MAGIC: Generating Self-Correction Guideline for In-Context Text-to-SQL

Arian Askari[1*], **Christian Poelitz**[2], **Xinye Tang**[3]

[1]Leiden University
[2]Microsoft Research Cambridge, UK,
[3]Microsoft Redmond
{cpoelitz, xinye.tang}@microsoft.com, a.askari@liacs.leidenuniv.nl

Abstract

Self-correction in text-to-SQL is the process of prompting large language model (LLM) to revise its previously incorrectly generated SQL, and commonly relies on manually crafted self-correction guidelines by human experts that are not only labor-intensive to produce but also limited by the human ability in identifying all potential error patterns in LLM responses. We introduce MAGIC, a novel **m**ulti-**ag**ent method that automates the creat**i**on of the self-**c**orrection guideline. MAGIC uses three specialized agents: a manager, a correction, and a feedback agent. These agents collaborate on the failures of an LLM-based method on the training set to iteratively generate and refine a self-correction guideline tailored to LLM mistakes, mirroring human processes but without human involvement. Our extensive experiments show that MAGIC's guideline outperforms expert human's created ones. We empirically find out that the guideline produced by MAGIC enhances the interpretability of the corrections made, providing insights in analyzing the reason behind the failures and successes of LLMs in self-correction.

Datasets —
https://huggingface.co/datasets/microsoft/MAGIC

Introduction

Converting natural language questions to SQL database queries, known as text-to-SQL, serves as a pivotal component for empowering non-expert data analysts in extracting desired information from relational databases using natural language (Qu et al. 2024). While large language models have shown a significant improvement in text-to-SQL and serve as state-of-the methods according to the leaderboards (Li et al. 2024c), they are prone to mistakes and even GPT4 has a notable accuracy gap of 30% within human (Li et al. 2023).

A solution for resolving the mistakes of LLM is the emerging concept of 'self-correction' (Madaan et al. 2023) that defines as the ability of LLMs in revising their previous mistakes under the hypothesis that recognizing errors is easier than avoiding them (Gou et al. 2024; Madaan et al. 2023). Self-correction in context of text-to-SQL is the process of prompting an LLM to revise its previously incorrectly generated SQL (Zhang et al. 2024; Chen et al. 2023).

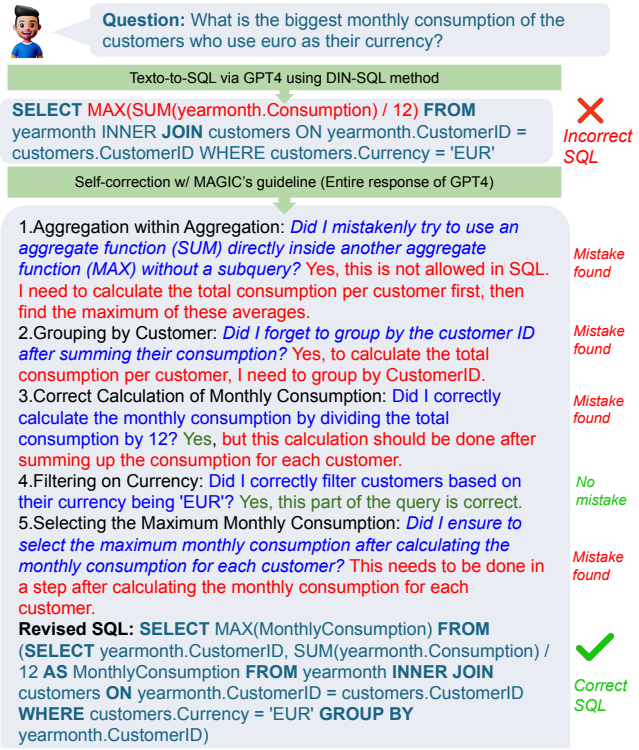


Figure 1: Example of self-correction using automatically generated guidelines by MAGIC

While self-correction for LLMs has been widely studied in variety of tasks including code generation with GPT4 (Shypula et al. 2024; Pan et al. 2023; Kamoi et al. 2024), it has been relatively unexplored in text-to-SQL. The common method for self-correcting in existing few-shot LLM based methods that achieve state-of-the-art effectiveness is designing a self-correction guideline by human and prompting it to GPT-4 together with its initial generated SQL (Pourreza and Rafiei 2023). This self-correction guideline commonly engineered based on the train set where an expert human analyzes common mistakes of the LLM on the training data and design a guideline to prevent common mistakes. This process is a time-consuming and challenging task that is limited to the ability of humans in identifying all the mistake patterns

[*] Work done during an internship at Microsoft.

that exist in the LLM responses (Pourreza and Rafiei 2023; Wang et al. 2024a; Talaei et al. 2024).

We propose MAGIC, a novel multi-agent self-correction guideline generation for text-to-SQL, that generates an effective self-correction guideline that outperforms human-written guidelines and yields to improving effectiveness of strong few-shot LLM based text-to-SQL methods. Similar to humans who engineer the self-correction guideline and apply it to LLMs during inference, with MAGIC, we first tackle guideline generation and then utilize the generated guideline during inference. By iterating over the incorrect generated SQLs by the initial text-to-SQL method, MAGIC automatically generates the self-correction guideline that is tailored to the mistakes of the initial system. Next, in inference, the self-correction guideline of MAGIC will be integrated to the initial text-to-SQL method, assisting it in preventing its common mistakes.

