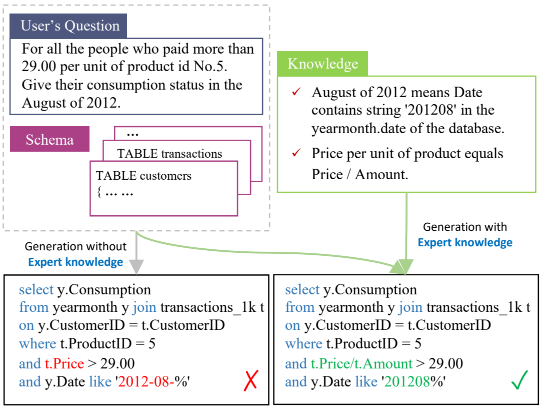
Knowledge-to-SQL: Enhancing SQL Generation with Data Expert LLM

Zijin Hong[1], Zheng Yuan[2], Hao Chen[2], Qinggang Zhang[2]
Feiran Huang[1†], Xiao Huang[2]
[1]Jinan University
[2]The Hong Kong Polytechnic University
hongzijin@stu2020.jnu.edu.cn
{yzheng.yuan,qinggangg.zhang}@connect.polyu.hk
sundaychenhao@gmail.com; huangfr@jnu.edu.cn; xiaohuang@comp.polyu.edu.hk

Abstract

Generating accurate SQL queries for user questions (text-to-SQL) has been a long-standing challenge since it requires a deep understanding of both the user's question and the corresponding database schema in order to retrieve the desired content accurately. Existing methods rely on the comprehensive capability of large language models (LLMs) to generate the SQL. However, some necessary knowledge is not explicitly included in the database schema and user question or has been learned by LLMs. Thus, the generated SQL of the knowledge-insufficient questions may be inaccurate, negatively influencing the text-to-SQL models' performance and robustness. To address this challenge, we propose the **Knowledge-to-SQL** framework, which employs tailored **Data Expert LLM (DELLM)** to provide helpful knowledge for all text-to-SQL models. Specifically, we introduce the detailed implementation of DELLM regarding table reading and the basic fine-tuning process. We further propose a *Preference Learning via Database Feedback (PLDBF)* strategy, refining the DELLM to generate more helpful knowledge for LLMs. Extensive experiments verify that DELLM can enhance the state-of-the-art approaches for text-to-SQL tasks. The corresponding code of DELLM is released for further research[1].



For all the people who paid more than 29.00 per unit of product id No.5. Give their consumption status in the August of 2012.

✓ August of 2012 means Date contains string '201208' in the yearmonth.date of the database.

✓ Price per unit of product equals Price / Amount.

```
select y.Consumption
from yearmonth y join transactions_1k t
on y.CustomerID = t.CustomerID
where t.ProductID = 5
and t.Price > 29.00
and y.Date like '2012-08-%'     ✗
```

```
select y.Consumption
from yearmonth y join transactions_1k t
on y.CustomerID = t.CustomerID
where t.ProductID = 5
and t.Price/t.Amount > 29.00
and y.Date like '201208%'     ✓
```

Figure 1: A sketch map illustrating the significance of incorporating expert knowledge in the text-to-SQL implementation. In the given example, the generation without expert knowledge makes mistakes in arithmetic reasoning and data conditions. Expert knowledge bridges the knowledge gap between the LLMs and the database, which assists the LLMs in generating accurate SQL.

1 Introduction

Generating SQL based on user questions (text-to-SQL) is currently one of the leading applications for large language models (LLMs). The most straightforward approach is to input the user questions and database schema into the LLMs and rely on their capability of natural language understanding to generate the SQL (Dou et al., 2022; Liu et al., 2023a; Pourreza and Rafiei, 2023). However, in real-world applications, user queries and database structure may contain customized or specialized knowledge (Yu et al., 2018; Li et al., 2023b), including arithmetic reasoning, domain knowledge, synonym explanation, etc. Since the knowledge is not explicit in either the user question or the database schema (Dou et al., 2022; Gan et al., 2021). In such cases, the intuitive approach may result in inaccurate or un-executable SQL unless a human expert provides the necessary helpful knowledge (so-called **"expert knowledge"**) to the LLMs to bridge the knowledge gap with database content (Li et al., 2023b; Gan et al., 2021). Given this challenge, it is valuable to develop a non-human data expert system that can automatically generate the required expert knowledge to assist SQL generation. This would significantly enhance the performance and robustness of text-to-SQL implementations.

