DataArc Homepage

SQL-R1: Training Natural Language to SQL Reasoning Model By Reinforcement Learning

Peixian Ma [1,2], Xialie Zhuang [1,3], Chengjin Xu [1,4], Xuhui Jiang [1,4], Ran Chen [1], Jian Guo [1]

[1] IDEA Research, International Digital Economy Academy

[2] The Hong Kong University of Science and Technology (Guangzhou),

[3] University of Chinese Academy of Sciences,

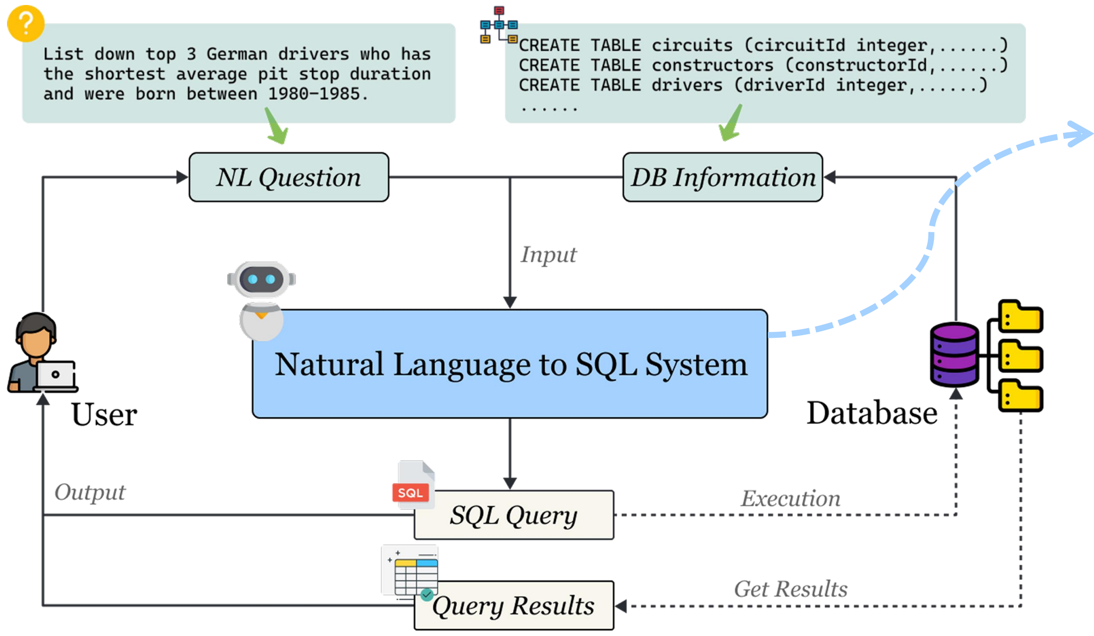
[4] DataArc Tech Ltd.

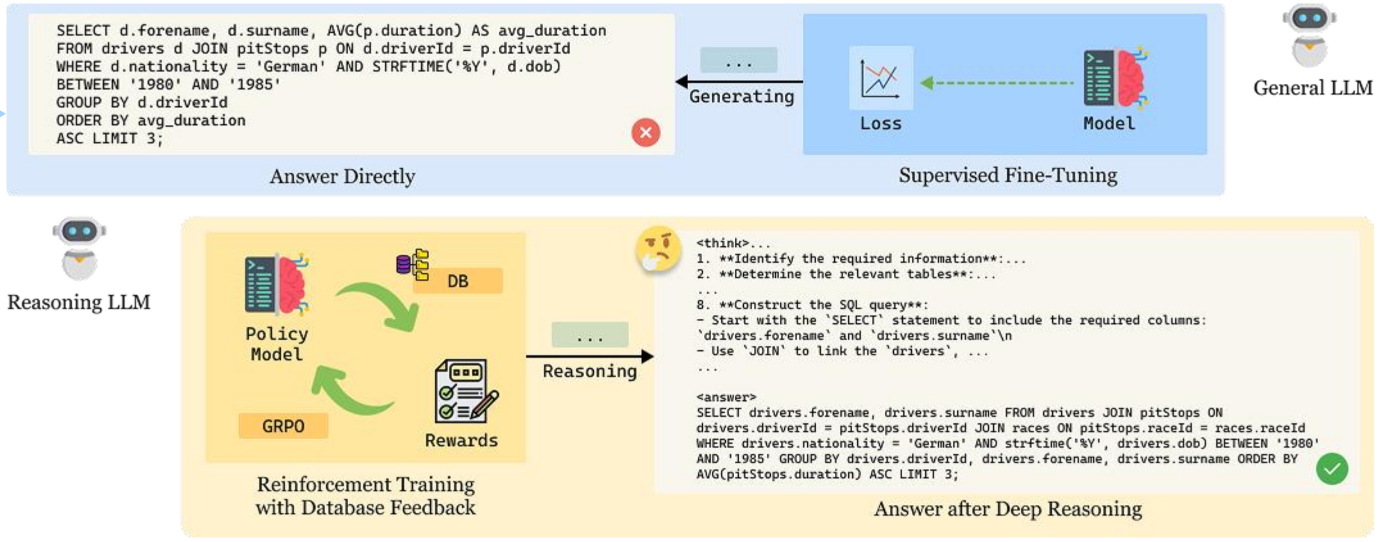
Corresponding author: Chengjin Xu (xuchengjin@idea.edu.cn)

