DIVER: A Robust Text-to-SQL System with Dynamic Interactive Value Linking and Evidence Reasoning

YAFENG NAN, Beijing University of Posts and Telecommunications, China
HAIFENG SUN, Beijing University of Posts and Telecommunications, China
ZIRUI ZHUANG*, Beijing University of Posts and Telecommunications, China
QI QI, Beijing University of Posts and Telecommunications, China
GUOJUN CHU, Beijing University of Posts and Telecommunications, China
JIANXIN LIAO, Beijing University of Posts and Telecommunications, China
DAN PEI, Tsinghua University, China
JINGYU WANG*, Beijing University of Posts and Telecommunications, China

In the era of large language models, Text-to-SQL, as a natural language interface for databases, is playing an increasingly important role. State-of-the-art Text-to-SQL models have achieved impressive accuracy, but their performance critically relies on expert-written evidence. This evidence typically clarifies schema and value linking that existing models struggle to identify. Such limitations stem from the ambiguity of user queries and, more importantly, the complexity of comprehending large-scale and dynamic database values. Consequently, in real-world scenarios where expert assistance is unavailable, existing methods suffer a severe performance collapse, with execution accuracy dropping by over 10%. This underscores their lack of robustness due to the reliance on expert assistance.

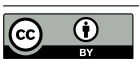
To address this, we propose DIVER, a robust system that ***automates*** evidence reasoning with dynamic interactive value linking. It leverages a *compatible toolbox* containing diverse tools to probe the database. Then, restricted by a *structured workspace (CoTF, Chain of Thoughts and Facts)*, it reflects based on probe results and selects a new tool for next round of probing. Through this automatically iterative process, DIVER identifies schema and value linking missed by existing methods. Based on these accurate linkings, DIVER is able to infer correct usage of SQL functions and formulas and generate high-quality evidence, achieving robust Text-to-SQL without expert assistance. Extensive experiments demonstrate that: 1) The DIVER system significantly enhances the robustness of various Text-to-SQL models, improving performance by up to 10.82% in Execution Accuracy (EX) and 16.09% in Valid Efficiency Score (VES). 2) Our dynamic interactive value linking significantly improves the robustness of existing systems and the accuracy of schema and value linking, especially when confronted with challenges posed by large-scale, dynamic database values.

CCS Concepts: • **Information systems** → **Structured Query Language**; • **Computing methodologies** → *Multi-agent planning*; *Machine translation*.

Additional Key Words and Phrases: Text-to-SQL, Multi-Agent, Large Language Models

*corresponding authors.

Authors' Contact Information: Yafeng Nan, nanyafeng@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Haifeng Sun, hfsun@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Zirui Zhuang, zhuangzirui@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Qi Qi, qiqi8266@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Guojun Chu, chuguojun@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Jianxin Liao, jxlbupt@gmail.com, Beijing University of Posts and Telecommunications, Beijing, China; Dan Pei, peidan@tsinghua.edu.cn, Tsinghua University, Beijing, China; Jingyu Wang, wangjingyu@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China.



ACM 2836-6573/2026/2-ART26
https://doi.org/10.1145/3786640

ACM Reference Format:
Yafeng Nan, Haifeng Sun, Zirui Zhuang, Qi Qi, Guojun Chu, Jianxin Liao, Dan Pei, and Jingyu Wang. 2026. DIVER: A Robust Text-to-SQL System with **D**ynamic **I**nteractive **V**alue Linking and **E**vidence **R**easoning. *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 26 (February 2026), 24 pages. https://doi.org/10.1145/3786640

1 Introduction

Text-to-SQL aims to translate natural language questions (NLQ) into executable SQL queries on relational databases [13, 21, 30]. It has always been regarded as an interface to databases, enabling non-technical users to query data using only natural language. As natural language serves as a core bridge in the current LLM-centric era, Text-to-SQL is playing an increasingly important role. More and more LLM-based agents are using Text-to-SQL interfaces to extract necessary information from databases. For instance, Business Intelligence (BI) and data analysis systems intensively leverage Text-to-SQL tools to query data and generate detailed analytical reports based on them . This places a higher demand on the full-process automation of the Text-to-SQL task [25, 37, 39, 44].

Real-world Text-to-SQL tasks often face challenges such as massive databases, dirty values, ambiguous user questions, and vague column names [7, 14, 20, 22] . A new generation of Text-to-SQL benchmarks, represented by BIRD-bench [20], is dedicated to constructing tasks closer to real-world scenarios, providing a solid data foundation for building practical natural language interfaces for databases. Numerous recent models have achieved remarkable progress on BIRD-bench. To improve Text-to-SQL accuracy, researchers have invested great amounts of effort to improve the underlying model architecture [9, 11, 15, 19, 24, 26, 27, 29, 34, 35].

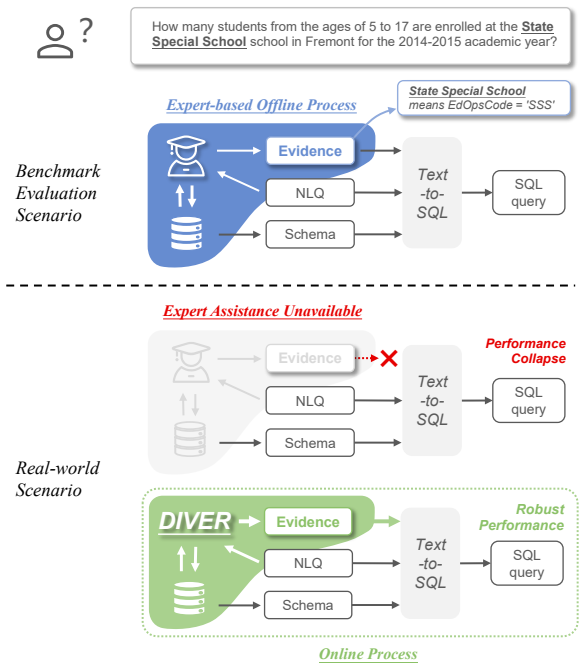


Fig. 1. Existing Text-to-SQL methods utilize an offline expert-written evidence to aid the model, achieving high but unrealistic performance. They suffer a severe performance collapse in real-world scenario where expert is unavailable. DIVER system provides a training-free, online process that acts as an automated expert, generating evidence to achieve robust performance.

However, we find that the impressive performance of existing models heavily relies on auxiliary of expert-written evidence. Specifically, different sizes of models all exhibit a significant performance **collapse** of over 10% in Execution Accuracy (EX) when external expert evidence is unavailable. This reveals that existing Text-to-SQL systems face severe performance degradation in real-world scenarios, as non-technical users cannot provide expert-level insights, and LLM-based agents also cannot rely on human assistance with the demand for high automation. Unfortunately, this heavy reliance on expert-written evidence has been largely overlooked in recent research. According to the latest BIRD leaderboard, among the 52 methods, only 5 methods report performance without evidence in their papers or on the leaderboard. The state-of-the-art among these 5 methods ranks only 33rd on the leaderboard. Therefore, **how existing Text-to-SQL models maintain robustness in real-world scenarios without expert assistance** is a pressing issue to be addressed.

This paper aims to address the critical challenge of maintaining robust Text-to-SQL performance in real-world scenarios without relying on expert-written evidence.

The Key Bottleneck Behind the Performance Collapse. To further explore the reason for such a collapse, we analyzed expert-written evidence on the BIRD-dev dataset, dividing it into VLE (value and schema linking evidence) and SQLE (SQL functions and formulas evidence). As shown in Figure 2, a significant gap exists between current value retrieval methods and expert-written evidence (c → e). Besides, the impact of this gap is profound, as SQL-related evidence is beneficial only when the value evidence is correct (c → d decrease vs. c → f increase). It is also important to note that this gap cannot be bridged by simply scaling up to larger LLMs such as GPT-4o. This is because the fundamental issue is not a deficiency in the model's reasoning capabilities, but rather a lack of access to complete, dynamic database values. Therefore, to achieve robust Text-to-SQL system without expert assistance, the key bottleneck lies in the incomplete value linking strategies. Current methods, relying on simple semantic similarity, often fail to handle ambiguous questions or non-semantic database content. For example, mapping "state special school" to its abbreviation "SSS" is difficult for existing methods.

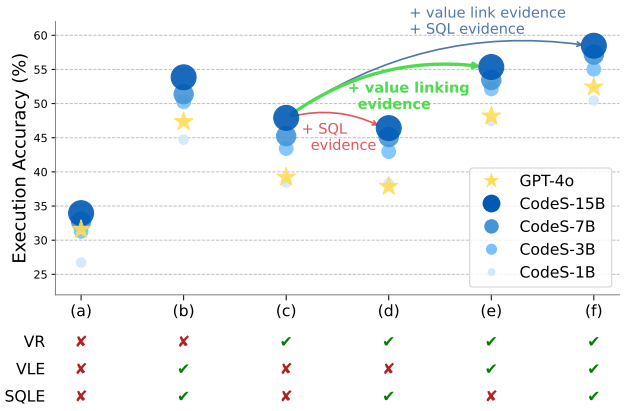


Fig. 2. Fine-grained analysis of expert-written evidence reveals that effective **value linking** is a critical prerequisite for robust Text-to-SQL performance. "VR" indicates the Value Retrieval module of methods. "VLE" represents expert-written value linking and schema linking evidence, and "SQLE" represents expert-written SQL-related Evidence such as function usage.

To tackle these challenges, We propose DIVER, a novel, training-free system that automates high-quality evidence generation through **D**ynamic **I**nteractive **V**alue-linking and **E**vidence **R**easoning.

DIVER mimics the iterative reasoning process of human experts using a multi-agent framework, comprising three key components: 1) the Break up Assistant decomposes complex NLQs into manageable sub-clauses for focused analysis. 2) The Look up Assistant conducts multi-turn interactive value linking and evidence reasoning with the database using a predefined toolbox in a structured Chain of Thoughts and Facts (CoTF) workspace. 3) The Evidence Assistant synthesizes structured information into natural language evidence tailored to different model preferences with style-aligned generation. This design enables DIVER to effectively handle ambiguous queries and large-scale values that are highly diverse and dynamic, thereby generating adaptive evidence that enhances the robustness of existing Text-to-SQL models. Through extensive experiments on 4 benchmarks and different scales of models, we show that DIVER consistently boosts the accuracy of state-of-the-art models. On BIRD benchmark, DIVER improves up to 10.82% in execution accuracy and 16.09% in valid efficiency score.

