Before Generation, Align it! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation

Ge Qu [1], **Jinyang Li** [1], **Bowen Li** [2], **Bowen Qin** [3], **Nan Huo**[1], **Chenhao Ma** [4], **Reynold Cheng** [1]
[1]The University of Hong Kong, [2] Shanghai AI Laboratory
[3] BAAI, [4]The Chinese University of Hong Kong, Shenzhen
{quge,jl0725}@connect.hku.hk, ckcheng@cs.hku.hk

Abstract

Large Language Models (LLMs) driven by In-Context Learning (ICL) have significantly improved the performance of text-to-SQL. Previous methods generally employ a two-stage reasoning framework, namely 1) schema linking and 2) logical synthesis, making the framework not only effective but also interpretable. Despite these advancements, the inherent bad nature of the generalization of LLMs often results in hallucinations, which limits the full potential of LLMs. In this work, we first identify and categorize the common types of hallucinations at each stage in text-to-SQL. We then introduce a novel strategy, Task Alignment (TA), designed to mitigate hallucinations at each stage. TA encourages LLMs to take advantage of experiences from similar tasks rather than starting the tasks from scratch. This can help LLMs reduce the burden of generalization, thereby mitigating hallucinations effectively. We further propose TA-SQL, a text-to-SQL framework based on this strategy. The experimental results and comprehensive analysis demonstrate the effectiveness and robustness of our framework. Specifically, it enhances the performance of the GPT-4 baseline by 21.23% relatively on BIRD dev and it yields significant improvements across six models and four mainstream, complex text-to-SQL benchmarks. For reproducibility, we release our code and prompt at https://github.com/quge2023/TA-SQL.

1 Introduction

In the age of big data, relational databases, as carriers for storing massive amounts of data, play a crucial role in information processing and data analysis. Text-to-SQL, which aims to convert natural language (NL) queries to executable SQL queries, facilitates access to ubiquitous relational data for a broader range of non-technical users, thereby attracting remarkable attention (Cai et al., 2018; Yu et al., 2018a; Wang et al., 2020; Cao et al., 2021).

Recently, Large Language Models (LLMs) have shown impressive success on a wide range of NLP tasks through in-context learning (ICL) (Dong et al., 2022), such as question answering (Nair et al., 2023; Nguyen et al., 2023), logic reasoning (Khot et al., 2023; Zhao et al., 2023), and code generation (Gu, 2023; Chen et al., 2023). The application of LLMs has also improved the performance of text-to-SQL to another level of intelligence (Dong et al., 2023; Pourreza and Rafiei, 2024; Gao et al., 2023). Delving into their crafted designs, these works generally approach text-to-SQL through a two-stage paradigm. The first stage, **Schema Linking**, involves the precise mapping of natural language queries to the relevant entities within a database schema (Lei et al., 2020; Wang et al., 2022; Liu et al., 2021). This meticulous alignment is crucial for the following execution of the query and provides transparency by illustrating how natural language queries are interpreted in relation to the database schema. The second step is **Logical Synthesis**, which refers to the process of generating accurate SQL queries based on the understanding of the logic of the natural language query and the structure of the database (Yin and Neubig, 2017).

Nevertheless, hallucination, a notorious problem in LLMs that refers to the generation of content that is irrelevant, erroneous, or inconsistent with user intents (Huang et al., 2023), remains a considerable barrier to current frameworks as a reliable automatic text-to-SQL parser. In this work, we first study and conclude primary hallucinations presented in the aforementioned two stages of current text-to-SQL frameworks and attribute them to two main categories: **schema-based hallucinations** and **logic-based hallucinations**, as shown in Table 1. Schema-based hallucinations refer to hallucinations in which LLMs might inaccurately identify schema structures, introduce unnecessary attributes, or fail to accurately represent or interpret database values. On the other hand, logic-based

Findings of the Association for Computational Linguistics: ACL 2024, pages 5456–5471
August 11-16, 2024 ©2024 Association for Computational Linguistics

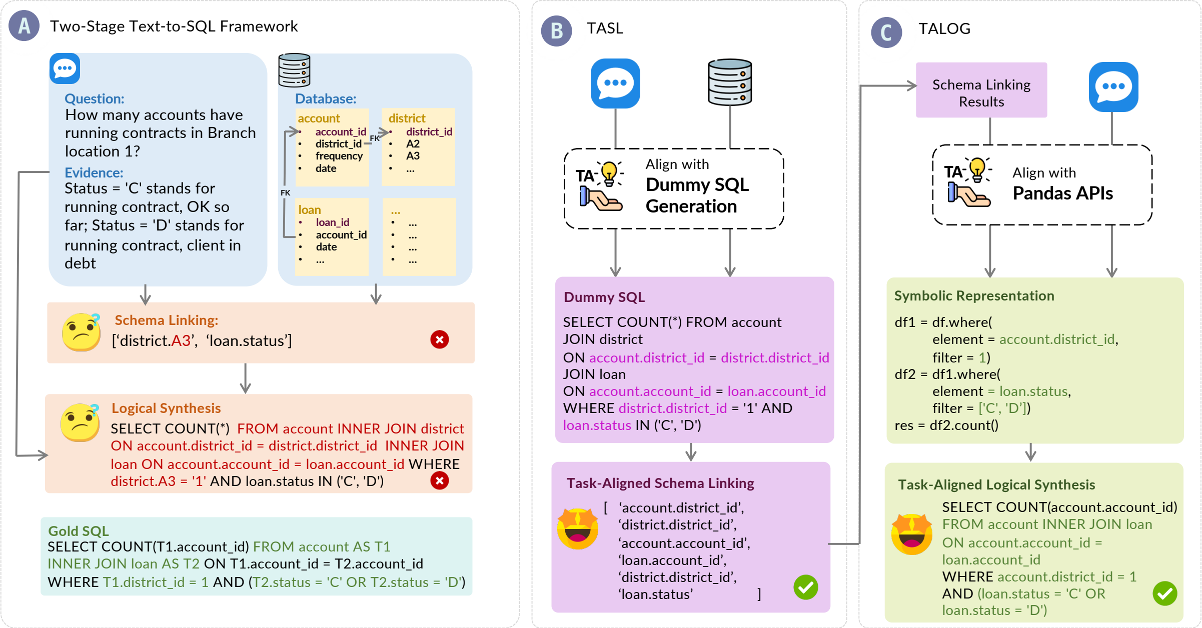
Schema-Based	**Definition**	**Example**
Schema Contradiction (30%)	Refers to the instance where incorrect SQL contradicts schema structure.	**Question:** *What language is the set of 180 cards that belongs to the Ravnica block translated into?* **Gold:** SELECT **T2.language** FROM sets AS T1 INNER JOIN **set_translations AS T2** ON WHERE T1.block = ‘Ravnica’ AND T1.baseSetSize = 180 **Wrong SQL:** SELECT **language FROM sets** WHERE baseSetSize = 180 AND block = ‘Ravnica’
Attribute Overanalysis (49%)	Refers to the instance where unnecessary attributes are introduced, leading to a contradiction with the intended result format.	**Question:** *Which player is the tallest?* **Gold:** SELECT **player_name** FROM Player ORDER BY height DESC LIMIT 1 **Wrong SQL:** SELECT player_name, **height** FROM Player ORDER BY height DESC LIMIT 1
Value Misrepresentation (24%)	Refers to the instance where the model imagines a reasonable but non-existent value format in the schema.	**Question:** *Give the race of the blue-haired men superhero.* **Gold:** SELECT ... WHERE colour.colour = **‘Blue’** AND gender.gender = **‘Male’** **Wrong SQL:** SELECT ... WHERE colour.colour = **‘blue’** AND gender.gender = **‘M’**
Logic-Based	**Definition**	**Example**
Join Redundancy (15%)	Refers to the instance where the SQL joins unnecessary tables for complex text-to-SQL cases.	**Question:** *Determine the bond type formed in the chemical compound containing element Tellurium.* **Gold:** SELECT T2.bond_type **FROM atom AS T1 INNER JOIN bond** AS T2 ON WHERE T1.element = ‘te’ **Wrong SQL:** SELECT bond_type **FROM bond INNER JOIN connected ON ... INNER JOIN atom** ON ... WHERE atom.element = ‘te’
Clause Abuse (25%)	Refers to the instance where clauses such as `GROUP BY` are abused, disrupting the correct order or limitation of results.	**Question:** *Among the posts that were voted by user 14, what is the id of the most valuable post?* **Gold:** SELECT post.Id ... WHERE votes.UserId = 14 ORDER BY post.FavoriteCount DESC LIMIT 1 **Wrong SQL:** SELECT post.Id FROM votes INNER JOIN posts ON ... WHERE votes.UserId = 14 **GROUP BY post.Id** ORDER BY post.FavoriteCount DESC LIMIT 1
Mathematical Delusion (17%)	Refers to the instance where the model fails to convert mathematical knowledge or logic into correct SQL functions, resorting to expressions such as imagined functions.	**Question:** *What is the percentage of the amount 50 received by the Student Club among members?* **Gold:** SELECT CAST(SUM(CASE WHEN income.amount = 50 THEN 1.0 ELSE 0 END) AS REAL) * 100 **/** COUNT(income.income_id) FROM ... WHERE member.position = ‘Member’ **Wrong SQL:** SELECT **DIVIDE(SUM(CASE WHEN income.amount = 50 THEN 1 ELSE 0 END), COUNT(member.member_id))** FROM ... WHERE member.position = ‘Member’

