ORANGE: An Online Reflection ANd GEneration framework with Domain Knowledge for Text-to-SQL

Yiwen Jiao[1*], Tonghui Ren[2*], Yuche Gao[3*], Zhenying He[1 ✉], Yinan Jing[1], Kai Zhang[1], and X.Sean Wang[1]

[1] Fudan University, China
ywjiao24@m.fudan.edu.cn, {zhenying,jingyn,zhangk,xywangCS}@fudan.edu.cn
[2] Tencent Cloud, China rayeeren@tencent.com
[3] University of Cambridge, United Kingdom yg473@cam.ac.uk

Abstract. Large Language Models (LLMs) have demonstrated remarkable progress in translating natural language to SQL, but a significant semantic gap persists between their general knowledge and domain-specific semantics of databases. Historical translation logs constitute a rich source of this missing in-domain knowledge, where SQL queries inherently encapsulate real-world usage patterns of database schema. Existing methods primarily enhance the reasoning process for individual translations but fail to accumulate in-domain knowledge from past translations. We introduce ORANGE, an online self-evolutionary framework that constructs database-specific knowledge bases by parsing SQL queries from translation logs. By accumulating in-domain knowledge that contains schema and data semantics, ORANGE progressively reduces the semantic gap and enhances the accuracy of subsequent SQL translations. To ensure reliability, we propose a novel nested Chain-of-Thought SQL-to-Text strategy with tuple-semantic tracking, which reduces semantic errors during knowledge generation. Experiments on multiple benchmarks confirm the practicality of ORANGE, demonstrating its effectiveness for real-world Text-to-SQL deployment, particularly in handling complex and domain-specific queries.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in translating natural language questions into executable SQL queries, significantly lowering the barrier for non-technical users to interact with databases [25, 26, 30, 32]. However, a critical challenge is the semantic gap between the general-purpose knowledge embedded in LLMs and the domain-specific semantics of the target database schema.

Unlike general text generation, Text-to-SQL requires in-domain reasoning that often goes beyond the database schema information alone. Crucial knowl-

---

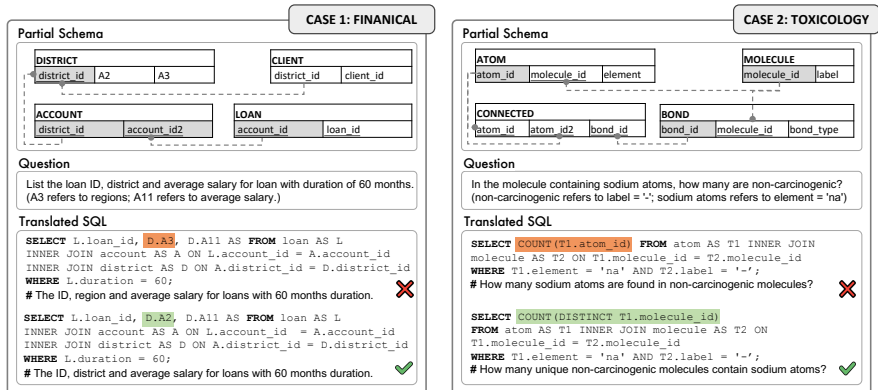* Equal contribution   ✉ Corresponding author

Fig. 1. A Text-to-SQL example.

edge about schema semantics, value distributions, and business logic is difficult to infer from limited schema definitions and is frequently absent from user questions. Consequently, this semantic gap leads to systematic errors, such as misinterpretation of column meanings or operational intent, resulting in SQL queries that are syntactically valid but semantically incorrect [8, 17]. As shown in Figure 1, Case 1 illustrates that when processing the query *average salary in each district*, a model lacking in-domain knowledge erroneously uses the *regions* column A3 instead of the correct *districts* column A2. In Case 2, the ATOM INNER JOIN MOLECULE operation produces an intermediate table, where each tuple no longer represents the intended molecule, but instead an atom associated with its molecule. This shift in tuple semantics causes aggregation errors, such as incorrectly using COUNT(ATOM.id) to count atoms rather than the correct operation COUNT(DISTINCT ATOM.molecule_id) to count molecules.

To effectively utilize LLMs for Text-to-SQL, the dominant paradigm focuses on enhancing the reasoning process of LLMs through techniques such as optimized prompting [20, 26, 37], task decomposition [43], refinement [6, 9], self-consistency voting [30, 32], and test-time scaling [15, 25]. While these methods improve the SQL translation performance, they lack a mechanism to acquire and accumulate domain-specific insights from the target database. Each query is treated as an isolated task, thus failing to leverage past translation experiences. Recent studies highlight the importance of in-domain knowledge and attempt to incorporate it by generating synthetic domain-specific question or leveraging historical logs [8, 10, 25]. However, synthetic data often misalign with real user intents and can introduce hallucinations, while log-based approaches typically require extensive manual annotation.

A more desirable solution is to create a Text-to-SQL system capable of using its translation logs to achieve evolution without human intervention [33, 40]. In this paper, we present ORANGE (**O**nline **R**eflection **AN**d **GE**neration), a self-evolutionary framework that parses and stores validated knowledge into database-specific memory, which is then leveraged for subsequent in-context learning process. ORANGE consists of three core components: **Knowledge**

Decomposition, **Knowledge Validation**, and knowledge-enhanced **Text-to-SQL Translation**. SQL operations such as `JOIN` and `Aggregation` alter tuple semantics, resulting in complex reasoning [12]. To accurately capture these shifts, we propose a nested Chain-of-Thought (CoT) approach that decomposes SQL queries into subcomponents for progressive annotation and explicitly tracks tuple-semantic shifting. This design ensures that the generated knowledge units faithfully reflect the database-specific semantics. Unlike prior methods, ORANGE relies solely on SQL queries from translation logs without original user queries, annotating the semantics of knowledge through tuple-semantic tracking and constructing a database-specific knowledge base. These verified in-domain knowledge can be reused to guide future predictions and improve the translation accuracy of ORANGE over time without manual intervention.