Our contributions can be summarized as follows:

- We identify and systematically analyze a critical but largely overlooked problem: the performance collapse of Text-to-SQL models in real-world scenarios due to their heavy reliance on expert-written evidence.
- We propose DIVER, a novel, training-free, and model-agnostic system that automates the generation of high-quality evidence by simulating an expert's iterative reasoning and database exploration process.
- We introduce a core methodology of multi-turn interactive value linking, enabled by a structured Chain of Thoughts and Facts (CoTF) workspace and an extensible and compatible toolbox, which robustly grounds the reasoning of evidence in verified database facts.
- We demonstrate through comprehensive experiments that DIVER significantly enhances the robustness of various text-to-SQL models on 4 benchmarks.

2 Related Work

Text-to-SQL. Text-to-SQL systems convert natural language queries (NLQs) to SQL for non-expert database access. Early approaches, such as RATSQL [35] and Global-GNN [1, 2], used graph neural networks for schema-query modeling. Pre-trained models like T5 [31] advanced this via fine-tuning, as in RESDSQL [18] and PICARD [32]. Recent LLM-based methods, including DIN-SQL [27], DAIL-SQL [9], and SQL-specific LLMs [11, 17, 19], leverage few-shot prompting and CoT reasoning. However, these models depend heavily on expert-written evidence, leading to significant accuracy drops (over 10% in Execution Accuracy) without it.

Schema Linking and Value Linking. Text-to-SQL converts natural language questions into executable SQL queries, a process that relies on schema linking (mapping question entities to database schema) and value linking (mapping them to actual database values). Early methods in the pre-trained language model (PLM) era mainly used explicit schema linking techniques such as relationally-aware embeddings [35], while value linking received less attention [3, 12]. With the rise of large language models (LLMs), the field shifted toward implicit linking, where models infer connections from rich schema information and retrieved data by semantic similarity matching [6, 33]. However, they face challenges in retrieving non-semantic values such as abbreviations or numbers. More importantly, semantic similarity does not always reflect logical relevance. For instance, retrieving a single similar value might result in an incorrect WHERE column = 'value' clause, overlooking cases that require a LIKE 'value%' pattern. These issues underscore the need for stronger value linking approaches that can handle the complexity and variability of dynamic, real-world databases.

Multi-agent Collaboration for Database. Multi-agent frameworks are widely used in database tasks. Systems like DB-GPT [2], DAgent [39], and Chat2Query [44] focus on Business Intelligence, treating Text-to-SQL as a black-box tool without enhancing its core process. Other methods target Text-to-SQL, using agents to decompose complex problems and boost accuracy [5, 16, 23, 28, 36, 38, 40]. Yet, incomplete value linking leaves them prone to hallucinations with ambiguous data. Inspired by Embodied AI [41, 43], DIVER introduces a paradigm where agents interact with the environment to explore and validate, enhancing reasoning.

3 Motivation

The strength of this straightforward evidence. As shown in Section 1, the expert-written evidence provided in the BIRD benchmark can significantly enhance model performance. This is because the evidence directly pinpoints complex schema and value links that are challenging for models to discover on their own. In the process of solving Text-to-SQL problems, experts engage in intensive reasoning and verification based on NLQ and database values, ultimately producing evidence that encapsulates critical and concise insights. It helps establish a **practical upper bound** and shows the level of performance achievable when a model is provided with all necessary information. With expert-written evidence, the task could be simplified to a more direct translation into SQL.

Limitations of Expert-written Evidence. However, for Text-to-SQL to be viable as a practical tool integrated into databases or employed by autonomous agents, relying on expert-written evidence

NLQ What is the highest eligible free rate for K-12 students in the schools in Alameda County?

Evidence Eligible free rate for K-12 = `Free Meal Count (K-12)` / `Enrollment (K-12)`

SQL
```
SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)`
FROM frpm
WHERE `County Name` = 'Alameda'
ORDER BY (CAST(`Free Meal Count (K-12)` AS REAL) / `Enrollment (K-12)`)
DESC LIMIT 1
```

Fig. 3. An Example of Evidence and Golden SQL Overlap.

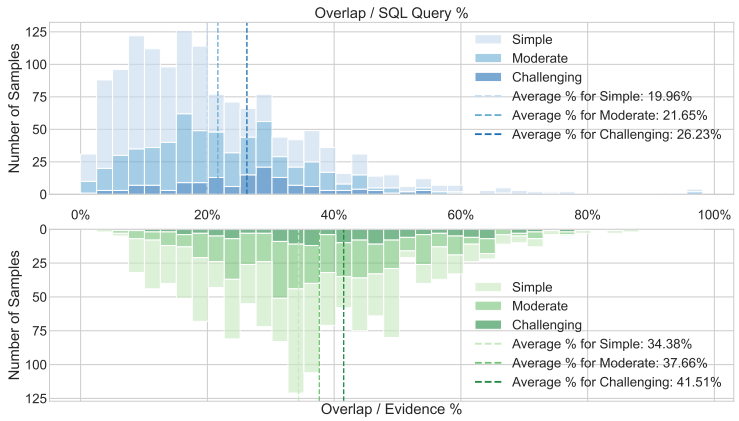


Fig. 4. Distribution of token overlap between expert-written evidence and the golden SQL query on the BIRD-dev dataset. The green plot shows significant answer leakage within the expert-written evidence, while the blue plot shows the degree to which this leakage directly contributes to the final SQL query.

raises questions about the generalizability and robustness. Moreover, the evidence is not conceptual guidance, but often has direct literally overlap with the golden SQL query, as shown in Figure. 3. This is a form of **answer leakage** that should be noticed. More broadly, we have quantified the overlap on the BIRD-dev dataset as shwon in Figure 4, revealing the severe answer leakage between evidence and the golden SQL. Besides, as shown in Table 1, the answer leakage present in the evidence creates a significant gap between the training environment and real-world scenarios. Consequently, models exhibit a substantial performance drop when evaluated in a more realistic, leakage-free setting.

Except for the limitaion that expert-written evidence does not exist in real-world scenarios, there are still other limitations. For example, evidence that is set out for human understanding may not fit with the LLM's own way of thinking. Besides, expert-written evidence is static and cannot fits different models that need different kinds of guidance, which is demonstrated in our experiments. These limitations show the importance of an automated approach to generate high-quality evidence.

Table 1. Performance on simulated real-world scenario that without answer leakage. Performance of models trained with evidence significantly drops, whereas models trained without evidence (Train w/o Evi.) exhibit greater robustness, which reveals unfair evaluation results with answer leakage. EX: Execution Accuracy; VES: Valid Efficiency Score.

	Train w/ Evi. →				Train w/o Evi. →			
	w/ Evi.		w/o Leak.		w/o Evi.		w/o Leak.	
	EX%	VES%	EX%	VES%	EX%	VES%	EX%	VES%
CodeS-1B	50.46	51.07	34.81	40.67	38.46	41.77	37.61	46.23
CodeS-3B	55.02	56.54	40.35	44.46	43.42	44.55	42.57	46.34
CodeS-7B	57.17	58.80	41.72	45.85	45.24	48.13	44.98	51.71

4 System Overview

To address the critical challenge of performance collapse in Text-to-SQL systems in real-world scenarios without expert assistance, we propose DIVER, a novel system for **D**ynamic **I**nteractive **V**alue-linking and **E**vidence **R**easoning. DIVER automates the generation of high-quality, model-adaptive evidence by mimicking the iterative and exploratory reasoning process of human experts. As shown in Figure 5, DIVER adopts a multi-agent collaborative framework, consisting of the Break up Assistant, Look up Assistant, and Evidence Assistant.

The *Break up Assistant* first segments the NLQ into a series of clauses that are semantically and lexically coherent, each designed to contain only a single entity to be explored. On one hand, this is crucial for complex NLQs with multiple intents, as it helps the Look up Assistant focus on single entity for deeper exploration. On the other hand, by decoupling the Break up Assistant with the Lookup Assistant, we address the risk of sub-problem "drifting" during subsequent multi-turn exploration.

The clauses segmented by the Break up Assistant are used to initialize the structured workspace of the Look up Assistant, which is structured along two dimensions: clauses and interaction turns. For each clause, the *Look up Assistant* engages in multi-turn interactive value linking and evidence reasoning with the database using a predefined toolbox. The interaction process includes formulating thoughts and hypotheses (Thought), invoking appropriate tools, receiving feedback from the tools, validating or adjusting hypotheses based on the feedback, and invoking new tools to validate revised thoughts. Once the Look up Assistant uncovers solid factual evidence for each

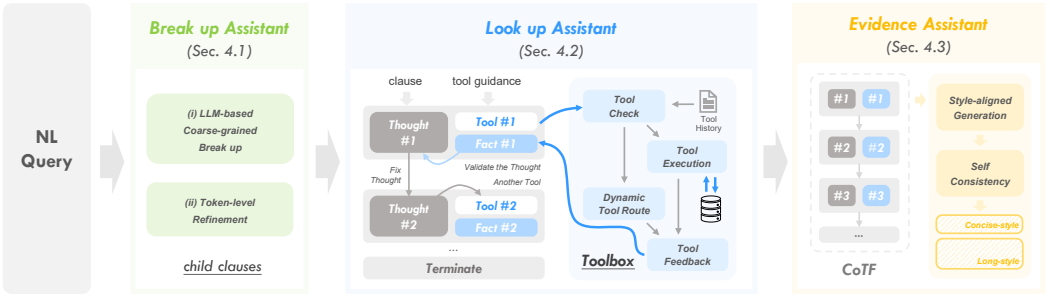


Fig. 5. An Overview of DIVER Workflow.

clause, the entire analysis process terminates, forming a structured, visual, and interpretable trace of reasoning and verification, which we refer to as the **Chain of Thoughts and Facts (CoTF)**.

Finally, the *Evidence Assistant* synthesizes the structured information, which is grounded in solid evidence from the CoTF, into a final piece of natural language evidence. Through style alignment and self-consistency, it can generate evidence in different styles to align with the specific preferences of existing Text-to-SQL models. Existing models can directly capture the necessary information without additional alignment. For instance, SFT-based models that is constrained by input length may prefer shorter and concise evidence, while prompt-based methods may benefit from evidence with richer context for more detailed reasoning.