Unlike previous studies that analyze self-correction on a simple method that uses GPT4 as its backbone or a low-effective open-source language model (Zhang et al. 2024), we employ a strong and effective open-source method based on GPT4, DIN-SQL (Pourreza and Rafiei 2023), as our initial text-to-SQL system aiming to compare our automatically generated guideline with an effective expert human-written guideline. Our experiments demonstrate that the generated guideline by MAGIC leads to self-explanatory self-correction, where the LLM asks itself questions before self-correcting. Figure 1 illustrates an example of where the MAGIC guideline assisted GPT4 to self-correct its previously incorrectly generated SQL. The process is interpretable as the LLM first starts with asking itself and then perform self-correction.

Our contributions are as follows: (i) We establish the task of self-correction guideline generation for text-to-SQL, and introduce MAGIC, a novel multi-agent method, that automates the generation of self-correction guideline and outperforms human-written guideline; yielding to improve effectiveness of a strong few-shot LLM based text-to-SQL method. (ii) We systematically analyze the impact of ours and existing self-correction methods to perform self-correction across different scenarios: Correcting incorrect queries; non-executable queries; and all queries. (iii) We reproduce all the baselines utilizing the same version of GPT-4 across all the experiments, and provide a comprehensive comparable insight on self-correction in text-to-SQL. (iv) We publish all code to reproduce our experiments as open source.[1]

We found that there can be a significant gap in terms of effectiveness when utilizing different versions of GPT-4. For instance, our replication of DIN-SQL (Pourreza and Rafiei 2023) on development set of BIRD (Li et al. 2023) achieves 56.52 compared to the original report of 50.72 in the paper, which is about 6 points higher than the reported number. This is our motivation for replicating and reproducing all the reported baselines using same GPT-4 and avoid copying numbers from previous papers.

[1]https://github.com/microsoft/SynQo

Related Work

LLMs for Text-to-SQL. Converting natural language questions into SQL queries, known as text-to-SQL, has been an active area of research within both the natural language processing (NLP) and database communities for many years (Zelle and Mooney 1996; Guo et al. 2019). Recently, text-to-SQL has benefited from the promising effectiveness of Large Language Models (LLMs) (Talaei et al. 2024; Pourreza and Rafiei 2023). Early methods utilized the zero-shot in-context learning capabilities of LLMs for SQL generation (Rajkumar, Li, and Bahdanau 2022). Building on this, subsequent models have improved LLM performance through task decomposition and techniques such as Chain-of-Thought (CoT) (Wei et al. 2022), including but not limited to models like DAIL-SQL (Gao et al. 2023), MAC-SQL (Wang et al. 2024a), C3 (Dong et al. 2023), self-consistency (Wang et al. 2022), and least-to-most prompting (Zhou et al. 2023). In addition, there are studies focusing on fine-tuning LLMs for text-to-SQL (Li et al. 2024d; Gao et al. 2023; Li et al. 2024b). We choose DIN-SQL (Pourreza and Rafiei 2023) as the initial text-to-SQL system due to two reasons: (i) high effectiveness and available open-source code; (ii) its self-correction component is a human expert written guideline based on prior mistakes of the LLM making it a suitable baseline for comparing the automatically self-correction guideline generated by MAGIC with expert human-written guideline. However, we would emphasize that our method is independent of how the in-context learning method is designed and is adaptable to the output of any method, as we only require the predicted SQL of the initial text-to-SQL system for tackling self-correction.

Self-correction in text-to-SQL. Pan et al. (2024) and (Kamoi et al. 2024) provide an extensive overview of research on self-correction in a variety of domains up to 2024. Here, we focus on self-correction in Text-to-SQL, where there has been relatively limited study. Self-debugging (Chen et al. 2023) generates additional explanations on the question and initial predicted SQL, which are then provided to an LLM for self-correction. DIN-SQL (Pourreza and Rafiei 2023) designs a human-written self-correction guideline and revises all the initially generated SQLs according to this guideline. Self-consistency is based on generating several candidates and employing a voting process, which has been integrated into Text-to-SQL by DAIL-SQL (Gao et al. 2023). The refiner component of MAC-SQL (Wang et al. 2024a) uses execution errors as signals for revising the SQL. To the best of our knowledge, we are the first study to address self-correction in Text-to-SQL by generating a self-correction guideline.

