International Journal of Multiphysics
Volume 18, No. 4, 2024
ISSN: 1750-9548

Enhancing Text-to-SQL Capabilities of Large Language Models by intermediate representations and Chain of Thought

Tingxuan Fu[1], Yu Fu[2], Haoran Li[2], Sijia Hao[1], Qiming Chen[1]
Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650031 (PR China)[1]
School of Computer & Information Technology, Northeast Petroleum University, Daqing 163318 (PR China)[2]

Abstract

While large language models with in-context learning have dramatically improved the performance of text-to-SQL tasks, the semantic gay between natural language and SQL queries has not yet been bridged. Although some intermediate representations are designed to reduce the difficulty of SQL query generation, the dilemma of problem decomposition is not effectively alleviated in complex scenarios. Our proposed solution is intended to address both of these issues. First of all, we use NatSQL as the intermediate representation to implement the task of Text-to-NatSQL. Secondly, we use samples with Chain-of-Thought information to fine-tune small and medium-scale LLMs to enhance their task decomposition and reasoning capabilities in complex scenarios. Experiment results demonstrate that our model achieves performance similar to or better than several competitive baselines on public datasets Spider.

1. Introduction

Text-to-SQL involves the automatic transformation of natural language (NL) questions into structured SQL statements. Previously, this task was done using machine learning models to complete the mapping between natural language and SQL queries. However, the emergence and rapid development of large language models (LLMs) have brought new dawn to the study of this problem. LLMs show great potential for understanding complex NL problems and generating complex SQL queries. Due to advanced inference techniques and contextual learning capabilities, LLMs have greatly outperformed traditional methods.

Since SQL is designed to query relational databases, it cannot adequately represent the meaning in natural language, so there is a mismatch problem in Text-to-SQL [1]. Since there is a semantic gap between natural language and its corresponding SQL query, some research has focused on designing efficient intermediate representations (IRs) to bridge this gap [1,2,3]. Instead of directly generating SQL statements, these methods train the model to generate more concise IRs that can be directly translated to corresponding SQL statements via a specially designed transpiler. Among them, NatSQL [3] provides simplified queries for other IRs, and also retains the high coverage of SQL structures, further eliminating the mismatch between NL and SQL. These processes greatly reduce the difficulty of natural language translation, making it easier to transition from NL to SQL. Therefore, NatSQL has been used in several important research results [4,5,6].

With Chain-of-Thought (CoT) prompting [7], LLMs are able to analyze the intermediate steps of problem solving with additional prompts in order to mimic human reasoning methods to answer more complex questions. While CoT prompting work well for large LMs, it doesn't necessarily provide the same benefits for smaller LMs. In order for a relatively small LMs to work effectively, a series of studies have taken the approach of fine-tuning with rationales (denoted as CoT fine-tuning) [8,9,10]. Normal CoT prompting does not perform well on Text-to-SQL mainly because the task involves complex inference and requires analysis of the query intentions as well as

the database schema. According to the characteristics of Text-to-SQL tasks, we propose a new paradigm, which is to use CoT to divide complex query tasks into simple and easy-to-handle subtasks.

Fine-tune a general-purpose LLMs with hundreds of training examples to apply it to a specific task [46,47]. While fine-tuning requires many more examples than few-shot learning, the number of examples is much smaller compared to LLMs pre-training that relies on a large corpus. While fine-tuning has a higher upfront cost, it has two advantages. First, it is possible to achieve the performance of an unfine-tuned closed-source model by fine-tuning the open-source model. Second, fine-tuning reduces the need for system prompts and few-shot examples, making it easier to ask questions in a more concise way. DAIL-SQL first applies supervised fine-tuning to the Text-to-SQL domain. Supervised fine-tuning has proven to be very beneficial for open-source LLMs in text-to-SQL.

Although LLMs have made encouraging progress in Text-to-SQL, there are still problems with Linguistic complexity and ambiguity, Schema understanding and representation, and cross-domain generalization. This paper aims to solve the complex mapping relationship between natural language and SQL queries. For complex query requirements, additional intermediate steps can be employed to bridge the semantic gap between natural language questions and SQL queries. While complex queries can benefit from listing the intermediate steps in a Chain-of-Thought style prompting, this approach may degrade the performance for simpler tasks [7]. We use NatSQL to address this issue. We divide Text-to-SQL into two steps, first implement Text-to-NatSQL, and then convert NatSQL to SQL.

Even though complex SQL is less difficult to generate with the help of NatSQL, it is still a big challenge. When a problem is complex and needs to be recursively inferred, it may be a better choice to divide it into several sub-problems [11,12,13,14]. Compared with vanilla CoT, we use a sub-problem partition strategy to decompose a complex problem into a series of simple sub-problems. With the help of NatSQL, these sub-problems are easier to describe and therefore easier to solve.

Our goal is to provide step-by-step inference capabilities for smaller LMs by using CoT samples for fine-tuning. However, the introduction of CoT has the following problems: firstly, the manually pre-designed CoT cannot meet the needs of complex and changeable practical application scenarios; Second, the limitation of context length makes the number of CoT examples limited. We introduce a batch of Text-to-NatSQL samples with CoT prompting to train small and medium-scale open-source LLMs, and want to explore two points: firstly, the feasibility of Text-to-NatSQL; Second, whether the learning samples with CoT prompting can be beneficial to the development of the reasoning ability of LLMs.

To summarize, our contributions are as follows:

• The idea of Text-to-NatSQL is clearly proposed, and the difficulty of complex operations such as nesting and collection in Text-to-SQL is reduced through intermediate representation.