Table 1: Definitions and Examples of schema-based and logic-based hallucinations.

hallucinations can also prevent LLMs from executing accurate `JOIN` operations, applying appropriate SQL clauses such as `GROUP BY` and nested sub-queries, or computationally reasoning in data science queries.

The aforementioned challenges reinforce the demand for a robust text-to-SQL framework to minimize hallucinations and improve overall performance while maintaining interpretability. We posit that hallucinations often arise when models misinterpret the decomposed stages of a task as entirely new challenges, for which they lack prior training. This situation is comparable to human experiences, where unfamiliarity with a task can lead to disorientation and a higher propensity for errors (Silva et al., 2016). Thus, just as experienced individuals can draw on familiar situations to reduce cognitive load and enhance task performance (Carbonell, 1993), we introduce **T**ask **A**lignment (**TA**), a novel strategy to mitigate hallucinations of LLMs in this way. TA fundamentally adjusts the approach of models to unfamiliar tasks by aligning them with tasks they have previously trained on. This method reduces the dependence of models on their generalization capability for generating responses from scratch, thereby significantly reducing the incidence of hallucinations.

We further propose a text-to-SQL framework named **TA-SQL**, which consists of a **T**ask-**A**ligned **S**chema **L**inking (**TASL**) module and a **T**ask-**A**ligned **LOG**ical synthesis (**TALOG**) module. TA is employed to mitigate hallucinations in these



Question:
How many accounts have running contracts in Branch location 1?

Evidence:
Status = 'C' stands for running contract, OK so far; Status = 'D' stands for running contract, client in debt

Schema Linking:

```
['district.A3', 'loan.status']
```

Logical Synthesis

```
SELECT COUNT(*) FROM account INNER JOIN district
ON account.district_id = district.district_id INNER JOIN
loan ON account.account_id = loan.account_id WHERE
district.A3 = '1' AND loan.status IN ('C', 'D')
```

Gold SQL

```
SELECT COUNT(T1.account_id) FROM account AS T1
INNER JOIN loan AS T2 ON T1.account_id = T2.account_id
WHERE T1.district_id = 1 AND (T2.status = 'C' OR T2.status = 'D')
```

Dummy SQL

```
SELECT COUNT(*) FROM account
JOIN district
ON account.district_id = district.district_id
JOIN loan
ON account.account_id = loan.account_id
WHERE district.district_id = '1' AND
loan.status IN ('C', 'D')
```

Task-Aligned Schema Linking

```
[  'account.district_id',
   'district.district_id',
   'account.account_id',
   'loan.account_id',
   'district.district_id',
   'loan.status'            ]
```

Symbolic Representation

```
df1 = df.where(
        element = account.district_id,
        filter = 1)
df2 = df1.where(
        element = loan.status,
        filter = ['C', 'D'])
res = df2.count()
```

Task-Aligned Logical Synthesis

```
SELECT COUNT(account.account_id)
FROM account INNER JOIN loan
ON account.account_id =
loan.account_id
WHERE account.district_id = 1
AND (loan.status = 'C' OR
loan.status = 'D')
```

Figure 1: An illustration of TA-SQL, utilizing the TASL (b) and TALOG modules (c), mitigates hallucinations that occur in each of the two stages of previous text-to-SQL frameworks (a).

two modules, respectively, enhancing the performance of the framework while preserving its interpretability. Experimental results on four text-to-SQL datasets and our comprehensive analysis demonstrate the effectiveness and robustness of TA-SQL. Specifically, TA-SQL relatively improves the performance of the GPT4 baseline in terms of Execution Accuracy (EX) by 21.23% and 14.86% on BIRD (Li et al., 2024) and SPIDER (Yu et al., 2018b), respectively. Moreover, our experimental results also illustrate that TA-SQL is a model-agnostic framework, exhibiting applicability to both mainstream closed-source LLMs and open-source weaker LLMs.

2 Preliminaries

Problem Definition Given a natural language question $\mathcal{Q} = \{q_1, ..., q_{|\mathcal{Q}|}\}$ with its corresponding database schema $\mathcal{D} = \langle \mathcal{C}, \mathcal{T} \rangle$, where $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$ and $\mathcal{T} = \{t_1, ..., t_{|\mathcal{T}|}\}$ represent columns and tables, $|\mathcal{C}|$ and $|\mathcal{T}|$ refer to the number of columns and tables in each database respectively. The goal of text-to-SQL is to generate the corresponding SQL query y.

In-Context Learning In-Context Learning (ICL) is a paradigm that allows language models to learn tasks with only a few examples in the form of demonstrations (Dong et al., 2022), or even without examples. It requires no additional training and is directly applicable to pre-trained LLMs. In this work, we only discuss hallucinations in ICL-based text-to-SQL frameworks. Few-shot prompting is a scenario in ICL where it uses task descriptions I and a set few-shot input-output (I/O) prompting demonstrations $S = \{(x_1, y_1), ..., (x_k, y_k)\}$ to assist LLM $\mathcal{M}$ to solve the input problem x which belongs to a task m by:

$$y = f_{\mathcal{M}}(x, I, S \mid m), \tag{1}$$

where $f_{\mathcal{M}}(\cdot \mid m)$ refers to a mapping function applied by LLM $\mathcal{M}$ when it generalizes task m from scratch. When I/O pairs are no longer provided, the scenario transitions to zero-shot prompting, where the model is expected to understand and complete the task relying solely on its pre-trained knowledge, and the output of zero-shot prompting could be represented as:

$$y = f_{\mathcal{M}}(x, I \mid m), \tag{2}$$

3 Methodology

3.1 Task Alignment

Inspired by the human approach of drawing upon relevant past experience when tracking unfamiliar tasks, we introduce Task Alignment (TA), a novel strategy designed to mitigate hallucinations. The fundamental idea is that LLMs have already acquired knowledge of various tasks during training

(Ouyang et al., 2022). We refer to tasks for which the basic rules have been mastered by LLMs as pre-trained tasks. Given a novel task m^n and one of its problems x, TA first retrieves the most related pre-trained task m^p from a set of LLM pre-trained tasks $\{m_1^p, m_2^p, ..., m_k^p\}$. In this study, we manually select m^p for each new task. The potential for LLMs to automatically select tasks is a valuable prospect for future research. It then leverages this pre-trained task to reconstruct the representation for the novel task m^n, aligning it with the representation that the LLMs are familiar with. The goal of TA is to solve the problem x by:

$$y = f_{\mathcal{M}}(x, I, S \mid m^n \rightarrow m^p), \tag{3}$$

where $f_{\mathcal{M}}(\cdot \mid m^n \rightarrow m^p)$ refers to the mapping function applied by LLM $\mathcal{M}$ when it applies experiences from aligned pretrained task m^p while generalizing m^n.