We conduct extensive experiments on three benchmarks [38, 19, 41] to evaluate ORANGE. The results show consistent accuracy improvement over baselines, indicating the robustness of the proposed self-evolutionary framework for Text-to-SQL. Our contributions are as follows:

1. We propose a self-evolutionary Text-to-SQL paradigm that accumulates and reuses in-domain knowledge without human intervention.
2. We introduce ORANGE, an online reflection and generation framework that constructs a reliable, domain-specific knowledge base from translation logs through nested CoT strategy with tuple-semantic tracking.
3. We conduct extensive experiments to validate the effectiveness of ORANGE.

2   Methods

2.1   Methodology Overview

Our work operates as a self-evolutionary framework that uses the translation logs to construct a reusable, in-domain knowledge base. To translate a natural language question X into a target SQL query Y based on the database schema $\mathcal{S}$, ORANGE parses the historical translated SQL queries $\mathcal{C}$ and maintains a memory $\mathcal{M}$ of verified $k = (k_x, k_y)$, where each unit consists of a Text-SQL pair. The system evolves $\mathcal{M}$ through successive translations, progressively enriching its domain understanding.

As illustrated in Figure 2, ORANGE operates with three stages:

1. **Knowledge Decomposition**: Extracting generated SQL queries from the logs, the PARSER adopts a nested CoT approach to parse SQL queries into knowledge units, each consisting of a semantically-aligned Text-SQL pair.
2. **Knowledge Validation**: The VALIDATOR verifies the correctness of knowledge units through probability scores, only retaining the reliable ones in $\mathcal{M}$.
3. **Knowledge-Enhanced Translation**: The CODER performs SQL translation as an in-domain In-Context Learning (ICL) task. It retrieves relevant demonstrations from $\mathcal{M}$, while incorporating multi-path generation with the self-consistency strategy to further enhance reliability.
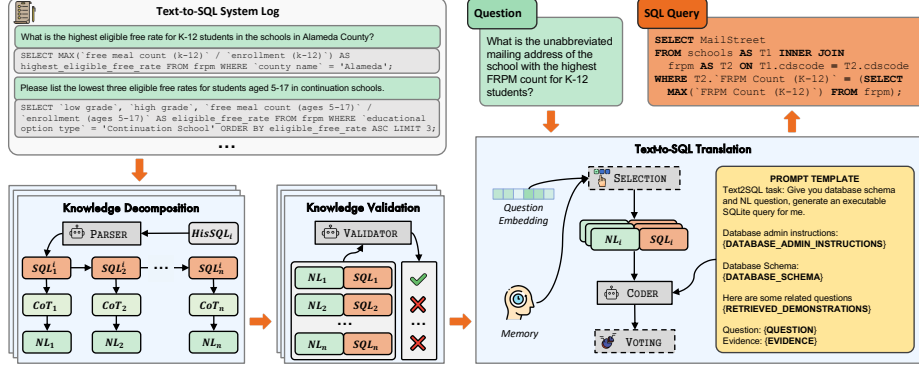
Fig. 2. Overview of ORANGE.

2.2   Knowledge Decomposition

This stage analyzes existing Text-to-SQL translation logs, which contain the generated candidate SQL queries for both the current question and previous tasks. For each question $\mathcal{X}$, its candidate SQL queries $\mathcal{C}$ are clustered by their execution results and are ranked by cluster size. We then select the first SQL query in each cluster to represent that cluster, denoted as $\mathcal{C}'$.

For each SQL in $\mathcal{C}'$, we employ a nested Chain-of-Thought (CoT) strategy that decomposes the SQL into sub-SQL components k_y and generates the corresponding question k_x. The resulting knowledge units, each is defined as a Text-SQL pair $k = (k_x, k_y)$, are then de-duplicated to serve as the output.

Nested CoT Reasoning We propose a nested Chain-of-Thought (CoT) strategy for SQL-to-SUBSQL-to-Text (SST) annotation. The approach uses an outer loop (SQL-to-SUBSQL) and an inner loop (SUBSQL-to-Text), which decomposes and interprets complex SQL step by step for better understanding.

In the outer loop, the PARSER decomposes the SQL in a *least-to-most* style. Simple queries with fewer operations form building blocks for more complex queries. This incremental approach allows partial reuse of earlier semantics. For each question X, the outer loop decomposes the candidate SQL C_i as:

$$P = \text{PARSER}(C_i, S, X_e), \tag{1}$$

where X_e is the evidence used in the original question. We only use X_e (not the entire X) to avoid leakage. The generated P is a sequence of triplets of the form (k_y^i, k_c^i, k_x^i) for $i = 1, \ldots, n$, where $\{k_y^i\}$ are parsed sub-SQL components from C_i, $\{k_c^i\}$ are the CoT reasoning content, and $\{k_x^i\}$ are the corresponding natural language questions.

In the inner loop, we apply CoT to each parsed SQL component. The PARSER focuses on tuple semantic shifts by tracking how each operation changes tuple semantics, which helps to interpret SQL more accurately. For each parsed sub-SQL k_y^i, the corresponding question is generated by:

$$(k_x^t, P^{>t}) = \text{PARSER}(C_i, S, X_e, P^{<t}, k_y^t, k_c^t), \tag{2}$$

where $P^{<t}$ is the previously generated (k_x, k_c, k_y) for C_i up to step t.