LLM-based Agents. LLM-based agents have been a promising area of study in both academic and industry communities for an extended period (Wang et al. 2024b), leading to extensive research exploring autonomous agents based on LLMs such as AutoGPT (Gravitas 2024), OpenAgents (Xie et al. 2023), and AutoGen (Wu et al. 2023). However, there are limited studies leveraging this concept for text-to-SQL, with MAC-SQL being the only multi-agent method focusing on addressing the Text-to-SQL task through a new multi-agent collaborative framework (Wang et al. 2024a). Inspired by the literature on multi-agent LLMs, in MAGIC, we employ a

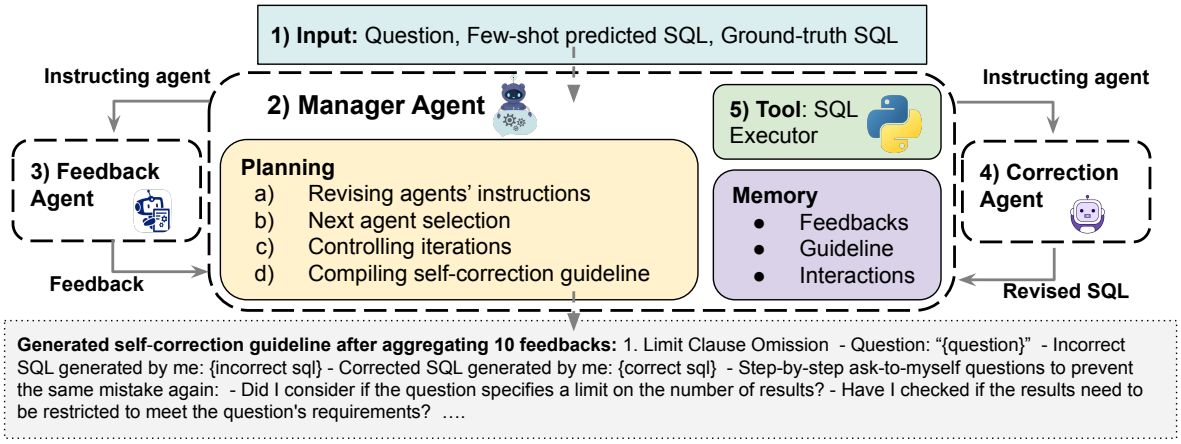


Figure 2: Illustration of our proposed method, MAGIC.

multi-agent collaborative framework that iteratively analyzes the failures of predicted SQLs by a text-to-SQL method and automatically generates a self-correction guideline tailored to the method's mistakes. To the best of our knowledge, there is no prior work in the literature that employs multi-agent methods for designing a self-correction guideline in text-to-SQL or other domains.

Self-Correction Guideline Generation

Tasks definition. Given a set of questions in natural language alongside the corresponding database schema and the initially generated incorrect SQL queries by a text-to-SQL system, the task of self-correction guideline generation is to generate a self-correction guideline that is tailored to the LLM's mistakes and prevents it from repeating them. This can be done by a human expert, an LLM, or a collaboration between a human expert and an LLM. The typical approach for self-correction involves writing a guideline for the Large Language Model manually by humans based on the LLM's common mistakes (Pourreza and Rafiei 2023; Talaei et al. 2024). The data used for guideline generation must not be shared with the evaluation data to prevent overfitting the guidelines on the test errors.

MAGIC

We tackle the task of generating self-correction guideline for text-to-SQL systems, focusing on the failures of an initial method, denoted as M, using the training dataset. A failure for M occurs when the predicted SQL is either not executable or its execution result differs from that of the ground truth SQL, denoted as s^{gt}. In this context, the predicted SQL that is either not executable or has a differing execution result is called an incorrect SQL, denoted as s'.

Our method, MAGIC, consists of three agents illustrated in Figure 2: the manager, feedback, and correction agents. These agents interact iteratively over the failures to generate the self-correction guideline. We describe this process in detail in the following. Please note that due to space limitations, the paper includes summarized prompts. We provide the complete prompts and the full automatically generated self-correction guideline by MAGIC in the technical appendix.

Feedback-correction cycle. Given each question, s^{gt}, and s', the manager agent starts a feedback-correction iteration cycle. At each iteration of this cycle, the manager requests the feedback agent for an explanation of the mistakes in s' by comparing it to s^{gt} (steps 2 to 3 in Figure 2). Next, the manager agent integrates the feedback received from the feedback agent to interact with the correction agent. The manager requests the correction agent to revise s' according to the provided feedback (step 3 to 4 in Figure 2). Subsequently, the correction agent generates a new revised SQL. At this stage, the manager identifies whether the revised SQL by the correction agent successfully leads to identical results with s^{gt}. To do so, the manager uses its SQL executor tool to execute the correction agent's revised SQL (step 5 in Figure 2). The manager ends this cycle if the stopping criteria are met. The stopping criteria are either revising s' successfully by the correction agent or reaching the maximum number of iterations. We set 5 as maximum number of iteration.

Revising agents' instruction. In the first iteration of the feedback-correction cycle, the manager uses two predefined prompts designed for its interactions with the feedback and correction agents, as illustrated in Figures 3 and 4 respectively. If the correction agent cannot successfully revise s' in the first iteration, the manager begins revising these predefined prompts. The prompt template of manager for revising the predefined prompts of feedback and correction agents are illustrated in Figure 5. We found empirically that this step plays an important role for the manager to adapt its interaction with agents. Manager revise the agent's instruction based on the previous response of the agent, previous prompt that is used for interacting with the agent, question, s', and s^{gt}.