• Combining CoT fine-tuning and NatSQL to significantly improve the performance of Text-to-SQL in complex scenarios by improving problem representation and problem decomposition.

• Experiments have proven the feasibility of our method. Although NatSQL has inherent drawbacks, our approach still achieves performance close to or better than several competing baselines.

2. Methodology

2.1 Research roadmap

The focus of this paper is on how to overcome the complexity inherent in SQL queries. According to the syntax of the SQL language, SQL queries can be complex, such as using multiple SQL keywords, containing nested sub-queries, containing set operations, various column selections or aggregations, the application of conditional statements, and the involvement of joins across multiple tables. This complexity creates a large semantic difference between natural language and SQL queries. Our general idea is to overcome this problem by introducing NatSQL and reducing the semantic gap between natural language and SQL queries. The second is to use CoT fine-tuning to reduce the difficulty of analyzing complex problems. Other issues in Text-to-SQL are beyond the scope of this study.

The research route of this paper is shown in Figure 1, and the relevant details are explained below.

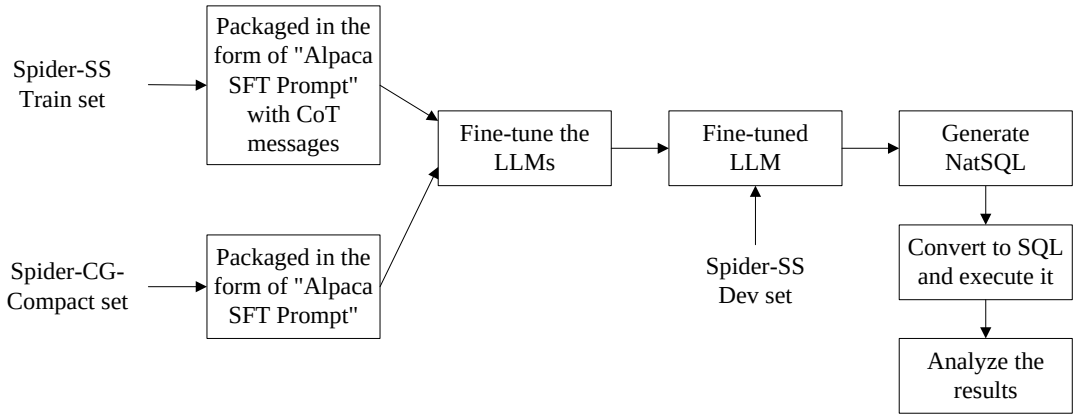

Figure 1. Research roadmap of this paper

2.2 The value of NatSQL

NatSQL retains the core functionality of SQL while being structurally easy to align with the NL problem, so using it can simplify the implementation of queries. Given the excellent performance of NatSQL [4,5,6], we choose it as our intermediate representation. Using NatSQL as an intermediary can significantly reduce the difficulty of implementing text-to-SQL in complex scenarios. Since there is already an existing algorithm for NatSQL-to-SQL [1], we need to use LLMs to solve the Text-to-NatSQL problem. The core of our solution is to fine-tune the LLMs implementation of Text-to-NatSQL.

In order to solve the Text-to-NatSQL problem by fine-tuning the LLMs, there must be enough high-quality learning samples. Moreover, in order to better describe the generation method of complex SQL queries, the query problem needs to be divided into several sub-problems in the learning sample, and the corresponding NatSQL clauses for each sub-problem are given. In order to achieve this idea, we introduce the CoT to illustrate the fine-tuning sample. All of this depends on the Spider-SS dataset [15].

2.3 Spider-SS dataset

As described in [15], the purpose of building Spider-SS is to obtain clause-level text-to-SQL data without the use of an alignment algorithm that is hard to build based on the complex large cross-domain text-to-SQL dataset. Spider-SS contains 7000 training and 1034 development examples. Since the Spider test set is not public, Spider-SS does not contain a test set. Spider-SS is implemented by using a sentence splitting algorithm to cut sentences into clauses, and then manually annotating the corresponding NatSQL clauses for each clause.List 1 is an example of the Spider-SS dataset, edited and cut for clarity.

List 1. An example from the Spider-SS dataset. "question_ type" is used to identify the division of the problem clause.

question: Find the id for the trips that lasted at least as long as the average duration of trips in zip code 94103
question_type: [1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]
NatSQL: select trip.id from trip where @ >= avg (trip.duration) and trip.zip_code = 94103
query: SELECT id FROM trip WHERE duration >= (SELECT avg(duration) FROM trip WHERE zip code = 94103)

As can be seen from List 1, in the public Spider-SS dataset, there is no "manual annotation of the NatSQL clause corresponding to each clause", only the final complete NatSQL statement is provided, but the partition method of each clause in the query question is retained, so we design an auxiliary algorithm to decompose the complete NatSQL statement into "NatSQL clause corresponding to each question clause". A detailed description of the algorithm is shown in Appendix A, and the processing results of the sample in List 1 are shown in Table 1.

Table 1. The correspondence between each problem clause obtained by the algorithm and each NatSQL clause

question clauses	NatSQL clauses
Find the id for the trips	SELECT id
that lasted at least as long as the average	@ >= avg (trip.duration)

duration of trips in zip code 94103	trip.zip_code = 94103

From the above results, it can be seen that a complex query problem is divided into several sub-problems, and the corresponding NatSQL clauses for each sub-problem are given. Next, we can construct a sample of CoT-style fine-tuning.