Tuple-Semantic Tracking In the inner loop of the nested CoT process, we use a tuple-semantic tracking method to generate correct knowledge units. For the molecular database in Figure 1, each tuple of `MOLECULE` represents information about a molecule, while after `ATOM INNER JOIN MOLECULE` operation, each tuple represents detailed information about an atom along with its associated molecule information. This shift in tuple semantics directly affects the meaning of aggregation operations: `COUNT(ATOM.id)` counts atoms and is equivalent to `COUNT(*)`, while `COUNT(DISTINCT ATOM.molecule_id)` counts molecules.

In ORANGE, this tuple-semantic tracking approach monitors how each SQL operation shifts tuple semantics. For each k_y^t, the PARSER infers the semantics of tuple step by step:

$$(k_c^t, k_x^t, P^{>t}) = \text{PARSER}(C_i, S, X_e, P^{<t}, k_y^t), \tag{3}$$

where k_c^t is the CoT reasoning content for tracking tuple semantics, and k_x^t is the question generated from k_c^t. This incremental inference process improves translation accuracy by ensuring consistency with the expected tuple semantics.

For example, the k_c^t of the first SQL in Figure 1:

```
This query counts the number of sodium atoms that are part of non-
carcinogenic molecules.
The INNER JOIN connects the atom and molecule tables based on their
shared molecule_id, ensuring that only sodium atoms from non-carcinogenic
 molecules are included in the count.The WHERE clause filters for both
sodium atoms and non-carcinogenic labels, and the COUNT function
aggregates these results into a single value, reflecting the total number
 of sodium atoms in non-carcinogenic molecules.
```

Knowledge De-duplication We merge all parsed knowledge units from $\mathcal{C}'$ into a unified set $\mathcal{K}_0$. Because $\mathcal{K}_0$ may contain duplicates or non-informative entries, we perform a de-duplication step. For each knowledge unit k_y^i in $\mathcal{K}_0$, we check whether its execution result is identical to that of any earlier unit $k_y^{<i}$ or is `NULL`. If so, we remove k_y^i from $\mathcal{K}_0$. The final de-duplicated set of knowledge units is $\mathcal{K}$.

2.3   Knowledge Validation

In the forward Text-to-SQL translation process, LLMs might produce incorrect SQL due to misunderstandings of S. We adopt a backward SQL-to-Text approach (Section 2.2) to annotate and interpret SQL semantics. However, the complexity of database semantics can still lead to incorrect knowledge units. While [12] uses human annotation to avoid such errors, this approach adds cost. We propose a probability-based filter to improve the reliability of generated knowledge units.

Probability-based Filter We use a probability-based filter to improve the quality of knowledge units. For each $k^t \in \mathcal{K}$ in $\mathcal{M}$, the probability is calculated as:
$$\begin{aligned} p(k^t \mid S, X) &= p(k^t \mid C_i, S, X) \cdot p(C_i \mid S, X) \\ &= p(k^t \mid C_i, S, X_e) \cdot p(C_i \mid S, X), \end{aligned} \tag{4}$$

where we assume each knowledge unit k^t comes from a unique SQL C_i, and each question X has unique evidence X_e.

We approximate by ignoring $p(k^t \mid C_i, S, X_e)$ because the nested CoT process reduces the chance of generating duplicate knowledge units, making $p(k^t \mid C_i, S, X_e)$ effectively constant. We then compute $p(C_i \mid S, X)$ from the probability of generating the same execution result, rather than matching the output token sequence, to avoid sparsity in exact SQL matching.

We remove knowledge units whose probability is below a threshold τ_0:

$$p(C_i \mid S, X) < \tau_0. \tag{5}$$

This step prevents low-quality knowledge units from entering $\mathcal{M}$.

2.4   Knowledge-Enhanced Translation

In the third step, we generate SQL based on $\mathcal{M}$. This process can be viewed as an in-domain ICL Text-to-SQL translation because the demonstrations come from the same database. Using domain-specific demonstrations boosts performance, and schema information is provided only once since all examples reference the same database. For the final SQL generation, we apply multi-path generation with a self-consistency strategy to ensure both robustness and accuracy.

In-domain Demonstration Selection In this step, we aim to identify the relevant knowledge units from the memory to serve as demonstrations. To ensure the retrieval of database-specific knowledge that may not be effectively captured by structural or syntactic similarity measures, we focus exclusively on semantic similarity. Specifically, we select the demonstrations exhibiting the highest semantic alignment and compute the similarity between the question X and a knowledge unit k^i as:

$$\mathrm{sim}(X, k^i) = \cos\left(\mathrm{EMB}(X), \mathrm{EMB}(k_x^i)\right), \tag{6}$$

where EMB converts sentences into vector representations. We implement EMB with Sentence-BERT and use FAISS to enable fast demonstration selection. A knowledge unit example is shown as:

```
Question: How many sodium atoms are found in non-carcinogenic molecules?
SQL: SELECT COUNT(T1.atom_id) FROM atom AS T1 INNER JOIN molecule AS T2
    ON T1.molecule_id = T2.molecule_id WHERE T1.element = 'na' AND T2.
    label = '-';
Exec_result: [[17]]
```

Exec_result is the SQL execution result. For large results, we retain only the top 3 tuples in the knowledge unit.