Another significant advantage of DIVER is that it is a *training-free* system. It leverages the inherent capabilities of LLMs in reasoning and tool usage. This makes DIVER an out-of-the-box system that can be seamlessly integrated into various existing Text-to-SQL systems to enhance their robustness and practicality in real-world scenarios.

5 System Design

5.1 Break up Assistant

Real-world user queries often exhibit a high degree of complexity, which can manifest in diverse syntactic structures, multiple query intents, and semantic ambiguity. Directly addressing such complex queries cause models to lose focus and struggle to conduct in-depth exploration. Therefore, we employ a "divide-and-conquer" strategy: by decomposing the NLQ into a series of sub-queries, the Lookup Assistant is able to perform more targeted reasoning.

We design a **two-stage process** to achieve this. First, an LLM performs a coarse-grained semantic segmentation, aiming for **each segment to correspond to a single SQL condition or entity**. Second, to ensure keeping the subtle semantics of NLQ, a token-level refinement module compares the segmentation result against the original NLQ word-by-word, **correcting any LLM-induced alterations or omissions**. The final segmentation is then used to initialize the Lookup Assistant's structured workspace.

5.2 Look up Assistant

The Look up Assistant is the core component of the DIVER system, designed to mimic the **iterative and exploratory reasoning process** employed by human experts when facing complex problems. It dynamically discovers and validates value linking through multi-turn interactions with the database, ultimately generating a high-quality, fact-grounded reasoning chain for the subsequent Evidence Assistant.

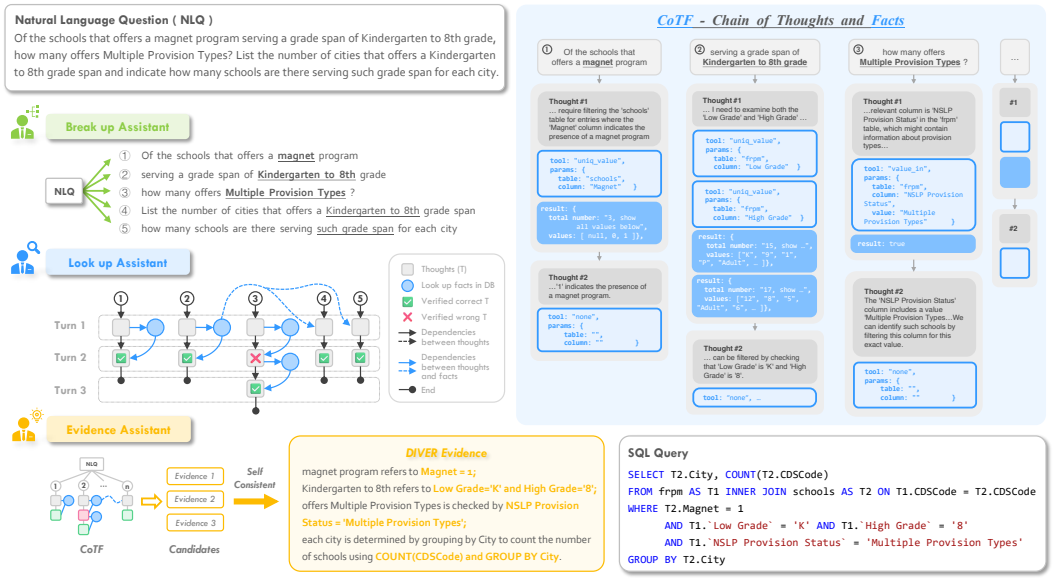


Fig. 6. A Detailed Example of DIVER Workflow and CoTF.

5.2.1 Chain of Thoughts and Facts: A Structured Workspace. The Look up Assistant operates within a structured workspace we term the **Chain of Thoughts and Facts (CoTF)**. As illustrated in Figure 6, CoTF is a JSON object that organizes the reasoning process along two dimensions: the sub-clauses from the Break up Assistant and the multi-turn interactions for each clause. Each interaction step contains a thought, a tool called, and the tool feedback. In the `thought` section, the LLM can freely output with natural language. In the `tool_called` section, the LLM is constrained with structured output to select a tool from our predefined toolbox.

Unlike the free-form and potentially ungrounded reasoning of a standard Chain of Thought (CoT), CoTF enforces a strict, verifiable loop:

(1) Grounded Reasoning: Every thought must be validated by a subsequent tool call that interacts with the database, ensuring the entire process is anchored in factual facts.
(2) Structured Actions: The tool-calling mechanism transforms abstract reasoning into a transparent and traceable sequence of actions, providing a solid evidence chain.

The complete CoTF JSON object is passed back to the LLM after each turn. This provides the Look up Assistant with a global view, enabling information sharing between different sub-tasks. A Fact verified through a tool call in one clause can be directly utilized by others. For instance, in Figure 6, clauses ④ and ⑤ reference entities mentioned in preceding clauses. The Look up Assistant can directly leverage the verified information about the "magnet program" and "grade span" without re-invoking tools.

5.2.2 Dynamic Interactive Value Linking: Thought-Verify-Refine. Based on the CoTF structured workspace, we design a toolbox with various tools to allow the Look up Assistant to explore the database and performing **Dynamic Interactive Value Linking**. This process is a **iterative "dialogue"**. Specifically, for a semantic entity in the NLQ (e.g., "Kindergarten to 8th grade"), the LLM first forms an initial **thought or hypothesis**, such as, "This might correspond to values in the `Low Grade` and `High Grade` column." It then selects the most appropriate tool from the toolbox (e.g.,

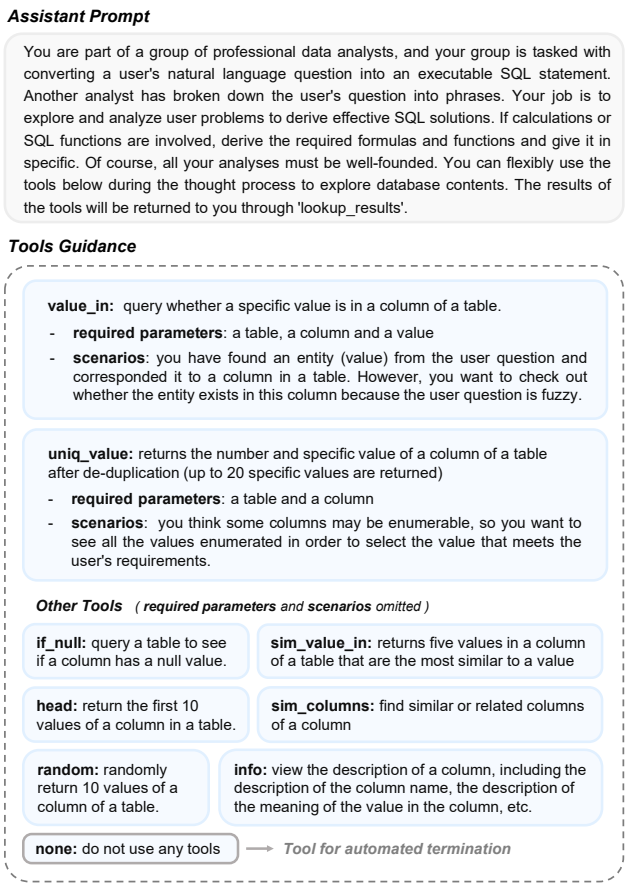


Fig. 7. An Example of Lookup Assistant Prompt and Tool Guidance Prompt

`uniq_value` or `sim_value_in`) to query the database and retrieve a **fact**, a raw value or schema item directly obtained from the database, without any correctness judgment about value linking and schema linking. The value linking or schema linking about this fact will be employed in the next thinking process: if the fact aligns with the user's intent (e.g., finding the value 'K' in the `Low Grade` column and the value '8' in the `High Grade` column), it is promoted to a **verified linking**, which is then summarized as part of the final evidence by evidence assistant. If the fact fails to support the hypothesis, the system uses the new information from this unsuccessful attempt to refine its thoughts, formulate a new hypothesis, and initiate the next round of interaction.

Exploring the Database via a Toolbox. Our tool design philosophy draws inspiration from the Embodied AI paradigm. In Embodied AI, an agent interacts with a physical environment through a series of atomic operations like "observe" and "grasp" to progressively understand the environment and complete tasks. This approach of decomposing complex tasks into atomic operations enhances system interpretability and extensibility. The Text-to-SQL task naturally possesses an "environment" to interact with (i.e., the database), and a universal interface (i.e., SQL). Unlike many open-domain QA tasks, the content retrieved from a database via SQL is **objective, truthful, and verifiable**, providing ideal conditions for constructing a reliable perception-action loop.

We designed a series of tools, each considered an "atomic operation" for interacting with the database, collectively defining the action space of the Look up Assistant. The benefits are twofold:

- **Tool as a Guider:** Tools transform the ambiguous goal of "understanding the database", which is difficult to describe precisely with prompts, into a series of concrete, executable, and verifiable steps. By understanding and autonomously calling these tools, the LLM naturally accomplishes the exploration and comprehension of the database.
- **Tool as an Explainer:** The chain of tool calls clearly records the evolution of the model's thoughts, making the process of value linking and evidence generation transparent and interpretable. Compared to the highly unconstrained CoT, a chain of tool calls is also easier to analyze and intervene in.

Specifically, as shown in Figure 7, we design eight tools covering three major functions: Value Probing (exact match, semantic similarity, and reverse linking from database values to NLQ), Data Inspection (sampling and null-value checks), and Schema Exploration (retrieving metadata and discovering related columns). For example, when a user asks about a "state special school", a semantic similarity model may fail to match its abbreviation "SSS". However, by inspecting the column content with `uniq_value` and retrieving metadata by `info`, DIVER can discover "SSS" and infer their relevance. As shown in Figure 7, to enable the LLM to understand and correctly use these tools, we have meticulously designed three parts of guidance for each: a brief functional description, a list of required parameters, and a detailed description of applicable scenarios. In our ablation study, we validate the critical role of these three guidance components for system performance.