2.4 Construction of SFT samples

Some works have attempted to explicitly incorporate CoT samples into the training corpus in order to enhance step-by-step reasoning ability and avoid over-fitting to monotonic sample templates [10,16,17]. The inclusion of CoT information in the training corpus can introduce more necessary knowledge and reasoning materials for LLMs, which will have a profound impact on the reasoning ability of LLMs. We use CoT to construct learning samples to see if small and medium-sized LLMs can increase the Breakthroughness in this area through samples with CoT information.

DAIL-SQL [18] proposes an Alpaca SFT Prompt for Text-to-SQL domain fine-tuning LLMs, which we extend in two ways. First, in order to provide more database information, we add field types and primary and foreign key information on the basis of the original table names and field names. The second is to add the analysis process of problem decomposition and sub-problem solving in CoT style. So, our example can essentially be summarized as a quadruple (DB Schema, Question, CoT Hint, NatSQL Statement).

List 2. An example of fine-tuning an LLM.

```
1 Below is a directive describing the task, paired with an input that provides further context. Tips for completing tasks step by step are also provided. Write a response to complete the request appropriately.
2
3 ### Instruction:
4 Write a NatSQL to answer the question " Find the id for the trips that lasted at least as long as the average duration of trips in zip code 94103".
5
6 ### Input:
7 DB Schema（Table name, field name, field type, and primary and foreign key information）
8
9 ### Tips:
10 NatSQL preserves the core functionalities of SQL, while simplifying the queries as follows:(1) dispensing with operators and keywords such as GROUP BY, HAVING, FROM, JOIN ON; (2) removing the need for nested subqueries and set operators, using only one SELECT clause in NatSQL.
11 Let's analyze it step by step. We break down the problem into sub-problems, and then solve them one by one.
12 Based on “Find the id for the trips”, we can use NatSQL to write the answer “SELECT id” .
13 Based on “that lasted at least as long as the average duration of trips”, we can use NatSQL to write the answer “@ >= avg ( trip.duration )” .
14 Based on “in zip code 94103”, we can use NatSQL to write the answer “trip.zip_code = 94103” .
15 Combining the above analysis, we can come up with the following results.
16 ### Response:
17 select trip.id from trip where @ >= avg ( trip.duration ) and trip.zip_code = 94103.
```

2.5 Selection of LLM models

We choose Llama-2 (LLaMA-2-CHAT-13B) and Code Llama models (CodeLlama-7b, CodeLlama-13b, CodeLlama-34b) as the fine-tuning objects. Llama 2 is one of the highest-performing open-source language models on the market, excelling in key benchmarks such as inference, coding, proficiency, and knowledge testing. Code Llama is based on Llama 2 and fine-tuned in Python, C++, Java, PHP, Typescript (JavaScript), C#, Bash and other languages to support code generation.

Some researchers have found that when the model size is relatively small (typically below 10 billion parameters), CoT doesn’t have a positive impact. However, as the model size increases, it can exhibit significant performance improvements [7,19,20,21]. So we tried different scales of the Code Llama model to explore the sensitivity of CoT fine-tuning to model size. Although Meta open-source the CodeLlama-70b model in January 2024, it is limited by our hardware resources and our research focus on medium-sized LLMs, so we do not consider this model.

The reason for the inclusion of the Llama-2-CHAT-13b model in the study is that we want to study two aspects: first, the effect of CoT fine-tuning on a model that have not been fine-tuned by code, second, the effect of NatSQL fine-tuning on models that have not been fine-tuned by code. Originally, our ideal comparison model is Llama-2-CHAT-33b, but unfortunately it is not open-sourced.

2.6 CoT Fine-tuning

Spider-SS contains 7000 training examples and 1034 development examples. For CoT fine-tuning, 7000 examples are sufficient, but for NatSQL fine-tuning, this number is seriously insufficient. Luckily, we can take advantage of the Spider-CG dataset. Spider-CG is a synthetic dataset, which is generated by recombining the sub-sentences of Spider-SS. In order to ensure the accuracy of the evaluation results, all the examples generated from the Spider development set in the Spider-CG training set were excluded, and finally 39479 examples remained, which are referred to as the Spider-CG Compact Training Set(Spider-CG-Compact) in this paper. It should be noted that the examples in the Spider-CG training set do not contain "decomposition of problem clauses" like the Spider-SS dataset, so the examples in the Spider-CG training set cannot be described in the CoT style of List 2 but can only be organized in the traditional "(DB Schema, Question, NatSQL Statement)" triplet.

To summarize, the fine-tuning sample set we used consists of two parts: one is composed of 7000 training examples in Spider-SS, written in the form of CoT-enhanced Alpaca SFT Prompt; one section consists of 39,479 examples from Spider-CG-Compact, written in the form of an Alpaca SFT Prompt.

We fine-tune LLM for 2 epochs with a batch size of 128, a learning rate of 1e-6, and a max context length of 2,048. The learning rate has a cosine decay and a linear warmup for the initial 3% of training. All LLMs are fine-tuned on a server with eight 64G A100 GPUs.

2.7 Calibration of Model obfuscation

After our fine-tuned Code Llama model, we face a problem: it is trained on both SQL and NatSQL examples, and a large part of the functions of the two are similar, so there will be confusion. In particular, the new @ and table.* placeholder features have been added to NatSQL. Therefore, we add the "Calibration instructions" shown in List 3 to the prompt.