Text-to-SQL Generation In the final Text-to-SQL step, we do not include the entire schema in the prompt. This schema linking strategy reduces the prompt length and inference cost for the CODER while focusing on relevant schema items eases the inference process. For each schema item $s_i \in \mathcal{S}$, we include s_i only if:

$$s_i \in \bigcup_{k \in \mathcal{D}} \{\mathrm{ES}(k_y)\}, \tag{7}$$

where ES extracts all schema items used in all of the input SQL.

Although ORANGE may rely on selected demonstrations $\mathcal{D}$ for schema linking and SQL translation, it performs well due to the coverage and reliability of $\mathcal{M}$. The knowledge parsing process includes all system logs, which contain candidate SQL for the current question, ensuring sufficient coverage. The VALIDATOR maintains the trustworthiness of $\mathcal{M}$. We formulate the prompt according to the template in Figure 2 and use multi-path generation with a self-consistency mechanism to select the most reliable SQL output.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate ORANGE on three datasets: BIRD [19], SPIDER [38], and SCIENCE [41].

BIRD includes 12,751 Text-to-SQL pairs from 95 large-scale databases across 37 professional fields, addressing noisy database values and leverages external knowledge for SQL generation.

SPIDER contains 10,181 natural language questions and 5,693 unique SQL from 206 databases spanning 138 domains for evaluation.

SCIENCE comprises three real-world scientific databases. Domain experts created 100/99/100 high-quality question-SQL pairs for each database.

Evaluation. We use execution accuracy (EX) as the primary evaluation metric. EX evaluates the accuracy of the SQL output by comparing the results of the predicted query with the gold query when executed on specific databases.

Baselines. We select several advanced prompting-based methods and compare ORANGE with these baseline models, including MAC-SQL [32], DIN-SQL [26], DAIL-SQL [13], PURPLE [28], CHESS [30], E-SQL [4], and RSL-SQL [5]. Details of these balines are shown in Appendix B.

Implementation Details. Considering the trade-off between model performance and cost efficiency, we implement ORANGE and the baseline methods using GPT-4o-mini. The translation history utilized by ORANGE is generated by CHESS based on GPT-4o-mini. To further demonstrate the scalability of ORANGE for the base LLM, we conduct experiments with different foundation models, including Qwen2.5 Coder (Qwen2.5 Coder-14B/32B-Instruct), Qwen3 Coder (Qwen3-Coder-30B-A3B-Instruct) and the non-thinking mode of DeepSeek-V3 (DeepSeek-V3.2-Exp). We set the probability-based filter threshold to $\tau = 0.3$ and use 30 demonstrations during the ICL SQL generation process.

3.2 Main Results

We evaluated all methods using GPT-4o-mini, and results are shown in Table 1. ORANGE achieves the best performance with an EX score of 65.12%, outperforming the strongest baseline by 3.13%. Notably, on the challenging Text-to-SQL tasks, ORANGE surpasses the best baseline by 6.24%, demonstrating its effectiveness in handling complex queries. The improvement can be

Table 1. EX score (%) on BIRD, SPIDER and SCIENCE dev datasets.

Method	BIRD				SPIDER					SCIENCE			
	Sim.	Mod.	Chall.	Total	Easy	Med.	Hard	Ex.Hard	Total	CORDIS	ONCOMX	SDSS	Total
DIN-SQL	52.43	31.61	25.69	43.61	83.9	79.1	68.4	60.2	75.4	51.00	52.53	7.00	36.79
MAC-SQL	60.11	46.67	35.42	53.72	91.1	83.4	66.1	78.4	78.4	51.00	55.56	14.00	40.13
DAIL-SQL	54.38	33.12	29.86	45.63	86.7	80.3	66.1	50.6	74.7	52.00	50.51	10.00	37.46
E-SQL	64.43	49.89	41.67	57.89	87.1	82.5	60.3	58.4	76.0	-	-	-	-
RSI-SQL	67.38	50.95	43.89	60.20	93.5	85.0	73.6	62.7	81.5	58.00	63.64	8.00	43.14
PURPLE	62.70	48.82	38.19	56.19	**96.8**	**89.7**	**75.9**	**67.5**	**85.5**	54.00	51.52	**33.00**	46.15
CHESS$_{UT}$	69.08	52.69	45.15	61.86	91.2	83.5	69.9	55.2	78.5	**62.00**	65.66	18.00	48.89
CHESS$_V$	69.92	52.69	43.06	61.99	91.5	83.6	69.5	56.0	78.7	55.00	58.59	14.00	42.47
ORANGE	**71.24**	**57.20**	**51.39**	**65.12**	91.1	87.7	73.6	60.8	81.8	**62.00**	**70.71**	31.00	**54.52**

attributed to the domain-specific knowledge bases constructed by ORANGE. In-domain demonstrations enable more precise semantic alignment with the target database, while stored partial SQL semantics facilitate efficient completion of complex queries through knowledge reuse.

The candidate SQL generation step of ORANGE is based on CHESS, thus the comparison with CHESS highlights the advantage of ORANGE. ORANGE significantly outperforms both CHESS variants (by 3.13-3.26%), showing that the knowledge parsing and reusing strategy provides greater benefits than SQL testing or voting strategies. Compared to other methods with the same base LLM, including CoT-based (DIN-SQL), multi-step reasoning (MAC-SQL), and out-domain methods (PURPLE), ORANGE also shows superior reasoning ability. Using the in-domain ICL strategy, ORANGE outperforms PURPLE and DAIL-SQL, which rely on SQL structural and semantic similarity correspondingly, demonstrating the in-domain demonstrations supply more relevant knowledge to assist the LLM in Text-to-SQL translation.