TA explicitly guides LLMs to approach unfamiliar tasks from the perspective of more familiar ones, alleviating the burden of from-scratch generalization and subsequently mitigating hallucinations.

3.2 TA-SQL

We further propose a robust text-to-SQL framework named TA-SQL. TA-SQL adheres to the two-stage paradigm of previous work, consisting of a task-aligned schema linking module and a task-aligned logic synthesis module. However, unlike previous works (Dong et al., 2023; Pourreza and Rafiei, 2024; Gao et al., 2023) that treat each decomposed module as an entirely new task for the LLM to generalize from scratch, we apply the TA strategy to each module to stimulate incremental generalization of LLMs. This design not only mitigates hallucinations effectively for better performance but also maintains the interpretability of the entire model. We introduce these two modules, respectively, in the following sections. The prompts employed within each module are displayed in Appendix A.

3.2.1 Task-Aligned Schema Linking Module (TASL)

Given a natural language question $\mathcal{Q}$ with corresponding database schema $\mathcal{D}$, schema linking is responsible for identifying references to columns, tables, and condition values in $\mathcal{Q}$. However, LLMs are not adept at the schema linking task. Therefore, when dealing with complex databases characterized by their extensive size and abundant semantic information, LLMs are highly prone to generating schema-based hallucinations. (Gan et al., 2023). Hallucinations at this stage would be influential negatively on the final performance due to the error propagation (Caselli et al., 2015).

As shown in Figure 1 (b), we design a TASL module for the schema linking stage. The schema linking task in this module is represented as first generating a dummy SQL query and then extracting related schema entities from it as the final output. Although the schema linking task is not familiar to LLMs, its downstream task, SQL generation, has been extensively exposed during training (Guo et al., 2024). Playing a similar role to negative sampling in the skip-gram algorithm (Mikolov et al., 2013), the objective of dummy SQL generation is not to create executable SQL directly for the final application. Instead, its primary function is to subtly leverage the successful experiences of schema entity selection during the generation process for LLMs.

3.2.2 Task-aligned Logical Synthesis Module (TALOG)

The TALOG module is responsible for reasoning the transformation logic from the NL query into SQL based on the results generated by the TASL module and accordingly producing accurate SQL. This process often involves multiple forms of logic, including SQL syntax reasoning, external knowledge reasoning, and computational reasoning. Such complexity presents a significant challenge for LLMs, leading to the emergence of logic-based hallucinations (Lee, 2023).

In fact, SQL serves as a tool for extracting values from Relational Database (RDB) for data analysis. It encapsulates various data analysis logics, such as data filtering, mathematical computation, and output synthesis. As such, we employ the TA in the capacity of a data scientist who addresses complex problems through step-by-step logical operations using pandas-like APIs (Zan et al., 2022) and generates symbolic representations that include reasoning processes, as shown in Figure 1 (c).

After logic alignment with data analysis processes, the remaining challenge is to ensure that LLMs are proficient in perceiving valid SQL syntax and structures. This proficiency is crucial for the generation of accurate SQL. To facilitate this, we replace conventional pandas API functions with symbolic functions that resemble SQL keywords, thereby enabling the symbolic representation to invoke them effectively.

4 Experiments

4.1 Experiment Settings

Datasets. We evaluate our text-to-SQL framework on four challenging benchmarks for cross-domain SQLs. (1) **Bird** (Li et al., 2024) is the most challenging lager-scale cross-domain text-to-SQL benchmark. It has two settings, with and without external knowledge, to highlight the new challenges brought by external knowledge. In this paper, we use its development set for evaluation, which contains 1534 pairs of text-to-SQL data and 11 databases, as the test set is not released. **(2) Spider**(Yu et al., 2018b) is a more standard cross-domain text-to-SQL benchmark. It contains 1034 examples, which cover 20 complex databases across multiple domains, in the development set. **(3) DK** (Gan et al., 2021a) requires text-to-SQL parsers to equip with the capability of domain knowledge reasoning. **(4) Realistic** removes and switches the obvious mentions of schema items in questions, making it closer to the real scenarios.

Metrics. Following the prior study (Yu et al., 2018b; Li et al., 2024), we use Execution Accuracy (EX) to measure the performance of our method. EX can reflect whether a predicted SQL is valid and return the exact result as the execution result of the ground truth SQL.

Models. We experiment our proposed method with both closed-sourced LLMs and open-sourced code generation models. For the closed-source LLMs, we experiment with GPT family models including ChatGPT (`gpt-3.5-turbo`) (Ouyang et al., 2022), GPT4 (`gpt-4-32k`) (Achiam et al., 2023), GPT4-Turbo (`gpt-4-turbo`), and Claude (`claude-2.0`) (Anthropic, 2023). For open-source weaker LLM models, we experiment with two most popular and strong baselines, CodeLlama (`codellama-34b-instruct`) (Roziere et al., 2023), and DeepSeek (`deepseek-coder-33b-instruct`) (Guo et al., 2024).

Compared Methods. We also compare our method with two SOTA ICL-based methods, that are, DIN-SQL (Pourreza and Rafiei, 2024) and DAIL-SQL (Gao et al., 2023) on both BIRD and SPIDER.

Implementation We implement the schema linking module with zero-shot prompts and the logical synthesis module with 6-shot prompts. For all models we used in this paper, we set the argument temperature and top-p as 0 and 1, respectively, to promise reproduction. The max_tokens (`max_new_tokens`) for closed-source LLMs and open-source weaker LLMs are both set as 800, respectively, for all modules.

Method	Dev	Test
	w/o knowledge	
Palm-2	18.77	24.71
Codex	25.42	24.86
ChatGPT	24.05	26.77
ChatGPT+COT	25.88	28.95
Claude-2	28.29	34.60
GPT-4	30.90	34.88
TA-SQL+GPT-4	**50.58** (↑ 19.68)	**54.38** (↑ 19.50)
	w/ knowledge	
Palm-2	27.38	33.04
Codex	34.35	36.47
ChatGPT	37.22	39.30
ChatGPT+COT	36.64	40.08
Claude-2	42.70	49.02
DIN-SQL+GPT-4 ♣	50.72	55.90
DAIL-SQL+GPT-4 ♣	54.76	56.08
GPT-4	46.35	54.89
TA-SQL+GPT-4	**56.19** (↑ 9.84)	**59.14** (↑ 4.25)

Table 2: Execution Accuracy (EX) (%) on BIRD. ♣ means the model uses self-consistency or re-modification mechanisms. ↑ is an absolute improvement.

4.2 Main Results

Results on BIRD. Table 2 displays the performance of TA-SQL and other competitive methods on the current most challenging text-to-SQL benchmark, BIRD. First, in the setting with oracle knowledge, we demonstrate that TA-SQL effectively mitigates hallucinations in the GPT4 baseline, resulting in a relative improvement of 21.23% in EX on the development set and 7.74% on the test set. This demonstrates that even the most powerful LLMs can produce severe hallucinations during the text-to-SQL process, thereby highlighting the value of hallucination mitigation research. ***Surprisingly, TA-SQL equipped with GPT4 outperforms the SOTA LLM-based method without fine-tuning by 2.61%*** even without the application of self-consistency or re-modification mechanisms. Furthermore, even in the setting without external knowledge, TA-SQL achieves performance comparable to the GPT4 baseline equipped with oracle external knowledge. This suggests that addressing hallucinations within the existing knowledge could

METHOD	SPIDER					DK					REALISTIC				
	easy	medium	hard	extra	all	easy	medium	hard	extra	all	easy	medium	hard	extra	all
GPT4	89.1	79.8	61.5	48.8	74.0	78.2	72.4	50.0	45.7	65.2	86.2	82.7	57.6	55.7	73.4
+ TA-SQL	**93.5**	**90.8**	**77.6**	**64.5**	**85.0**	**84.5**	**78.0**	**64.9**	**54.3**	**72.9**	**88.1**	**87.7**	**72.7**	**59.8**	**79.5**

Table 3: Execution Accuracy (EX) across queries of varying levels of difficulty on SPIDER, DK, and REALISTIC.

be a promising and cost-effective solution, rather than resorting to the addition of manually extracted external knowledge from heterogeneous resources with much more effort.