Robust Self Correction via Tool Feedback. Although the structured workspace and toolbox reduce the occurrence of LLM hallucinations, the model may still err when generating tool parameters (like table or column names) or get stuck in ineffective reasoning loops when facing complex problems. To address this, we designed an **in-loop feedback mechanism** that endows the system with powerful self-correction capabilities and robustness. This mechanism includes three types of feedback:

- **Standard Feedback:** When a tool is called for the first time and executes successfully, the system returns its execution result. In the next turn of thought, the LLM can use this new "fact" to confirm or refine its previous hypothesis.
- **Corrective Feedback:** When a tool call fails due to incorrect parameters (e.g., a non-existent table or column name), the system captures the exception and feeds the error message back to the LLM. This mechanism enables the system to automatically recover from common LLM hallucinations without complex hard-coded rules. According to our statistics on the BIRD-dev dataset, this mechanism successfully corrected 94 erroneous tool calls.
- **Guiding Feedback:** When the system detects that the model is repeatedly calling the same tool (with identical tool and parameters), it provides a "Route Guide" based on our predefined rules. For instance, for repeated calls to `value_in`, the feedback might be, "This tool has been called before. Consider using `sim_value_in` for a fuzzy search." Such repetitive behavior typically indicates that the model has entered an unproductive reasoning loop. Owing to the structured nature of CoTF, this state can be explicitly identified and and corresponding guidance can be provided. Since the suggestion is fed back as a contextual prompt, it preserves the assistant's decision-making freedom while effectively steering it out of the stagnated state.

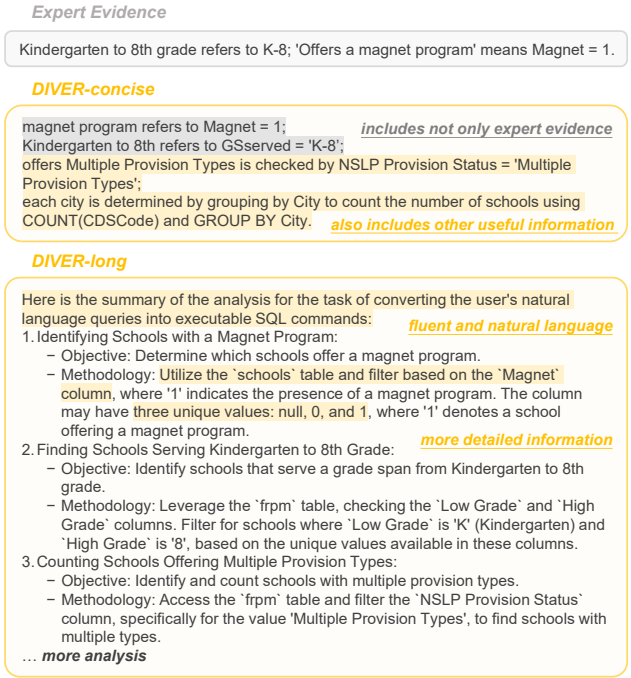


Fig. 8. An Example of Different Evidence Styles of DIVER.

5.3 Evidence Assistant

After the Look up Assistant's process, the CoTF contains a wealth of accurate, verified, but highly structured information. However, existing Text-to-SQL models exhibit different preferences for the textual style of evidence due to variations in their model architectures and training details. For example, for a method like ChatGPT+CoT that relies on internal reasoning, overly concise evidence may degrade performance by mismatching the LLM's thought process, whereas evidence containing detailed analysis and verification might be more effective. Conversely, for models fine-tuned on the BIRD dataset (like CodeS), verbose evidence could lead to a performance drop due to distribution mismatch.

Evidence Assistant is designed to **bridge the gap between the structured CoTF and the specific style of evidence required by the downstream Text-to-SQL model**. We designed two evidence generation styles, as shown in Figure 8:

- **DIVER-long:** A **long-form evidence** style that includes detailed analysis, thoughts, and verification processes. Its language is fluent and natural, making it more suitable for models that require rich context for reasoning.
- **DIVER-concise:** A **concise evidence** style that closely emulates the original expert-written evidence in BIRD, focusing on directly stating the core conclusions about value linking, schema linking, and function usage.

Style-aligned Generation. To achieve style-aligned evidence generation, we employ a few-shot learning approach. In the Style-aligned Generation module, we provide the Evidence Assistant with five carefully constructed "CoTF → concise evidence" pairs as in-context examples. Since the LLM's task is primarily to learn and imitate the format and linguistic style of the concise evidence, rather

than performing complex reasoning from scratch, a 5-shot context is sufficient for it to accurately generalize to new CoTF data.

Self-Consistency. In the process of synthesizing the CoTF into natural language evidence, the LLM itself may produce hallucinations, such as omitting critical information from the CoTF or adding unsubstantiated details. Following common self-consistency strategies, we generate three different evidence. These three versions are then compared, and they supplement one another to produce a final, comprehensive version of the evidence.

6 Experiments

6.1 Experimental Setup

6.1.1 Datasets. We conduct our main experiments on 4 text-to-SQL benchmarks, including two widely used datasets, BIRD [20] and Spider [42], along with two more challenging datasets for robustness evaluation, DR.Spider [4] and Spider-DK [8]. As DIVER is a training-free system, we directly evaluate it with the development sets of BIRD and Spider, containing 1,534 and 1,034 samples respectively. ***Spider-dev*** comprises 166 databases spanning over 100 diverse domains, yet it is relatively basic and simple with small-scale database and clear user queries and database schemas. In contrast, ***BIRD*** poses greater challenges, particularly in terms of database scale (each database contains on average around 549K rows, compared to Spider's 2,000 rows), SQL function usage, and the presence of dirty values. ***Spider-DK*** and ***DR.Spider*** are derived from the Spider dataset by applying transformations and perturbations to the original samples, simulating real-world text-to-SQL scenarios in order to evaluate model's robustness. Spider-DK contains 535 transformed questions. DR.Spider is more challenging, featuring **17 carefully designed perturbation subsets** covering databases, user questions, and SQL queries, enabling a comprehensive assessment of model robustness across diverse conditions.

6.1.2 Evaluation Mertics. For all benchmarks, we apply **Execution accuracy (EX)** to evaluate whether the predicted and ground-truth SQL queries yield the same execution results on specific database instances. For the BIRD benchmark, **Valid Efficiency Score (VES)** is a special metric to evaluate SQL query performance by considering both accuracy and execution speed. Yet, preliminary experiments indicated that VES could be highly susceptible to fluctuations based on varying hardware, software, and system status. Hence, for BIRD, EX still serves as the stable and dependable metric.

6.1.3 Models. To evaluate the performance improvement brought by our DIVER system to existing Text-to-SQL methods, we selected the following SOTA (State-of-the-Art) models with various scales: the CodeS series (1B, 3B, 7B, 15B) [19], the XiYanSQL-QwenCoder series (3B, 7B, 14B) [11], DIN-SQL [27], and DAIL-SQL [10].

CodeS and **XiYanSQL-QwenCoder** are both open-source models in the text-to-SQL domain, achieving SOTA among models of comparable sizes. **SFT-CodeS** is a version of the CodeS model that has been subsequently fine-tuned on the BIRD dataset or Spider dataset. We use the corresponding fine-tuned version directly to evaluate DIVER-generated evidence on BIRD-dev and Spider-dev. For Spider-DK and DR.Spider, we follow the experimental setup of the original paper [19] and utilize the Spider-version SFT-CodeS for evaluation. In contrast, XiYanSQL-QwenCoder is a in-context-learning-based model trained on mixed datasets, which could be utilized with prompt and few shots for all datasets. **DIN-SQL** and **DAIL-SQL** are widely compared methods built upon closed-source large-scale foundation models such as GPT-4. As GPT-4o now outperforms GPT-4 while being more cost-effective, we adopt GPT-4o as the foundation model for these two methods in our experiments.

Table 2. Performance enhancement on SOTA Text-to-SQL models with DIVER system. "Original" denotes the baseline performance without BIRD evidence of each method. "+ DIVER-long" and "+ DIVER-concise" represent the results with two sytles of DIVER-generated evidence. Blue texts indicate the specific improvement of DIVER over the baseline.

Method	Date	Original		+ DIVER-long		+ DIVER-concise	
		EX%	VES%	EX%	VES%	EX%	VES%
with Close-source Foundation Models							
GPT-4o	2024-08	32.14	33.42	42.96 (+10.82)	49.51 (+16.09)	42.57 (+10.43)	44.58 (+11.16)
DIN-SQL + GPT-4o	2023-12	38.14	45.06	44.72 (+6.58)	50.11 (+5.05)	44.26 (+6.12)	51.24 (+6.18)
DAIL-SQL + GPT-4o	2024-01	36.96	41.67	45.31 (+8.35)	**54.30** (+12.63)	**46.22** (+9.26)	50.51 (+8.84)
with Open-source Foundation Models							
SFT CodeS-1B	2023-10	38.46	41.77	43.48 (+5.02)	45.57 (+3.80)	44.33 (+5.87)	48.76 (+6.99)
SFT CodeS-3B	2023-10	43.42	44.55	46.81 (+3.39)	48.11 (+3.56)	48.63 (+5.21)	49.93 (+5.38)
SFT CodeS-7B	2023-10	45.24	48.13	49.48 (+4.24)	51.61 (+3.48)	49.41 (+4.17)	52.43 (+4.3)
SFT CodeS-15B	2023-10	47.91	49.60	**50.91** (+3.00)	**53.15** (+3.55)	49.74 (+1.83)	52.96 (+3.36)
XiYanSQL-QwenCoder-3B	2025-04	35.27	38.78	43.02 (+7.75)	49.98 (+11.2)	39.11 (+3.84)	44.18 (+5.40)
XiYanSQL-QwenCoder-7B	2025-04	39.63	40.32	43.35 (+3.72)	48.90 (+8.58)	41.33 (+1.70)	41.44 (+1.12)
XiYanSQL-QwenCoder-14B	2025-04	41.33	41.95	46.02 (+4.69)	49.78 (+7.83)	44.26 (+2.93)	47.66 (+5.71)
Code Unavailable Methods (just as a reference)							
ExSL + granite-20b-code	2024-05	51.69	56.11	-	-	-	-
AskData + GPT-4o	2025-03	65.91	63.34	-	-	-	-

To utilize DIVER evidence in these models, we uniformly replaced the expert-written evidence from the BIRD dataset with the long-style or concise-style evidence generated by DIVER system. For Spider, Spider-DK, and DR.Spider, we directly concatenate the DIVER evidence to the user question. For most models, their foundation models possess large context length, allowing them to directly process the long-style evidence. However, CodeS presents a special case. During its schema filtering, it utilizes the BERT model to retrieve relevant schemas with both NLQ and evidence. Since the length of long-style evidence exceeds BERT's context length limit, when evaluating long-style evidence on CodeS, we use only the NLQ for schema filtering and incorporated the DIVER-generated evidence during the SQL generation stage.