Existing methods primarily focus on fully exploring the comprehensibility of the pre-trained language models (PLMs). As the forerunner, T5-based methods (Scholak et al., 2021; Li et al., 2023a; Rai

[†]Corresponding author

[1]https://github.com/Rcrossmeister/Knowledge-to-SQL

Findings of the Association for Computational Linguistics: ACL 2024, pages 10997–11008
August 11-16, 2024 ©2024 Association for Computational Linguistics

et al., 2023) initially attempted to train PLMs from scratch to generate SQL based on user questions and database schema. Following the popularity of proprietary LLMs, particularly ChatGPT and GPT-4 (OpenAI, 2023), DIN-SQL (Pourreza and Rafiei, 2023) utilizes LLMs to decompose the process of SQL generation into several sub-tasks. It uses LLMs to first accomplish these sub-tasks and then aggregates the results to generate the final SQL. Similarly, DAIL-SQL (Gao et al., 2024) designed a strategy for sampling few-shot instances, then using few-shot prompting for proprietary LLMs to accomplish the generation of the required SQL. Most recently, MAC-SQL (Wang et al., 2023) assigns different roles to three LLM agents, specifically selector, decomposer, and refiner, then uses these agents to make different contributions to the text-to-SQL process.

However, recent methods mainly focus on designing more sophisticated structures in order to improve text-to-SQL performance. Although achieving promising performance, there is a lack of emphasis on the necessity of expert knowledge (Li et al., 2023b). As depicted on the left of Figure 1, relying solely on the user's question and the database schema as input without expert knowledge may cause inaccurate conditions in the generated SQL. For example, the incorrect condition "y.Date like "2012-08-%"" could potentially result in the SQL returning empty data. In contrast, on the right of the figure, when provided with accurate expert knowledge for assistance, the LLMs are able to rectify the condition and generate valid SQL.

Nevertheless, generating expert knowledge from the question and schema faces the following challenges: **1. Question & Database Specialization**: The generated expert knowledge should be specialized for the given question and database. It is challenging for the data expert to understand the question and database and provide helpful knowledge. **2. Table Content Awareness** (Cheng et al., 2023): The data expert should be able to read the content of the table in order to determine whether it is necessary to provide detailed content examples in the knowledge. **3. Performance Enhancement** (Li et al., 2023b): The generated knowledge should be helpful for the text-to-SQL models. It is challenging to ensure that expert knowledge can contribute to more accurate SQL generation.

By addressing the aforementioned challenges, we present a detailed design of **Knowledge-to-SQL** framework for enhancing the SQL generation of LLMs. Specifically, we propose a well-designed **Data Expert Large Language Model (DELLM)**, which comprises a table reading module and a knowledge-oriented supervised fine-tuning process. Furthermore, we introduce *Preference Learning via Database Feedback (PLDBF)* to further refine the helpfulness of the generated expert knowledge for LLM-based text-to-SQL (Christiano et al., 2017; Rafailov et al., 2023; Hong and Liu, 2024). Specifically, PLDBF provides preferences to DELLM based on two tailored criteria: 1) the extent to which the generated knowledge aids in retrieving more accurate content, and 2) the degree to which the generated knowledge assists LLM in producing more precise SQL queries. In summary, our contributions can be listed as follows:

- We highlight the significance of expert knowledge and present the knowledge-to-SQL framework for improving SQL generation.
- We introduce a well-designed Data Expert Large Language Model (DELLM), along with customized structure, fine-tuning technique, and preference-tuning training strategies.
- We release the training and evaluation code of DELLM as open source for future research.
- We validate the effectiveness of our approach on the BIRD and Spider datasets, demonstrating that DELLM can generally enhance the performance of common LLM-based text-to-SQL implementations.

