R^3 : “This is My SQL, Are You With Me?” A Consensus-Based Multi-Agent System for Text-to-SQL Tasks

Hanchen Xia[*][🍑][🍋], **Feng Jiang**[*][🍋], **Naihao Deng**[🥑], **Cunxiang Wang**[🍋], **Guojiang Zhao**[🫐]
Rada Mihalcea[🥑], **Yue Zhang**[🍋]
[🍑] School of Mathematical Science, Shanghai Jiao Tong University
[🍋] School of Engineering, Westlake University
[🥑] University of Michigan [🫐] Carnegie Mellon University

Abstract

Large Language Models (LLMs) have demonstrated strong performance on various tasks. To unleash their power on the Text-to-SQL task, we propose $\boldsymbol{R}^3$ (Review-Rebuttal-Revision), a consensus-based multi-agent system for Text-to-SQL tasks. $\boldsymbol{R}^3$ outperforms the existing single LLM Text-to-SQL systems as well as the multi-agent Text-to-SQL systems by 1.3% to 8.1% on Spider and Bird. Surprisingly, we find that for Llama-3-8B, $\boldsymbol{R}^3$ outperforms chain-of-thought prompting by over 20%, even outperforming GPT-3.5 on the development set of Spider.

1 Introduction

Text-to-SQL, the task of converting natural language to SQL queries, enables non-technical users to access databases with natural language (Deng et al., 2022; Katsogiannis-Meimarakis and Koutrika, 2023). Recently, Large Language Models (LLMs) have made significant progress on various tasks (Touvron et al., 2023; OpenAI, 2023).

Although researchers have proposed various methods to enhance the reasoning abilities of LLMs (Wei et al., 2022; Yao et al., 2023; Besta et al., 2024), However, they are still facing challenges with Text-to-SQL tasks (Li et al., 2023; Hong et al., 2024). The LLM-based multi-agent system leverages collective intelligence from a group of LLMs and have achieved exceptional performance across various tasks (Park et al., 2023; Hong et al., 2023; Xu et al., 2023), but little work explores using them on Text-to-SQL. The existing multi-agent Text-to-SQL system first decomposes the task into multiple subtasks which are then accomplished step-by-step by agents (Wang et al., 2023). While achieving remarkable performances, such a decomposition-based system necessitates extensive manual prompt engineering and logic design.

We propose $\boldsymbol{R}^3$, a consensus-based multi-agent system for Text-to-SQL tasks. The proposed system draws inspiration from the peer-review mechanism, featuring one agent as the SQL-writer and several reviewers automatically generated by the LLM. Once the generated SQL query is tested to be executable, the system will step into a review process, where we use the execution results to guide the SQL-writer and reviewers to refine the SQL. Through rounds of “review”, “negotiation or rebuttal”, and “revision”, SQL-writer and reviewers will finally achieve consensus and deliver a solution with collective agreement (see Figure 1).

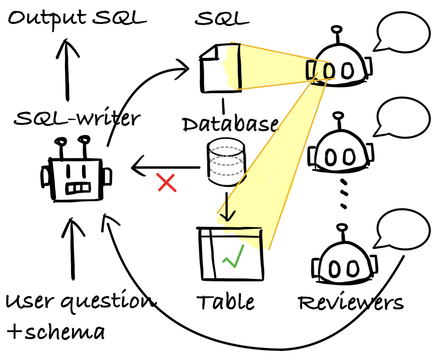

Figure 1: $\boldsymbol{R}^3$ Architecture. n reviewer agents, each with distinct characteristics, are created to review the generated SQL and its execution result. The process continues until the master node (SQL-writer) and the other nodes reach a consensus, at which point the system outputs the final SQL.

We test $\boldsymbol{R}^3$ on the popular Spider and Bird benchmarks. $\boldsymbol{R}^3$ outperforms the existing single LLM as well as the multi-agent Tex-to-SQL systems by 1.3% to 8.1% on Spider and Bird. Surprisingly, we find that for Llama-3-8B, $\boldsymbol{R}^3$ outperforms chain-of-thought prompting by over 20%, even outperforming GPT-3.5 on the Spider-Dev set. Our contributions can be summarized as follows:

1. To the best of our knowledge, $\boldsymbol{R}^3$ is the first Text-to-SQL system to use the execution result for SQL refinements, and the first Text-to-SQL system to equip agents with memory sequences

to enhance SQL generation.

2. $\boldsymbol{R}^3$ offers a consensus-based multi-agent system for Text-to-SQL tasks. Using very succinct prompts, it achieves strong performance compared to other systems. In addition, it effectively helps open-source LLMs such as Llama-3-8B on SQL generation.