List 3. Calibration instructions of Model obfuscation

```
    Notice that @ is a place holder of NatSQL. When we need to use a field to accomplish a certain function but cannot determine which field it is, use @ to represent this field.
    Notice that the '*' keyword does not appear in the WHERE condition without an aggregation function, so NatSQL uses it to represent a table.
```

3. Experiments

3.1 Experimental Setup

- Datasets

We benchmark text-to-SQL on Spider [22]. We also evaluate the robustness of the model in three more challenging benchmarks: Spider-DK [23], Spider-Syn [24], and Spider-Realistic [25]. Spider-DK, Spider-Syn, Spider-Realistic are variants derived from the original Spider dataset and are used to simulate problems that users might encounter in real-world scenarios. Our fine-tune training example consists of 7000 examples in Spider-SS and 39479 examples in Spider-CG-Compact. Due to the limitations of the Spider submission platform, we are unable to evaluate our models using its test set. Therefore, the main evaluation is carried out on its publicly available development set.

- Evaluation Metrics

For Spider-family benchmarks (including Spider, Spider-DK, Spider-Syn, Spider-Realistic), we consider two prevalent evaluation metrics: execution accuracy (EX) [22] and test-suite accuracy (TS) [26]. The EX metric evaluates whether the predicted and ground-truth SQL queries yield the same execution results on the database. The TS metric assesses if the generated SQL query consistently passes the EX evaluations across multiple database instances, which are derived from automated database augmentations.

• Baselines

Our baselines are drawn from SOTA text-to-SQL approaches listed on the official leaderboards of benchmarks, including open-source and closed-source models, covering fine-tuning, zero-shot, few-shot, and more.

3.2 Evaluation

In order to show the effect of "Text-to-NatSQL" and "CoT fine-tuning", the experimental process should try to exclude the influence of other methods, so except for the "Calibration of Model obfuscation" mentioned in Section 2.7, we do not use auxiliary methods such as "Schema linking simplification", "self-consistency", "self-correction" to improve the accuracy.

• Evaluation results of the Spider's dev set

Table 2 shows the EX and TS evaluation results of the four fine-tuned LLMs versus the other methods on the Spider's dev set under the 3-shot in-context learning setting.

Table 2. The EX and TS evaluation results of the four fine-tuned LLMs versus the other methods on the Spider's dev set.

Methods	Features	EX(%)	TS(%)
SQL-PaLM [27]	Fine-tuning	87.3	83.5
SFT CodeS-7B [28]	Fine-tuning	85.4	80.3
RESDSQL-3B + NatSQL [6]	Fine-tuning	84.1	73.5
T5-3B + PICARD [29]	Fine-tuning	79.3	69.4
DAIL-SQL + GPT-4 + Self-Consistency [18]	Fine-tuning + few-shot	83.6	72.8
SQL-PaLM [27]	few-shot	82.7	77.3
ACT-SQL + GPT-4 [30]	few-shot	82.9	74.5
C3 + ChatGPT [31]	Zero-shot	81.8	-
DIN-SQL + GPT-4 [5]	few-shot	74.2	-
GPT-4 [5]	few-shot	67.4	-
Ours			
SFT Llama-2-CHAT-13b	Fine-tuning + few-shot	79.6	73.7
SFT CodeLlama-7b	Fine-tuning + few-shot	72.4	63.1
SFT CodeLlama-13b	Fine-tuning + few-shot	83.2	76.3
SFT CodeLlama-34b	Fine-tuning + few-shot	84.8	77.6

As can be seen from the results in Table 2, our method has obtained good results. Specifically, SFT CodeLlama-34b (few-shot) is slightly better than RESDSQL-3B + NatSQL, which also applies to NatSQL, and slightly worse than SFT CodeS-7B, which also uses fine-tuning. The performance of SFT CodeLlama-13b (few-shot) is slightly worse than that of SFT CodeLlama-34b (few-shot), indicating that the scale of 13B is already a good choice for CoT. SFT Llama-2-CHAT-13b (few-shot) has a significant gap with SFT CodeLlama-13b (few-shot), indicating that Code pre-training on CodeLlama is very valuable. The gap between SFT CodeLlama-7b (few-shot) and SFT CodeLlama-13b (few-shot) is large, indicating that the scale of 7B is not enough for CoT.

• Evaluation on Robustness Benchmarks

We evaluate the robustness of our method on three Spider variants (Spider-DK, Spider-Syn, and Spider-Realistic) and compare it to other methods, as shown in Table 3.

Table 3. Evaluates of our approach on three Spider variants for robustness

Methods	Features	Spider-Syn		Spider-Realistic		Spider-DK
		EX(%)	TS(%)	EX(%)	TS(%)	EX(%)
RESDSQL-3B + NatSQL [6]	Fine-tuning	76.9	66.8	81.9	70.1	66.0
T5-3B + PICARD [29]	Fine-tuning	69.8	61.8	71.4	61.7	62.5
ChatGPT [32]	few-shot	58.6	48.5	63.4	49.2	62.6
SQL-PaLM [27]	few-shot	74.6	67.4	77.6	72.4	66.5
SQL-PaLM [27]	Fine-tuning	70.9	66.4	77.4	73.2	67.5
Ours						
SFT Llama-2-CHAT-13b	Fine-tuning + few-shot	68.7	60.4	72.3	63.1	62.9
SFT CodeLlama-7b	Fine-tuning + few-shot	59.2	47.9	61.7	47.6	59.4
SFT CodeLlama-13b	Fine-tuning + few-shot	73.7	64.2	78.5	71.7	66.2
SFT CodeLlama-34b	Fine-tuning + few-shot	72.1	66.3	77.3	70.8	65.7

As can be seen from Table 3, SFT CodeLlama-13b and SFT CodeLlama-34b achieve similar performance to RESDSQL-3B + NatSQL and SQL-PaLM. Our model is trained on Spider but test on its variants, and these results demonstrate the method's good generalization ability in challenging scenarios.