On the relatively simple SPIDER benchmark, ORANGE substantially improves over both CHESS variants, confirming its general reliability. While ORANGE doesn't achieve the best performance, this can be attributed to the reliance on CHESS for candidate generation and is further discussed in Section 3.5.

To evaluate the performance of ORANGE on more complex domain-specific scenarios, we employ ORANGE on the SCIENCE benchmark, which features three specialized domains and demands comprehensive semantic knowledge. ORANGE outperforms CHESS$_{UT}$ by 5.63% and is the only method that exceeds an EX score of 50, highlighting its strong domain adaptability for specialized databases. As E-SQL involves numerous full data retrieval operations, which are infeasible on large-scale databases in SCIENCE, its experimental result is not reported.

3.3 Hyper-parameter Analysis

We analyze two key hyper-parameters in ORANGE: the number of in-context demonstrations (*shot num*) and the probability threshold τ for knowledge filtering in the probability-based filter. We set *shot num* = 30 and $\tau = 0.3$ by default and vary one parameter while fixing the other. Figure 3 illustrates how performance varies under different configurations.

The left plot shows the EX score trend under different *shot num* values. The EX score of ORANGE initially increases as *shot num* grows but eventually de-
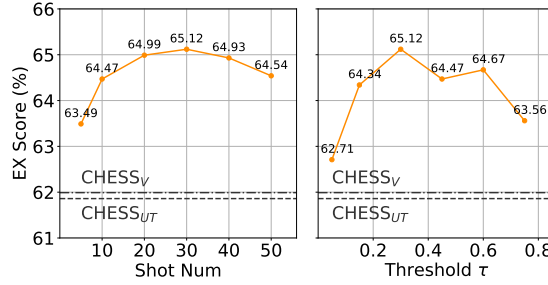
Fig. 3. EX score (%) on Bird dev under various hyper-parameters of ORANGE.

creases. This pattern reflects a trade-off: too few shots may omit demonstrations with useful knowledge, while too many introduce irrelevant examples, adding noise and leading to a performance drop.

Similarly, the right plot indicates that the EX score first rises with stricter filtering (higher τ) but eventually decreases. A low threshold retains incorrect knowledge units, whereas an overly high one discards semantically correct units, both degrading translation quality.

3.4   Ablation Study

To evaluate the contributions of different modules in ORANGE, we conduct the ablation study.

– *History* removes knowledge from other historical translation tasks in the same database and only use candidates for the current task to generate knowledge units. The performance degradation observed in the absence of knowledge from other tasks highlights the self-evolution capability of ORANGE.

Table 2. Ablation study of ORANGE.

Strategy	EX Score (%)
ORANGE	65.12
- History	63.62 (-1.50)
- Validator	62.71 (-2.41)
- Ranking	62.32 (-2.80)
- ALL	61.99 (-3.13)
- Schema Linking	64.47 (-0.65)

– *Validator* removes the validator but retains all knowledge units. The notable performance decrease demonstrates the importance of knowledge unit validation, which enables ORANGE to exclude incorrect knowledge units.

– *Ranking* replaces the ranking-based demonstration selection with random sampling, resulting in a 2.8% performance loss, indicating that selecting relevant knowledge units as demonstrations is crucial for enhancing in-domain translation accuracy of ORANGE.

– *ALL* removes the entire ORANGE framework and uses only the voting results (equal to CHESS$_V$). This ablation shows a performance drop of 3.13%, emphasizing the comprehensive impact of the ORANGE strategy.

– *Schema Linking* causes a minor performance decline. Nonetheless, the schema linking strategy is still an appropriate design, as it improves performance while reducing the cost for LLMs.

Table 3. EX score (%) of various methods with different prior SQL generator on BIRD, SPIDER and SCIENCE dev datasets.

Method	BIRD		SPIDER		SCIENCE	
	EX	Diff.	EX	Diff.	EX	Diff.
PURPLE	56.19	(-8.09)	85.5	(-0.3)	46.15	(-7.03)
PURPLE+ORANGE	64.28		85.8		53.18	
CHESS$_{UT}$	61.86	(-3.26)	78.5	(-3.3)	48.49	(-6.03)
CHESS$_V$	61.99	(-3.13)	78.7	(-3.1)	42.47	(-12.05)
CHESS+ORANGE	65.12		81.8		54.52	

3.5   Dependency on Prior Generation Process

As discussed in Section 3.2, while ORANGE achieves strong performance on complex, domain-specific benchmarks such as BIRD and SCIENCE, the results on the SPIDER dataset are constrained. This limitation is primarily caused by the reliance on the quality of candidate SQL generation process. Due to the suboptimal performance of CHESS on the SPIDER dataset, the performance of ORANGE is accordingly constrained.

To further evaluate the effectiveness of ORANGE and assess its robustness across different prior generation models, we conduct an additional experiment using PURPLE as the candidate SQL generator. As shown in Table 3, when integrated with PURPLE, ORANGE achieves a new state-of-the-art on the SPIDER dataset, improving from 85.5% to 85.8%. More impressively, on the BIRD and SCIENCE datasets, the PURPLE+ORANGE combination leads to even more substantial performance gains, demonstrating the robustness of ORANGE under different prior generation conditions.

3.6   Self-Evolutionary Performance

To explore the long-term learning capacity and model scalability of ORANGE, we examine its performance under various historical knowledge settings. We simulate three deployment scenarios: (1) Self-Only Context, (2) Accumulated History, and (3) All History and report the number of available knowledge units under each scenario.

Self-Only Context acts as a cold-start situation without prior knowledge, corresponding to the baseline performance of ORANGE without knowledge accumulation. Only SQL candidates from the current translation task are available for ICL demonstration selection.