Results on SPIDER and its Variant Datasets As shown in Table 3, TA-SQL effectively enhances the EX performance of the GPT4 baseline by 14. 86%, 11. 80%, and 8. 31% on the SPIDER and its variant datasets, DK and REALISTIC, respectively, with improvements across all difficulty levels. This suggests that TA-SQL, as a general method, is not only useful in complex text-to-SQL scenarios that closely mirror the real world but also delivers robust performance on standard text-to-SQL benchmarks where the context is relatively simple.

Results on Model Agnosticism. TA-SQL is proved to be model-agnostic since it can work among mainstream closed-source LLM and open-source weaker language models, as shown in Table 4. TA-SQL can improve the performance across queries of varying difficulty levels for closed-source LLMs. However, we observe that the performance gains brought by TA-SQL for weaker models (CodeLlama, DeepSeek) are relatively limited. This can be attributed to their constrained capabilities in generalizing and instruction-following (Qi et al., 2023), which limit the effectiveness of TA-SQL in diminishing hallucinations for challenging queries (Shen et al., 2024).

MODEL	SIM.	MOD.	CHALL.	TOTAL
Closed-Source LLM				
GPT4	54.35	34.64	31.70	46.35
+TA-SQL	63.14	48.60	36.11	56.19
GPT4-turbo	59.35	38.92	27.78	50.19
+TA-SQL	60.54	40.86	38.19	52.48
Claude	51.34	30.07	23.24	42.47
+TA-SQL	56.97	39.78	27.78	48.89
ChatGPT	47.60	22.44	18.31	37.22
+TA-SQL	51.57	33.76	25.69	43.74
Open-Source weaker LLM				
DeepSeek	51.68	29.03	18.06	41.66
+TA-SQL	53.41	32.04	19.44	43.74
CodeLlama	34.81	15.48	11.11	26.73
+TA-SQL	37.30	13.33	11.11	27.57

Table 4: Execution Accuracy (EX) of TA-SQL employing various models as the backend. SIM., MOD., CHALL. represent the levels of query difficulty and are the abbreviations of simple, moderate, and challenging, respectively.

METHOD	SIM.	MOD.	CHALL.	TOTAL
TA-SQL	63.14	48.60	36.11	**56.19**
w/o Schema Linking	58.35	37.92	32.04	**49.77** (↓ **6.42**)
w/o Logical Synthesis	61.59	39.57	32.64	**52.41** (↓ **3.78**)

Table 5: Imperative analysis for the two-stage paradigm on BIRD development set. ↓ is an absolute decrease.

4.3 Imperative of Two-stage Paradigm

We conduct an imperative analysis of the two-stage paradigm. Table 5 illustrates that the two-stage paradigm not only makes the text-to-SQL framework interpretable, but also impacts the overall performance of the framework. Specifically:

The schema linking module constitutes a prerequisite for the success of TA-SQL. Firstly, the removal of the schema linking module disrupts the interpretability of the framework, preventing it from correcting hallucinations through more flexible methods such as human-computer interaction. Secondly, through quantitative analysis, we discover that the removal of the schema linking module leads to a significant performance decline across queries of varying difficulty levels (↓ 6.42% in total). This is attributed to the fact that more accurate schema linking results not only reduce schema-based hallucinations but also facilitate the subsequent logical synthesis module to conduct more granular and complex reasoning based on these results.

The logical synthesis module determines the upper bound for the performance of the entire framework. This is evidenced by the observation that, relative to the performance decline on simple queries (↓ 1.55%), the removal of the logical synthesis module has a more obvious impact on moder-

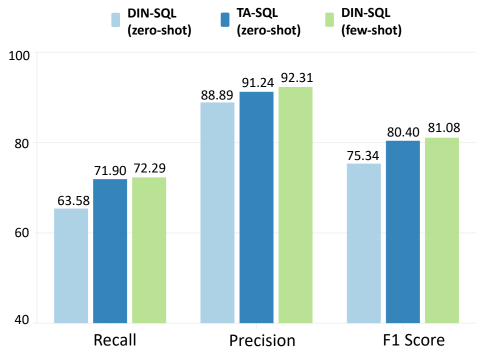


Figure 2: Results of different schema linking modules on BIRD-dev.

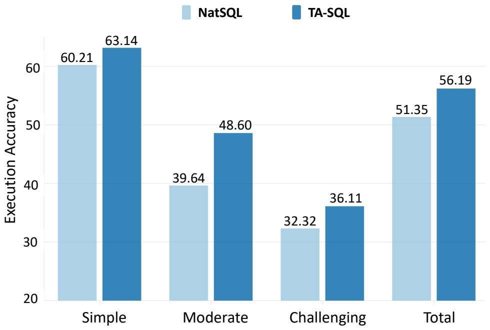


Figure 3: Results of different logical synthesis modules on BIRD-dev.

ate (↓ 10.68%) and challenging queries (↓ 3.74%). Relatively challenging queries often contain more complex analytical intentions, involving mathematical computations, multi-step reasoning, or compositional generalization. The symbolic representation produced by the logical synthesis module effectively guides analytical reasoning processes, thereby raising the upper bound of the framework's ability to solve complex problems.

4.4 Ablation Study

After validating the imperative of the two-stage paradigm, we further conduct an ablation study to evaluate the effectiveness of implementing the TA strategy within these two stages. The schema linking module and logical synthesis module, following the customized designs in DIN-SQL (Pourreza and Rafiei, 2024) and NatSQL(Gan et al., 2021b), are implemented, respectively, for comparison.

Results in the Schema Linking Stage. We implement the schema linking module of DIN-SQL in both the zero-shot and few-shot settings for comparison. Three metrics are introduced to facilitate a more intuitive comparison of the schema linking results: (1) **Recall** computes the ratio of instances in which the schema linking outcomes encompass all ground truth schema elements of this instance. (2) **Precision** quantifies the accuracy of the linked schema. (3) **F1 Score** represents a harmonic mean of recall and precision. The detailed definitions of these metrics are presented in Appendix B.

Figure 2 illustrates that, even when following the design of the SOTA method DIN-SQL, zero-shot schema linking tasks confuse LLMs, since it requires LLMs to comprehend and generalize this unfamiliar task from scratch. This confusion can be alleviated by human-annotated example demonstrations in the few-shot setting (F1 Score: 75.34 → 81.08). However, the TASL module in the zero-shot setting can directly achieve results that are competitive with the few-shot schema linking module in DIN-SQL without human intervention. This suggests that TA can effectively guide the model to align with pre-trained knowledge to tackle unfamiliar tasks without relying on additional external information.

Results in the Logical Synthesis Stage. We implement another logical synthesis module with a classic customized symbolic representation designed in NatSQL (Gan et al., 2021b) and refer to it as NatLOG module. Same as TALOG, it is also implemented in the 6-shot setting. As shown in Figure 3, the TALOG module exhibits superior performance across various levels of difficulty compared with the NatLOG module. This is because the custom symbolic representation includes new rules that the model needs to understand and learn from scratch, thereby increasing the emergence of hallucinations during the logical synthesis process.