The 39th Conference on Neural Information Processing Systems

Motivation & Research Questions

NEURAL INFORMATION PROCESSING SYSTEMS





General LLMs for NL2SQL

Difficult to ***think*** and ***reason*** the relationship between ***Schema*** and ***Sematic***

Capability gap between different LLMs and ***privacy security*** concerns

Supervised Fine-Tuning LLMs for NL2SQL

Strong dependence on ***training data*** from target database

Instability in ***domain adaptation*** and ***generalization*** in new database

Multi-component Workflow for NL2SQL

High ***token consumption*** and ***computation latency***

Strong reliance on the capabilities of ***foundation model*** (e.g., Claude, GPT)

Research Questions & Contributions

1. Can we design a specific reinforcement learning algorithm for the NL2SQL task and successfully train a NL2SQL reasoning model?
2. For the RL-based NL2SQL reasoning model, do we need to perform a specific form of cold start on it?
3. Can we deploy sustainable data engineering for training robust and efficient NL2SQL reasoning models?

SQL-R1

Data Processing for Training

Training Dataset for RL: SynSQL-Complex-5k

Randomly sampled ***5K data pairs*** from SynSQL-2.5M, whose complexity are ***Complex***

Training Dataset for SFT: SynSQL-200k

Sampling ***200k data*** from the SynSQL-2.5M with each level comprising ***50k data pairs***.