3. We provide a detailed error analysis of $\boldsymbol{R}^3$ on the existing Text-to-SQL benchmarks, shedding light on future research on the Text-to-SQL task.

2 Architecture

SQL-Writer (SW). We task SW agents to: (1) compose the original SQL query based on the user question and database schema; (2) ensure that the SQL query is executable, and correct it when errors occur; (3) respond to reviewer agents' feedback and revise the SQL query accordingly. Specifically, we prompt SW agent through Prompt 1 in Appendix A.9. For task (1), we feed the Prompt 1 to SW agent directly. Given a user question x and the database schema $\mathcal{S}$, task (1) can be formalized as:

$$y = \texttt{LLM}(x, \mathcal{S}),$$

where y is the generated SQL query. For (2) and (3), we maintain a dialogue history $\mathcal{H}$ initially set to $\mathcal{H} = [(x, \mathcal{S}), y]$. Specifically, if an error e occurs, we append e to the history $\mathcal{H} \leftarrow \mathcal{H} + e$ and get y' through:

$$y' = \texttt{LLM}(\mathcal{H}).$$

We then concatenate y' with the history $\mathcal{H} \leftarrow \mathcal{H} + y'$. In addition, considering the length limitation of LLMs' context window, we truncate the history $\mathcal{H}$ when the prompt is longer than the context limit.

Reviewers (REs). We generate the reviewer agent's professions using an LLM (see Prompt 3 in Appendix A.9) based on the database schema and the content of the SQL query, for instance, "Senior Database Engineer specialized in writing various clauses" and "Data Analyst in the automotive industry", etc. We incorporate these professions in the system prompt for the reviewer agent to make them focus on different aspects of the SQL query. These reviewer agents are prompted to provide their professional comments based on the database schema, the user's question, the predicted SQL, and its execution result in the table format.

Overall Architecture. After several rounds of "negotiation" between the SQL-writer and reviewer agents, we decide whether there is a consensus by checking if the SQL-writer agent generates the same SQL query as in the previous round. When there is a consensus, we terminate the negotiation loop and output the final SQL query. Algorithm 1 depicts the overall process of our system.

```
Given $x$ (user question), $\mathcal{S}$ (schema)
$y = \texttt{LLM}(x, \mathcal{S})$
$\mathcal{H} = [(x, \mathcal{S}), y]$
$i = 0$
$j = 0$
while $i <= \text{MaxReviewTurns}$ do
    while $j <= \text{MaxDebugTurns}$ do
        Try:
            $\mathcal{T} = \texttt{Database}(y)$
            break
        Except Exception as $e$:
        $j \leftarrow j + 1$
            $\mathcal{H} \leftarrow \mathcal{H} + e; \quad y' = \texttt{LLM}(\mathcal{H})$
            $\mathcal{H} \leftarrow \mathcal{H} + y'$
    end
    $r = \texttt{LLM}(x, \mathcal{S}, y, \mathcal{T})$
    $\mathcal{H} \leftarrow \mathcal{H} + r; \quad y'' = \texttt{LLM}(\mathcal{H})$
    $\mathcal{H} \leftarrow \mathcal{H} + y''$
    if $y == y''$ then
        break
    else
        $y \leftarrow y''$
    end
    $i \leftarrow i + 1$
end
```

Algorithm 1: $\boldsymbol{R}^3$-Loop

Appendix A.9 provides the detailed prompts we use in $\boldsymbol{R}^3$. In addition, we incorporate:

1. Program of Thoughts (PoT) (Chen et al., 2023) to prompt the SQL-writer agent to generate Python code before SQL query (see Prompt 2 in Appendix A.9). Therefore, the agents may leverage Python in their reasoning process for better SQL query generation.

2. k-shots example selection based on similarity of the user question embeddings. Specifically, when our system infers the SQL query in the test set, we select the k most similar use questions and their corresponding SQL queries from the training set (k-shots) and use them for in-context learning.