4.5 Fine-grained Case Study

To illustrate the effectiveness of TA-SQL qualitatively, we conduct an analysis of its performance across databases in the BIRD development set. TA-SQL demonstrates a significant impact on databases `debit_card_specializing`, `toxicology`, and `california_schools`, yielding relative EX increases of 59.05%, 43.1%, and 32.13%, respectively. ***From the perspective of database schema***, as shown in Table 6 Case 1, GPT4 selects `satscores.dname` and `AVG(AvgScrRead)` as final attributes of the generated SQL, which contradicts the intent of the NL query. This suggests that GPT4 struggles to map NL query entities to the database schema

Case 1	`california_school`	
Question	*Which active district has the highest average score in Reading?*	
Gold	SELECT T1.District FROM schools AS T1 INNER JOIN satscores AS T2 ON T1.CDSCode = T2.cds WHERE T1.StatusType = 'Active' ORDER BY T2.AvgScrRead DESC LIMIT 1	
GPT4	SELECT dname, AVG(AvgScrRead) FROM satscores JOIN schools ON satscores.cds = schools.CDSCode WHERE StatusType = 'Active' GROUP BY dname ORDER BY AVG(AvgScrRead) DESC LIMIT 1	✗

TA-SQL	`df1 = df.where(element = schools.StatusType, filter = 'Active')`
	`df2 = df1.orderby(by = satscores.AvgScrRead, desc).limit(1)`
	`res = df2.select(schools.District)`
	SELECT schools.District FROM satscores INNER JOIN schools ON satscores.cds = schools.CDSCode WHERE schools.StatusType = 'Active' ORDER BY satscores.AvgScrRead DESC LIMIT 1
		✓

Case 2	`debit_card_specializing`	
Question	*What is the percentage of the customers who used EUR in 2012/8/25?*	
Gold	SELECT CAST(SUM(IIF(T2.Currency = 'EUR', 1, 0)) AS FLOAT) * 100 / COUNT(T1.CustomerID) FROM transactions_1k AS T1 INNER JOIN customers AS T2 ON T1.CustomerID = T2.CustomerID WHERE T1.Date = '2012-08-25'	
GPT4	SELECT (SELECT COUNT(CustomerID) * 100.0 / (SELECT COUNT(CustomerID) FROM customers) FROM transactions_1k AS T1 INNER JOIN customers AS T2 ON T1.CustomerID = T2.CustomerID WHERE customers.Currency = 'EUR' AND transactions_1k.Date = '2012-08-25')	✗

TA-SQL	`df1 = df.where(element = transactions_1k.Date, filter = '2012-08-25')`
	`df2 = df1.where(element = customers.Currency, filter = 'EUR')`
	`res = df.select(cast(df2.count(), real) * 100 / df1.count())`
	SELECT CAST(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM transactions_1k WHERE Date = '2012-08-25') AS REAL) FROM transactions_1k AS T1 INNER JOIN customers AS T2 ON T1.CustomerID = T2.CustomerID WHERE transactions_1k.Date = '2012-08-25' AND customers.Currency = 'EUR'
		✓

Table 6: Case study: two illustrative cases from BIRD-dev.

within complex contexts, leading to the occurrence of schema contraction and attribute overanalysis, which fall under schema-based hallucinations. However, TA-SQL can correspond the question entities to the correct column names through precise retrieval of tables and columns by the TASL module, coupled with more granular schema-related reasoning, such as element selection, by the TA-LOG module. ***From the perspective of query difficulty***, the capacity of TA-SQL to mitigate logic-based hallucinations can yield a more substantial effect within databases that contain complex queries, which require multiple logical operations. In Case 2, GPT4 manifested erroneous computational logic, which is an instance of mathematical delusion, suggesting GPT4's limited capability when confronted with complicated multi-step reasoning. Conversely, TA-SQL clearly demonstrates the data manipulation process, thereby equipping it with the capacity to manage complex logic.

4.6 Discussion about Hallucinations in Text-to-SQL Systems

Given that this is the first attempt to systematically study hallucinations in text-to-SQL systems, we define them following the survey (Ji et al., 2023). All types of hallucinations shown in Table 1 are categorized as **Intrinsic Hallucinations**, which occur when the generated SQL query contradicts the information or intent expressed in the natural language query, the underlying database schema, or SQL syntax.

It is worth noting that the distinction between hallucinations we define and errors is quite subtle. While hallucinations can lead to the occurrence of errors, they may not always directly result in errors. For instance, joining redundant tables can sometimes produce the same executed results, which are considered correct in the current SQL evaluation system. More importantly, errors are typically detected after the final SQLs are outputted and executed, usually when the entire workflow is completed. However, hallucinations can be observed and mitigated before final result generations (i.e., during schema linking or logical synthesis phases). Our proposed method originates from this problem definition and achieves equivalent or better performance than error-corrected methods, as demonstrated in the experiment section. Figure 4 shows the fine-grained performance of TA-SQL on mitigating hallucinations across each category.

5 Related Work

Text-to-SQL The development of a successful cross-domain text-to-SQL parser fundamentally involves the creation of an encoder for learning representations of questions and schema and a decoder for generating SQL queries (Qin et al., 2022). For instance, RATSQL (Wang et al., 2020), SDSQL (Hui et al., 2021), LGESQL (Cao et al., 2021), S^2SQL (Hui et al., 2022), and Proton (Wang et al., 2022) have advanced the representation learning of natural language questions and database schema using a relational graph neural network. The intro-

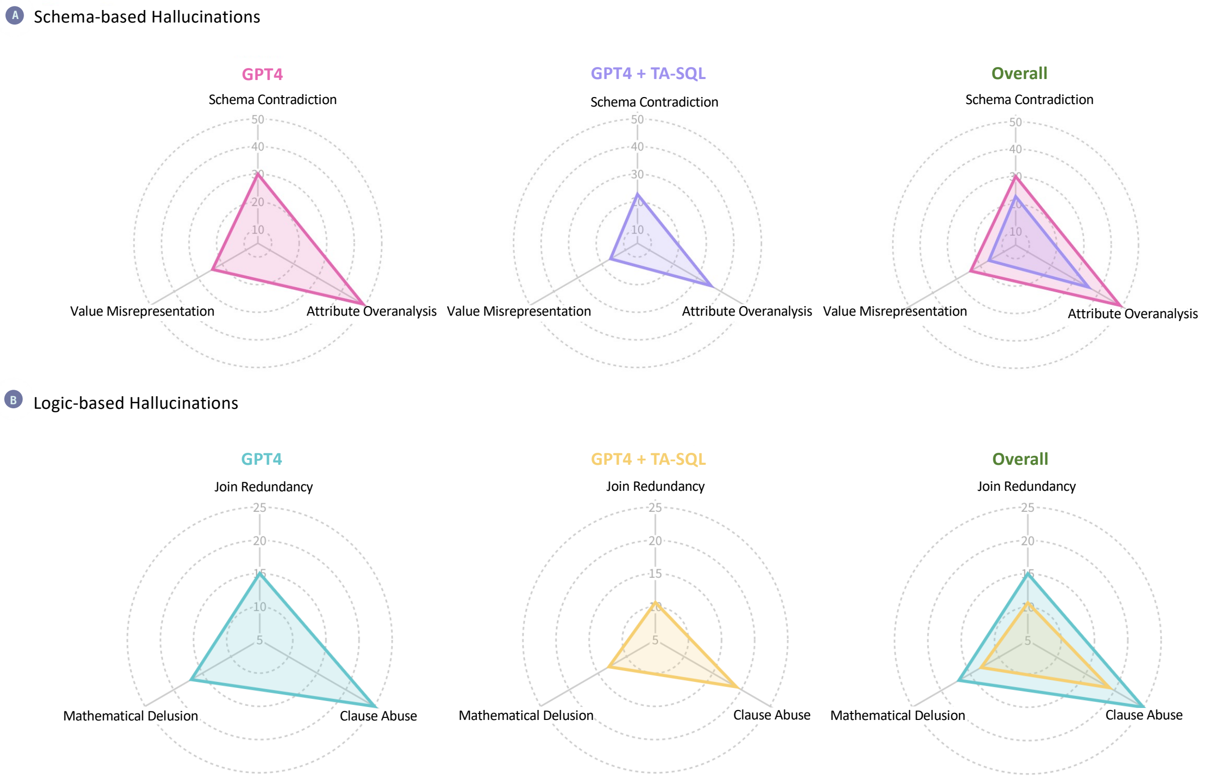


Figure 4: Performance of fine-grained categorical hallucination mitigation on BIRD.

duction of sequence-to-sequence pre-trained language models (PLMs) such as T5 (Raffel et al., 2020) and NQG-T5 (Shaw et al., 2021) significantly transforms text-to-SQL tasks, given their adaptability and generative capabilities across diverse datasets. These models demonstrate impressive results through fine-tuning with minimal effort. Besides, PICARD (Scholak et al., 2021) designs a constrained decoder to reject inadmissible tokens at the decoding step. RASAT (Qi et al., 2022) further improves the structural information encoding of T5 by integrating schema alignment into the encoder, while Graphix (Li et al., 2023b) has equipped T5 with multi-hop reasoning. RESDSQL (Li et al., 2023a) enhances T5 by decoupling the schema linking and the skeleton parsing.