6.1.4 Implementation Details. For all 4 benchmarks, SQLite serves as the database engine for hosting and management. However, we design and implement DIVER system in a compatible manner. It can be adapted to different SQL engines with rewriting less than 100 lines of code.

The specific version of the model we have chosen for DIVER is `gpt-4o-2024-08-06`. For the `sim_value_in` and `sim_columns` tools, we employ the BERT model to assess semantic similarity. For the Breakup, Lookup, and Evidence Assistants, we set the temperature to 0.2, 0.7, and 0.7, respectively. During the dynamic interactive value linking process, we set the maximum number of turns to 5, at which point the exploration is forcibly terminated to prevent infinite loops. For the Evidence Assistant, we use 5 shots for the style alignment module and 3 candidates for self-consistency.

All experiments are employed on a server equipped with 1 NVIDIA A100 (80G) GPU, 1 NVIDIA A800 (80G) GPU, 40 Intel Xeon Silver 4210R CPUs, and the Ubuntu 22.04.1 operating system.

6.2 Performance Evaluation

6.2.1 Accuracy Improvement on BIRD. In this section, we conducted experiments on the BIRD-dev and report two metrics: Execution Accuracy (EX) and Valid Execution Score (VES). For each

Table 3. Improvement of DIVER-concise on DR.Spider benchmark. For every perturbation (row), the top three performers are marked with 🏆, 🥉, and 🎗. For every method (column), red cells (■, ■, ■) highlight three lowest scores. The change after applying DIVER is shown with arrows (↑, ↓). Cell colors represent the magnitude of improvement: ■ for 0-2%, ■ for 2-5%, ■ for 5-10%, and ■ for 10+%.

Type	Perturbation	# Samples	RESDSQL-3B +NatSQL	ChatGPT +ZeroNL2SQL	SFT-CodeS-1B		SFT-CodeS-3B		SFT-CodeS-7B		SFT-CodeS-15B	
					EX%	+ DIVER	EX%	+ DIVER	EX%	+ DIVER	EX%	+ DIVER
DB	schema-synonym	2619	68.3	69.8🎗	58.5	65.7 ↑ 7.2	64.3	69.2 ↑ 4.9	67.2	73.9🥉 ↑ 6.7	66.9	74.0🏆 ↑ 7.1
	schema-abbreviation	2853	70.0	74.8	68.6	71.5 ↑ 2.9	75.0	77.4 ↑ 2.4	76.8	79.7🥉 ↑ 2.9	78.7🎗	80.2🏆 ↑ 1.5
	Dbcontent-equivalence	382	40.1	56.8🎗	53.9	58.4🏆 ↑ 4.5	47.9	55.0 ↑ 7.1	46.9	58.4🏆 ↑ 11.5	47.6	57.6🥉 ↑ 10.0
	Average	-	59.4	67.1	60.3	65.2 ↑ 4.9	62.4	67.2🎗 ↑ 4.8	63.6	70.7🏆 ↑ 7.1	64.4	70.6🥉 ↑ 6.2
NLQ	keyword-synonym	953	72.4	74.0🏆	60.9	65.1 ↑ 4.2	70.9	70.4 ↓ 0.5	73.0🎗	72.6 ↓ 0.4	73.5🥉	72.2 ↓ 1.3
	keyword-carrier	399	83.5	88.2	86.5	88.5 ↑ 2.0	91.2🎗	88.5 ↓ 2.7	91.5🥉	90.2 ↓ 1.3	91.7🏆	89.2 ↓ 2.5
	column-synonym	563	63.1	62.7	56.0	58.1 ↑ 2.1	60.0	63.8🥉 ↑ 3.8	63.2	63.8🥉 ↑ 0.6	64.7🏆	63.6🎗 ↓ 1.1
	column-carrier	579	63.9	71.7	67.4	68.4 ↑ 1.0	74.4	73.8 ↓ 0.7	80.7🏆	80.0🥉 ↓ 0.7	79.1	79.3🎗 ↑ 0.2
	column-attribute	119	71.4	70.6	47.9	64.7 ↑ 16.8	67.2	73.1🎗 ↑ 5.9	63.0	77.3🥉 ↑ 14.3	68.9	79.0🏆 ↑ 10.1
	column-value	304	76.6🏆	76.0🎗	72.4	71.1 ↓ 1.3	75.0	71.4 ↓ 3.6	73.7	72.0 ↓ 1.7	76.3🥉	75.3 ↓ 1.0
	value-synonym	506	53.2	70.6	59.7	66.2 ↑ 6.5	67.0	66.8 ↓ 0.2	72.7🏆	71.2 ↓ 1.6	71.9🎗	72.5🥉 ↑ 0.6
	multitype	1351	60.7	66.4	57.5	61.2 ↑ 3.7	66.5	65.8 ↓ 0.7	69.5🏆	68.7🎗 ↓ 0.8	69.4🥉	67.8 ↓ 1.6
	others	2819	79.0	79.4	74.9	74.7 ↓ 0.2	78.5	78.0 ↓ 0.5	81.5🏆	80.1🎗 ↓ 1.4	81.2🥉	80.0 ↓ 1.2
	Average	-	69.3	73.2	64.8	68.7 ↑ 3.9	72.3	72.4 ↑ 0.1	74.3	75.1🎗 ↑ 0.8	75.2🥉	75.4🏆 ↑ 0.2
SQL	comparison	178	82.0🏆	73.6	60.7	70.2 ↑ 9.5	69.7	72.5 ↑ 2.8	77.5🥉	74.7🎗 ↓ 2.8	71.9	71.9 =
	sort-order	192	85.4🏆	80.2	69.8	72.9 ↑ 3.1	79.2	81.3 ↑ 2.1	81.8	83.9🎗 ↑ 2.1	84.9🥉	83.9🎗 ↓ 1.0
	nonDB-number	131	85.5	92.4🏆	84.7	84.0 ↓ 0.7	87.8	84.7 ↓ 3.1	90.1🥉	88.6🎗 ↓ 1.6	84.0	87.8 ↑ 3.8
	DB-text	911	74.3	80.7🏆	67.1	74.1 ↑ 7.0	77.2	76.6 ↓ 0.6	80.5🥉	80.4🎗 ↓ 0.2	80.7🏆	79.9 ↓ 0.8
	DB-number	410	88.8🏆	86.1🎗	80.5	76.6 ↓ 3.9	85.1	86.1🎗 ↑ 1.0	84.9	86.8🥉 ↑ 1.9	85.9	85.6 ↓ 0.3
	Average	-	83.2🏆	82.6	72.6	75.6 ↑ 3.0	79.8	80.2 ↑ 0.4	83.0🥉	82.9🎗 ↓ 0.1	81.5	81.8 ↑ 0.3
All	**Global average**	-	71.7	74.9	66.3	70.1 ↑ 3.8	72.8	73.8 ↑ 1.0	75.0	76.6🏆 ↑ 1.6	75.1🎗	76.5🥉 ↑ 1.4

baseline model, we report its original performance without evidence assistance and the enhanced performance after applying two variants of DIVER-generated evidence, **DIVER-long** and **DIVER-concise**. The results are summarized in Table 2. DIVER provides significant improvements across all models. Key observations include:

(1) **Consistent Gains Across Scales:** DIVER-long and DIVER-concise deliver consistent performance gains for models ranging from small-scale (e.g., CodeS-1B) to large-scale closed-source SOTA models (e.g., GPT-4o), with a maximum improvement of 10.82%.

(2) **Style-specific Strengths:** DIVER is capable of automatically generating different forms of evidence tailored to different models. Experimental results show that DIVER-long achieves superior performance for prompt-based models as well as larger-scale models (e.g., CodeS-15B), as these models can leverage the richer information contained in long-style evidence. In contrast, DIVER-concise is more suitable for smaller-scale models (e.g., CodeS-1B), which typically exhibit lower inherent generalization ability and higher sensitivity to format alignment.

6.2.2 Robustness Improvement. In this section, we evaluate the DIVER's improvement on robustness on three datasets: Spider-dev, Spider-DK, and DR.Spider. Table 4 and Table 3 show the results. In summary, while DIVER is less impactful for over-simplified datasets like Spider-dev, it brings significant robustness improvements under complex perturbations in Spider-DK and DR.Spider, with marked benefits for both large and small models.

Due to the Spider-dev's simplicity such as small databases, simple schemas, and direct value linking, most baseline models already achieve high accuracy on it (e.g., SFT-CodeS-15B: 84.9%), leaving little room for improvement. Therefore, introducing additional DIVER evidence to Spider-dev occasionally causes slight performance drops, suggesting that it may become unnecessary or even distracting to apply DIVER to overly simple environments.

Table 4. Performance comparison on Spider-dev and Spider-DK with DIVER-long and DIVER-concise.

Method	Spider-dev			Spider-DK		
	Original	+ DIVER-long	+ DIVER-concise	Original	+ DIVER-long	+ DIVER-concise
DIN-SQL + GPT-4o	82.9	79.4 ↓ 3.5	82.6 ↓ 0.3	55.9	69.3 ↑ 13.4	66.0 ↑ 10.1
DAIL-SQL + GPT-4o	83.3	80.8 ↓ 2.5	82.9 ↓ 0.4	69.7	71.6 ↑ 1.9	72.8 ↑ 3.1
XiYanSQL-QwenCoder-3B	82.8	79.4 ↓ 3.4	80.5 ↓ 2.3	72.3	74.2 ↑ 1.9	73.8 ↑ 1.5
SFT CodeS-1B	77.9	79.0 ↑ 1.1	78.9 ↑ 1.0	64.7	69.0 ↑ 4.3	68.8 ↑ 4.1
SFT CodeS-3B	83.4	83.4 =	83.4 =	71.8	76.1 ↑ 4.3	73.6 ↑ 1.8
SFT CodeS-7B	85.4	84.6 ↓ 0.8	84.9 ↓ 0.5	72.0	76.6 ↑ 4.6	75.0 ↑ 3.0
SFT CodeS-15B	84.9	83.8 ↓ 1.1	85.0 ↑ 0.1	70.7	74.8 ↑ 4.1	74.4 ↑ 3.7

In contrast, baseline models face severe accuracy drops on Spider-DK and DR.Spider due to more complex variations and perturbations. For example, DB perturbations such as renaming a column or changing field values poses challenge for existing value-linking strategies. As shown in Table 3, SFT-CodeS-15B only achieves 47.6% of EX under DBcontent-equivalence perturbation of DR.Spider, revealing that current text-to-SQL systems struggle with dynamic database in real-world settings.