2 Related Work

2.1 LLMs for Text-to-SQL

The text-to-SQL task focuses on translating natural language questions into SQL queries. Recent advances in this field have shown a growing interest in using large language models (LLMs) paired with prompt engineering. Chain of Thought (CoT) prompting (Wei et al., 2022b), an effective prompt engineering technique, has found considerable utility in the text-to-SQL domain. Numerous researchers conducted empirical studies on prompt organization for text-to-SQL using CoT (Rajkumar et al., 2022; Gao et al., 2024; Chang and Fosler-Lussier, 2023; Li et al., 2023b; Zhang et al., 2023a). In DAIL-SQL (Gao et al., 2024), the authors introduced a novel approach for selecting pertinent few-shot examples, considering both similarities in the user's question and SQL query. Moreover, ACT-SQL (Zhang et al., 2023a) employed a cost-efficient method to automatically generate CoT instances, alleviating the need for manual prompt creation. For enhancing the reasoning capabilities of LLMs, works like C3 (Dong et al., 2023b) and DIN-SQL (Pourreza and Rafiei, 2023) have proposed a paradigm for decomposing main tasks into multiple sub-tasks, which have been specifically designed for zero-shot and few-shot text-to-SQL tasks respectively. Most recently, MAC-SQL (Wang et al., 2023) assigns multiple LLM agents with different roles for the text-to-SQL process. Additionally, to assist text-to-SQL with expert knowledge, authors in (Li et al., 2023b) engaged human experts to annotate helpful knowledge for each text-to-SQL instance. However, this mode of human-led annotation proves to be highly labor-consuming for large-scale tasks. In this paper, we focus on exploring automated expert knowledge generation for assisting text-to-SQL using LLMs.

2.2 Prompt Engineering in Text-to-SQL

Previous studies have highlighted the importance of prompt engineering in optimizing the performance of LLMs (Radford et al., 2019; Liu et al., 2023b). The effective prompt design significantly improves the efficiency and quality of LLM outputs (Wei et al., 2022a). Research initiatives like CoT (Wei et al., 2022b) and retrieval augmented generation (RAG) (Lewis et al., 2020) integrate contextual information to enhance LLMs' understanding of natural language reasoning. In the text-to-SQL task, recent investigations (Rajkumar et al., 2022) have refined the prompts for LLMs, resulting in high uniformity input formats. These prompts typically include user questions, database schema, and task instructions. However, comprehending complex input schema and correlating them with user questions poses a significant challenge for LLMs, stemming from diverse aspects of prompt design. Leading approaches (Rajkumar et al., 2022; Pourreza and Rafiei, 2023) have incorporated techniques such as few-shot sampling (Gao et al., 2024) and CoT (Li et al., 2023b; Wang et al., 2023) guidelines to address this challenge. Our proposed framework can also be viewed as a particular prompt engineering, with the purpose of generating expert knowledge as input prompts to assist LLMs' understanding of text-to-SQL.

3 Proposed Method

We adhere to the two criteria mentioned to guide knowledge generation: 1) Enhancing accurate database execution. 2) Improving precise SQL generation. Our proposed method aims to develop a system called DELLM for expert knowledge generation that satisfies the above criteria. As depicted in Figure 2, our framework consists of the following three primary modules:

- The supervised fine-tuned (SFT) model: It takes the user question, database schema, and relevant tables from the database as input and converts them into generated knowledge.
- The preference learning (PL) framework: It refines the model by aligning the feedback from SQL query executions on the database with the contributions from ground-truth SQL.
- An off-the-shelf text-to-SQL model: It predicts the SQL by inputting the user question, database schema, and generated knowledge.

3.1 Supervised Fine-Tuning of DELLM

This module aims to generate expert knowledge based on the user question and the database schema. Assuming the user question and the corresponding database schema are Q and $\mathcal{S}$ respectively; the goal is to match the relevant sub-tables $\mathcal{T}_\alpha$ and generate knowledge K^{gen} with the above inputs.

Table Reading. In our study, we incorporate the task of table reading to generate expert knowledge. When dealing with large databases, inputting the complete database tables poses a challenge regarding input length limitation and redundancy. To address this, we utilize semantic techniques to match the relevant table. This allows us to extract the most pertinent table for the given question as the input prompt for the subsequent SFT model. Let $\mathcal{S}$ represent the database schema, defined as:

$$\mathcal{S} = \{\mathcal{T}_1(c_{11}, ..., c_{1m_1}), ..., \mathcal{T}_i(c_{i1}, ..., c_{im_i}), ...\}, \tag{1}$$

where $\mathcal{T}_i$ denotes the i^{th} table in the database, and c_{ij} represents the j^{th} column in table $\mathcal{T}_i$, m_i denotes the number of column included in $\mathcal{T}_i$. Given a schema with n tables, we can denote the collection of all table's columns as $\mathcal{C} = \{\{c_{11}, ..., c_{1m_1}\}, ..., \{c_{n1}, ..., c_{nm_n}\}\}$. For each column $c_{ij} \in \mathcal{C}$, we can obtain a sub-collection of relevant columns denoted as $\mathcal{C}_\alpha$:

$$\mathcal{C}_\alpha = \{c_{ij} \mid \text{sim}(Q, c_{ij}) > \alpha\}, \tag{2}$$

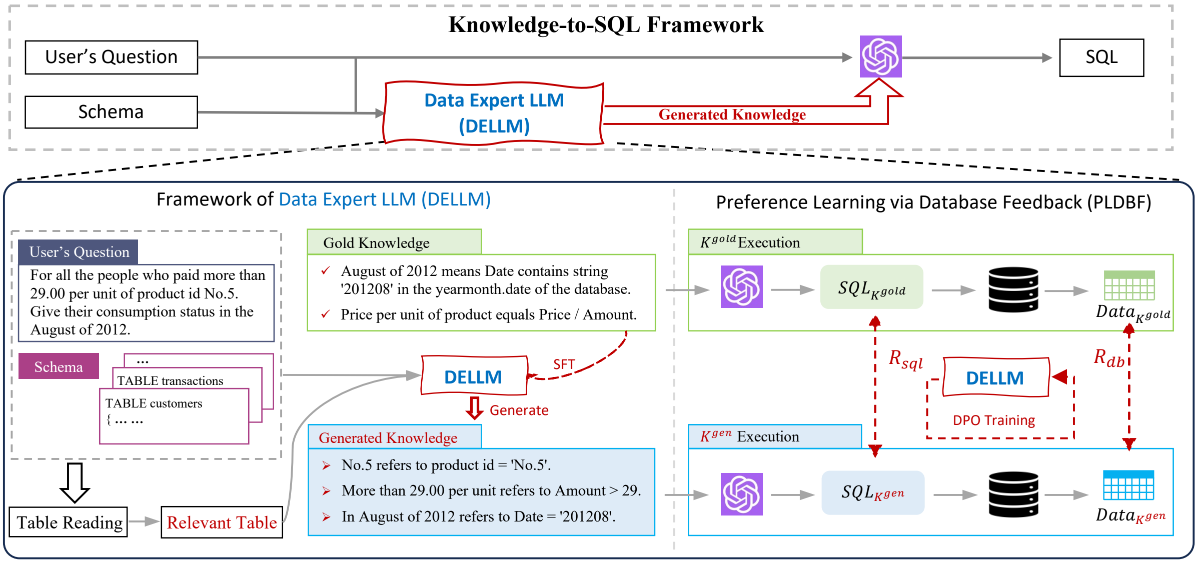

Knowledge-to-SQL Framework
User's Question
Schema
Data Expert LLM (DELLM)
Generated Knowledge
SQL
Framework of Data Expert LLM (DELLM)
User's Question
For all the people who paid more than 29.00 per unit of product id No.5. Give their consumption status in the August of 2012.
Schema
...
TABLE transactions
TABLE customers
{ ... ...
Table Reading
Relevant Table
Gold Knowledge
✓ August of 2012 means Date contains string '201208' in the yearmonth.date of the database.
✓ Price per unit of product equals Price / Amount.
DELLM
SFT
Generate
Generated Knowledge
➢ No.5 refers to product id = 'No.5'.
➢ More than 29.00 per unit refers to Amount > 29.
➢ In August of 2012 refers to Date = '201208'.
Preference Learning via Database Feedback (PLDBF)
K^{gold} Execution
$SQL_{K^{gold}}$
$Data_{K^{gold}}$
R_{sql}
DELLM
DPO Training
R_{db}
K^{gen} Execution
$SQL_{K^{gen}}$
$Data_{K^{gen}}$

Figure 2: The overview of our approach. The upper is the overall knowledge-to-SQL framework. The details of DELLM are presented at the bottom. On the left side, we have the framework of DELLM, including supervised fine-tuning (SFT) and table reading. On the right side, we introduce preference learning via database feedback (PLDBF), which is employed to further refine the performance of DELLM.

where $\text{sim}(\cdot)$ denotes the semantic similarity calculator, and $\text{sim}(Q, c_{ij}) > \alpha$ represents the similarity between column c_{ij} and query Q exceeding the threshold α. The relevant sub-tables can be obtained by:

$$\mathcal{T}_\alpha = \{\mathcal{T}_i(c_{ij}) \mid c_{ij} \in \mathcal{C}_\alpha\}. \tag{3}$$

Supervised Fine-tuning. The knowledge generation process can be represented as follows:

$$K^{gen} = \pi(Q, \mathcal{S}, \mathcal{T}_\alpha), \tag{4}$$

$\pi(\cdot)$ represents a text generation model we utilize as a backbone model for the SFT process. As defined earlier, the model is required to generate the knowledge K^{gen}. The objective function of SFT can be defined as:

$$\begin{aligned}\mathcal{L}_{SFT} &= -\log \Pr(K^{gold} \mid Q, \mathcal{S}, \mathcal{T}_\alpha) \\ &= -\sum K^{gold} \log(K^{gen}),\end{aligned} \tag{5}$$

where K^{gold} denotes the gold (ground-truth) knowledge annotated by human experts. The SFT process aims to minimize the cross-entropy loss in Eq. 5 defined above. Consequently, we can obtain a fine-tuned model π^{SFT}, which possesses the ability to generate knowledge in a preliminary manner.