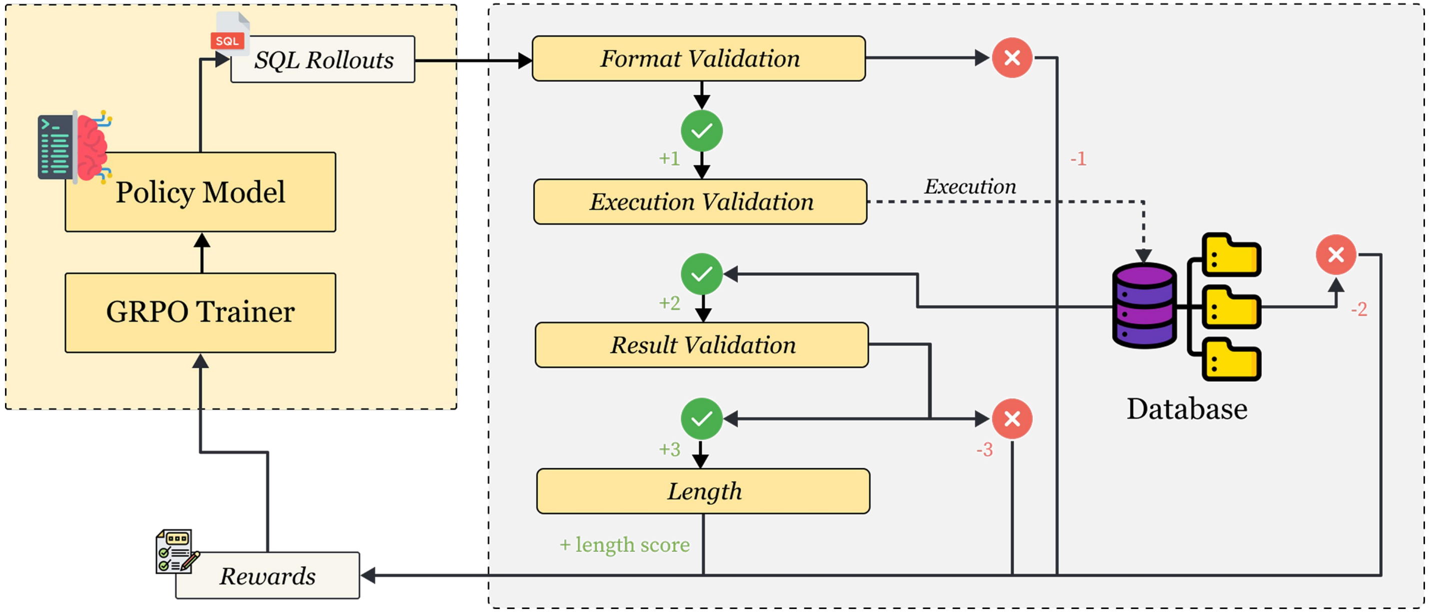
Reward Function for GRPO

$$S_f = \begin{cases} 1, & \text{if format is correct} \\ -1, & \text{if format is incorrect} \end{cases}$$

$$S_e = \begin{cases} 2, & \text{if SQL candidate is executable} \\ 0, & \text{if format is incorrect} \\ -2, & \text{if SQL candidate is not executable} \end{cases}$$

$$S_r = \begin{cases} 3, & \text{if query result is correct} \\ 0, & \text{if format is incorrect or SQL candidate is not executable} \\ -3, & \text{if query result is incorrect} \end{cases}$$

$$S_l = \begin{cases} 0.5 \times S_{tl} + S_{al}, & \text{if query result is correct and } len_{response} <= \texttt{MAX LENGTH} \\ 0.5 + S_{al}, & \text{if query result is correct and } len_{response} > \texttt{MAX LENGTH} \\ 0, & \text{other cases} \end{cases}$$



Main Results

NEURAL INFORMATION PROCESSING SYSTEMS

Table 1: Execution accuracy (%) of different NL2SQL methods on Spider and BIRD benchmark.

NL2SQL Method	Base Model	Candidate Selection	Spider (Dev)	Spider (Test)	BIRD (Dev)
CodeS [14]	CodeS-15B	-	84.9	79.4	57.0
DTS-SQL [13]	Deepseek-Coder-7B	-	85.5	84.4	55.8
CHESS [7]	Deepseek-Coder-33B	-	-	87.2	61.5
Alpha-SQL [29]	Qwen2.5-Coder-7B	Self-Consistency	84.0	-	66.8
SQL-o1 [30]	Qwen2.5-Coder-7B	Self-Consistency	84.7	85.1	66.7
OmniSQL [22]	Qwen2.5-Coder-7B	Self-Consistency	85.5	88.9	66.1
DeepRetrieval [31]	Qwen2.5-Coder-7B	-	-	76.1	56.0
Reasoning-SQL [32]	Qwen2.5-Coder-14B	Self-Consistency	81.4	-	65.3
C3-SQL [10]	GPT-3.5-Turbo	Self-Consistency	82.0	82.3	-
DIN-SQL [8]	GPT-4	-	82.8	85.3	-
DAIL-SQL [16]	GPT-4	Self-Consistency	83.6	86.2	54.8
MAC-SQL [9]	GPT-4	Self-Consistency	86.8	82.8	59.4
SuperSQL [33]	GPT-4	Self-Consistency	84.0	87.0	58.5
MCTS-SQL [34]	GPT-4o	-	88.7	86.6	69.4
OpenSearch-SQL [35]	GPT-4o	Self-Consistency	-	87.1	69.3
CHASE-SQL [36]	Gemimi-1.5-Pro	-	-	87.6	73.0
SQL-R1 (Ours)	**Qwen2.5-Coder-3B**	**Self-Consistency**	**78.1**	**78.9**	**54.6**
SQL-R1 (Ours)	**Qwen2.5-Coder-7B**	**Self-Consistency**	**87.6**	**88.7**	**66.6**
SQL-R1 (Ours)	**Qwen2.5-Coder-14B**	**Self-Consistency**	**86.7**	**88.1**	**67.1**

Table 2: Execution accuracy (%) on Spider-DK, Spider-Syn, Spider-Realistic benchmark.