Guideline generation. Given each successful revision by the correction agent, the manager stores the corresponding successful feedback that led to this in its memory. The manager then aggregates these stored feedbacks to generate the self-correction guideline batch-by-batch using a predefined prompt for guideline generation (Figure 6). Each batch con-

```
"question": "{question}",
"Correct SQL": "{Correct SQL}",
"Incorrect SQL": "{Incorrect SQL}". The mistakes are:
```

Figure 3: The predefined prompt for interaction of manager with feedback agent.

```
- Schema Overview: {schema}
- Question: {question}
- Predicted SQL: "'sql {Incorrect SQL} "'
- Expert Human Feedback: {feedback}
```

Figure 4: The predefined prompt for interaction of manager with correction agent.

sist of k feedbacks. In our preliminary experiments, we determined that a feedback batch size of 10 is optimal. In the first batch, the manager begins without any pre-existing guideline and generate an initial guideline. For subsequent batches, the manager updates the already generated guideline. This guideline generation process is triggered at the end of a feedback-correction cycle if the manager has accumulated a new batch of successful feedbacks that has not yet been utilized for guideline generation. The self-correction guideline generated by our proposed method, MAGIC, automatically and from scratch, is presented in the technical appendix.

Efficiency. Previous approaches to craft self-correction guidelines needed extensive expert human work. With MAGIC, we can efficiently generate, empirically shown better guideline, in less than 2 hours.

Agents access to information

Although the manager agent has access to all the available information during generation of self-correction guideline, the agents has limited access to the information about the task. During the iterations, the correction agent consistently receives only s'. This ensures that a successful self-correction can indicate that the feedback could effectively cover the mistakes that exist in the s'. This is important since the goal is generating a guideline that can prevent from initial text-to-SQL system errors. As the result, the self-correction agent should focus on correcting the initial system prediction.

```
Manager, review the following prompt for the agent and generate a revised prompt.
Agent description: {Agent description}.
Previous output and prompt that was not useful: {agent_outputs[-1]} and {agent_prompts[-1]}
Revise prompt (Return the entire prompt):
```

Figure 5: The prompt template of manager for revising the predefined prompts of feedback or correction agent.

```
# Recent mistakes to add to Guideline:
{batch_of_successful_feedbacks}
# Updated Guideline (Return the entire guideline):
```

Figure 6: The prompt of manager for self-correction guideline generation.

Experimental Setup

Datasets. The Spider (Yu et al. 2018) dataset is a collection of 10,181 questions and 5,693 unique complex SQL queries across 200 databases in 138 domains, with each domain featuring multiple tables. It is divided into training, development, and test sets with 8,659, 1,034, and 2,147 examples, respectively, across 146, 20, and 34 distinct databases, ensuring no overlap between sets. Queries are classified into four difficulty levels based on complexity factors such as SQL keywords, nested subqueries, and use of column selections and aggregations. The BIRD dataset (Li et al. 2023) comprises 12,751 unique question-SQL pairings across 95 relative large databases (33.4 GB) in 37 professional domains like blockchain and healthcare. BIRD introduces external knowledge as an additional resource for generating accurate SQL queries to bring more complexity into the task.

Metrics. We employ two key metrics from existing literature: Execution Accuracy (EX) and Valid Efficiency Score (VES). Execution Accuracy measures the correctness of a predicted SQL query by comparing its execution output with that of the ground truth SQL query. A SQL query is deemed correct if its execution results match those of the ground truth, allowing for a precise assessment of the model's performance given the possibility of multiple valid SQL queries for a single question. The Valid Efficiency Score evaluates the efficiency of executing the generated SQL queries, focusing on both their accuracy and execution time, but only considers queries that produce correct results, i.e., those whose execution outcomes are consistent with the reference query. This dual-metric approach provides a comprehensive evaluation of the model's ability to generate both accurate and efficient SQL queries. Given our primary focus on enhancing accuracy, we prioritize Execution Accuracy (EX) as the main metric in our experiments, while also reporting the Valid Efficiency Score (VES) in selected cases to offer additional insights. Furthermore, we do not report VES results for the SPIDER dataset because computing VES for this dataset is less meaningful. The execution times for SPIDER are so short that any time differences often round to zero, rendering VES comparisons inconclusive. This is why previous studies also omit VES for SPIDER dataset (Pourreza and Rafiei 2023; Talaei et al. 2024). The SPIDER dataset's databases contain significantly fewer entries compared to the BIRD dataset, where the databases are, on average, 140 times larger. This substantial difference in scale makes VES a more relevant metric for BIRD, where efficiency considerations are more significant.