3.2 Feedback From Database Execution and Ground-truth SQL Contribution

To enable our model to generate helpful knowledge for accurate database execution and contribute effectively to SQL queries, we employ the preference learning framework. This framework refines the model's capabilities by aligning feedback from database executions and the contributions of ground-truth SQL.

Feedback From Database Execution. In Section 3.1, we obtained a model π^{SFT} via SFT. We utilize the knowledge generated by this model to assist an off-the-shelf text-to-SQL model $g(\cdot)$ in SQL generation.

$$\hat{Y}^{gen} = g(\mathcal{S}, Q, K^{gen}), \tag{6}$$

where $\hat{Y}^{gen}$ represents the SQL query generated using the knowledge K^{gen}. For every instance in the training set, we obtain a predicted SQL; the collection of these SQL queries is denoted by $\hat{\mathcal{Y}}^{gen}$. Similarly, we obtain a predicted SQL collection $\hat{\mathcal{Y}}^{gold}$ using the manually annotated ground-truth knowledge K^{gold}. To interact with the database execution, we execute the SQL collections $\hat{\mathcal{Y}}^{gen}$ and $\hat{\mathcal{Y}}^{gold}$, respectively, to obtain the result sets $\hat{V}^{gen}$ and $\hat{V}^{gold}$. An indicator function is defined to evaluate the execution results. We have:

$$\mathbb{1}_{db}(V, V') = \begin{cases} 1, & V = V' \\ 0, & V \neq V' \end{cases}, \tag{7}$$

then, we utilize this indicator function as the condition for annotating the preference knowledge pairs

Ground-Truth SQL: **SELECT** 'Free Meal Count (Ages 5-17)' / 'Enrollment (Ages 5-17)' **FROM** from **WHERE** 'Educational Option Type' = 'Continuation School' **AND** 'Free Meal Count (Ages 5-17)' / 'Enrollment (Ages 5-17)' **IS NOT NULL ORDER BY** 'Free Meal Count (Ages 5-17)' / 'Enrollment (Ages 5-17)' **ASC LIMIT** 3
Contributing Knowledge: Eligible free rates for students aged 5-17 = 'Free Meal Count (Ages 5-17)' / 'Enrollment (Ages 5-17)'.
Non-contributing Knowledge: Continuation schools refer to EdOpsCode = 'C', lowest three eligible free rate refer to MIN('Percent (%) Eligible Free (Ages 5-17)').

Table 1: Example of ground-truth SQL contribution. We highlight the key content in SQL and the corresponding sub-knowledge included in the contributing and non-contributing knowledge.

with the feedback of database execution:

$$\mathcal{P}^{db}_{\{K_w,K_l\}} = \{K_i^{gold}, K_i^{gen} \mid \mathbb{1}_{db}(\hat{V}_i^{gold}, \hat{V}_i^{gen}) = 0\}. \tag{8}$$

This feedback requires the generated knowledge to align with database execution.

Feedback From Ground-truth SQL Contribution. Knowledge plays a decisive role in text-to-SQL that the final predicted SQL will include sub-knowledge content introduced by the given knowledge. An example is shown in Table 1. However, the incorrect sub-knowledge may lead to a misprediction in generating SQL queries. A contributing knowledge indicates that every included sub-knowledge is helpful for the predicted SQL, the corresponding content should be valid. Thus, we focus on studying the contribution of generated knowledge to the ground-truth SQL.

Noting that a generated knowledge K comprises several sub-knowledge $k_1, k_2, ...$, can be represented as $K = \{k_1, k_2, ...\}$. We introduce another indicator function:

$$\mathbb{1}_{sql}(K, Y) = \begin{cases} 1, & k \in Y, \forall k \in K \\ 0, & k \notin Y, \exists k \in K \end{cases}, \tag{9}$$

where Y is the ground-truth SQL. We check whether every sub-knowledge is contained by Y to judge whether the knowledge contributes. Then, we collect the preference knowledge pairs based on the feedback of ground-truth SQL contribution, formulated as:

$$\mathcal{P}^{sql}_{\{K_w,K_l\}} = \{K_j^{gold}, K_j^{gen} \mid \mathbb{1}_{sql}(K_j^{gold}, Y) = 1, \tag{10}$$
$$\mathbb{1}_{sql}(K_j^{gen}, Y) = 0\}.$$

This feedback necessitates the generated knowledge to become correctly SQL contributing and reduce the redundancy in the knowledge generation process.