- Evaluations on the Spider's dev set across various difficulty levels

We compare the execution accuracy performance results across the four SQL difficulty levels officially classified by Spider. It can be seen that our approach has obvious value for complex reasoning tasks. From the table 3, it can be seen that the performance of our method is significantly improved compared with SQL-PaLM and RESDSQL-3B + NatSQL for tasks at the Medium, Hard, and Extra levels, especially for the Hard and Extra tasks. This fully illustrates the importance of introducing NatSQL and using samples with CoT for complex SQL generation. On easy-level tasks, there is a slight drop in performance, which we speculate is due to the imperfect parsing from NatSQL to SQL creating additional errors.

Table 4. Execution accuracy (EX %) on the Spider's dev set across various difficulty levels

Methods	Features	Easy	Medium	Hard	Extra	All
DIN-SQL + GPT-4 [5]	few-shot	91.1	79.8	64.9	43.4	74.2
GPT-4 [5]	few-shot	86.7	73.1	59.2	31.9	67.4
SQL-PaLM [27]	few-shot	93.5	84.8	62.6	48.2	77.3
RESDSQL-3B + NatSQL [6]	Fine-tuning	91.6	75.2	65.5	51.8	73.7
Ours						
SFT Llama-2-CHAT-13b	Fine-tuning + few-shot	89.9	87.7	78.7	43.4	79.6
SFT CodeLlama-7b	Fine-tuning + few-shot	90.3	80.9	64.4	31.3	72.4
SFT CodeLlama-13b	Fine-tuning + few-shot	91.1	89.5	79.3	58.4	83.2
SFT CodeLlama-34b	Fine-tuning + few-shot	90.7	90.8	85.1	59.6	84.8

3.3 Ablation studies

We performed extensive ablation studies on the execution accuracy of the Spider dataset to assess the impact of each critical component. The results are shown in Table 5.

- Comparison between zero-shot and few-shot

Using the zero-shot method (-w/o few-shot) results in a slight decrease in performance. Although some studies indicate that after fine-tuning, few-shot makes little sense. However, in the current scenario, it is proved that even if a learning sample containing CoT is used, the few-shot method still has some value. We try that the few-shot example does not contain CoT information, and the result is not as good as the zero-shot method. Based on this, we speculate that CoT plays a role in few-shot.

- Whether to enable Calibration of Model obfuscation for new features of NatSQL such as @

If you do not enable Calibration of Model obfuscation (-w/o Calibration of Model obfuscation) for new features of NatSQL such as @, the performance of SFT CodeLlama will be significantly degraded, but there will be almost no impact on SFT Llama-2. This proves our guess: the SFT CodeLlama model confuses NatSQL with SQL statements and needs additional hints to fix it.

- Whether the dataset Spader-CG is added to fine-tuning

In this test, we wanted to explore whether the 7000 samples of the Spider-SS dataset could achieve satisfactory learning results for the model. If the dataset Spider-CG does not add fine-tuning (-w/o Spider-CG), the performance of the simple difficulty will be basically unchanged, and the performance of the complex difficulty will be significantly reduced. We analyze that although the samples of the dataset Spider-CG do not contain CoT information, more samples are more beneficial for the model to learn the parsing of complex NatSQL statements.

- Whether CoT prompts are included in the learning sample

When the learning sample does not contain CoT information (-w/o CoT in examples), the performance deteriorates significantly, especially the performance of complex difficulty decreases by more than 10%, and the overall performance decreases by 6%. Obviously, it is of great significance to include CoT information in the sample,

especially in complex difficulty scenarios, and helping the model to decompose the problem can significantly reduce the difficulty of the model to learn and achieve better performance.

Table 5. Ablation studies on Spider's dev set in the 3-shot in-context learning manner (Execution accuracy %)

Methods	Easy	Medium	Hard	Extra	All
SFT Llama-2-CHAT-13b(original)	89.9	87.7	78.7	43.4	79.6
-w/o Few-shot	88.7	86.1	74.7	41.6	77.7
-w/o Calibration of Model obfuscation	89.5	87.9	79.3	43.4	79.7
-w/o Spider-CG	90.3	87.4	77.0	41.0	78.9
-w/o CoT in examples	81.0	78.5	63.2	26.5	68.2
SFT CodeLlama-7b(original)	90.3	80.9	64.4	31.3	72.4
-w/o Few-shot	89.5	79.4	59.8	27.1	70.1
-w/o Calibration of Model obfuscation	87.5	78.5	59.2	25.9	69.0
-w/o Spider-CG	89.9	80.5	57.5	24.1	69.8
-w/o CoT in examples	84.7	76.5	56.3	23.5	66.6
SFT CodeLlama-13b(original)	91.1	89.5	82.2	55.4	83.2
-w/o Few-shot	89.9	88.6	81	54.2	82.1
-w/o Calibration of Model obfuscation	88.7	87.7	79.3	53.0	81.0
-w/o Spider-CG	91.1	89.2	78.7	51.2	81.8
-w/o CoT in examples	87.5	86.5	70.7	41.0	76.8
SFT CodeLlama-34b(original)	90.7	90.8	86.8	57.8	84.8
-w/o Few-shot	89.9	89.9	85.1	56.0	83.6
-w/o Calibration of Model obfuscation	89.1	88.6	83.3	55.4	82.5
-w/o Spider-CG	90.3	90.4	82.8	51.8	82.9
-w/o CoT in examples	87.9	87.0	71.8	41.6	77.4

4. Analysis

4.1 Error analysis

We analyze 157 error results generated by the SFT CodeLlama-34b (Fine-tuning + few-shot) model on the Spider's development dataset, as shown in Figure 2. For comparison, we use the exact same error categories as C3 [31]. If a sample contains errors of multiple categories, it will be counted in all these categories.

