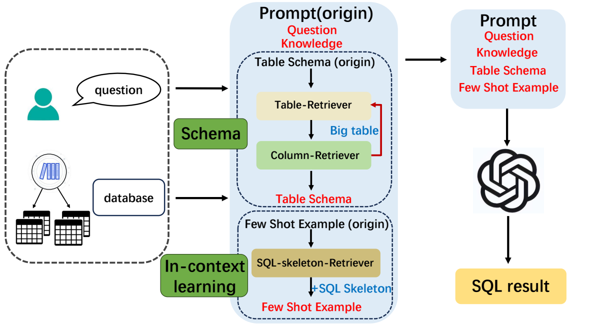

Figure 2: Framework of the RB-SQL. Table-Retriever filter tables from database and Column-Retriever further filter columns. SQL-skeleton-Retriever is used to choose similar few-shot examples and add SQL skeleton into example organization.

lem in LLMs. We observe that the approach of guiding through the skeleton related to SQL syntactic can alleviate hallucination. Previous studies focused on integrating the skeleton information of SQL into sequence-to-sequence models for modeling (Li et al., 2023a), without explicitly utilizing the syntactic advantages of the skeleton to guide correct SQL generation process.

To address the above issues, we consider using Dense Passage Retrieval (DPR) models to retrieve relevant tables, columns and examples from original databases for prompt engineering. Existing DPR models tend to calculate similarity directly between *question* and *document* without involving other data types, while recent research (Wang et al., 2022) points out that DPR models can also be used for retrieving answers from *table*. Therefore, we are motivated to design Retrieval-Based (RB) models on the basis of DPR, which further calculate similarity between *SQL question* and *certain SQL data types (table, column, SQL skeleton)* instead of using *document*. We also improve in-context learning (ICL) approach to benefit from retrieval effectiveness. As shown in Fig 1 (b), We use RB-models to separately retrieve *table*, *column*, *SQL skeleton* that have high similarity with our target question. This pre-processing method helps decrease redundant information in schema and search out few-shot examples with high reference value (similar SQL skeleton) for in-context learning.

In this paper, we propose a retrieval-based text-to-SQL framework named RB-SQL, which mainly contains three independent RB-models to separately calculate similarity between *question* and certain SQL data types (*tables*, *columns*, *SQL skeleton*). Table-Retriever aims to retrieve tables that are most relevant to the question from the massive tables in database. Column-Retriever further retrieves columns in the previous retrieved tables to reduce the number of selected columns. The goal of Table-Retriever and Column-Retriever is to play a pre-filtering role in text-to-SQL task, which not only reduces redundant information and minimizes the impact of excessive tables and columns (including their mutual effects) but also accelerates the efficiency of subsequent SQL generation. SQL-skeleton-Retriever is used for searching few-shot examples having similar SQL skeleton with questions. Besides, we introduce SQL skeleton into the stage of example organization between question and gold SQL, which enhances the in-context learning process. We conduct comprehensive evaluations on two cross-domain text-to-SQL datasets BIRD and Spider, experimental results indicate RB-SQL outperforms several baselines.

To summarize, our contributions are as follows:

- We propose RB-SQL, a novel retrieval-based framework for LLMs in text-to-SQL.
- We introduce three independent RB-models to refine SQL schema and select relevant examples for in-context learning. We also introduce SQL skeleton as an intermediate and effective step in the prompt example organization to guide correct SQL generation.
- Experimental results demonstrate that our proposed model outperforms several baselines on BIRD and Spider datasets.

2 Related Work

2.1 LLM for text-to-SQL

Recently, LLMs have shown remarkable improvement for various NLP tasks (Gao et al., 2024; Wang et al., 2024). Many researchers utilize LLMs in text-to-SQL tasks to further improve the performance. It is the most important tasks to properly design and use prompts to better guide LLMs for SQL generation, as it directly affects the accuracy. For example, Tai *et al.* (Tai et al., 2023) studied how to enhance the inference ability of LLMs through chain-of-thought style prompt, including the original chain-of-thought prompt and least-to-most prompt. Chang *et al.* (Chang and Fosler-Lussier, 2023) comprehensively investigated the impact of prompt constructions across various settings when constructing the prompt for text-to-SQL inputs. DAIL-SQL (Gao et al., 2024) consider both question and queries to select few-shot example, use a new example organization strategy to trade-off in terms of quality and quantity, and adopt Code Representation Prompt as the question representation. Additionally, some researchers propose novel frameworks for simplifying databases, query decomposition and other prompt engineering approach, like C3-SQL (Dong et al., 2023b) and DIN-SQL (Pourreza and Rafiei, 2023). More recently, Wang *et al.* propose MAC-SQL (Wang et al., 2023), a framework centered on multi-agent collaboration that can be utilized for more intricate data scenarios and a broader spectrum of error types for detection and correction.