Recently, large language models (LLMs) (Ouyang et al., 2022; Chowdhery et al., 2023; Anthropic, 2023) have attracted considerable attention due to their robust reasoning and domain generalization capabilities. Models like DIN-SQL (Pourreza and Rafiei, 2024) and DAIL-SQL (Gao et al., 2023) with few-shot demonstrations, along with the evolution of language models to language agents (Deng et al., 2024; Gu et al., 2024), have pioneered text-to-SQL solutions to a new level of intelligence.

Hallucination One of the most prominent challenges is hallucination, a phenomenon where a model generates information that is not present or inferred from the input (Ji et al., 2023). This issue is particularly serious in text generation tasks, as evidenced by the factual consistency problems of dialog generation (Dziri et al., 2021; Rashkin et al., 2021; Shuster et al., 2021) when using LLMs. As one of the important techniques in database applications, hallucination can result in the text-to-SQL generation of erroneous or non-sensical SQL queries.

6 Conclusion

In this research, we first systematically identify and classify common hallucination types in text-to-SQL. Subsequently, we propose Task Alignment (TA), a novel strategy to mitigate hallucinations in Large Language Models (LLMs) during the text-to-SQL process. Based on this strategy, we further propose TA-SQL, a framework to mitigate hallucinations at each stage of this process. Experimental results and comprehensive analysis show the importance of hallucination research in text-to-SQL and data science and suggest promising directions for future work.

7 Limitations

Our findings suggest that TA is particularly adept at handling complex cases where the knowledge evidence supplied by the BIRD database is explicit and the questions poised are unequivocally answerable, as shown in Section 4.5. On the contrary, in the databases `codebase_community`, `student_club`, and `thrombosis_prediction`, we observe that the clarity of questions and the sufficiency of knowledge evidence render it more effective to directly generate SQL queries from annotated data, bypassing the need for a multi-step calibration process. Furthermore, the selection of familiar tasks in each phase of the text-to-SQL conversion process is currently conducted by human prior knowledge rather than an automated mechanism capable of identifying and retrieving relevant and familiar tasks for TA applications. This gap highlights a notable avenue for future research.

8 Acknowledgement

We thank all constructive comments from anonymous reviewers. Reynold Cheng, Ge Qu, Jinyang Li, and Nan Huo were supported by the Hong Kong Jockey Club Charities Trust (Project 260920140), the University of Hong Kong (Project 109000579), and the HKU Outstanding Research Student Supervisor Award 2022-23. Chenhao Ma was partially supported by NSFC under Grant 62302421, Basic and Applied Basic Research Fund in Guangdong Province under Grant 2023A1515011280. Ge Qu and Jinyang Li were supported by HKU Presidential PhD Scholar Programme. Ge Qu was also funded by Hong Kong PhD Fellowship Scheme.

9 Ethical Statement

All datasets employed in this work are publicly accessible, ensuring the transparency and reproducibility of our findings. Furthermore, the output generated by our investigations is structured as SQL queries—a programming language format—rather than natural language text, which could potentially involve harmful or biased content. Our team meticulously examines each output to confirm the absence of politically sensitive or biased material. Finally, regarding our analysis of open-source models utilizing GPUs, such as Deepseek and Codellama, it is notable that our approach involves only model inference without training.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2023. Introducing Claude.

Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. 2018. An encoder-decoder framework translating natural language to database queries. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: line graph enhanced text-to-sql model with mixed local and non-local relations. 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*.

Jaime G. Carbonell. 1993. *Derivational analogy: a theory of reconstructive problem solving and expertise acquisition*.

Tommaso Caselli, Piek Vossen, Marieke van Erp, Antske Fokkens, Filip Ilievski, Rubén Izquierdo, Minh Le, Roser Morante, and Marten Postma. 2015. When it's all piling up: investigating error propagation in an NLP pipeline. In *Proceedings of the Workshop on NLP Applications: Completing the Puzzle, WNACP 2015, co-located with the 20th International Conference on Applications of Natural Language to Information Systems (NLDB 2015), Passau, Germany, June 17-19, 2015*.

Hailin Chen, Amrita Saha, Steven Chu-Hong Hoi, and Shafiq Joty. 2023. Personalized distillation: Empowering open-sourced llms with adaptive learning for code generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems.*

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234.*

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306.*

Nouha Dziri, Andrea Madotto, Osmar Zaïane, and Avishek Joey Bose. 2021. Neural path hunter: Reducing hallucination in dialogue systems via path grounding. 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.*

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021a. 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.*

Yujian Gan, Xinyun Chen, and Matthew Purver. 2023. Re-appraising the schema linking for text-to-sql. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023.*

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John H. Drake, and Qiaofu Zhang. 2021b. Natural SQL: making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021.*

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

Qiuhan Gu. 2023. Llm-based code generation method for golang compiler testing. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023.*

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. 2024. Middleware for llms: Tools are instrumental for language agents in complex environments. *arXiv preprint arXiv:2402.14672.*

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196.*

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232.*

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022.*

Binyuan Hui, Xiang Shi, Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2021. Improving text-to-sql with schema dependency learning. *arXiv preprint arXiv:2103.04399.*

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.*

Minhyeok Lee. 2023. A mathematical investigation of hallucination and creativity in gpt models. *Mathematics.*

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *AAAI.*

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial*

Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. 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.*

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021.*

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*

Inderjeet Nair, Shwetha Somasundaram, Apoorv Saxena, and Koustava Goswami. 2023. Drilling down into the discourse structure with llms for long document question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023.*

Minh Thuan Nguyen, Khanh-Tung Tran, Nhu-Van Nguyen, and Xuan-Son Vu. 2023. Vigptqa - state-of-the-art llms for vietnamese question answering: System overview, core models training, and evaluations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023.*

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems.*

Chengwen Qi, Bowen Li, Binyuan Hui, Bailin Wang, Jinyang Li, Jinwang Wu, and Yuanjun Laili. 2023. An investigation of llms' inefficacy in understanding converse relations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023.*

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: integrating relational structures into pretrained seq2seq model for text-to-sql. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022.*

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629.*

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*

Hannah Rashkin, David Reitter, Gaurav Singh Tomar, and Dipanjan Das. 2021. Increasing faithfulness in knowledge-grounded dialogue with controllable features. 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.*

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950.*

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing.*

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).*

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324.*

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021.*

Yasin N. Silva, Isadora Almeida, and Michell F. Queiroz. 2016. SQL: from traditional databases to big data. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE 2016, Memphis, TN, USA, March 02 - 05, 2016*.

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

Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*.