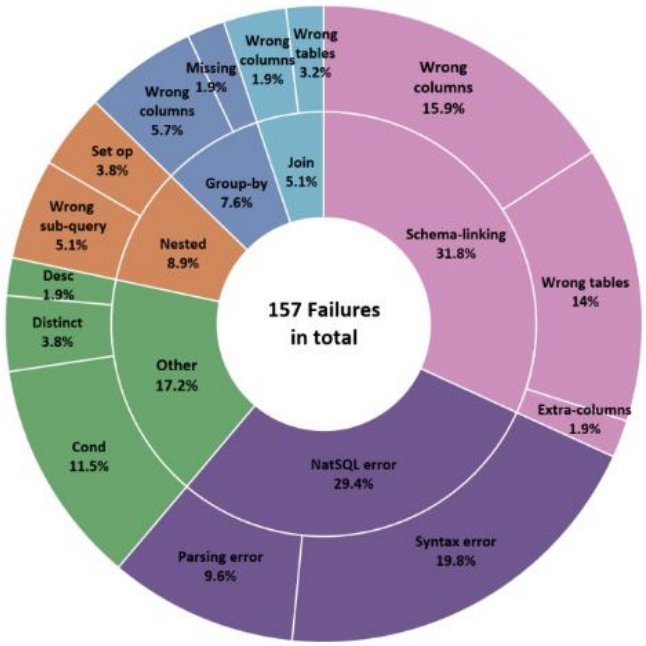

Figure 2. Error Distributions SFT CodeLlama-34b on dev set

Table 6. Comparative analysis of various types of errors generated by our method and the C3

Categories	Sub-categories	Ours %	C3 %
Schema-linking	Wrong tables	14	8
	Wrong columns	15.9	18

	Extra columns	1.9	3
Join	Wrong tables	3.2	17
	Wrong columns	1.9	5
Group-by	Wrong columns	5.7	8
	Missing	1.9	2
Nested	Set op	3.8	7
	Wrong sub-query	5.1	9
NatSQL error	Parsing error	9.6	-
	Syntax error	19.7	-
Other	Cond	11.5	15
	Distinct	3.8	5
	Desc	1.9	3

By analyzing the data in Figure 2 and Table 6, we can draw the following conclusions.

Due to the imperfection of NatSQL, a special category "NatSQL error" appears in our errors, which is divided into two subcategories, "Parsing error" and "Syntax error". The "Parsing error" is caused by the imperfection of the existing NatSQL parser [15]. The "Syntax error" is due to the imperfect design of NatSQL, which makes some SQL statements unable to be represented by NatSQL. There were 46 "NatSQL error" errors, accounting for 4.4% of the entire Spider development set. This ratio is consistent with the description in [15]. "NatSQL error" accounts for 29.4% of all errors, which is a high percentage, indicating that the definition and parsing of NatSQL needs to be further improved in order to further improve performance.

Of all the error categories, "schema-linking" dominates. Since our research focuses on the complex mapping of natural language and SQL queries, we do not specifically deal with "schema-linking", which is understandable that such errors occur more often.

Compared with C3, the proportion of "Join" and "Nested" errors is significantly reduced, which should be attributed to the strong expression ability of NatSQL in nested queries and multi-table joins, and can better handle some complex SQL queries.

Compared to C3, the proportion of "group-by" errors is slightly reduced. The reason for this is that the root cause of this type of error is a misunderstanding of the semantics of the problem, resulting in an error in field selection, which is not something that NatSQL can solve.

Compared with C3, the proportion of "Other" errors is reduced, and the reason for this is that CoT is used to decompose tasks, which reduces the difficulty of generating SQL statements.

In short, the proportion of "structural errors" has decreased a lot, proving the value of using NatSQL.

4.2 Significance of CoT information in the sample

From the above ablation results, it can be seen that when the learning sample does not have CoT information (-w/o CoT in examples), the performance of Hard and Extra difficulty is significantly reduced (about 10%). The dataset Spider-CG does not include fine-tuning (-w/o Spider-CG), and the performance is also reduced in Hard and Extra difficulties (2%-5%). The comparison between the two shows that learning samples with CoT information is more valuable than increasing the number of learning samples. Here is a question for further research: in the context of fine-tuning the model to recognize NatSQL, it is not enough to use only 7000 samples with CoT information from the Spider-SS dataset, so how many samples are needed?

From the data of SFT Llama-2-CHAT-13b in Table 5, it can be seen that it is significantly more sensitive to learning samples with or without CoT information, that is, since Llama-2-CHAT-13b is not pre-trained in SQL queries, the CoT information in learning samples is more valuable for it to learn NatSQL. Here's a hypothesis to be confirmed: the CoT information in the sample can speed up the learning of the model, and it can also effectively reduce the number of learning samples.

In the ablation studies described above, the contrast between -w/o few-shot and -w/o CoT in examples is an interesting topic about how the application of CoT information can lead to better results. The results are clear, and the CoT information placed in the sample is more helpful for the performance improvement. How to automatically generate learning samples with CoT information is an unsolved problem. It is very difficult to have a sample and then add CoT information to it. Our future research direction is to automatically generate CoT information first, and then generate the corresponding learning samples.