Accumulated History simulates real-world scenarios, where knowledge is incrementally obtained from sequentially processed tasks. Each translation can only utilize the knowledge derived from the completed tasks.

All History is the default setting of ORANGE, using the full translation history of the target database to provide comprehensive knowledge coverage, which showcases the performance of ORANGE with ample historical data.

As shown in Table 4, ORANGE demonstrates clear evolutionary improvement as historical knowledge accumulates. *All History* achieves the best performance, and even *Self-Only Context* outperforms other baseline methods. This

Table 4. EX score (%) on BIRD dev and knowledge units statistics of ORANGE under different historical knowledge settings.

History	EX score (%) on BIRD dev				KU Count		
	Simple	Moderate	Challenging	Total	Average	Min	Max
Self-Only Context	70.49	54.41	49.31	63.62	3.01	0	11
Accumulated History	70.49	56.77	50.69	64.47	228.8	1	617
All History	71.24	57.20	51.39	65.12	459.3	222	617

Table 5. EX score(%) on BIRD dev with different base models of ORANGE.

Base Model	Simple	Moderate	Challenging	Total
GPT-4o-mini	71.24	57.20	51.39	65.12
Qwen2.5-Coder-14B-Instruct	74.16	56.90	54.48	67.08
Qwen2.5-Coder-32B-Instruct	73.19	60.99	52.41	67.54
Qwen3-Coder-30B-A3B-Instruct	74.38	60.78	55.86	68.51
DeepSeek-V3.2-Exp	**76.11**	**61.21**	**62.07**	**70.27**

progressive enhancement highlights the the long-term adaptability and self-evolution capability of ORANGE through continuous knowledge integration, offering a scalable advantage in real-world applications.

3.7 Scalability with Different Foundation Models

To assess the architectural independence and scalability of our framework, we conducted experiments with different foundation models.

As shown in Table 5, ORANGE demonstrates consistent performance improvements across foundation models. While GPT-4o-mini registers at 65.12% on the BIRD dataset, Qwen family models reach up to 68.51%, with DeepSeek-V3 further advancing to 70.27%. The most notable gain occurs in challenging tasks, where DeepSeek-V3 shows a 10.68% improvement over GPT-4o-mini, confirming its effectiveness in employing advanced models for complex reasoning.

4 Related Works

Enhancing the reasoning process of LLMs is crucial for generating accurate SQL, especially for complex queries in the Text-to-SQL translation tasks. [27, 22, 29] This is often achieved by decomposing complex questions into simpler, intermediate steps, as exemplified by Chain-of-Thought (CoT) prompting [35, 18] and its variants like Least-to-Most Prompting [43]. To enhance robustness, Self-consistency explores multiple reasoning paths and selects the most frequent answer through majority voting [34]. A related stream focuses on iterative refinement, where the model improves its output through self-correction. This can be guided by model-generated critique (Self-improvement) [31, 39], execution feedback from the database (Self-debugging) [2, 30], or other generated auxiliary information [42, 36]. While these strategies improve the SQL translation performance, they can not acquire and accumulate past translation experiences and fail to equip the model with domain-specific insights.

Recognizing the limitations of relying solely on intrinsic reasoning, several researches highlight the utilization of in-domain knowledge. SQL-aligned demonstrations [14, 24, 23, 28] semantically similar examples [1] are incorporated in In-context prompting. Some methods explicitly provide domain-specific instructions or demonstrations [11, 16, 21, 3]. To overcome the limited number of demonstrations, other approaches leverage larger-scale knowledge sources, such as generating synthetic domain-specific question-SQL pairs [7] or utilizing historical query logs [25, 10]. However, In-context demonstrations are constrained by the finite context windows of LLMs and struggle with knowledge scalability. Synthetic data generation inevitably fails to fully capture the real intent of user queries, and the generation process itself often introduces additional noise. Meanwhile, log-based approaches typically require extensive manual annotation or human interaction, thereby necessitating a more scalable, reliable, and automated mechanism for in-domain knowledge integration.

5   Conclusion

We propose ORANGE, a self-evolutionary Text-to-SQL method that enhances complex reasoning by parsing and validating in-domain knowledge from translation history. Through continual memory updates after each translation, ORANGE expands its knowledge of database-specific semantics, progressively improving translation accuracy without human intervention. This approach offers a practical and scalable solution for deploying Text-to-SQL in real-world scenarios.

Several future directions are also envisioned. Since ORANGE is compatible with most Text-to-SQL methods, researchers can experiment with alternative SQL generation techniques during the cold-start phase. While our nested CoT strategy demonstrates notable accuracy improvement, it remains adaptable to various reasoning paradigms. Further advancements in LLM-based reasoning could refine the quality of knowledge generation and filtering, leading to more robust Text-to-SQL translation.

A   Case Study

We compare different models on the BIRD dataset and analyze their outputs.

Figure 4 shows SQL queries generated by CHESS and ORANGE, along with their corresponding NL questions, for three databases: CALIFORNIA_SCHOOLS, FINANCIAL, and TOXICOLOGY. In these examples, ORANGE comprehends natural language semantics more accurately and generates more precise SQL.

In the CALIFORNIA_SCHOOLS database, CHESS fails to capture a critical semantic detail: the term *direct* indicates that the funding type of schools should be *Directly funded*. Consequently, when selecting schools from FRPM, the query should include FRPM.`Charter Funding Type`='Directly funded', instead of using FRPM.`Charter School (Y/N)`=1 alone.