However, DIVER yields large improvements on difficult perturbations. Across models, DIVER sets new state-of-the-art scores for multiple perturbations, especially in DB-level perturbation, confirming the effectiveness of the dynamic interactive value linking in handling dynamic databases. Furthermore, for small-scale models like SFT-CodeS-1B that inherently lack robustness and generalization, DIVER strongly improves its performance of global average from only 66.3% to 70.1% (Table 3), narrowing the gap with much larger 3B models.

While minor drops occur in a few perturbations where baselines are already strong, DIVER consistently improves the average performance, demonstrating its own robustness that it provides helpful information without introducing excessive noise.

6.2.3 Dynamic Interactive Value Linking vs. Existing Value Linking. To conduct a detailed comparison between the dynamic interactive value linking in the DIVER system and existing methods, we designed a rigorous evaluation focusing on linking accuracy. For each sample, we extract schema items (tables and columns) and values from both the ground-truth SQL and the predicted SQL, forming a golden set of entities and a predicted set of entities. To ensure the accuracy of this extraction, we utilize sqlparse library to parse SQL queries. We then compute the F1-score for both schema linking and value linking as follows: **Precision** is the fraction of correctly predicted entities out of all predicted entities ($\frac{|\text{Golden Set} \cap \text{Predicted Set}|}{|\text{Predicted Set}|}$), measuring the accuracy of the predictions. **Recall** is the fraction of correctly predicted entities out of all entities in the golden set ($\frac{|\text{Golden Set} \cap \text{Predicted Set}|}{|\text{Golden Set}|}$), measuring the completeness of the predictions. The **F1-score** is the harmonic mean of these two metrics ($2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$), providing a single, balanced measure of performance, which is reported finally.

We compare DIVER against two representative schema linking and value linking techniques:

- CodeS - Schema Filter and Value Retriever: Uses a classifier for schema pruning and a "coarse-to-fine" BM25+LCS search for values.
- XiYanSQL - M-Schema: Employs LLM-based retrieval for keywords and column selection via few-shot pruning.
- CodeS + Unique Values: an additional baseline to further investigate the role of value-related evidence. Our abaltion study shows that uniq_value tool plays an important role, therefore, in this setup, we augment the CodeS retriever with up to 20 unique values for each retrieved column, following DIVER's uniq_value setup.

Table 5. Comparison of F1-Scores for schema and value linking across different methods and difficulty levels. The best results are **emphasized**.

Method	Simple		Moderate		Challenging		Overall	
	Schema	Value	Schema	Value	Schema	Value	Schema	Value
DIVER	**88.06**	**84.29**	**84.89**	72.13	**83.22**	**72.87**	**86.64**	**79.53**
CodeS	87.23	84.03	83.66	**73.21**	80.26	67.58	85.49	79.21
+ uniq	78.88	82.65	77.63	72.54	75.56	66.58	78.19	78.07
XiYanSQL	80.74	75.42	75.14	62.13	72.69	64.53	78.28	70.37

For DIVER and CodeS, we utilize the SFT-CodeS-3B as the base Text-to-SQL model. For XiYanSQL, we utilize the XiYanSQL-QwenCoder-3B. The experiment is employ on the BIRD-dev dataset. All SQL queries were generated in an expert-free setting, without access to expert-written evidence.

As shown in Table 5, the experimental results fully demonstrate the effectiveness of dynamic interactive value linking. The results show that DIVER's interactive value linking surpasses existing methods in both schema and value linking accuracy in most cases, with only a slight decline in value linking on "moderate" questions.

Notably, DIVER achieves a particularly significant advantage on "challenging" questions. This is because challenging problems in BIRD often feature ambiguous user queries alongside highly complex data structures, where values and schema are difficult to comprehend even with description. These makes it difficulty for methods that solely rely on semantic similarity. In contrast, DIVER's interactive approach allows it to explore the database dynamically and deeply understand both the ambiguous user intent and the complex data formats.

As for CodeS + Unique Values, the augmented unique values not only failed to close the gap with DIVER but also degraded CodeS's performance, especially on schema linking. This stems from two factors: (1) the retrieval module of CodeS cannot exploit unique values to filter ambiguous or semantically weak column names due to context length constraints, and (2) large amounts of irrelevant unique values, especially from non-categorical columns (e.g., IDs), introduce noise that impact both linking and SQL generation. This underscores DIVER's core advantage: rather than injecting more and more values, it distills them into concise, noise-free evidence that directly benefits SQL generation.

6.3 Ablation Study

To analyze the contribution of each key component within **DIVER** system, we conduct a thorough ablation study. To ensure experimental efficiency while maintaining result reliability, experiments in ablation study were performed on a smaller, representative subset of the BIRD-dev, which we refer to as `small-dev`. This subset was constructed via stratified sampling 10% data from the full BIRD-dev set, preserving the original data distribution with respect to question difficulty and different database. Besides, we employ SFT-CodeS-3B and DIVER-concise for the ablation experiments, unless stated otherwise. We organize the ablation study of the DIVER system around its three main components: the Breakup Assistant, the Lookup Assistant, and the Evidence Assistant.

6.3.1 Ablation of Breakup Assistant. Here, we quantify the impact of two key components: the **Breakup Assistant** as a whole, and its internal **Token-level Refinement** module. We compare the performance of the complete DIVER system against two ablated versions: (1) **w/o Break up Assistant**, where we bypass the decomposition process and use the original, complete NLQ in all subsequent steps. This means the Lookup Assistant now process the entire NLQ, which is more

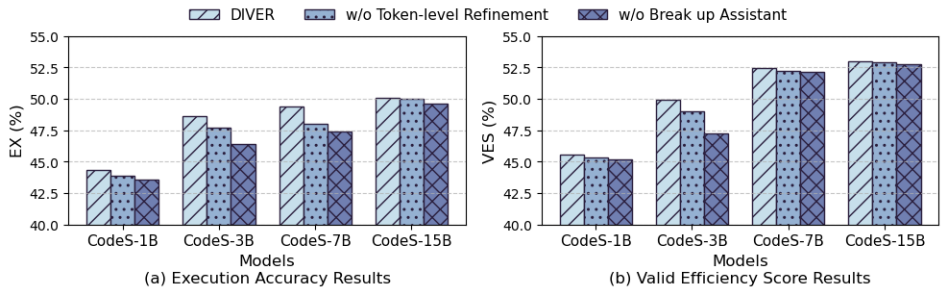


Fig. 9. Abaltion Results of Break up Assistant. DIVER-concise is utilized in this experiment.

complex. So we set the maximum number of turns to 10 to ensure the process is not terminated before all necessary information can be explored. For the Evidence Assistant, we constructs 5 style-alignment shots based on the complete NLQ to ensure the style-alignment performance; (2) **w/o Token-level Refinement**, where we use the sub-clauses generated by the LLM without postprocessing. This means the sub-clauses may contain errors such as omissions or unintended alterations of the original NLQ. The experiments are conducted on the `SFT-CodeS` model family, from 1B to 15B, to assess the impact across different model scales.

Figure 9 presents the results, both ablated versions lead to a noticeable degradation in performance across all model sizes. The removal of the Break up Assistant causes the most significant performance drop, reaching up to over 2%, which represents almost half of the total gains brought by full DIVER. This underscores the critical importance of the NLQ decomposition stage. The Token-level Refinement module also proves to be a valuable contributor. Without it, LLM hallucinations can cause the generated sub-clauses to omit or alter words from the original user question. By applying token-level refinement, the hallucination issue is effectively solved, resulting in a corresponding improvement in performance.

6.3.2 Ablation of Lookup Assistant. The Lookup Assistant is central to DIVER's ability to ground its reasoning in the database's actual content. In this section, we conduct two sets of ablation studies to analyze its key components, including tools, route guidance, tool prompt, and CoTF.

Ablation of Toolbox. Toolbox is the core component that enables Lookup Assistant to interact with the database. It includes 8 tools with prompts for description, parameters, and usage scenarios, plus a dynamic routing mechanism to avoid unproductive reasoning loops. Ablation study analyzes these components' impacts. The `none` tool, which serves as a stop signal for the assistant to terminate exploration, is not considered as an exploration tool thus excluded from this ablation study.

We examine tool calling counts across experiments to detect substitutions between tools. By linking changes in usage and performance, we assess each tool's significance and irreplaceability.

Toolbox is the core component that facilitates the Lookup Assistant interacting with the database. It comprises 8 distinct tools for exploration, each accompanied by a tool prompt including its description, required parameters, and applicable scenarios to help the LLM understand its function. The `none` tool, which serves as a stop signal for the assistant to terminate exploration, is not considered as an exploration tool thus excluded from this ablation study. Furthermore, the toolbox incorporates a dynamic routing mechanism that that steers the LLM away from ineffective reasoning loops and towards more productive tools. Accordingly, our ablation study of the toolbox is designed to analyze the impact of these specific components.

To better illustrate the utilization of each tool, we analyze the calling counts of each tool in every experiments. An interesting aspect of this analysis is that by ablating a single tool and observing the corresponding shifts in the calling counts of others, we can identify the tools act as substitutes. By correlating these shifts in tool usage with the overall change in performance, we can precisely evaluate the significance and irreplaceability of each tool.

The results are presented in Table 6. *Ablation of single tool* shows the impact of removing each tool individually, including removing its prompt and removing it from callable tools. We observe that removing any single tool leads to a degradation in both EX and VES scores, demonstrating the effectiveness of the Lookup Assistant. This suggests the difficulty of understanding the database with single value retrieval strategy. Moreover, the varying frequencies of tool calling highlight the autonomous decision-making capability of the Lookup Assistant. Compared to static combinations of tools for information retrieval, dynamic and self-directed tool calling provides more precise information.