NL2SQL Method	Base Model	Spider-DK	Spider-Syn	Spider-Realistic
SENSE [37]	CodeLlama-7B	77.9	72.6	82.7
ROUTE [38]	Llama3-8B	74.6	77.4	80.9
SQL-o1 [30]	Llama3-8B	78.7	72.6	82.7
OmniSQL [22]	Qwen2.5-Coder-7B	77.8	69.6	78.0
SQL-PaLM [39]	PaLM-2	67.5	70.9	77.4
PURPLE [40]	GPT-4	75.3	74.0	79.9
SQL-R1 (Ours)	**Qwen2.5-Coder-3B**	**70.5**	**66.4**	**71.5**
SQL-R1 (Ours)	**Qwen2.5-Coder-7B**	**78.1**	**76.7**	**83.3**
SQL-R1 (Ours)	**Qwen2.5-Coder-14B**	**79.3**	**78.5**	**86.2**

Table 3: Execution accuracy (%) of different complexity levels on BIRD-Dev dataset.

NL2SQL Method	Base Model	Simple	Moderate	Challenging	All
CodeS [14]	Codes-15B	65.8	48.8	42.4	58.5
DAIL-SQL [16]	GPT-4	63.0	45.6	43.1	55.9
SuperSQL [33]	GPT-4	66.9	46.5	43.8	58.5
SQL-R1 (Ours)	**Qwen2.5-Coder-7B**	**72.1**	**60.8**	**51.0**	**66.6**
SQL-R1 (Ours)	**Qwen2.5-Coder-14B**	**72.4**	**59.7**	**56.5**	**67.1**

Table 6: Execution accuracy (%) of different models on Spider2.0 SQLite subset.

Model	Accuracy (%)
Qwen2.5-Coder-7B	2.2
GPT-4o	15.6
Deepseek-V3	15.6
OmniSQL-7B	10.4
SQL-R1-7B	**20.0**

SQL-R1 achieves superior performance on ***Spider*** and ***BIRD*** benchmarks, outperforming methods based on general LLMs and multi-component workflows. This confirms the feasibility of ***RL algorithm for NL2SQL*** tasks.

Main Results

NEURAL INFORMATION PROCESSING SYSTEMS

Table 4: Execution accuracy (%) of models with different cold start strategy. The *Reasoning Instruction* column is applied represents the instruction applied SFT process.

Model	SFT Data	Reasoning Instruction	Spider (Dev)	Spider (Test)	BIRD (Dev)
Qwen2.5-Coder-7B	-	✗	77.4	79.4	58.2
Qwen2.5-Coder-7B	SynSQL-200K	✓	82.7	83.3	57.0
Qwen2.5-Coder-14B	-	✗	87.0	88.0	66.1
OmniSQL-7B [22]	SynSQL-2.5M	✗	85.5	88.9	66.1
OmniSQL-14B [22]	SynSQL-2.5M	✗	86.2	88.3	65.9
SQL-R1 + Qwen2.5-Coder-7B	-	✗	84.5	86.1	**63.1**
SQL-R1 + Qwen2.5-Coder-7B	SynSQL-200K	✓	84.7	86.4	**59.2**
SQL-R1 + Qwen2.5-Coder-14B	-	✗	86.7	88.1	**67.1**
SQL-R1 + OmniSQL-7B	SynSQL-2.5M	✗	87.6	88.7	**66.6**
SQL-R1 + OmniSQL-14B	SynSQL-2.5M	✗	86.4	87.6	**66.6**