5. Related Work

Early text-to-SQL methods were represented by sequence-to-sequence architectures. IRNet [1] utilizes a bidirectional LSTM neural architecture to encode natural language questions and uses a self-attention mechanism to encode the database schema representation. Models such as RAT-SQL [33] and RASAT [34] use the self-attention mechanism to improve the integration method of database schema information, capture its relationship with natural language questions, and reduce the difficulty of semantic parsing. RESDSQL [6] proposes a rank-enhanced coding and skeleton-aware decoding framework for decoupling schema linking and skeleton parsing. Its encoder is injected by the most relevant schema items, which can ease the schema linking effort during SQL parsing. Its decoder first generates the skeleton and then generates the actual SQL query, which can effectively help SQL parsing. Graph neural networks have also been tried to mine the relationship between natural language questions and database schemas [35,48].

Recently, the text-to-SQL field has benefited from recent LLM methodological innovations and has made significant progress. Many prompting techniques have been used to further extend the capabilities of LLMs, such as Chain of Thought [7], Least-to-Most [11]. However, these general-purpose prompting methods have not achieved a leading performance in text-to-SQL tasks, so task decomposition and prompt optimization methods have become the focus of research. DIN-SQL [5] reduces the difficulty of generation by breaking down text-to-SQL tasks into subtasks, and completes schema linking and self-correction with advanced prompting approaches. DAIL-SQL [18] comprehensively uses the similarity of the problem and SQL query to select similar few-shot demonstrations. When DAIL-SQL calculates the similarity of the problem, it first blocks the domain-specific vocabulary and then calculates the Euclidean distance from the example problem. When calculating SQL similarity, SQL queries are predicted first, and then the similarity between queries is calculated. With this approach, the few-shot examples are guaranteed good similarity with both questions and SQL queries. MAC-SQL [36] adopts a multi-agent collaboration scheme, which decomposes the text-to-SQL task and hands it over to three agents (Selector, Decomposer, and Refiner) for collaboration. The Selector filters the data table based on the user problem to reduce the length of the context, the Decomposer breaks down the user problem into subproblems and provides solutions, and the Refiner verifies the correctness of the SQL and fixes the defective SQL. C3 [31] is a ChatGPT-based zero-shot approach that consists of three key components: Clear Prompting, Calibration with Hints, and Consistent Output. The first two components are essentially prompt optimization, and the third component is used to achieve self-consistency, i.e., the generated SQL queries are voted on based on the execution result. PET-SQL [37] proposes a two-stage framework, which first uses the few-shot method to prompt the LLM to generate preliminary results, parses the entities of the results, completes the extraction and compression of schema linking information, then uses the newly generated linked schema to instruct the LLM to generate the final SQL, and finally uses the cross-consistency test results.

DTS-SQL [38] introduces a two-stage fine-tuning approach that decomposes the task into two simpler tasks, separating schema linking and SQL generation, and employing two smaller LLMs with a parameter size of 7 billion each, enabling small open-source models to achieve performance comparable to large models. Blar-SQL [39] fine-tunes the Llama-2 and Code Llama open-source models, using the Llama-2 model for schema linking because it has better world understanding, and Code llama for SQL generation because it has a better programming background. At the same time, SQL generation not only relies on the schema linking output, it also uses the original schema in order to increase the chance of correcting potential errors. Open-SQL [40] combination of supervised fine-tuning, Chain-of-Thought, and few-shot learning, the performance of open-source LLMs for text-to-SQL tasks is significantly enhanced, and the potential of incorporating CoTs to enhance the performance of supervised fine-tuning in open-source LLMs is evaluated.

Intermediate representations have been used in many studies to generate SQL queries with complex structures [2,33,41,42]. SemQL [41] removes keywords JOIN ON, FROM, and GROUP BY, which are usually hard to find counterparts for in the text descriptions, and merges the HAVING and WHERE clauses. On this basis, NatSQL [3] removes the need for nested subqueries and set operators, uses only one SELECT clause. NatSQL makes schema linking easier by reducing the required number of schema items that are normally not mentioned in the NL question. The distinctive features of NatSQL have enabled it to be well applied in some studies [5,14,43,44]. In order to improve the ability of text-to-SQL models to understand NL problems, it is an attractive research direction to design appropriate IRs to better represent the mapping relationship between complex natural language and SQL queries.

6. Conclusion

In this paper, we validate two ideas. First of all, Text-to-NatSQL is a feasible method, and our experiments have shown that it shows excellent performance in complex scenarios such as nesting and set operations. Secondly, fine-tuning small and medium-sized LLMs using samples with Chain of Thought (CoT) can significantly increase the model's inference ability in complex scenarios. Although the inherent flaws of NatSQL make our model not acquire SOTA on Spider, our experiments show that both ideas are promising research directions.

References

[1] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. arXiv preprint arXiv:1905.08205, 2019.

[2] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1653 – 1663, Brussels, Belgium. Association for Computational Linguistics.

[3] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver,John R. Woodward, John Drake, Qiaofu Zhang. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. arXiv preprint arXiv:2109.05153, 2019.

[4] Aseem Arora, Shabbirhussain Bhaisaheb, Harshit Nigam, Manasi Patwardhan, Lovekesh Vig, Gautam Shroff. Adapt and Decompose: Efficient Generalization of Text-to-SQL via Domain Adapted Least-To-Most Prompting. arXiv preprint arXiv:2308.02582, 2023.

[5] Mohammadreza Pourreza and Davood Rafiei. Din-SQL: Decomposed in-context learning of text-to-SQL with self-correction. arXiv preprint arXiv:2304.11015,2023.

[6] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 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. 13067 – 13075.

[7] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. arXiv preprint arXiv:2201.11903, 2022.

[8] Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. CoRR, abs/2210.09261.