2.2 Dense passage retrieval

Given a collection of M text passages, the goal of DPR is to index all the passages in a low-dimensional and continuous space, such that it can retrieve efficiently the top-k passages relevant to the input question (Karpukhin et al., 2020). Early researchers apply representation-focused rankers, which independently compute an embedding for question and another for document and estimate relevance as a single similarity score between two vectors (Zamani et al., 2018). There are also some researchers use all-to-all interaction, which models the interactions between words within as well as across question and document at the same time, as in BERT's transformer architecture (Nogueira and Cho, 2019). However, the performance of the former architecture need to be further improved, while the latter architecture has the relatively slower running efficiency. Therefore, Omar *et al.* propose late interaction as a paradigm for efficient and effective neural ranking (Khattab and Zaharia, 2020).

3 Problem Definition

Text-to-SQL is the task of converting a natural language question Q into a correct SQL query Y, which is capable of retrieving relevant data from a database. The database can be represented as $D = \{T_1, T_2 ... T_m\}$, m is the number of tables in the database. For $T = \{C_1, C_2 ... C_n\}$, C_i refers to columns in table T, n is the number of columns in the table. When dealing with complex database values, we may use external knowledge evidence K to support our model understand the inner relationship between question and database better. Ultimately, the process of text-to-SQL could be

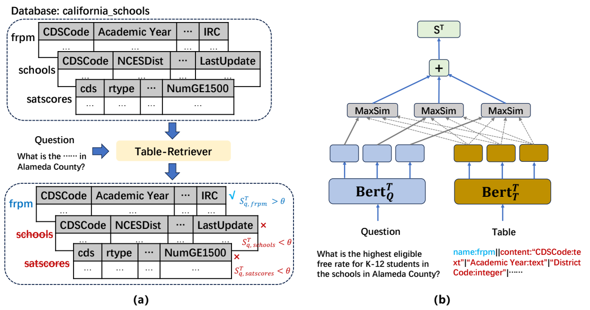

Figure 3: (a) The workflow of Table-Retriever. The module calculate similarity of question with tables and retrieve highly relevant tables for question. (b) Framework of Table-Retriever. We use BERT to encode question and table separately with MaxSim-based late interaction to calculate the similarity score.

formulated as follows:

$$Y = f\left(Q, D, K \mid \theta\right) \quad (1)$$

where $f(\cdot|\theta)$ can represent a model or neural network with the parameter θ.

4 Methodology

4.1 Proposed Model

Inspired by ColBERT (Khattab and Zaharia, 2020), we propose a retrieval-based text-to-SQL framework for constructing prompt, which consists of Table-Retriever (TR), Column-Retriever (CR) and SQL-Skeleton-Retriever (SR). TR, CR and SR are three different RB-models. TR filters out irrelevant tables which reduces the first interference at the database tables level. CR aims to continuously reduce the interference caused by columns (ex.too many columns in a table) and obtain appropriate numbers of relevant columns. TR and CR jointly complete SQL schema construction and involve schema linking and are served as a SQL pre-processing function. Furthermore, SR selects few-shot examples with similar SQL skeleton for questions, which provides syntactic guidance to generate more syntactically correct SQL results. What's else, we introduce SQL skeleton into example organization, which enhances the in-context learning process of LLMs.

4.2 Schema construction

4.2.1 Table-Retriever

Table-Retriever is a module for retrieving highly correlated tables for each question. Omar Khattab *et al.* (Khattab and Zaharia, 2020) discover that a model employing contextualized late interaction over deep LMs is efficient for retrieval. In our model, we use BERT as encoders and MaxSim-based late interaction to calculate the similarity of question q and table t. As shown in Fig 3 (b), we first convert the tables into continuous text by directly concatenating table name, column names and column types as $\{t_{text} = name : n||"c_1 : ty_1"|"c_2 : ty_2"|...|"c_n : ty_n"\}$, n is table name, c_i is column name, ty_i is data type of c_i. We use q as the input of $Bert_Q$, which computes a contextualized representation of each token. Then, we pass the output representations through a 1D-CNN layer with no activations, which is used for dimension reduction. Following the settings of ColBERT (Khattab and Zaharia, 2020), we typically fix the output size m to be much smaller than BERT's fixed hidden dimension, which we discuss later. After that, we normalize the output embeddings so each has L2 norm equal to one:

$$O_q^T = \text{Normalize}\left(\text{CNN}\left(\text{BERT}_Q^T\left(q\right)\right)\right) \quad (2)$$