The most critical tools appear to be **`sim_columns`** and **`uniq_value`**. Removing `uniq_value` results in the largest performance drop in EX (from 48.70 to 44.81), underscoring the importance of reverse value linking. Through `uniq_value`, the assistant gains insight into the values contained within a column, enabling it to map these values back to entities mentioned in the natural language query (NLQ). This is particularly important when the user intent is ambiguous or database values lack sufficient semantic. Besides, we observe a notable increase in the calling of tools such as `value_in`, `head`, `random`, and `if_null`, indicating that the assistant attempts to compensate by gathering more information through alternative tools, which also leads to an increase in interaction turns (from 2.38 to 2.67). Nevertheless, both EX and VES scores still drop significantly, further emphasizing the irreplaceable role of `uniq_value`.

Similarly, removing `sim_columns` leads to the same degradation in EX score, demonstrating its central role in the DIVER system. As discussed in the Method section, when the assistant lacks a clear plan for the next step, it often falls into inefficient reasoning loops—such as repeatedly querying the same column even when no useful information is returned. In addition to using the tool router to guide reasoning, `sim_columns` serves

Table 6. Ablation study of the toolbox. The bar chart shows the distribution of counts of calling each tool. The tools, from left to right, are: ① `value_in`, ② `sim_value_in`, ③ `uniq_value`, ④ `head`, ⑤ `random`, ⑥ `if_null`, ⑦ `info`, and ⑧ `sim_columns`.

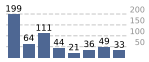
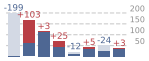
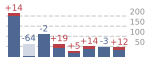
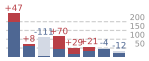
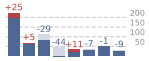
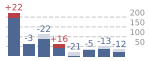
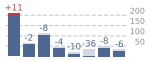
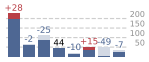
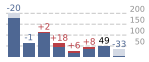
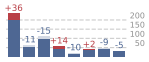
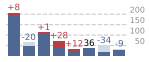
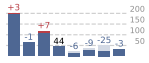
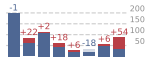
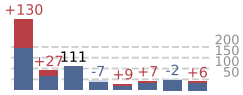
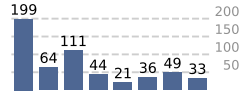
	#Tool Calling	#Turns	EX%	VES%
DIVER		2.38	48.70	50.49
Ablation of single tool				
w/o value_in		2.57	48.05	50.15
w/o sim_value_in		2.81	47.40	48.11
w/o uniq_value		2.67	44.81	46.73
w/o head		2.66	45.45	46.91
w/o random		2.70	48.05	48.45
w/o if_null		2.62	46.75	48.03
w/o info		2.70	46.10	46.36
w/o sim_columns		2.68	44.81	46.32
Ablation of tool route and prompt				
w/o tool route		2.68	48.70	51.95
w/o description		2.76	48.05	48.98
w/o parameter		2.50	45.45	46.74
w/o scenario		2.21	44.81	47.98

Table 7. Ablation study of the CoTF (Chain of Thoughts and Facts). Table 6 explains the bar chart.

Components		#Tool Calling	#Turns	EX%	VES%
CoT	CoF				
✗	✗	-	-	30.52	31.17
✓	✗	-	-	32.47	33.12
✗	✓		3.31	47.40	48.37
✓	✓		2.38	48.70	50.49

as a guidance tool itself. Based on the idea of "tool as a guider", its presence reminds the assistant to switch columns for querying. When `sim_columns` is removed, the frequencies of other tools do not significantly increase, possibly because the assistant becomes trapped in a reasoning loop, and chooses to terminate the process after receiving repeated calling feedback about redundant tool use.

Other tools also reveal valuable insights. For example, ablating `value_in` has limited impact on EX and VES, as tools like `sim_value_in` can almost entirely substitute it in value retrieval. Nevertheless, we retain `value_in` in the DIVER system because it is more efficient and lightweight compared to `sim_value_in`. Although `sim_value_in` has its limitations, it remains indispensable as a fuzzy search tool that supports broader value retrieval, especially when column values are not enumerable (e.g., usernames or geographic names). The removal of `if_null` also results in a noticeable performance drop. We observe an increase in `value_in` calling (+11), likely as a partial compensation. However, this number still falls short compared to the original 36 calling of `if_null`, suggesting that the replacement is inadequate. This again illustrates the importance of tool itself that serve as guides in the reasoning process.

Overall, our results confirm the effectiveness of leveraging tools as both guiders and explainers. Furthermore, they show that each tool is crucial for complete understanding of database values, requiring multiple retrieval strategies to work together in a complementary fashion.

Ablation of tool route and prompt in Table 6 shows the impact of route mechanism and each aspect of tool prompt. Disabling `tool route` does not decrease the EX score and even increase the VES to 51.95. However, it results in an increase in calling `value_in` and decrease other calling of tools, showing a more concentrated distribution of tool-calling. The most severe degradation comes from removing the `scenario` examples, underscoring the importance of it for guiding the complex tool-use behavior. We can also observe that, the calling of tools becomes more more uniform, indicating the assistant's insufficient understanding of the tools.

Ablation of the CoTF. Chain of Thought and Fact (CoTF) integrates LLM reasoning (*Thought*) with grounded evidence from tool use (*Fact*). To validate the effectiveness of CoTF, we compare four distinct configurations as shown in Table 7.

- **Full DIVER (`CoT` ✓, `CoF` ✓):** Our proposed system, which combines reasoning and tool-based fact-checking. It achieves the highest performance (48.70% `EX`, 50.49% `VES`) with the greatest efficiency (2.38 average turns).

Table 8. Ablation study of components in Evidence Assistant (EA). "SA" denotes style-alignment module.

	Format	#Avg Length	EX%	VES%
DIVER-long	NL	373	**49.35**	**52.02**
w/o EA	json	763	43.51	45.09
DIVER-concise	NL	71	**48.70**	**50.49**
w/o SA	NL	124	45.45	46.59
w/o EA	NL template	322	41.56	43.40
	json	174	42.21	44.27

- **w/o CoT (`CoT` ×, `CoF` ✓):** In this setting, we disable the model's ability to output its reasoning chain ("thought") and force it to directly select a tool. The performance drops significantly. Crucially, the model resorts to a less efficient, brute-force exploration strategy, evidenced by the sharp increase in both total tool calls (+130) and average turns (3.31). This demonstrates that the reasoning step is vital for formulating an efficient and effective exploration plan.
- **w/o CoF (`CoT` ✓, `CoF` ×):** This configuration represents a traditional Chain-of-Thought approach where the model can reason but cannot use tools to verify facts or explore the database. While it can generate a plausible-sounding plan, it cannot ground its reasoning in the actual database content, often leading to unexecutable SQL due to factual inaccuracies (e.g., hallucinating column or value names).
- **Baseline (`CoT` ×, `CoF` ×):** This is a standard text-to-SQL baseline where the model generates SQL directly from the NLQ and schema, guided by a few-shot prompt to ensure output format compliance. This approach lacks any interactive exploration of facts and deep thinking, achieving the lowest performance.

In summary, the results powerfully demonstrate the synergistic relationship between reasoning and fact-checking. Removing either component cripples the system: without thought (`CoT`), exploration becomes inefficient and reactive; without facts (`CoF`), reasoning becomes ungrounded and prone to hallucination. Our CoTF architecture successfully leverages both to achieve superior performance and efficiency.

6.3.3 Ablation of Evidence Assistant. Evidence Assistant aims to synthesize structured CoTF into different styles of summary, providing high-quality, contextualized evidence for the final Text-to-SQL model. To validate the effectiveness of it, we conducted a comprehensive ablation study, with results presented in Table 8.

We conducted ablation studies for two target evidence styles: long and concise. For the long-style evidence, we design a setting without the Evidence Assistant (DIVER-long w/o EA), where the JSON-formatted CoTF is passed directly to the Text-to-SQL model.

For the concise-style evidence, the DIVER system includes a style-alignment module (SA) designed to ensure that the generated evidence aligns with the expert-written evidence. To ablate the SA module, we prompt the Evidence Assistant to generate evidence as concise as possible (DIVER-concise w/o SA). Besides, we also ablate the Evidence Assistant for concise-style evidence. We explored two alternative approaches. First, leveraging the structured nature of CoTF, we design concise natural language (NL) templates for each tool. For example, the template for the `uniq_value` tool is `("Table 'table' column 'column' has total unique values, samples: ', '.join(values).")`, where table and column are parameters for calling the tool, and total and values are the results returned by the tool. Second, we directly extracted the "facts" section from CoTF, which consists of the JSON for all tool calls and their return results. It is important to note that this setup differs from the

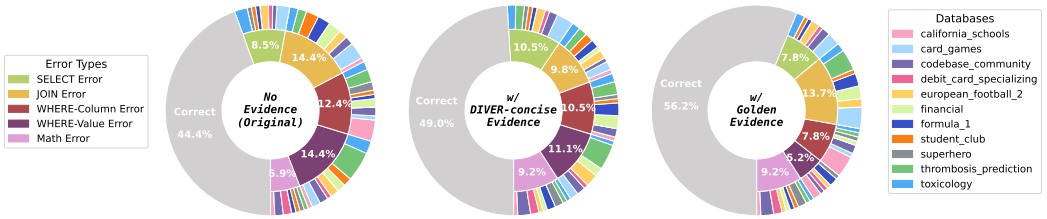


Fig. 10. Distribution of error types for baseline model, DIVER-concise, and upper bound.

w/o CoT setting in the ablation study of CoTF. In this setting, the Lookup Assistant is allowed to output the CoT part, but we only use the CoF to generate SQL, whereas w/o CoT does not allow the Lookup Assistant to output the CoT part at all.

The experimental results demonstrate the effectiveness of the Evidence Assistant. For the long style evidence, summarizing the CoTF into natural language yields significantly more compact evidence, reducing the length by half compared to the raw JSON CoTF. Furthermore, the raw JSON CoTF is not limited to necessary information. It also including noise from trial-and-error steps, presenting a greater challenge for the Text-to-SQL model, leading to a sharp decline in performance.

For the concise style evidence, the style-alignment module achieves the most streamlined evidence (average length of only 71) while maintaining high quality. In contrast, prompting the model to be concise is unstable, resulting in a longer average length (124) and a slight drop in both EX and VES scores. As for the two w/o EA settings, they suffer from a similar problem as the DIVER-long w/o EA setting: neither the NL template nor the raw fact JSON can filter out irrelevant content from the CoTF. Even with a carefully designed concise NL template, the average length still reached 322, as it included substantial noise that negatively impacted performance.