CHESS aligns questions with schema descriptions based solely on restrained individual evidence. As illustrated in the case of FINANCIAL, due to the absence

Fig. 4. Case Study.

of explicit mention of `A2`, CHESS incorrectly selects `A3` (represents regions) for *district*. In contrast, ORANGE can leverage its domain-specific knowledge and identify that `A2` corresponds to districts and `A3` to regions, thus correctly using `A2` in the generated SQL.

Operations such as `JOIN` and `GROUP BY` can alter tuple semantics during aggregation. For instance, `ATOM INNER JOIN MOLECULE` transforms each tuple to represent an atom associated with a specific molecule, rather than the atom alone. When counting atoms, ORANGE recognizes this semantic shift and uses `molecule_id`, while CHESS incorrectly uses `atom_id`.

B Baseline Details

MAC-SQL [32] proposes a multi-step reasoning approach to address complex questions, which are decomposed into smaller, more manageable sub-questions and subsequently solved by different agents. To further enhance performance, the refiner agent employs external tools for SQL execution and iteratively refines faulty SQL queries according to the feedback.

DIN-SQL [26] incorporates task classification and problem decomposition to handle the complex SQL generation task. It categorizes input questions into three distinct types based on the presence of sub-queries and multi-table `JOIN` operations. For each category, tailored prompts are employed to reduce mismatch issues during SQL translation. DIN-SQL uses a standard in-context learning (ICL) framework without relying on explicit similarity-based retrieval metrics.

DAIL-SQL [13] focuses on the example selection and organization in few-shot prompting strategies. It employs DAIL Selection, a retrieval method that extracts demonstrations based on semantic similarity, considering both questions and queries to better align the retrieved demonstrations with the target query.

PURPLE [28] tackles the difficulty of generating SQL queries involving complex logical operator compositions. To enhance the SQL-writing capabilities of LLMs, PURPLE masks specific values and highlights logical operations within SQL queries during demonstration selection. It adopts a retrieval strategy

grounded in SQL structural similarity, enabling the model to better generalize to intricate SQL logic patterns.

CHESS [30] adopts a pipeline that involves retrieving relevant entities and context, optimizing schema, generating SQL candidates, and ultimately selecting the final SQL from them. CHESS provides two SQL selection strategies: Unit Testing (denoted as CHESS$_{UT}$), which selects the query with the most consistent execution results, and Voting (denoted as CHESS$_V$), which involves multiple LLMs voting and ranking the candidates.

E-SQL [4] integrates schema information directly into the question representation, rather than conducting dependent schema linking. This approach is claimed to effectively narrow the gap between natural language queries and database structures.

RSL-SQL [5] seeks to balance the risks of overlooking important information in a complex schema and the inefficiencies of using a simplified schema. It generates SQL queries in two scenarios: one with the full schema and one with a simplified schema enriched by extra context and selects the final SQL from the generated candidates.

References

1. An, S., Zhou, B., Lin, Z., Fu, Q., Chen, B., Zheng, N., Chen, W., Lou, J.G.: Skill-based few-shot selection for in-context learning. arXiv preprint arXiv:2305.14210 (2023)
2. Andrew, J.J., Vincent, M., Burgun, A., Garcelon, N.: Evaluating llms for temporal entity extraction from pediatric clinical text in rare diseases context. In: Proceedings of the First Workshop on Patient-Oriented Language Processing (CL4Health)@ LREC-COLING 2024. pp. 145–152 (2024)
3. Arora, A., Bhaisaheb, S., Nigam, H., Patwardhan, M., Vig, L., Shroff, G.: Adapt and decompose: Efficient generalization of text-to-sql via domain adapted least-to-most prompting. arXiv preprint arXiv:2308.02582 (2023)
4. Caferoğlu, H.A., Ulusoy, Ö.: E-sql: Direct schema linking via question enrichment in text-to-sql. arXiv preprint arXiv:2409.16751 (2024)
5. Cao, Z., Zheng, Y., Fan, Z., Zhang, X., Chen, W., Bai, X.: Rsl-sql: Robust schema linking in text-to-sql generation. arXiv preprint arXiv:2411.00073 (2024)
6. Cen, J., Liu, J., Li, Z., Wang, J.: Sqlfixagent: Towards semantic-accurate SQL generation via multi-agent collaboration. CoRR **abs/2406.13408** (2024)
7. Chang, S., Fosler-Lussier, E.: Selective demonstrations for cross-domain text-to-sql. arXiv preprint arXiv:2310.06302 (2023)
8. Chen, K., Chen, Y., Koudas, N., Yu, X.: Reliable text-to-sql with adaptive abstention. Proceedings of the ACM on Management of Data **3**(1), 1–30 (2025)
9. Chen, X., Lin, M., Schärli, N., Zhou, D.: Teaching large language models to self-debug. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net (2024)
10. Chu, Z., Wang, Z., Qin, Q.: Leveraging prior experience: An expandable auxiliary knowledge base for text-to-sql. arXiv preprint arXiv:2411.13244 (2024)
11. Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Lin, J., Lou, D., et al.: C3: Zero-shot text-to-sql with chatgpt. arXiv preprint arXiv:2307.07306 (2023)