We use converted table as the input of $Bert_T$, the rest of steps are the same as above, so we can get the output representations of table as follow:

$$O_t^T = \text{Normalize}\left(\text{CNN}\left(\text{BERT}_T^T\left(t_{text}\right)\right)\right) \quad (3)$$

Next, we employ the output embeddings O_q^T and O_t^T to conduct late interaction. Concretely, we apply each token embedding of O_q^T to calculate dot-products similarity with every embedding of O_t^T and obtain the maximum value. We add these value together and acquire the final similarity score of question q and table t:

$$S_{q,t}^T = \sum_{i \in [|O_q^T|]} \max_{j \in [|O_t^T|]} O_{q_i}^T \cdot O_{t_j}^T \quad (4)$$

Fig 3 (a) shows the process of table retrieval. We input a question and tables from a database into Table-Retriever module, then we get similarity scores of q with each table. If the score is higher than a threshold θ, we assume the table is relevant to this question. On the contrary, it is not. Table-Retriever module is used for retrieving highly relevant tables to help reduce the burden of inference for LLMs.

4.2.2 Column-Retriever

Column-Retriever is the downstream module of Table-Retriever. Given the retrieved tables output by Table-Retriever, Column-Retriever can retrieve

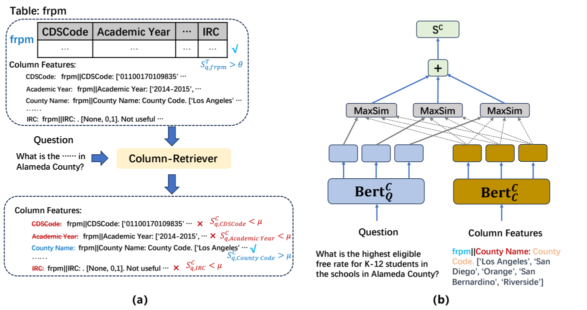

Figure 4: (a) The workflow of Column-Retriever. The module retrieve highly relevant columns for question. (b) Framework of Column-Retriever.

highly correlated columns for each question, such as those that are identical or semantically similar to certain entities in the question, and can further filters out redundant information of schema. As shown in Fig 4 (b), the framework of Column-Retriever is the same as Table-Retriever, which is designed to calculate the similarity of question q and column c. We convert column features into continuous text c_{text} by concatenating table name t_{name}, column name c_{name}, column description c_{desc}, examples $[e_1...e_i]$, value description v_{desc} and other knowledge k, as $\{c_{text} = t_{name}||c_{name} : c_{desc}[e_1...e_i]v_{desc}k\}$. Then we use q and c_{text} as the input of $Bert_Q$ and $Bert_C$, and obtain output embeddings O_q^C and O_c^C through similar process with Table-Retriever:

$$O_q^C = \text{Normalize}\left(\text{CNN}\left(\text{BERT}_Q^C\left(q\right)\right)\right) \quad (5)$$

$$O_c^C = \text{Normalize}\left(\text{CNN}\left(\text{BERT}_C^C\left(c_{text}\right)\right)\right) \quad (6)$$

In late interaction, we acquire the similarity score by the sum of MaxSim value in the same way:

$$S_{q,c}^C = \sum_{i \in [|O_q^C|]} \max_{j \in [|O_c^C|]} O_{q_i}^C \cdot O_{c_j}^C \quad (7)$$

Fig 4 (a) shows the process of column retrieval. We input the target question and column features (from the retrieved tables after TR) into Column-Retriever module to obtain similarity scores of q with each column. Only if the scores are higher than a threshold μ, we reserve the related columns. It can further reduce the overall length of the schema in the prompt and eliminates potential interference information, therefore improving the execution performance and accuracy of LLMs.

4.2.3 Specialized handling of Large tables

In practical applications, some tables may have too many columns that the converted tables t_{text} are so

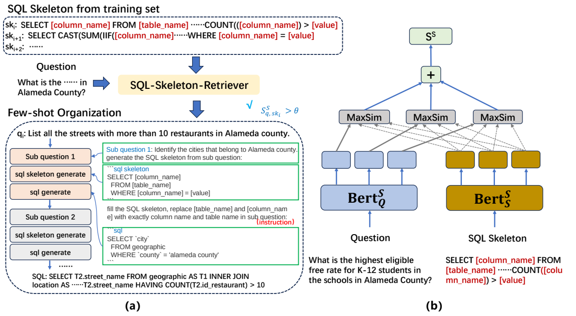

Figure 5: (a) The workflow of SQL-Skeleton-Retriever. (b) Framework of SQL-Skeleton-Retriever.

long. Since we use BERT as our encoder, which is not able to handle over 512 tokens, we need a specialized design to handle large tables. In our method, if the length of t_{text} is over 512, we firstly use Column-Retriever to perform coarse filtering with smaller threshold μ', which can shorten the table by reducing the number of columns. Then we pass the shortened table back to Table-Retriever. All the following steps are the same as before.