By Eq. 8 and 10, we obtain two preference knowledge set $\mathcal{P}^{db}$ and $\mathcal{P}^{sql}$. To get the final preference learning dataset $\mathcal{D}$, we combine corresponding input of K_l according to Eq. 4, denoted by I_l and the preference pair (K_w, K_l) by:

$$\mathcal{D} = \{(I_l, K_w, K_l) \mid (K_w, K_l) \in \mathcal{P}^{db} \cup \mathcal{P}^{sql}\}. \tag{11}$$

3.3 Preference Learning with PLDBF

Typically, preference fine-tuning is employed subsequent to SFT for further refinement. In our scenario, the preference learning framework utilizes a direct preference optimization (DPO) (Rafailov et al., 2023) algorithm. For each preference pair $(I_l, K_w, K_l) \in \mathcal{D}$, the objective function of this process can be formulated as:

$$\begin{aligned} &\mathcal{L}_{\text{PL}}(\pi^{DPO}; \pi^{SFT}) \\ &\quad = -\mathbb{E}_\pi[\log \sigma(\beta R(K_w) - \beta R(K_l))], \end{aligned} \tag{12}$$

specifically, $\mathbb{E}_\pi = \mathbb{E}_{(I_l,K_w,K_l)\sim\mathcal{D}}$ and $R(K) = \log(\pi^{DPO}(K|I_l)/\pi^{SFT}(K|I_l))$. where $R(\cdot)$ is the reward implicitly defined by the target model π^{DPO} and reference model π^{SFT}. Then, we obtain the DPO refined model π^{DPO}, which can generate expert knowledge to assist in accurate database execution and contribute to precise SQL generation.

3.4 Knowledge-to-SQL with DELLM

Finally, the user question, the database schema, and the relevant table are given during the testing phase. With PL-refined DELLM π^{DPO}, we collect the generated expert knowledge and combine it with the question, schema, and task instruction to assist an off-the-shelf text-to-SQL model for SQL generation. We enumerate each question and the corresponding schema and get the predicted SQL query as the result.

4 Experiments

In this section, we will empirically evaluate our proposed Knowledge-to-SQL framework. After introducing the experimental setups, the experimental results are discussed in five parts:

- *Main Result:* The purpose of our framework is to assist a text-to-SQL model in generating accurate SQL. We compare the originally predicted SQL with the predicted SQL incorporating the generated expert knowledge; the comparative performance is evaluated on various models or methods on different benchmarks.

- *Ablation Study:* We conduct the ablation studies to verify the efficiency and robustness of our proposed framework. By predicting SQL using the knowledge generated by variations of DELLM, we discuss the influence of different modules.

- *Comprehensive Evaluation:* We first look deep into how the generated knowledge assists different types (difficulties) of the question. Then, we compare the improvement brought by the generated knowledge and the ground-truth knowledge. Then conclude the comparative results based on the two results above.

- *Performance on Partial Training Data:* We discuss the scenario with partial training data and analyze our advantage on practical scenarios with a limited budget.

- *Statistical Analysis:* To visualize the influence of DELLM from a data statistical perspective, we calculate the ratio of various influences brought by incorporating the generated knowledge and provide corresponding discussions and analyses.

4.1 Experimental Setup

Dataset. Our experiments are conducted on two widely recognized dataset BIRD (Li et al., 2023b) and Spider (Yu et al., 2018). BIRD benchmark was released most recently, which annotated high-quality text-to-SQL instances with 95 databases on a large scale. BIRD is a leading benchmark focusing on massive and real database content, first introducing knowledge reasoning between natural language questions and database content. As the evaluation metric, BIRD introduces a new metric verifying the balance of efficiency and accuracy during the execution. The knowledge introduced by BIRD is annotated by human experts who are native speakers of English with degrees above the bachelor's level and have database-related knowledge. Spider benchmark is assessed frequently to evaluate the performance of text-to-SQL across multiple databases, which include 200 distinct databases and 138 domains. Since the test set of these two datasets is not publicly available, we evaluate our method's efficacy on the accessible development (dev) set.