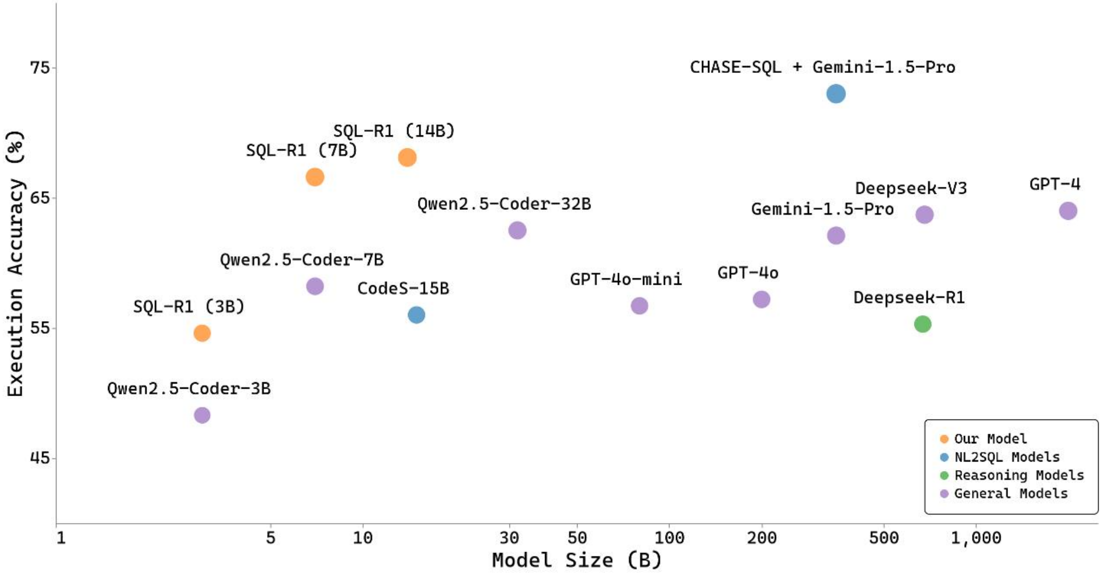


Figure 2: Performance and model scale on the BIRD-Dev dataset.

SFT cold-start training is ***not universally essential*** for RL-based NL2SQL models. Its effectiveness is contingent upon the origin and volume of the training data.

Synthetic data engineering plays a pivotal role in augmenting the reasoning capabilities of NL2SQL models, especially when utilized in conjunction with RL.

Ablation Studies

Table 5: Ablation study of reward components on BIRD-Dev dataset.

Reward Function	**Accuracy (%)**
Qwen2.5-Coder-7B	58.2
$S_f + S_e + S_r + S_l$	**63.1**
- w/o S_f (Format Score)	60.4 (↓ 2.7)
- w/o S_e (Execution Score)	60.7 (↓ 2.4)
- w/o S_r (Result Score)	62.4 (↓ 0.7)
- w/o S_l (Length Score)	61.0 (↓ 2.1)

Table 10: Ablation study of different setting of reward components.

Reward Type	**Set to 1.0**	**Increase by 50%**
S_e (Execution Score)	60.0	-
S_r (Result Score)	61.3	-
S_l (Length Score)	59.7	60.6

Table 9: Ablation study of database value retrieval for RL training on BIRD-Dev dataset.

Base Model	**Use Database Values**	**Accuracy (%)**
Qwen2.5-Coder-7B	-	58.2
SQL-R1 + Qwen2.5-Coder-7B	✗	61.9
SQL-R1 + Qwen2.5-Coder-7B	✓	**63.1**

Case Study

Question: Tally the toxicology element of the 4th atom of each molecule that was carcinogenic.

Let's break down the question and the database schema step by step to generate the appropriate SQL query.

1. **Identify the Relevant Tables and Columns:**

- We need to identify the molecules that are carcinogenic (label = '+').
- We need to find the 4th atom of each molecule.
- We need to determine the element of the 4th atom.
- We need to tally the toxicology element.

2. **Filter Molecules by Carcinogenicity:**

- We will use the 'molecule' table to filter molecules where 'label = '+''.

3. **Find the 4th Atom of Each Molecule:**

- We will use the 'atom' table and the 'connected' table to find the 4th atom. The 4th atom's 'atom_id' will be the 7th character of the 'atom_id' column (since 'atom_id' follows the pattern 'TR000_1').

4. **Determine the Element of the 4th Atom:**

- We will use the 'atom' table to get the 'element' of the 4th atom.

5. **Tally the Toxicology Element:**

- We will use the 'element' column from the 'atom' table to tally the toxicology element.

6. **Construct the SQL Query:**

- We will join the 'molecule', 'atom', 'connected', and 'atom' tables to get the required information.

Here is the SQL query that addresses the question:

```sql
SELECT a.element FROM molecule m JOIN atom a ON m.molecule_id
= a.molecule_id JOIN connected c ON a.atom_id = c.atom_id
WHERE m.label = '+' AND a.atom_id LIKE 'TR000_4' GROUP BY
a.element; ✗
```

Explanation:

- 'SELECT a.element': Selects the element of the 4th atom.
- 'FROM molecule m': Joins with the 'molecule' table to filter molecules by their label.
- 'JOIN atom a ON m.molecule_id = a.molecule_id': Joins with the 'atom' table to get the atoms of each molecule.
- 'JOIN connected c ON a.atom_id = c.atom_id': Joins with the 'connected' table to find the connections between atoms.
- 'WHERE m.label = '+' AND a.atom_id LIKE 'TR000_4'': Filters the atoms to only include the 4th atom ('atom_id' starting with 'TR000_4').
- 'GROUP BY a.element': Groups the results by the element to tally the number of each element.