4.3 In-context Learning

LLMs can perform better for text-to-SQL through in-context learning, in which only a few examples are provided in the input prompts (Gao et al., 2024). To enhance the SQL generation capabilities of LLM, we specialized design the examples selection and examples organization for in-context learning in the following.

4.3.1 Example Selection

According to prior studies (Dong et al., 2023a), in-context learning is essentially learning from analogy, so it is effective to select examples that are similar with the target question. In our method, we apply a RB-model SQL-Skeleton-Retriever as the example selection module. As shown in Fig 5 (b), the framework of SQL-Skeleton-Retriever is the same as the RB-model above, the input for BERT-based encoders are question q and SQL skeleton sk. sk is the original SQL which is masked specific content by [column_name], [table_name] and [value] token. As we have introduced RB-model in detail , here we directly provide the formula of SQL-Skeleton-Retriever:

$$O_q^S = \text{Normalize}\left(\text{CNN}\left(\text{BERT}_Q^S\left(q\right)\right)\right) \quad (8)$$

$$O_{sk}^S = \text{Normalize}\left(\text{CNN}\left(\text{BERT}_S^S\left(sk\right)\right)\right) \quad (9)$$

$$S_{q,sk}^S = \sum_{i \in [|O_q^S|]} \max_{j \in [|O_{sk}^S|]} O_{q_i}^S \cdot O_{sk_j}^S \quad (10)$$

Before we select few-shot examples for in-context learning, we first translate all the SQL queries from our training set into SQL skeletons as a candidate set $SK = \{sk_1, sk_2 ... sk_n\}$. To conduct k-shot examples selection for a target question q, we apply SQL-Skeleton-Retriever to retrieve top-k SQL skeletons from SK. Then we trace the source and find the original samples corresponding to these skeletons as our final selected k-shot examples.

4.3.2 Example Organization

The example organization plays an important role in in-context learning which guides LLMs to think step by step and finally generate SQL result. There are two advanced methods for text-to-SQL parsing: chain-of-thought prompt and least-to-most prompt (Zhou et al., 2023). The former provide thinking process to obtain an answer, while the latter decompose complex question into progressively refined sub-questions and solve them one by one. Inspired by the previous work, we find it is efficient to decompose complex questions into multiple simple steps and provide the human like thinking process as detailed as possible.

Based on the above, we introduce SQL skeleton as an intermediate step in in-context learning, which conforms to human way of thinking. Fig 5 (a) illustrates our organization process. Given the selected few-shot question q_i, we first decompose it into sub-questions as the way of (Zhou et al., 2023). Then, we generate SQL skeleton (original SQL masked by [column_name], [table_name] and [value]), which guides LLMs to think about the structures of SQL first. Next, we prompts the model to extract exact values from the sub-question and fill the SQL skeleton to obtain gold SQL query. After all the sub-question solved, we finally obtain the SQL query of q_i. In conclusion, generating and filling SQL skeleton provide more detailed inference steps for in-context learning, which enhance the performance of LLM.

4.4 Error Correction

Error correction module is designed to automatically correct errors after generating SQL queries, because the generated SQL usually contains certain accidental errors such as missing keywords or syntax errors. Thus, we need an error correction module to optimize the initial SQL generation results by automatically amending specific errors.

We firstly execute the initial SQL results to obtain preliminary execution results (PER). Whether

Datasets	Train	Dev	Test
BIRD	9428	1534	1789
Spider	8659	1034	2147

Table 1: The statistics of BIRD and Spider datasets.

to use the error correction module will be evaluated based on execution feedback. When the PER are empty or certain errors occur during the process, we need integrate the SQL results and error information together as input to generate a correct SQL using LLMs. This iterative process continues until the PER is error-free or a predefined maximum number of correction attempts has been reached. The appendix A introduces this module in detail.

5 Experimental Setup

This section mainly introduces experimental setups. Table 1 shows the statistics of two datasets. The appendix B contains the experimental settings.