Evaluation Settings. To make a fair comparison, we follow the metric as previous work (Gao et al., 2024; Wang et al., 2023) for evaluation. We consider two metrics: 1) Execution Accuracy (**EX**), which is the ratio of questions that the execution results of the predicted SQL queries match exactly with those of the ground-truth SQL queries, compared to the total number of questions. 2) Valid Efficiency Score (**VES**), designed to evaluate the efficiency of SQL generated by text-to-SQL systems. The VES metric considers both the efficiency and accuracy of execution results and incorporates running time for a more comprehensive evaluation.

Implementations. We select the widely-used LLaMA-2-13b (Touvron et al., 2023) with official configuration as our backbone model for knowledge generation. The learning rate is set as 5e-05, selected from the interval [1e-05, 1e-04]. The semantic similarity calculation of the table reading process is implemented by Faiss[2] (Douze et al., 2024). The other hyper-parameters are discussed in the Appendix A. Noting that the Spider dataset does not have annotated knowledge samples, we utilize the human-annotated knowledge in BIRD as ground-truth knowledge to train the model and evaluate the main performance in both the BIRD and Spider datasets. The evaluation of the Spider dataset aims to verify our effectiveness in cross-domain databases. The ablation study and further discussions are conducted on the BIRD dataset.

Baselines. We select the comparative baselines on the BIRD dataset based on the official benchmark page[3]. Specifically, to verify the effectiveness of DELLM on the open-source model, we utilize 1)

[2] https://github.com/facebookresearch/faiss/

[3] https://bird-bench.github.io/

	Models	EX		VES	
		w/o knowledge	w/ DELLM	w/o knowledge	w/ DELLM
BIRD	T5-3B	10.37	16.68 (+6.31)	13.62	20.84 (+7.22)
	GPT-3.5-Turbo	27.64	33.31 (+5.67)	28.64	36.12 (+7.48)
	GPT-4	33.25	37.94 (+4.69)	35.92	42.15 (+6.23)
	Claude-2	30.05	35.53 (+5.48)	32.97	39.71 (+6.74)
	GPT-3.5-Turbo + CoT	27.25	32.79 (+5.54)	29.16	35.51 (+6.35)
	DAIL-SQL + GPT-4	40.89	45.81 (+4.92)	45.13	51.59 (+6.46)
	MAC-SQL + GPT-4	43.65	48.92 (+5.27)	48.07	54.78 (+6.71)
Spider	GPT-3.5-Turbo	67.89	69.60 (+1.71)	68.33	70.16 (+1.83)
	GPT-4	70.02	71.68 (+1.66)	71.03	72.82 (+1.79)

Table 2: Experimental results for text-to-SQL on different benchmarks with and without knowledge generated by our proposed DELLM. The number in the bracket denotes the improvement in execution accuracy (EX) and valid efficiency score (VES) brought by DELLM's knowledge compared to the baseline performance without knowledge.

T5-3B (Raffel et al., 2020), using the fine-tuning-based method that incorporates the knowledge as fine-tuning input. Then, for the off-the-shelf models, we utilize 2) GPT-3.5-Turbo and 3) GPT-4 (OpenAI, 2023), a widely-used powerful proprietary model with zero-shot text-to-SQL prompt for SQL generation; 4) Claude-2 (Anthropic, 2023), another well-recognized proprietary model. For the up-to-date prompt engineering techniques, we compare 5) GPT-3.5-Turbo + CoT (Li et al., 2023b), a simple CoT prompt engineering on text-to-SQL; 6) DAIL-SQL (Gao et al., 2024), encoding structure knowledge and selects few-shot instances based on similarity matching; 7) MAC-SQL (Wang et al., 2023), a novel LLM-based multi-agent collaborative framework designed for the text-to-SQL task. On the Spider dataset, as introduced above, the evaluation is conducted as a cross-domain verification. We utilize proprietary models GPT-3.5-Turbo and GPT-4 for validation.

4.2 Main Results

Table 2 gives the experimental results. The number in the brackets represents the improvement of the metric by prompting the LLMs with knowledge generated by DELLM.

BIRD Results. The knowledge generated by DELLM obtained promising results on both metrics in assisting different models/prompting techniques in the text-to-SQL task, which obtained around 5% improvement on EX and 6% in VES around all comparative baselines. Specifically, 1) The knowledge significantly improves the model's performance with simple prompting. We owe this phenomenon to the PL-refinement based on the database execution and ground-truth SQL contribution feedback, which enable DELLM to generate more helpful knowledge to assist SQL generation. The straightforward prompt challenges the LLMs' capability for question and schema understanding, especially when the question is challenging. DELLM generates the corresponding expert knowledge, reducing the difficulty of understanding, which leads to substantial improvement. 2) When assisting prompting techniques, the knowledge also achieves promising results. The prompting technique surpasses the simple prompt by focusing on providing high-quality few-shot instances or decomposition, which is a various angle from knowledge that assists in enhancing SQL generation. This means that DELLM can potentially improve the state-of-the-art performance from a knowledge-assisting angle when facing the scenario without annotated knowledge. Although the well-designed techniques achieve solid performance, there is still space for improvement in generating helpful knowledge.