Baselines. We reproduce DIN-SQL using the publicly available original implementation (Pourreza and Rafiei 2023). For self-correction, we employ an extensive set of base-

DIN-SQL	Scenario of Self-Correction Application											
	Correcting incorrect SQLs				SQLs with Exec error				All SQLs			
	S	M	C	T	S	M	C	T	S	M	C	T
W/o self-correction	63.14	49.03	38.19	56.52	63.14	49.03	38.19	56.52	63.14	49.03	38.19	56.52
Self-correction w/o guideline												
Self-Debugging (Chen et al. 2023)	63.57	50.11	39.67	57.24	63.35	49.03	39.58	56.78	63.03	49.25	38.89	56.58
Self-Consistency (Wang et al. 2022)	64.56	50.79	40.09	58.08	63.24	49.03	38.19	56.58	64.00	49.68	37.50	57.17
Multiple-Prompt (Lee et al. 2024)	64.22	50.75	40.02	57.86	63.35	49.03	38.19	56.65	63.68	**50.11**	39.58	57.30
Self-correction w/ guideline												
Human Expert G, MAC-SQL (Wang et al. 2024a)	63.18	49.19	38.30	56.60	63.20	49.03	38.19	56.58	47.80	46.22	35.30	46.14
Human Expert G, DIN-SQL (Pourreza and Rafiei 2023)	64.32	50.32	39.58	57.76	63.58	49.03	38.89	56.98	62.49	48.17	37.50	55.80
MAGIC G **(ours)**	**65.84**†	**51.61**†	**40.28**†	**59.13**†	**63.69**†	**50.15**†	**39.59**†	**57.32**†	**65.75**†	49.46	**41.67**†	**58.55**†

Table 1: Effectiveness results in terms of Execution Accuracy (EX) on the DEV set of the BIRD dataset. All the experiments have been reproduced by us. 'S', 'M', 'C', and 'T' refer to different levels of difficulty: Simple, Medium, Challenging, and Total, respectively. The baseline for Natural Language to SQL conversion is DIN-SQL and 'Human Expert G' refers to self-correction guideline (prompt). 'MAGIC refers to the guideline that is automatically generated by our proposed method, MAGIC, using the train set of BIRD dataset. Significance is shown with † for MAGIC compared to both 'Human Expert' G baseliens. Statistical significance was measured with a paired t-test ($p < 0.05$) with Bonferroni correction for multiple testing.

lines categorized into guideline-dependent and guideline-independent methods. For guideline-independent methods, we reproduce self-debugging (Chen et al. 2023) based on the written prompts in the paper. For self-consistency (Wang et al. 2022; Gao et al. 2023), we generate 20 SQL queries given the SQL generation prompt of DIN-SQL instead of generating only one SQL and select the final SQL based on voting. In voting, we execute all SQL queries and select the most frequently returned result's SQL. If one result is produced by various SQL queries, we select the most efficient one by comparing them against each other. For the Multiple-Prompt baseline, we follow the approach in (Lee et al. 2024) by reordering candidate tables in the prompt and generating up to 20 different combinations, employing a voting mechanism similar to our self-consistency implementation. For human-expert baselines, we utilize the self-correction guideline from DIN-SQL (Pourreza and Rafiei 2023), written by human experts, and the revision guideline from MAC-SQL (Wang et al. 2024a), that is also written by human experts and are designed specifically for correcting SQL queries with execution errors. However, we also adopt their approach for correcting incorrect SQL queries and for correcting all initially predicted SQL queries, regardless of whether they are correct, similar to the scenarios analyzed in (Chen et al. 2023) and (Pourreza and Rafiei 2023) respectively. Our method, MAGIC, is designed to generate self-correction guidelines for text-to-SQL tasks without relying on fine-tuned models. We exclude baselines that depend on fine-tuning (Talaei et al. 2024; Pourreza and Rafiei 2024; Li et al. 2024a) to ensure fair comparisons and to highlight MAGIC's effectiveness in scenarios where fine-tuning is not feasible due to high computational costs.

Results

This section addresses the following research questions (RQs): (i) RQ1: What is the effectiveness of MAGIC's self–correction guideline compared to the existing state-of-the-art baselines for self-correction in text-to-SQL, particularly across different error scenarios and datasets? (ii) RQ3: What is the impact of manager agent intervention on reducing the number of iterations and increasing the number of corrected SQLs?

Main results (RQ1). Table 1 and 2 present the results of self-correction with the guidelines generated by MAGIC on the development set of BIRD and SPIDER, respectively. Overall, the guidelines generated by our proposed method outperform all the baselines in terms of total effectiveness measured by execution accuracy and demonstrate significant improvements in self-correction performance across the BIRD and SPIDER datasets, outperforming self-correction guidelines written by expert humans. For example, MAGIC's guidelines improve the DIN-SQL (Pourreza and Rafiei 2023) baseline from 56.52 to 59.13 in terms of execution accuracy and outperform the self-correction guideline of (Wang et al. 2024a) as well, as presented in Tables 5 and 1 respectively.