5.1 Datasets

- **BIRD** (Li et al., 2023c) represents a pioneering, cross-domain dataset that examines the impact of extensive database contents on text-to-SQL parsing. BIRD contains over 12,751 unique question-SQL pairs, 95 big databases with a total size of 33.4 GB. We test and verify the effect of our proposed method on development set, as the test set is not accessible.

- **Spider** (Yu et al., 2018) is a large-scale complex and cross-domain semantic parsing and text-to-SQL dataset. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. Inspired by BIRD, we generate extra evidence for Spider, which we illustrate in appendix C.

5.2 Evaluation Metrics

Following BIRD (Li et al., 2023c), we utilize execution accuracy (EX) and valid efficiency score (VES) to evaluate text-to-SQL models.

- **Execution Accuracy (EX)** (Li et al., 2023c) is defined as the proportion of questions in the evaluation set for which the execution results of both the predicted and ground-truth inquiries are identical, relative to the overall number of queries.

Model	BIRD	
	EX	VES
ChatGPT + CoT	36.64	42.30
GPT-4	46.35	49.77
DIN-SQL + GPT-4	50.72	58.79
DAIL-SQL + GPT-4	54.76	56.08
RB-SQL + GPT-4	**58.07**	**59.72**

Table 2: EX and VES on dev set of BIRD dataset.

Model	Spider	
	EX (dev)	EX (test)
C3 + ChatGPT	81.80	82.30
DIN-SQL + GPT-4	82.80	85.30
DAIL-SQL + GPT-4	84.40	**86.60**
RB-SQL + GPT-4	84.91	85.68
+ Generated Evidence	**85.89**	**86.73**

Table 3: EX on both dev and test set of Spider.

- **Valid Efficiency Score (VES)** (Li et al., 2023c) is designed to measure the efficiency of valid SQLs generated by models. It is worth noting that the term "valid SQLs" refers to predicted SQL queries whose result sets align with those of the ground-truth SQLs.

5.3 Baselines

- **GPT-4** (OpenAI, 2023) uses simple zero-shot text-to-SQL prompt for SQL generation.

- **DIN-SQL** (Pourreza and Rafiei, 2023) decompose the task into smaller sub-tasks and feed the solutions of those sub-problems into LLMs to generate the final SQL query.

- **DAIL-SQL** (Gao et al., 2024) consider both question and queries to select few-shot example, apply a new example organization strategy to trade-off in terms of quality and quantity, and adopt Code Representation Prompt as the question representation.

- **C3-SQL** (Dong et al., 2023b) is a novel zero-shot text-to-SQL method based on ChatGPT, which provides a systematic treatment from the perspective of model input, model bias, and model output.

6 Results and Analysis

6.1 Overall Results

The overall results of all the models on BIRD and Spider are shown in Table 2 and Table 3. We can

Method	BIRD	
	EX	VES
(1) RB-SQL + GPT-4	**58.07**	**59.72**
(2) GPT-4	46.35($\downarrow$ 11.72)	49.77($\downarrow$ 9.95)
(3) + Table-Retriever & Column-Retriever	54.06($\downarrow$ 4.01)	56.11($\downarrow$ 3.61)
(4) + SQL skeleton(example organization)	54.48($\downarrow$ 3.59)	56.38($\downarrow$ 3.34)
(5) + SQL-Skeleton-Retriever(example selection)	55.19($\downarrow$ 2.88)	56.81($\downarrow$ 2.91)
(6) + Error correction	**58.07**($\downarrow$ 0.0)	**59.72**($\downarrow$ 0.0)

Table 4: Results of ablation study on BIRD. "+" means adding module on the basis of the previous row.

learn from the results that our proposed-RB-SQL achieves better performance than several competitive baselines on the two datasets.

In Table 2, we report the performance of RB-SQL and other competitive baselines on development set of BIRD. Firstly, as a more powerful LLM, GPT-4 achieves better performance than Chatgpt with chain-of-thought. Then, we can find the recent researches DIN-SQL and DAIL-SQL beat GPT4 in both execution accuracy and valid efficiency score, while the former performs better in valid efficiency score and the latter performs better in execution accuracy. Finally, our proposed RB-SQL outperforms all the baselines in both metrics. Specifically, RB-SQL achieves at least 3.31% improvement in execution accuracy and 0.93% in valid efficiency score than the state-of-the-art. On the other hand, Table 3 shows the execution accuracy of RB-SQL and other baselines on development set and test set of Spider. Inspired by BIRD (Li et al., 2023c), external knowledge evidence is helpful for mapping the natural language instructions into counterpart database values. Thus, we generate evidence for the Spider in advance. With the generated extra evidence, RB-SQL reaches the new state of the art by at least 1.49% on the development set and by 0.13% on the test set, which further demonstrate the high efficiency of RB-SQL framework.