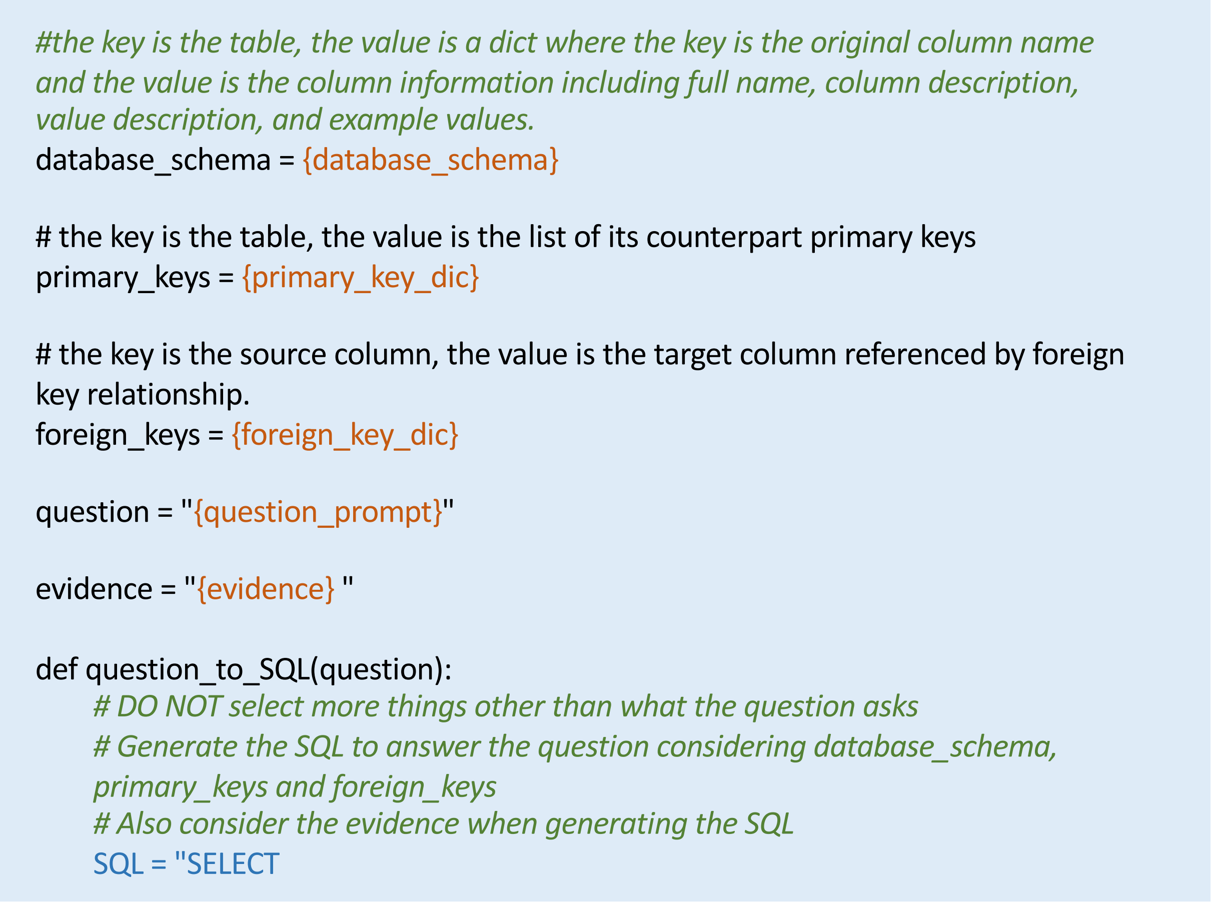
Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proc. of NAACL*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018b. 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*.

Daoguang Zan, Bei Chen, Zeqi Lin, Bei Guan, Yongji Wang, and Jian-Guang Lou. 2022. When language model meets private library. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*.

Hongyu Zhao, Kangrui Wang, Mo Yu, and Hongyuan Mei. 2023. Explicit planning helps language models in logical reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*.



```
#the key is the table, the value is a dict where the key is the original column name
and the value is the column information including full name, column description,
value description, and example values.
database_schema = {database_schema}

# the key is the table, the value is the list of its counterpart primary keys
primary_keys = {primary_key_dic}

# the key is the source column, the value is the target column referenced by foreign
key relationship.
foreign_keys = {foreign_key_dic}

question = "{question_prompt}"

evidence = "{evidence} "

def question_to_SQL(question):
    # DO NOT select more things other than what the question asks
    # Generate the SQL to answer the question considering database_schema,
    primary_keys and foreign_keys
    # Also consider the evidence when generating the SQL
    SQL = "SELECT
```

Figure 5: The prompt of generating dummy SQLs.

A TA-SQL Recipe

A.1 Pythonic Prompt

Unlike previous works that used natural language to construct prompts, all prompts used in TA-SQL are pythonic prompts. Considering that text-to-SQL is fundamentally a code generation task, pythonic prompts can exhibit data structures more clearly and express constraints and requirements more concisely.

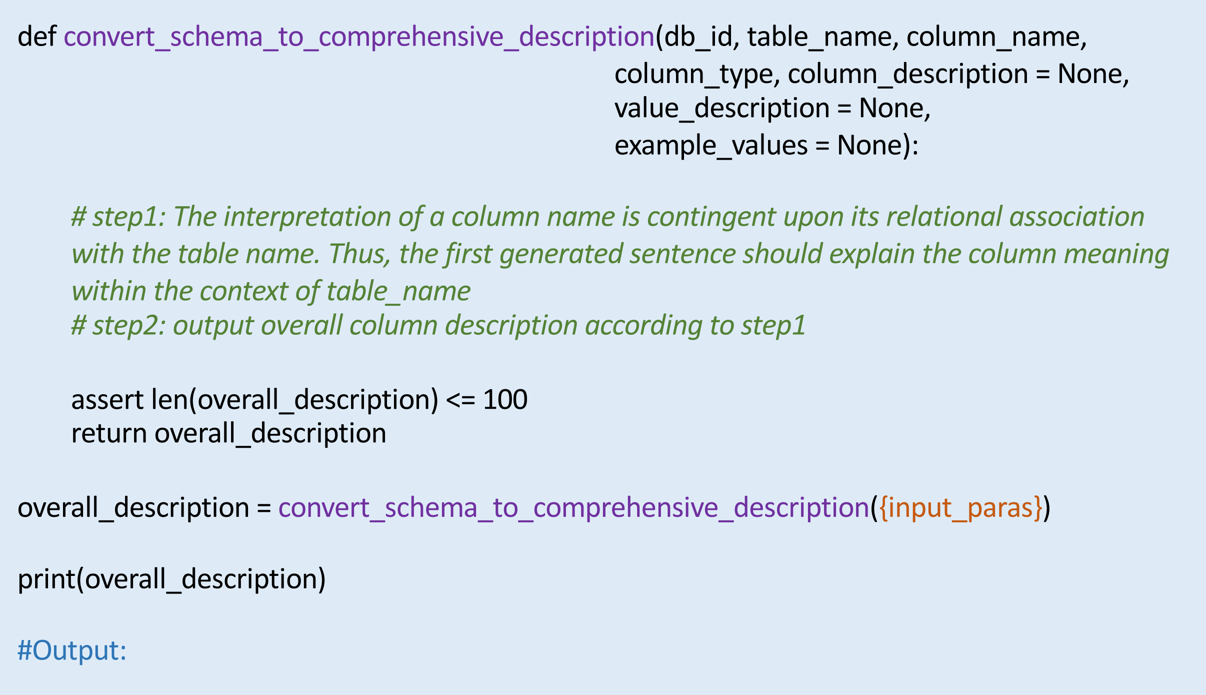
A.2 TASL Module

The schema linking task in TASL module is represented as first generating a dummy SQL query and then extracting related schema entities from it as the final output. As discussed in Section 4.4, we implement the TASL module in the zero-shot setting, leveraging the efficient employment of TA. The zero-shot prompt used to generate dummy SQL in this module is presented in Figure 5.

Specifically, we employ a Python dictionary to represent the database schema, where the key is the `table_name.column_name` entity (e.g. `account.account_id`), and the value is the comprehensive description of the corresponding column. However, directly concatenating all related information, such as column type, original column description, and value description, as the final comprehensive description might lead to an excessively lengthy prompt. This verbosity could potentially confuse LLMs. Therefore, to prevent such issues, as a preparatory step to generating dummy SQL, we first prompt LLMs to generate a succinct description for each column, drawing upon the aforementioned related information. These succinct descriptions then serve as the value for each column within the database schema dictionary. The prompt used to generate succinct column descriptions is presented in Figure 6.