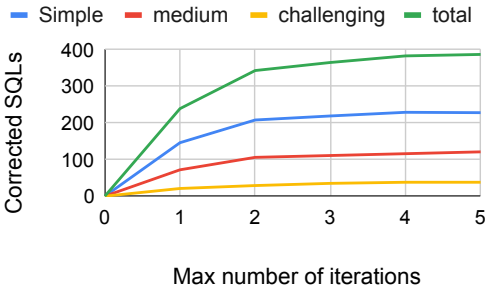
We experiment with self-correction in three scenarios, as shown in Table 1: (i) correcting incorrect SQLs, assuming the availability of a correctness oracle. This setup is common in recent studies focusing on self-correction in text-to-SQL (Chen et al. 2023; Zhang et al. 2024). (ii) SQLs with execution errors: this scenario is common in self-correction where there is no available correctness oracle, but the goal is to avoid generating incorrect and non-executable SQL (Wang et al. 2024a). (iii) All SQLs: this setup is common in methods where self-correction is part of a multi-step model, employing a self-correction guideline on all predictions (Pourreza

Correction Method	EX
w/o self-correction	78.62
Self-correction w/o guideline	
Self-Consistency (Wang et al. 2022)	81.64
Multiple-Prompt (Lee et al. 2024)	80.22
Self-correction w/ guideline	
Human's G (Wang et al. 2024a)	81.15
Human's G (Pourreza and Rafiei 2023)	80.35
MAGIC's G (Ours)	**85.66**

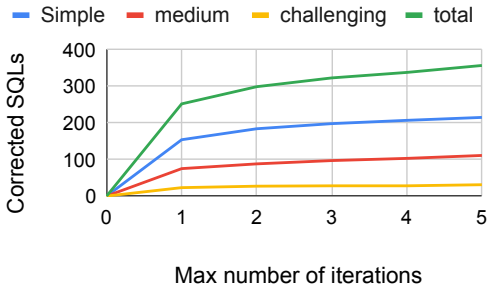
Table 2: Effectiveness results for self-correcting incorrect SQL queries on the SPIDER development set. We compared MAGIC's generated guideline (MAGIC's G) against two top self-correction baselines: self-consistency and multiple-prompt, as well as expert human self-correction guidelines.

	# of aggregated batch of feedbacks				
	0	1	5	10	39 (All)
EX	56.52	57.4	58.8	59.13	59.13

Table 3: Impact of number of aggregation levels of feedbacks on the effectiveness of generated self-correction guideline on the development set of BIRD dataset.



a) MAGIC w/ manager.



b) MAGIC w/o manager.

Figure 7: Analyzing impact of maximum number of iterations on the total number of corrected SQLs, that were initially incorrect, with and without manager being involved in MAGIC. The analysis is conducted on the train set of the BIRD dataset.

and Rafiei 2023).

We find that applying self-correction to all predicted SQLs, as in the human-expert written guideline for self-correction in (Wang et al. 2024a), reduces effectiveness. Intuitively, when self-correction is applied for all predicted SQLs, there is a risk that the LLM changes its previously correct prediction to an incorrect one, as suggested by previous work (Li et al. 2024e). Therefore, designing a proper self-correction guideline is crucial, or it is safer to apply self-correction where a correctness oracle can identify whether the initial response is correct or not. This oracle can be a human user who minimally interacts with the system and only determines whether the executed query results meet their expectations. Regarding the "SQLs with execution errors" scenario, only 46 queries predicted by the initial text-to-SQL system (Pourreza and Rafiei 2023) are non-executable, which limits the effectiveness of self-correction for these cases, with several methods achieving the same effectiveness (49.03) for medium query difficulty. It is noteworthy that although we have not attempted to combine MAGIC with other baselines, as our focus is on analyzing different self-correction methods rather than integrating them, some baselines could potentially enhance MAGIC if combined, such as Self-consistency (Wang et al. 2022) and Multiple-Prompt (Lee et al. 2024).

Impact of Feedback Quantity (RQ2). Table 3 presents the results of the generated self-correction guideline effectiveness by MAGIC, where different numbers of batches of feedback are aggregated for generating the guideline. We found that after aggregating 10 batches of feedback, each batch consist of 10 feedbacks, the generated guideline by MAGIC outperforms existing self-correction baselines, including those with guidelines written by humans. This shows the efficiency of process of MAGIC for self-correction guideline generation. Furthermore, we found that questions for which MAGIC cannot provide feedback to self-correct are often controversial. For example, when an additional aggregation function column is reported alongside the main column name that should be reported, such as 'SELECT city, MAX(population) FROM cities' where the MAX(population) should not be included in the predicted query according to the ground truth.

Feedback generation effectiveness (RQ3). Figure 7 presents an analysis of the number of corrected SQL queries with and without the inclusion of the manager agent, by evaluating the performance of the MAGIC framework across a range of maximum iteration counts used as stopping criteria, from one to five. When the manager agent is excluded, the feedback agent controls the maximum iteration count and utilizes the SQL executor tool to compare the execution results of the revised SQL with the ground truth SQL. This comparison underscores the significant role of the manager agent in correcting a larger number of incorrect SQL queries within fewer iterations and ultimately correcting more SQL queries when both setups reach the maximum of five iterations. It is important to note that excluding the manager agent from MAGIC is equivalent to the critic-refine cycle method (Kamoi et al. 2024), that has been employed in other tasks, such as code generation. However, unlike MAGIC, the critic-refine (Kamoi et al. 2024) does not produce any guidelines as a result of its feedback generation process and has not been applied to text-to-SQL tasks to the best of our knowledge.