6.2 Ablation Study

To study the impact of the modules in RB-SQL, we evaluate it by conducting a set of ablation studies. We use BIRD as the representative because it is larger dataset with more tables and rows in databases. Row(1) represents the experiment results of the whole RB-SQL framework with GPT-4, while in the following rows, we start with GPT-4 and add Table-Retriever & Column-Retriever, SQL skeleton organization, SQL-Skeleton-Retriever and error correction module row by row to compare the efficacy of each module in RB-SQL framework. For comparison, the last row(6) represent the same framework as the whole RB-SQL after adding all modules. The results are shown in Table 4.

Firstly, let's pay attention to the comparison of rows(2)(3). After adding Table-Retriever & Column-Retriever modules, the execution accuracy raise by 7.71% and the valid efficiency score raise by 6.34%. The results imply the importance of tables and columns retrieval, and demonstrate that concise and direct table schema is efficient for prompt engineering. Secondly, experiments on rows(3)(4) illustrate the advantage of introducing SQL skeleton into example organization. By adding SQL skeleton into in-context learning, we provide more detailed instruction for LLM to learn and generate SQL query step by step. As a result, the execution accuracy raise by 0.42% and the valid efficiency score raise by 0.27%. Furthermore, the comparison of rows(4)(5) shows the performance improvement brought by SQL-Skeleton-Retriever module, which provides few-shot examples that have high similar SQL skeleton with our target query. Combine with the SQL skeleton step in example organization, the retrieved examples make the LLM easier to imitate and learn the generative process. The execution accuracy raise by 0.71% and the valid efficiency score raise by 0.43%. The experiment of row(6) increase the error correction module on the basis of row(5). We rerun samples with empty execution results or syntax errors for up to specific rounds or make simple corrections by rules. The execution accuracy raise by 2.88% and the valid efficiency score raise by 2.91%.

In conclusion, the ablation study proves that all the modules in RB-SQL framework play important roles for performance enhancement. Compare with GPT-4, the whole RB-SQL framework make further improvement by 11.72% in execution accuracy and 9.95% in valid efficiency score.

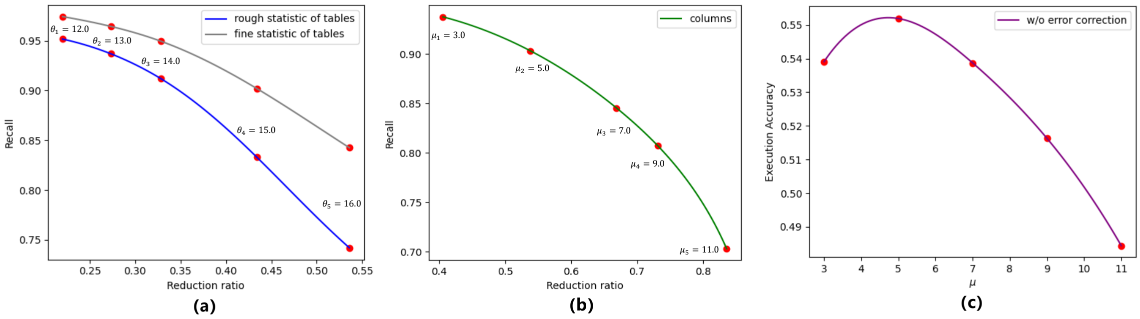

Figure 6: (a) Recall and reduction ratio with different θ in Table-Retriever. (b) Recall and reduction ratio with different μ in Column-Retriever. (c) Execution accuracy of LLM with different μ while the θ is fixed.

RB-SQL	BIRD		Spider	
	EX	VES	EX(dev)	EX(test)
0-shot	56.77	58.17	74.10	82.30
1-shot	56.96	58.65	82.13	84.66
3-shot	**58.07**	**59.72**	84.75	85.85
5-shot	57.88	59.61	**86.89**	**86.73**

Table 5: Results of RB-SQL with different number of few-shot examples on the dev set of BIRD and Spider.

7 Discussion

7.1 Hyper-parameter of Retrievers

Here we study how θ & μ influence the performance of Table-Retriever and Column-Retriever on the development set of BIRD. Figure 6 (a) shows the trade-off of recall and reduction ratio of Table-Retriever by tuning threshold θ (fine statistic means the recall of gold tables, while coarse statistic means the recall of all gold tables for each question). Specifically, with the growth of θ, the reduction ratio of invalid tables increase, but the recall of gold tables decrease. Similarly, as shown in Figure 6 (b), we fix θ=13.0 and modify μ. With the growth of μ, the reduction ratio of invalid column increase, while the recall of gold columns decrease. The appearance demonstrates that a higher confidence threshold may filter out both invalid and gold tables/columns, which will lead to a decrease in recall and an increase in reduction ratio.