[9] Jason Wei, Yi Tay, and Quoc V Le. Inverse scaling can become u-shaped. arXiv preprint arXiv:2211.02011, 2022.

[10] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. CoRR, abs/2210.11416.

[11] Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. CoRR, abs/2205.10625.

[12] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. CoRR, abs/2210.03350.

[13] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. CoRR, abs/2210.02406.

[14] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. CoRR, abs/2302.04761.

[15] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver. Measuring and Improving Compositional Generalization in Text-to-SQL via Component Alignment. Findings of the Association for Computational Linguistics: NAACL 2022, pages 831 – 843. July 10-15, 2022 ©2022 Association for Computational Linguistics.

[16] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. CoRR, abs/2212.10071.

[17] Ping Yu, Tianlu Wang, Olga Golovneva, Badr AlKhamissy, Gargi Ghosh, Mona T. Diab, and Asli Celikyilmaz. ALERT: adapting language models to reasoning tasks. CoRR, abs/2212.08286.

[18] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation. arXiv preprint arXiv:2308.15363, 2023.

[19] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261,2022.

[20] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. CoRR, abs/2212.08410.

[21]Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. arXiv preprint arXiv:2301.12726,2023.

[22] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, DongxuWang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and et al. 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. 3911 – 3921.

[23] Yujian Gan, Xinyun Chen, and Matthew Purver. Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021. 8926 – 8931.

[24] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. Towards Robustness of Text-to-SQL Models against Synonym Substitution. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021. 2505 – 2515.

[25] Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-Grounded Pretraining for Text-to-SQL. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021. 1337 – 1350.

[26] Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020. 396 – 411.

[27] Ruoxi Sun, Sercan Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha,Pengcheng Yin, and Tomas Pfister. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL. CoRR abs/2306.00739 (2023). arXiv preprint arXiv:2306.00739,2023.

[28] Haoyang Li,Jing Zhang,Hanbing Liu,Ju Fan,Xiaokang Zhang,Jun Zhu,Renjie Wei,Hongyan Pan,Cuiping Li,Hong Chen. CodeS: Towards Building Open-source Language Models for Text-to-SQL. arXiv preprint arXiv:2402.16347,2024.

[29] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021. 9895 – 9901.

[30] Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu and Kai Yu. ACT-SQL: In-Context Learning for Text-to-SQL with Automatically-Generated Chain-of-Thought. arXiv preprint arXiv: 2310.17342,2023.

[31] X. Dong, C. Zhang, Y. Ge, Y. Mao, Y. Gao, J. Lin, D. Lou. C3: Zero-shot text-to-SQL with ChatGPT. arXiv preprint arXiv:2307.07306, 2023.

[32] Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. A comprehensive evaluation of ChatGPT’s zero-shot Text-to-SQL capability. CoRR abs/2303.13547 (2023). arXiv preprint arXiv:2303.13547,2023.

[33] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Ratsql:Relation-aware schema encoding and linking for text-to-sql parsers. arXiv preprint arXiv:1911.04942, 2019.

[34] Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang,Quanshi Zhang, and Zhouhan Lin. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-SQL. arXiv preprint arXiv:2205.06983, 2022.

[35] Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. Sadga: Structure-aware dual graph aggregation network for text-to-SQL. Advances in Neural Information Processing Systems, 34:7664 – 7676, 2021.

[36] B. Wang, C. Ren, J. Yang, X. Liang, J. Bai, L. Chai, Z. Yan, Q.-W. Zhang, D. Yin, X. Sun, and Z. Li. Macsql:A multi-agent collaborative framework for text-to-SQL. arXiv preprint arXiv:2312.11242, 2024.

[37] Z. Li, X. Wang, J. Zhao, S. Yang, G. Du, X. Hu, B. Zhang, Y. Ye, Z. Li, R. Zhao, and H. Mao. Pet-SQL: A prompt-enhanced two-stage text-to-SQL framework with cross-consistency. arXiv preprint arXiv:2403.09732, 2024.

[38] Mohammadreza Pourreza,Davood Rafiei. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. arXiv preprint arXiv:2402.01117,2024.

[39] JOSé MANUEL DOMíNGUEZ,BENJAMíN ERRAZURIZ, PATRICIO DAHER. Blar-SQL: Faster, Stronger, Smaller NL2SQL. arXiv preprint arXiv:2401.02997,2024.

[40] Xiaojun Chen,Tianle Wang,Tianhao Qiu,Jianbin Qin,Min Yang. Open-SQL Framework: Enhancing Text-to-SQL on Open-source Large Language Models. arXiv preprint arXiv:2405.06674,2024.

[41] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. arXiv preprint arXiv:1905.08205, 2019.

[42] Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cícero Nogueira dos Santos, and Bing Xiang. Learning contextual representations for semantic parsing with generation-augmented pre-training. CoRR, abs/2012.10309.

[43] Aseem Arora, Shabbirhussain Bhaisaheb, Harshit Nigam, Manasi Patwardhan, Lovekesh Vig, Gautam Shroff. Adapt and Decompose: Efficient Generalization of Text-to-SQL via Domain Adapted Least-To-Most Prompting. arXiv preprint arXiv:2308.02582,2023.

[44] Wang, Bailin, et al. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020.

[45] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416,2022.

[46] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. arXiv preprint arXiv:2212.00193,2022.

[47] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. arXiv preprint arXiv:2212.10071,2022.

[48] Yu, T.; Yasunaga, M.; Yang, K.; Zhang, R.;Wang, D.; Li, Z.; and Radev, D. R. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. arXiv preprint arXiv:1810.05237,2018.