	BIRD			Spider	
	EX ↑	VES ↑	AVG$_i$ ↓	EX ↑	AVG$_i$ ↓
MAGIC$_{\max_i=2}$					
w/o manager	73.01	68.29	1.17	81.40	1.11
w/ manager	78.94	80.22	**1.12**	86.52	1.05
MAGIC$_{\max_i=5}$					
w/o manager	78.60	79.58	2.87	83.28	2.22
w/ manager	**81.81**	**84.19**	2.41	**91.78**	**2.15**

Table 4: The effectiveness for generating feedbacks on the train set of BIRD dataset. The max_i and AVG_i refers to the maximum number of iterations and average number of iterations till stopping criteria triggers respectively.

Method	EX
Zero-shot GPT-4 (OpenAI 2023b)	40.18
Zero-shot GPT-4 + MAGIC G (Ours)	48.19
Few-shot CoT GPT-4 (Li et al. 2023)	45.66
Few-shot CoT GPT-4 (Li et al. 2023) + MAGIC G (Ours)	50.38
DIN-SQL (Pourreza and Rafiei 2023)	56.52
DIN-SQL (Pourreza and Rafiei 2023) + MAGIC G (Ours)	59.13
MAC-SQL (Wang et al. 2024a)	59.39
MAC-SQL (Wang et al. 2024a) + MAGIC G (Ours)	**61.92**

Table 5: The effectiveness of self-correction guideline generated by MAGIC across different methods.

We further analyze the impact of manager agent in terms of overall performance of self-correction considering all the queries. Table 4 measures the effectiveness of MAGIC in terms of execution accuracy (EX) to determine how many could be addressed correctly in total with the already corrected predictions by LLM and self-corrected ones with MAGIC. We also measure Valid Efficiency Score (VES) to determine whether the self-correction by MAGIC results in optimized SQLs compared to the ground truth. As it can be observed, MAGIC with the manager agent outperforms the exclusion of manager agent in all setups in terms of all metrics across the BIRD and SPIDER datasets. Results show that MAGIC with manager agent is able to generate more efficient SQLs compared to exclusion of manager agent, e.g., 84.19 vs. 79.58 in terms of VES for w/ manager and w/o manager on the BIRD dataset where the maximum number of iteration is set to 5. We manually analyzed the revised prompt by manager agent and found out during its interactions with the agents where it revises the prompts, the manager agent tends to add instructions for self-correction to optimize the efficiency of SQL, which might be the reason behind generating more efficient SQLs.

Discussion

Applicability to other methods. We analyze the applicability of the guideline generated by MAGIC based on the mistakes of the DIN-SQL (Pourreza and Rafiei 2023) method on three other methods that generate SQL in a single step: zero-shot GPT-4 (OpenAI 2023a), few-shot CoT GPT-4 (Li et al. 2023), and multi-agent (MAC-SQL (Wang et al. 2024a)) as shown in Table 5. As we can observe, MAGIC's guidelines improve the zero-shot GPT-4 baseline from 40.18 to 48.19, Few-shot CoT GPT-4 from 45.56 to 50.38, and multi-agent baseline from 59.39 to 61.92 in terms of execution accuracy, respectively.

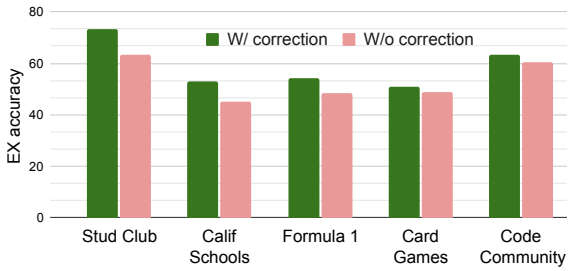


Figure 8: Analyzing impact of MAGIC's correction guideline across different databases of BIRD dataset.

Source of examples in guideline. We analyzed the relationship between the generated guideline and the feedback that contributed to its creation. Specifically, we examined whether any examples in the guideline directly mirrored instances from the feedback or outputs of the correction agent. Our findings indicate that no example was directly copied from the feedback to be used in the guideline. Instead, our empirical observations reveal that the manager agent tends to aggregate multiple pieces of feedback to construct new examples into the self-correction guideline.

Database analysis. Figure 8 illustrates the impact of the self-correction guidelines generated by MAGIC across different databases in the development set of the BIRD dataset. We found that while self-correction improves effectiveness across all databases, some databases benefit more from MAGIC's guidelines. This variability could be due to the capability of the LLM, the differing levels of challenging information in the databases, or other factors.

Conclusions

This paper presents a new perspective on self-correction in in-context learning for text-to-SQL translation. It proposes a novel method for generating self-correction guidelines, called MAGIC. The motivation behind this approach is to overcome the limitations of existing methods that generate self-correction guidelines by hand, a time-consuming task. Additionally, it addresses the important and costly task of automatically fixing incorrect SQL generated by humans. This work showcases the potential of leveraging LLMs to generate their own self-correction guidelines and highlights the significance of guideline generation in text-to-SQL. We emphasize the importance of improving self-correction methods in text-to-SQL and addressing them as a separate task. The findings of this study contribute to advancing the state-of-the-art in text-to-SQL translation, as our method can be applied to fix issues in any method and provide valuable insights for future research in this domain.