Furthermore, we design a set of experiments to explore how confidence threshold of Table-Retriever and Column-Retriever influence the final performance of LLM. Here we use μ in Column-Retriever as the representative. Figure 6 (c) shows the execution accuracy of LLM with the tuning of μ while the θ is fixed, the settings of θ and μ is the same as Figure 6 (b). In order to study the impact for LLM clearly, we experiment without post-processing error correction module. We can easily find the execution accuracy first increase and then decrease with the growth of μ. As shown in table, when μ=5.0, we get the best LLM performance. The results indicates Table-Retriever and Column-Retriever with too small θ and μ may not decrease invalid tables and columns adequately, while too large θ and μ may lead to low recall of gold tables and columns. Thus, it is important to fine tune θ and μ to obtain a suitable value.

7.2 Number of Few-shot Examples

Table 5 shows the impact on different number of few-shot examples. As the number of shots increase from 0 to 5, the EX and VES of BIRD first increase and then decrease, reaching maximum value at 3-shot, while RB-SQL achieves the best results on Spider at 5-shot. The results indicates that few-shot examples are helpful for LLM generating SQL query, but excessive examples may lead to a decrease in efficiency and performance.

8 Conclusion

In this paper, we systematically propose a retrieval-based framework (RB-SQL) by constructing efficient SQL generation prompt to improve the LLMs' reasoning performance. We design three independent retrieval-based models to alleviate the drawback of redundant tables and columns which cause excessive redundancy, and retrieve similar samples for few-shot example selection. Then, we also introduce SQL skeleton in example organization to achieve more fine-grained SQL generation process. Through comprehensive experiments, the results demonstrate the effectiveness of retrieving and filtering valid information in advance for constructing LLM's prompt engineering, and the rationality of using skeleton to guide the correct SQL generation.

Limitations

In our work, we did not design more adaptable RB-models for different input structure or skillfully integrate pre-trained models and LLMs for more refined prompt engineering. Moreover, we introduce SQL skeleton as only an extra step into example organization process, which can lead to better results with more detailed steps and instructions.

Ethics Statement

In this work, all of the datasets, models, code and related documents are not associated with any ethical concerns.

References

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *CoRR*, abs/2305.11853.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023a. A survey for in-context learning. *CoRR*, abs/2301.00234.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Lu Chen, Jinshu Lin, and Dongfang Lou. 2023b. C3: zero-shot text-to-sql with chatgpt. *CoRR*, abs/2307.07306.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4524–4535. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. *CoRR*, abs/1908.08113.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1254–1262. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. RESDSQL: decoupling schema linking and skeleton parsing for text-to-sql. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 13067–13075. AAAI Press.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 13076–13084. AAAI Press.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023c. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *CoRR*, abs/1901.04085.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *CoRR*, abs/2208.13629.

Ruoxi Sun, Sercan Ö. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. 2023. Sql-palm: Improved large language model adaptation for text-to-sql. *CoRR*, abs/2306.00739.

Chang-Yu Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain of thought style prompting for text-to-sql. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5376–5393. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7567–7578. Association for Computational Linguistics.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. MAC-SQL: A multi-agent collaborative framework for text-to-sql. *CoRR*, abs/2312.11242.

Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. 2022. Table retrieval may not necessitate table-specific model design. *CoRR*, abs/2205.09843.

Zihan Wang, Xinzhang Liu, Shixuan Liu, Yitong Yao, Yuyao Huang, Zhongjiang He, Xuelong Li, Yongxiang Li, Zhonghao Che, Zhaoxi Zhang, Yan Wang, Xin Wang, Luwen Pu, Huihan Xu, Ruiyu Fang, Yu Zhao, Jie Zhang, Xiaomeng Huang, Zhilong Lu, Jiaxin Peng, Wenjun Zheng, Shiquan Wang, Bingkai Yang, Xuewei He, Zhuoru Jiang, Qiyi Xie, Yanhan Zhang, Zhongqiu Li, Lingling Shi, Weiwei Fu, Yin Zhang, Zilu Huang, Sishi Xiong, Yuxiang Zhang, Chao Wang, and Shuangyong Song. 2024. Telechat technical report. *CoRR*, abs/2401.03804.

Hefeng Wu, Yandong Chen, Lingbo Liu, Tianshui Chen, Keze Wang, and Liang Lin. 2023. Sqlnet: Scale-modulated query and localization network for few-shot class-agnostic counting. *CoRR*, abs/2311.10011.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 497–506. ACM.

Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. HIE-SQL: history information enhanced network for context-dependent text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2997–3007. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

A Error Correction

In this paper, we classify errors into five categories: syntax errors, schema linking errors, internal errors, empty results, and mismatched results. Our error correction module dedicates to resolving the first four types of errors. Specifically, syntax errors, internal errors and empty results are caused by a variety of complex reasons, while schema linking errors account for the largest proportion and are easily perceived by LLMs. Thus, we focus on discussing this type of error in the following. Among the types of schema linking errors, the most frequent ones are forging columns and forging tables. There are two reasons for this type of error. On the one hand, LLMs may produce hallucinations. On the other hand, some related tables and columns may be filtered out during the retrieval process, which force LLMs to forge schema information in order to match the semantics of the query. To handle the issues above, we further enhance our correction module. In particular, we substitute the filtered schema with a full schema when the output of LLMs explicitly signals the absence of schema components or after multi iterations of error correction process.

B Experimental settings

We reproduce all baselines with their original experimental settings. For three RB-models, We use the popular transformers library for pre-trained BERT. Similar to previous work (Khattab and Zaharia, 2020), we fine-tune all RB-models with learning rate 3×10^{-6} with a batch size 32. We fix the number of embeddings per question at 32 with [mask] tokens padding or truncating it to the first 32 tokens. Our RB-models embedding dimension m is set to 128. In condition, we adopt L2 normalization for output dimension, and cosine similarity as the final similarity score. We construct training set for Table-Retriever and Column-Retriever by paring each positive one with negative ones in the same database, and paring each positive SQL skeleton with random 100 negative SQL skeletons for SQL-Skeleton-Retriever as [+,-]. Taking BIRD as an example, we finally construct training sets for three RB-models with size of 181416, 288444 and 942800 (we provide the processed training sets in `https://anonymous.4open.science/r/Anonymize-A5E7/RB-model`). Finally, we train models for a maximum of 5 epochs which is enough for convergence.

As we have analysed in section 7.1, for achieving the best retrieval effects, hyper-parameter θ & μ should be neither too large, nor too small. Thus, we use grid-search strategy to tune the hyper-parameters. We tune θ in $\{11,12,13,14,15,16\}$ and μ in $\{1,3,5,7,9,11\}$, and we finally obtain the best result at θ=13 and μ=5.

We use a single Tesla V100 GPU with 32 GiBs of memory on a server to pre-train RB-models, the total number of parameters for each model is approximately 220 million, the training time is about 3~6 hours for each. All the experiments utilize gpt-4-turbo version, the context window is 128000, the temperature is set to 0.1. We enable five threads to run RB-SQL (approximately 200-500 samples for each according to the size of dataset), it costs about 4~6 hours to generate all results.

C Evidence generation for Spider

Inspired by BIRD, we find that evidence of database provides extra knowledge that can help SQL generation process. Thus, we generate evidence for Spider by using LLM. Concretely, we give out [question], [schema] and instructions to guide gpt-4 generate evidence for each sample. We show detailed instructions and an example as follows (`https://anonymous.4open.science/r/Anonymize-A5E7/prompt_for_evidence.txt`):

C.1 Instruction:

Given a [Database schema] description and the [Question], you need to use valid SQLite and understand the database knowledge, and then generate the [Evidence] of the [Question].
When generating [Evidence], we should always consider constraints:
[Constraints]
1. Map the entities or metadata from user questions to the schema.
2. Take into account the examples in the schema and convert the natural language descriptions in user input into the standard format in the database.
3. Evidence should be a single sentence describing the relationship between user queries and the schema.

C.2 An example:

[Question]
How many singers do we have?
[Database schema]
Table: stadium [Stadium_ID,Location,...]

Table: singer [Singer_ID,Name,...]
Table: concert [concert_ID,concert_Name,...]
Table: singer_in_concert [concert_ID,Singer_ID]
[Foreign keys]
concert.‘Stadium_ID‘ = stadium.‘Stadium_ID‘
singer_in_concert.‘Singer_ID‘ = singer.‘Singe_ID‘
singer_in_concert.‘concert_ID‘=concert.‘concert_ID‘
[Evidence]
The total number of singers is represented by the count of distinct ‘Singer_ID‘ in the table singer.

D Prompt details of RB-SQL

We provide an example in `https://anonymous.4open.science/r/Anonymize-A5E7/prompt_case.txt` to illustrate the prompt details of RB-SQL, which contains 3-shot examples and all the instructions.