3 Experiments and Results

We conduct experiments on two cross-domain Text-to-SQL benchmarks, Spider and Bird detailed in Table 5 in Appendix A.1. We employ test-suite execution evaluation[1] (Zhong et al., 2020), the standard evaluation protocol for Spider, and the official SQL

[1]github.com/taoyds/test-suite-sql-eval

Model	Method	SD	ST	BD
GPT-3.5	- (Li et al., 2023)	72.1	-	37.22
	C3 (Dong et al., 2023)	81.8	82.3	-
	MAC (Wang et al., 2023)	80.6	75.5	50.56
	$\boldsymbol{R}^3$ 5-shot	81.4	81.1	52.15
GPT-4	DAIL (Gao et al., 2023)	83.6	86.6	-
	PET (Li et al., 2024)	82.2	87.6	-
	DIN (Pourreza and Rafiei, 2023)	82.8	85.3	50.72
	MAC (Wang et al., 2023)	86.8	82.8	59.39
	$\boldsymbol{R}^3$ 5-shot	**88.1**	**89.9**	**61.80**

Table 1: Execution accuracy across various models and methods. We use the GPT-3.5-Turbo in our experiment. "SD", "ST", "BD" represent Spider-Dev, Spider-Test, Bird-Dev, respectively. For detailed description of baseline methods mentioned above, see Appendix A.2.

Model	Method	SD	ST
GPT-3.5	- (Li et al., 2023)	72.1	-
Llama-3-8B	CoT	52.1	53.5
	$\boldsymbol{R}^3$ 0-shot	72.8	72.6
Llama-3-70B	$\boldsymbol{R}^3$ 0-shot	79.7	80.3

Table 2: Execution accuracy comparison when we equip Llama-3 models with $\boldsymbol{R}^3$on Spider-Dev ("SD") and Spider-Test ("ST").

execution accuracy evaluation for Bird[2]. Table 1 compares $\boldsymbol{R}^3$'s performance with existing baseline methods when we employ different foundation LLMs. Our best performed system achieves 88.1%, 89.9%, and 61.8% on the Spider-Dev, Spider-Test, and Bird-Dev respectively, surpassing the existing multi-agent Text-to-SQL systems. In addition, we test our system with open-source Llama-3 models on Spider and report the results in Table 2. To our surprise, with the help of $\boldsymbol{R}^3$, zero-shot Llama-3-8B outperforms GPT-3.5 performance reported by Li et al. (2023) on Spider-Dev set. This demonstrates the effectiveness of our proposed $\boldsymbol{R}^3$ system.

3.1 Ablation Studies

We conduct an ablation study on the impact of CoT, PoT with one or three reviewer agents in the discussion and report the results in Table 3. The results in Table 3 show that the n-Reviewer(s) Loop (nR-Lp) plays a major role in performance improvement, with the 3R-Lp configuration significantly outperforming the 1R-Lp setup. The proposed $\boldsymbol{R}^3$ system achieves a 10.54% improvement over the baseline GPT-4 + CoT. We provide the statistical significant test for these results in Appendix A.4. Appendix A.3 provides a sensitivity analysis of the impacts of the k value in k-shots.

[2]bird-bench.github.io/

	GPT-3.5-Turbo		GPT-4	
	Spider	Bird	Spider	Bird
CoT	78.2	37.22	79.7	53.30
PoT	78.5	36.96	80.0	54.61
1R-Lp + CoT	78.3	44.13	82.3	57.89
1R-Lp + PoT	79.3	46.35	85.4	58.34
$\boldsymbol{R}^3$: 3R-Lp + PoT	**81.4**	**52.15**	**88.1**	**61.80**

Table 3: Ablation Studies on Spider-Dev and Bird-Dev (Execution Accuracy). The 1-Reviewer Loop (1R-Lp) represents that only one reviewer agent participates in the discussion, while the 3-Reviewers Loop (3R-Lp) represents three in the dicussion, which is also the default configuration of $\boldsymbol{R}^3$. We conduct all the experiments here under the 5-shot setting.

We conducted case studies on 244 instances from the Spider-Dev dataset where the CoT fail but $\boldsymbol{R}^3$succeed when combined with Llama-3-8B. The findings are as follows:

1. **Corrected non-executable SQL queries, 51%.** The LLM equipped with memory module (see Section 2) excels at correcting non-executable SQL queries.

2. **Refinement based on reviewers' comments, 44%.** The nR-Lp functions as an enhanced Self-Consistency (SC) (Wang et al., 2022). On the one hand, it avoids the hallucinations caused by high temperatures (Renze and Guven, 2024). On the other hand, the nR-Lp considers feedback from all agents, unlike the voting process in SC, which consistently disregards minority opinions.

3. **Refinement based on the output table, 27.5%.** LLMs may not experts in SQL writing, but they are full-skilled data readers. The information that reviewers observe from execution results greatly assists the SQL-writer in refining the SQL.

3.2 Error Analysis

In total, GPT-4+$\boldsymbol{R}^3$ fails to generate the gold SQL queries for 123 instances in Spider-Dev. Table 4 shows the error case distribution for our system on Spider-Dev (more in Appendices A.6 and A.7). Note that though we have spotted issues with the