References

Chen, X.; Lin, M.; Schärli, N.; and Zhou, D. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Dong, X.; Zhang, C.; Ge, Y.; Mao, Y.; Gao, Y.; Lin, J.; Lou, D.; et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Gao, D.; Wang, H.; Li, Y.; Sun, X.; Qian, Y.; Ding, B.; and Zhou, J. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Gou, Z.; Shao, Z.; Gong, Y.; yelong shen; Yang, Y.; Duan, N.; and Chen, W. 2024. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *The Twelfth International Conference on Learning Representations*.

Gravitas, S. 2024. AutoGPT.

Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; and Zhang, D. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.

Kamoi, R.; Zhang, Y.; Zhang, N.; Han, J.; and Zhang, R. 2024. When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs. *arXiv preprint arXiv:2406.01297*.

Lee, D.; Park, C.; Kim, J.; and Park, H. 2024. MCS-SQL: Leveraging Multiple Prompts and Multiple-Choice Selection For Text-to-SQL Generation. *arXiv preprint arXiv:2405.07467*.

Li, B.; Luo, Y.; Chai, C.; Li, G.; and Tang, N. 2024a. The Dawn of Natural Language to SQL: Are We Fully Ready? *arXiv preprint arXiv:2406.01265*.

Li, H.; Zhang, J.; Liu, H.; Fan, J.; Zhang, X.; Zhu, J.; Wei, R.; Pan, H.; Li, C.; and Chen, H. 2024b. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *arXiv preprint arXiv:2402.16347*.

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Cao, R.; Geng, R.; Huo, N.; Zhou, X.; Ma, C.; Li, G.; Chang, K. C. C.; Huang, F.; Cheng, R.; and Li, Y. 2024c. BIRD-SQL Leaderboard.

Li, J.; Hui, B.; QU, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; Zhou, X.; Ma, C.; Li, G.; Chang, K.; Huang, F.; Cheng, R.; and Li, Y. 2023. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. 2024d. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Li, L.; Chen, G.; Su, Y.; Chen, Z.; Zhang, Y.; Xing, E.; and Zhang, K. 2024e. Confidence Matters: Revisiting Intrinsic Self-Correction Capabilities of Large Language Models. *CoRR*, abs/2402.12563.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*.

OpenAI. 2023a. GPT-4 Technical Report. arXiv:2303.08774.

OpenAI, R. 2023b. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5).

Pan, L.; Saxon, M.; Xu, W.; Nathani, D.; Wang, X.; and Wang, W. Y. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

Pan, L.; Saxon, M.; Xu, W.; Nathani, D.; Wang, X.; and Wang, W. Y. 2024. Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Automated Correction Strategies. *Transactions of the Association for Computational Linguistics*, 11: 484–506.

Pourreza, M.; and Rafiei, D. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 36339–36348. Curran Associates, Inc.

Pourreza, M.; and Rafiei, D. 2024. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. *arXiv preprint arXiv:2402.01117*.

Qu, G.; Li, J.; Li, B.; Qin, B.; Huo, N.; Ma, C.; and Cheng, R. 2024. Before Generation, Align it! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation. *arXiv preprint arXiv:2405.15307*.

Rajkumar, N.; Li, R.; and Bahdanau, D. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Shypula, A. G.; Madaan, A.; Zeng, Y.; Alon, U.; Gardner, J. R.; Yang, Y.; Hashemi, M.; Neubig, G.; Ranganathan, P.; Bastani, O.; and Yazdanbakhsh, A. 2024. Learning Performance-Improving Code Edits. In *The Twelfth International Conference on Learning Representations*.

Talaei, S.; Pourreza, M.; Chang, Y.-C.; Mirhoseini, A.; and Saberi, A. 2024. CHESS: Contextual Harnessing for Efficient SQL Synthesis. arXiv:2405.16755.

Wang, B.; Ren, C.; Yang, J.; Liang, X.; Bai, J.; Chai, L.; Yan, Z.; Zhang, Q.-W.; Yin, D.; Sun, X.; and Li, Z. 2024a. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. arXiv:2312.11242.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Zhang, S.; Zhu, E.; Li, B.; Jiang, L.; Zhang, X.; and Wang, C. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Xie, T.; Zhou, F.; Cheng, Z.; Shi, P.; Weng, L.; Liu, Y.; Hua, T. J.; Zhao, J.; Liu, Q.; Liu, C.; et al. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; Zhang, Z.; and Radev, D. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3911–3921. Brussels, Belgium: Association for Computational Linguistics.

Zelle, J. M.; and Mooney, R. J. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, 1050–1055.

Zhang, B.; Ye, Y.; Du, G.; Hu, X.; Li, Z.; Yang, S.; Liu, C. H.; Zhao, R.; Li, Z.; and Mao, H. 2024. Benchmarking the Text-to-SQL Capability of Large Language Models: A Comprehensive Evaluation. *arXiv preprint arXiv:2403.02951*.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q. V.; et al. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.