7 Error Analysis

To further understand the strengths and limitations of DIVER, we conducted a detailed error analysis on BIRD-small-dev set used in ablation study, with 154 samples in total. As illustrated in Figure 10, the analysis compares the error distribution of the baseline model, the model augmented with DIVER-concise evidence, and an upper-bound performance with manually crafted golden evidence.

The primary observation is that DIVER significantly reduces the most critical and frequent errors. The baseline model's failures are concentrated in JOIN errors (14.4%) and WHERE-Value errors (14.4%). With DIVER's assistance, JOIN errors are substantially cut down to 9.8%, and WHERE-Value errors are reduced to 11.1%. This demonstrates DIVER's effectiveness in providing crucial evidence for establishing complex table relationships and performing accurate value linking, leading to an overall accuracy boost from 44.4% to 49.0%.

However, a performance gap still exists between DIVER and the Golden Evidence upper bound. Database-level error distribution reveals that many of the persistent errors are in databases that require deep domain-specific knowledge, such as financial, thrombosis_prediction, and toxicology. This suggests that the remaining errors stem from a lack of understanding of specialized domain logic, which represents a direction for future work.

8 Conclusion

We present DIVER, an expert-free Text-to-SQL system that automates evidence via dynamic interactive value-linking and reasoning. Using a toolbox and Chain of Thoughts and Facts (CoTF) workspace, DIVER grounds thoughts in database facts, handling ambiguous queries and complex

values. Experiments show up to 10.82% accuracy gains, with interactive linking outperforming static methods. This enhances real-world robustness and motivates future work on advanced value-linking and expert-free systems.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grants (62471055, 62321001, U23B2001, 62101064, 62171057, 62201072), the High-Quality Development Project of the MIIT(2440STCZB2584), the Ministry of Education and China Mobile Joint Fund (MCM20200202, MCM20180101), the Project funded by China Postdoctoral Science Foundation (2023TQ0039, 2024M750257, GZC20230320), the Fundamental Research Funds for the Central Universities (2024PTB-004), the 2025 Education and Teaching Reform Project Funding at Beijing University of Posts and Telecommunications (2025YZ005).

References

[1] Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 4560–4565. doi:10.18653/v1/P19-1448

[2] Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global Reasoning over Database Structures for Text-to-SQL Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3659–3664. doi:10.18653/v1/D19-1378

[3] Ursin Brunner and Kurt Stockinger. 2021. ValueNet: A Natural Language-to-SQL System That Learns from Database Information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 2177–2182. doi:10.1109/ICDE51399.2021.00220

[4] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. 2023. Dr.Spider: A Diagnostic Evaluation Benchmark towards Text-to-SQL Robustness. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=Wc5bmZZU9cy

[5] Minghang Deng, Ashwin Ramachandran, Canwen Xu, Lanxiang Hu, Zhewei Yao, Anupam Datta, and Hao Zhang. 2025. ReFoRCE: A Text-to-SQL Agent with Self-Refinement, Format Restriction, and Column Exploration. arXiv:2502.00675 [cs] doi:10.48550/arXiv.2502.00675

[6] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Lu Chen, Jinshu Lin, and Dongfang Lou. 2023. C3: Zero-Shot Text-to-SQL with ChatGPT. arXiv:2307.07306 [cs] doi:10.48550/arXiv.2307.07306

[7] Yujian Gan, Xinyun Chen, and Matthew Purver. 2021. Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 8926–8931. doi:10.18653/v1/2021.emnlp-main.702

[8] Yujian Gan, Xinyun Chen, and Matthew Purver. 2021. Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 8926–8931. doi:10.18653/v1/2021.emnlp-main.702

[9] 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:2308.15363 [cs] doi:10.48550/arXiv.2308.15363

[10] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.* 17, 5 (Jan. 2024), 1132–1145. doi:10.14778/3641204.3641221

[11] Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, Jinyang Gao, Liyu Mou, and Yu Li. 2025. A Preview of XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. arXiv:2411.08599 [cs] doi:10.48550/arXiv.2411.08599

[12] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 4524–4535. doi:10.18653/v1/P19-1444

[13] George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A Survey on Deep Learning Approaches for Text-to-SQL. *VLDBJ* 32, 4 (July 2023), 905–936. doi:10.1007/s00778-022-00776-8

[14] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2024. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. In *ICLR 2025 Oral*. arXiv. arXiv:2411.07763 [cs] doi:10.48550/arXiv.2411.07763

[15] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024. The Dawn of Natural Language to SQL: Are We Fully Ready? *Proc, VLDB Endow,* 17, 11 (July 2024), 3318–3331. doi:10.14778/3681954.3682003

[16] Boyan Li, Jiayi Zhang, Ju Fan, Yanwei Xu, Chong Chen, Nan Tang, and Yuyu Luo. 2025. Alpha-SQL: Zero-Shot Text-to-SQL Using Monte Carlo Tree Search. arXiv:2502.17248 [cs] doi:10.48550/arXiv.2502.17248

[17] Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. OmniSQL: Synthesizing High-Quality Text-to-SQL Data at Scale. arXiv:2503.02240 [cs] doi:10.48550/arXiv.2503.02240

[18] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. In *AAAI*. arXiv. arXiv:2302.05965 [cs] doi:10.48550/arXiv.2302.05965

[19] Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. CodeS: Towards Building Open-Source Language Models for Text-to-SQL. In *Proceedings of the ACM on Management of Data (SIGMOD'24, Vol. 2)*. 127:1–127:28. doi:10.1145/3654930

[20] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2024. Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, 42330–42357.

[21] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A Survey of NL2SQL with Large Language Models: Where Are We, and Where Are We Going? arXiv:2408.05109 [cs] doi:10.48550/arXiv.2408.05109

[22] Kyle Luoma and Arun Kumar. 2025. SNAILS: Schema Naming Assessments for Improved LLM-based SQL Inference. *SIGMOD* 3, 1 (Feb. 2025), 77:1–77:26. doi:10.1145/3709727

[23] Peixian Ma, Boyan Li, Runzhi Jiang, Ju Fan, Nan Tang, and Yuyu Luo. 2024. A Plug-and-Play Natural Language Rewriter for Natural Language to SQL. arXiv:2412.17068 [cs] doi:10.48550/arXiv.2412.17068

[24] Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The Death of Schema Linking? Text-to-SQL in the Age of Well-Reasoned Language Models. arXiv:2408.07702 [cs] doi:10.48550/arXiv.2408.07702

[25] Parth Parikh, Oishik Chatterjee, Muskan Jain, Aman Harsh, Gaurav Shahani, Rathin Biswas, and Kavi Arya. 2022. Auto-Query - a Simple Natural Language to SQL Query Generator for an e-Learning Platform. In *2022 IEEE Global Engineering Education Conference (EDUCON)*. 936–940. doi:10.1109/EDUCON52537.2022.9766617

[26] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. 2024. CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. arXiv:2410.01943 [cs] doi:10.48550/arXiv.2410.01943

[27] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-Context Learning of Text-to-SQL with Self-Correction. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, 36339–36348.

[28] Mohammadreza Pourreza and Davood Rafiei. 2024. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. arXiv:2402.01117 [cs] doi:10.48550/arXiv.2402.01117

[29] 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 *EMNLP 2022*. arXiv. arXiv:2205.06983 [cs] doi:10.48550/arXiv.2205.06983

[30] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. *TKDE* (Aug. 2022). arXiv:2208.13629 [cs] doi:10.48550/arXiv.2208.13629

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs] doi:10.48550/arXiv.1910.10683

[32] 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*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 9895–9901. doi:10.18653/v1/2021.emnlp-main.779

[33] Ruoxi Sun, Sercan O. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. 2023. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL. arXiv:2306.00739 [cs] doi:10.48550/arXiv.2306.00739

[34] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2024. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. In *COLING 2025*. arXiv. arXiv:2312.11242 [cs] doi:10.48550/arXiv.2312.11242

[35] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2021. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *ACL 2020*. arXiv. arXiv:1911.04942 [cs] doi:10.48550/arXiv.1911.04942

[36] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break It down: A Question Understanding Benchmark. *TACL* (Jan. 2020). arXiv:2001.11770 [cs] doi:10.48550/arXiv.2001.11770

[37] Yang Wu, Yao Wan, Hongyu Zhang, Yulei Sui, Wucai Wei, Wei Zhao, Guandong Xu, and Hai Jin. 2024. Automated Data Visualization from Natural Language via Large Language Models: An Exploratory Study. *Proc, ACM Manag, Data* 2, 3 (May 2024), 115:1–115:28. doi:10.1145/3654992

[38] Xiangjin Xie, Guangwei Xu, Lingyan Zhao, and Ruijie Guo. 2025. OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-Shot and Consistency Alignment. arXiv:2502.14913 [cs] doi:10.48550/arXiv.2502.14913

[39] Wenyi Xu, Yuren Mao, Xiaolu Zhang, Chao Zhang, Xuemei Dong, Mengfei Zhang, Jun Zhou, and Yunjun Gao. 2025. DAgent: A Relational Database-Driven Data Analysis Report Generation Agent. arXiv:2503.13269 [cs] doi:10.48550/arXiv.2503.13269

[40] Yicun Yang, Zhaoguo Wang, Yu Xia, Zhuoran Wei, Haoran Ding, Ruzica Piskac, Haibo Chen, and Jinyang Li. 2025. Automated Validating and Fixing of Text-to-SQL Translation with Execution Consistency. *SIGMOD* 3, 3 (Feb. 2025).

[41] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR 2023 (ICLR)*. arXiv. arXiv:2210.03629 [cs] doi:10.48550/arXiv.2210.03629

[42] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. 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*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 3911–3921. doi:10.18653/v1/D18-1425

[43] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. 2024. Building Cooperative Embodied Agents Modularly with Large Language Models. In *The Twelfth International Conference on Learning Representations (International Conference on Learning Representations)*. arXiv. arXiv:2307.02485 [cs] doi:10.48550/arXiv.2307.02485

[44] Jun-Peng Zhu, Peng Cai, Boyan Niu, Zheming Ni, Kai Xu, Jiajun Huang, Jianwei Wan, Shengbo Ma, Bing Wang, Donghui Zhang, Liu Tang, and Qi Liu. 2024. Chat2Query: A Zero-Shot Automatic Exploratory Data Analysis System with Large Language Models. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 5429–5432. doi:10.1109/ICDE60146.2024.00420

Received July 2025; revised October 2025; accepted November 2025