12. Fan, Y., He, Z., Ren, T., Guo, D., Chen, L., Zhu, R., Chen, G., Jing, Y., Zhang, K., Wang, X.S.: Gar: A generate-and-rank approach for natural language to SQL translation. In: 39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023. pp. 110–122. IEEE (2023)
13. Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., Zhou, J.: Text-to-sql empowered by large language models: A benchmark evaluation. arXiv preprint arXiv:2308.15363 (2023)
14. Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., Zhou, J.: Text-to-sql empowered by large language models: A benchmark evaluation. arXiv preprint arXiv:2308.15363 (2023)
15. Gao, Y., Liu, Y., Li, X., Shi, X., Zhu, Y., Wang, Y., Li, S., Li, W., Hong, Y., Luo, Z., et al.: Xiyan-sql: A multi-generator ensemble framework for text-to-sql. arXiv preprint arXiv:2411.08599 (2024)
16. Gu, Z., Fan, J., Tang, N., Zhang, S., Zhang, Y., Chen, Z., Cao, L., Li, G., Madden, S., Du, X.: Interleaving pre-trained language models and large language models for zero-shot nl2sql generation. arXiv preprint arXiv:2306.08891 (2023)
17. Hong, Z., Yuan, Z., Chen, H., Zhang, Q., Huang, F., Huang, X.: Knowledge-to-sql: Enhancing sql generation with data expert llm. arXiv preprint arXiv:2402.11517 (2024)
18. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in neural information processing systems **35**, 22199–22213 (2022)
19. Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Geng, R., Huo, N., Zhou, X., Ma, C., Li, G., Chang, K.C., Huang, F., Cheng, R., Li, Y.: Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023 (2023)
20. Li, Z., Wang, X., Zhao, J., Yang, S., Du, G., Hu, X., Zhang, B., Ye, Y., Li, Z., Zhao, R., et al.: Pet-sql: A prompt-enhanced two-round refinement of text-to-sql with cross-consistency. arXiv preprint arXiv:2403.09732 (2024)
21. Liu, A., Hu, X., Wen, L., Yu, P.S.: A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. arXiv preprint arXiv:2303.13547 (2023)
22. Liu, X., Tan, Z.: Divide and prompt: Chain of thought prompting for text-to-sql. arXiv preprint arXiv:2304.11556 (2023)
23. Nan, L., Zhao, Y., Zou, W., Ri, N., Tae, J., Zhang, E., Cohan, A., Radev, D.: Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. arXiv preprint arXiv:2305.12586 (2023)
24. Poesia, G., Polozov, O., Le, V., Tiwari, A., Soares, G., Meek, C., Gulwani, S.: Synchromesh: Reliable code generation from pre-trained language models. arXiv preprint arXiv:2201.11227 (2022)
25. Pourreza, M., Li, H., Sun, R., Chung, Y., Talaei, S., Kakkar, G.T., Gan, Y., Saberi, A., Ozcan, F., Arik, S.Ö.: CHASE-SQL: multi-path reasoning and preference optimized candidate selection in text-to-sql. CoRR **abs/2410.01943** (2024)
26. Pourreza, M., Rafiei, D.: DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023 (2023)

27. Pourreza, M., Rafiei, D.: Din-sql: Decomposed in-context learning of text-to-sql with self-correction. Advances in Neural Information Processing Systems **36** (2024)
28. Ren, T., Fan, Y., He, Z., Huang, R., Dai, J., Huang, C., Jing, Y., Zhang, K., Yang, Y., Wang, X.S.: Purple: Making a large language model a better sql writer. arXiv preprint arXiv:2403.20014 (2024)
29. Tai, C.Y., Chen, Z., Zhang, T., Deng, X., Sun, H.: Exploring chain-of-thought style prompting for text-to-sql. arXiv preprint arXiv:2305.14215 (2023)
30. Talaei, S., Pourreza, M., Chang, Y., Mirhoseini, A., Saberi, A.: CHESS: contextual harnessing for efficient SQL synthesis. CoRR **abs/2405.16755** (2024)
31. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
32. Wang, B., Ren, C., Yang, J., Liang, X., Bai, J., Zhang, Q., Yan, Z., Li, Z.: MAC-SQL: A multi-agent collaborative framework for text-to-sql. CoRR **abs/2312.11242** (2023)
33. Wang, D., Dou, L., Zhang, X., Zhu, Q., Che, W.: Improving demonstration diversity by human-free fusing for text-to-sql. arXiv preprint arXiv:2402.10663 (2024)
34. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
35. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)
36. Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., Choi, Y.: Generating sequences by learning to self-correct. arXiv preprint arXiv:2211.00053 (2022)
37. Xie, Y., Jin, X., Xie, T., Lin, M., Chen, L., Yu, C., Cheng, L., Zhuo, C., Hu, B., Li, Z.: Decomposition for enhancing attention: Improving llm-based text-to-sql through workflow paradigm. arXiv preprint arXiv:2402.10671 (2024)
38. Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., Radev, D.R.: Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (eds.) Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018. pp. 3911–3921. Association for Computational Linguistics (2018)
39. Zelikman, E., Wu, Y., Mu, J., Goodman, N.: STar: Bootstrapping reasoning with reasoning. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022)
40. Zhang, T., Chen, C., Liao, C., Wang, J., Zhao, X., Yu, H., Wang, J., Li, J., Shi, W.: Sqlfuse: Enhancing text-to-sql performance through comprehensive llm synergy. arXiv preprint arXiv:2407.14568 (2024)
41. Zhang, Y., Deriu, J., Katsogiannis-Meimarakis, G., Kosten, C., Koutrika, G., Stockinger, K.: Sciencebenchmark: A complex real-world benchmark for evaluating natural language to SQL systems. Proc. VLDB Endow. **17**(4), 685–698 (2023)
42. Zheng, C., Liu, Z., Xie, E., Li, Z., Li, Y.: Progressive-hint prompting improves reasoning in large language models. arXiv preprint arXiv:2304.09797 (2023)
43. Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al.: Least-to-most prompting enables complex reasoning in large language models. arXiv preprint arXiv:2205.10625 (2022)