A.3 TALOG Module

TALOG module employs pandas APIs to guide LLMs in conducting step-by-step logical reasoning. However, there is a natural gap between the conventional pandas APIs and the ultimate execution language, SQL. To bridge this, we replace the conventional pandas API functions with symbolic



```
def convert_schema_to_comprehensive_description(db_id, table_name, column_name,
                                                column_type, column_description = None,
                                                value_description = None,
                                                example_values = None):

    # step1: The interpretation of a column name is contingent upon its relational association
    with the table name. Thus, the first generated sentence should explain the column meaning
    within the context of table_name
    # step2: output overall column description according to step1

    assert len(overall_description) <= 100
    return overall_description

overall_description = convert_schema_to_comprehensive_description({input_paras})

print(overall_description)

#Output:
```

Figure 6: The prompt of succinct column description generation.

functions that resemble SQL keywords, thereby ensuring a precise translation from generated symbolic representations to SQLs in subsequent steps. To facilitate the model's understanding of this substitution, we provide a few demonstrations, specifically six shots, for the model to learn from, thereby generating the desired symbolic representations. Figure 7 presents the prompts used within TALOG for generating symbolic representations.

B Details of Evaluation Metrics for Schema Linking

Recall. Recall P is defined as the proportion of queries for which the linked schema outputted by the schema linking module contains all the ground truth schema, relative to the overall number of queries. It is noteworthy that since the schema retrieved by the TASL module would replace the original complete database schema as the input for the subsequent TALOG modules, the recall determines the upper bound of the EX for the final generated SQL. Considering the ground truth schema set as S_n of the n^{th} query, and the linked schema set as $\hat{S_n}$, EM could be computed by:

$$R = \frac{\sum_{n=1}^{N} \mathbb{I}(\hat{S}_n, S_n)}{N} \tag{4}$$

where $\mathbb{I}(\hat{S}_n, S_n)$ is an indicator function, which can be represented as

$$\mathbb{I}(\hat{S}, S) = \begin{cases} 1, & \hat{S} \supseteq S \\ 0, & \hat{S} \not\supseteq S \end{cases} \tag{5}$$

Precision. Precision P quantifies the accuracy of the linked schema. Considering the ground truth schema set as S_n with length L_n of the n^{th} query, and the linked schema set as $\hat{S_n}$ with length $\hat{L}_n$, precision is computed by:

$$P = \frac{\sum_{n=1}^{N} p_n}{N}, p_n = \frac{\sum_{j=1}^{\hat{L}_n} \mathbb{I}(\hat{s}_j \in S_n)}{\hat{L}_n} \tag{6}$$

F1 score. F1 score $F1$ represents a harmonic mean of recall and precision. It offers an evaluation of schema linking results, taking into account both precision and recall as:

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{7}$$

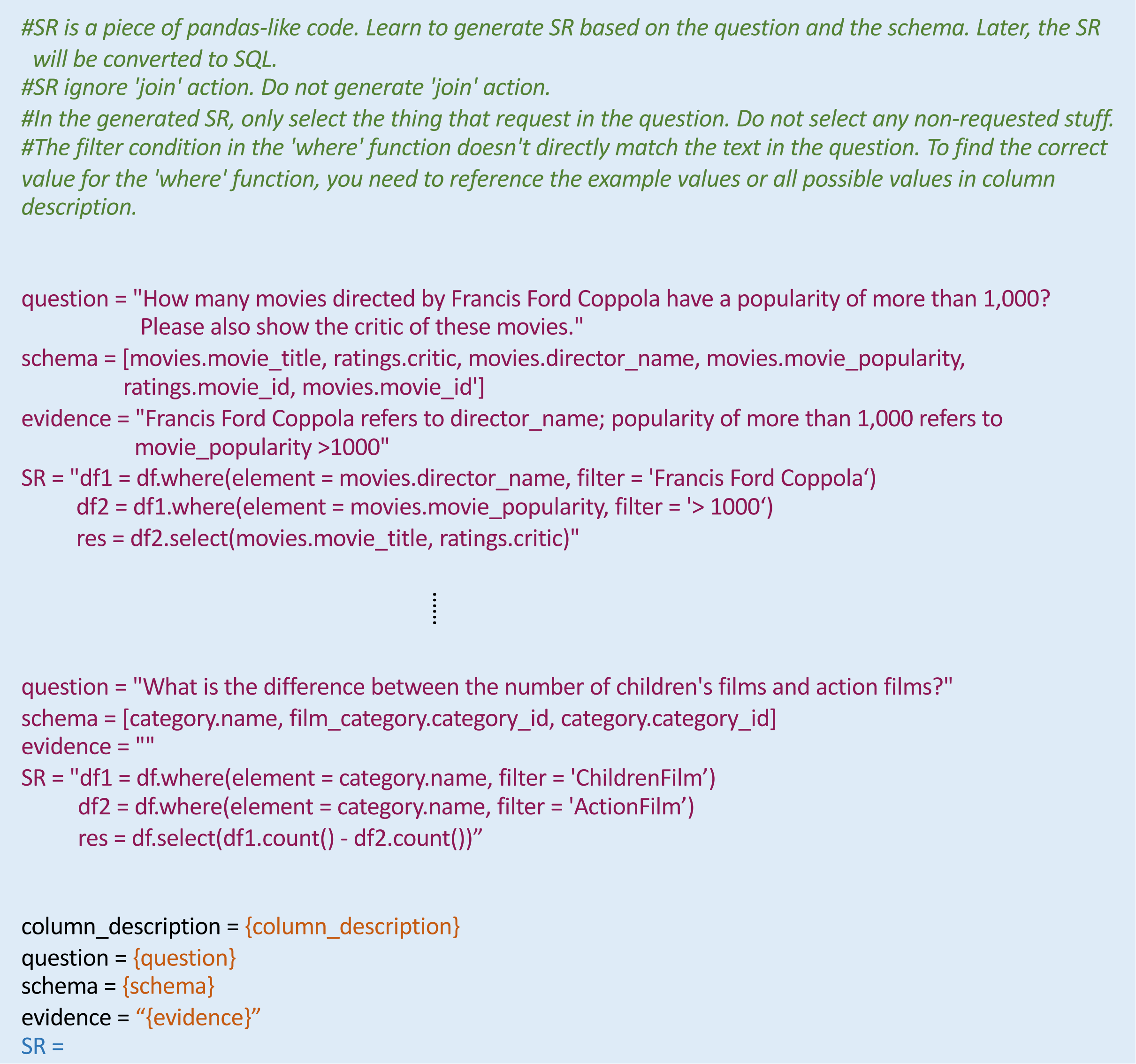
C Implement Details

The open-source models are implemented using `Pytorch`[1], `Transformers`[2], and `vllm`[3]. To

[1] https://pytorch.org/

[2] https://huggingface.co/docs/transformers/en/installation

[3] https://github.com/vllm-project/vllm



```
#SR is a piece of pandas-like code. Learn to generate SR based on the question and the schema. Later, the SR
  will be converted to SQL.
#SR ignore 'join' action. Do not generate 'join' action.
#In the generated SR, only select the thing that request in the question. Do not select any non-requested stuff.
#The filter condition in the 'where' function doesn't directly match the text in the question. To find the correct
value for the 'where' function, you need to reference the example values or all possible values in column
description.


question = "How many movies directed by Francis Ford Coppola have a popularity of more than 1,000?
               Please also show the critic of these movies."
schema = [movies.movie_title, ratings.critic, movies.director_name, movies.movie_popularity,
             ratings.movie_id, movies.movie_id']
evidence = "Francis Ford Coppola refers to director_name; popularity of more than 1,000 refers to
               movie_popularity >1000"
SR = "df1 = df.where(element = movies.director_name, filter = 'Francis Ford Coppola‘)
        df2 = df1.where(element = movies.movie_popularity, filter = '> 1000‘)
        res = df2.select(movies.movie_title, ratings.critic)"

                                          ⋮

question = "What is the difference between the number of children's films and action films?"
schema = [category.name, film_category.category_id, category.category_id]
evidence = ""
SR = "df1 = df.where(element = category.name, filter = 'ChildrenFilm’)
        df2 = df.where(element = category.name, filter = 'ActionFilm’)
        res = df.select(df1.count() - df2.count())”


column_description = {column_description}
question = {question}
schema = {schema}
evidence = “{evidence}”
SR =
```

Figure 7: The prompt of generating symbolic representations.

expedite the inference process, we also implemented deepspeed[4]. The DeepSeek and CodeLlama models are accessed via huggingface[5].

[4] https://github.com/microsoft/DeepSpeed

[5] https://huggingface.co/models