This query will provide the tally of the toxicology element of the 4th atom of each carcinogenic molecule.

Question: Tally the toxicology element of the 4th atom of each molecule that was carcinogenic.

<think>

To answer the given question, we need to follow these steps:

1. **Identify the Molecules that are Carcinogenic**: According to the question, molecules that are carcinogenic have a label of '+' in the 'molecule' table.
2. **Identify the 4th Atom of Each Molecule**: The 4th atom is identified by the 'atom_id' where the substring of 'atom_id' starting from the 7th character and having a length of 1 is '4'.
3. **Retrieve the Element of the 4th Atom**: Once we have identified the 4th atom, we need to retrieve the element associated with it from the 'atom' table.
4. **Tally the Toxicology Element**: This step is not explicitly clear from the question, but it seems to imply that we need to count or list the elements of the 4th atoms of the carcinogenic molecules. Let's break down the steps into SQL queries:

Step 1: Identify the Molecules that are Carcinogenic

```sql
SELECT molecule_id FROM molecule WHERE label = '+';
```

Step 2: Identify the 4th Atom of Each Molecule

To find the 4th atom, we need to filter the 'atom_id' where the substring starting from the 7th character and having a length of 1 is '4'. We can use the 'SUBSTR' function for this.

```sql
SELECT atom_id FROM atom WHERE SUBSTR(atom_id, 7, 1) = '4';
```

Step 3: Retrieve the Element of the 4th Atom

We need to join the 'atom' table with the 'connected' table to find the 4th atom and then retrieve its element.

```sql
SELECT atom.element FROM atom JOIN connected ON atom.atom_id
= connected.atom_id WHERE SUBSTR(atom.atom_id, 7, 1) = '4';
```

Step 4: Tally the Toxicology Element. To tally the toxicology element, we need to count or list the unique elements of the 4th atoms. We can use the 'GROUP BY' clause to achieve this.

```sql
SELECT atom.element FROM atom JOIN connected ON atom.atom_id
= connected.atom_id WHERE SUBSTR(atom.atom_id, 7, 1) = '4'
GROUP BY atom.element;
```

Combining All Steps: We need to combine all the steps into a single query that filters the carcinogenic molecules, identifies the 4th atom, retrieves its element, and tallies the unique elements.

```sql
SELECT atom.element FROM atom JOIN connected ON atom.atom_id
= connected.atom_id JOIN molecule ON atom.molecule_id =
molecule.molecule_id WHERE SUBSTR(atom.atom_id, 7, 1) = '4'
AND molecule.label = '+' GROUP BY atom.element;
```

</think>

<answer>

```sql
SELECT atom.element FROM atom JOIN connected ON atom.atom_id
= connected.atom_id JOIN molecule ON atom.molecule_id =
molecule.molecule_id WHERE SUBSTR(atom.atom_id, 7, 1) = '4'
AND molecule.label = '+' GROUP BY atom.element; ✓
```

</answer>

Conclusion

NEURAL INFORMATION PROCESSING SYSTEMS

Explicit NL2SQL Reasoning Model: We propose SQL-R1, a NL2SQL reasoning model trained on a few NL2SQL data~(e.g., 5K) currently, which can achieve 88.6% and 66.6% accuracy on the leading benchmark Spider-Test and BIRD, respectively, and can output a detailed explicit reasoning process.

Training Strategy for NL2SQL Reasoning Model: We extensively explored the impact of cold-start training on SQL-R1, developing a training strategy that integrates SFT and RL. Our findings highlight the strategy of using synthetic data to enhance model performance and robustness, offering key insights into optimizing NL2SQL reasoning model training.

DataArc Homepage

Thanks !

Homepage: *https://dataarctech.github.io/SQL-R1/*

arXiv: *https://arxiv.org/abs/2504.08600*

Github: *https://github.com/DataArcTech/SQL-R1*

Huggingface: *https://huggingface.co/MPX0222forHF/SQL-R1-7B*