Spider Results. The database scenario of Spider is less challenging. The performance on the same metrics is overall better, as we can see from the results. Similar to BIRD, 1) The knowledge substantially improves the performance of proprietary models. 2) The improvement brought by the knowledge is significantly lower than the BIRD dataset. We owe this result to the divergent complexity and characteristics of the database structure and the different difficulties of the user question. The question in the Spider dataset may not be as challenging as

Models	EX	VES
GPT-4 + DELLM	37.94	42.15
w/o *table reading*	37.23 (-0.71)	41.30 (-0.85)
w/o *db feedback*	36.25 (-1.69)	40.46 (-2.07)
w/o *sql feedback*	36.91 (-1.03)	41.12 (-1.55)

Table 3: Ablation study on variations of DELLM.

Models	EX	VES
GPT-4	33.25 ± 0.61	35.92 ± 1.03
+ DELLM$_{PPO}$	35.28 ± 1.18	40.03 ± 1.81
+ DELLM$_{DPO}$	37.94 ± 0.57	42.15 ± 0.95

Table 4: Ablation study on the utilized PL algorithms.

in the BIRD. As the number of questions that need to be assisted decreases, the improvement declines respectively.

4.3 Ablation Study

Table 3 presents the results of the ablation study for our framework in the BIRD dev set with proprietary LLM GPT-4. The table lists different variations of DELLM, including excluding database execution feedback (*db feedback*) or ground-truth SQL contributing feedback (*sql feedback*) and also removing the *table reading* process. The column presents the evaluation metrics.

As we can see in Table 3, removing any components leads to a performance decrease on both metrics. The performance declined the most when excluding the database feedback, indicating that assisting the predicted SQL in database execution is the priority in generating helpful knowledge. Overall, the ablation study demonstrates that each component of our method plays an irreplaceable role in achieving good performance, as their removal decreased the performance as intuitively expected.

We also discuss the choice of PL algorithm. Apart from the utilized direct preference optimization (DPO) (Rafailov et al., 2023) in our framework, we evaluate a variation of DELLM, which is PL-refined by proximal policy optimization (PPO) (Schulman et al., 2017).

Given the results in Table 4, we can conclude: 1) PPO works for DELLM, which can also generate helpful knowledge to improve the performance of the text-to-SQL model. 2) DPO outperforms PPO, one potential reason is that PPO relies on a reward model to give numeric rewards for each generated knowledge. Since the knowledge generation task is non-deterministic, even if the provided knowledge is correct, the downstream SQL generation may be wrong. In this case, the DPO algorithm with implicit rewards performs better.

Model	Simp.	Mod.	Chall.	All
GPT-3.5-Turbo	35.58	14.60	17.61	27.64
GPT-3.5-Turbo + D	43.09	18.30	17.61	33.31
GPT-3.5-Turbo + E	50.27	31.81	20.42	41.98
GPT-4	41.05	21.13	21.13	33.25
GPT-4 + D	47.16	24.18	21.83	37.94
GPT-4 + E	54.01	36.38	31.69	46.67

Table 5: The execution accuracy (EX) for questions with different difficulty. Simp. denotes Simple, Mod. denotes Moderata, and Chall denotes Challenging.

4.4 Comprehensive Evaluation

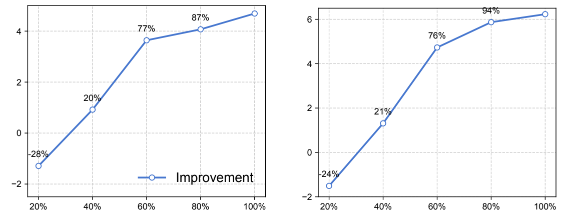
To reach a comprehensive conclusion, we discuss SQL generation at different difficulty levels and compare the knowledge generated by DELLM to the ground-truth knowledge. Table 5 shows the results on the BIRD dev set, decomposing different difficulty levels of the user question. In the table, D and E represent the knowledge generated by our proposed DELLM (D) and the ground-truth knowledge annotated by human experts (E).

From the results, 1) There is a gap between the DELLM-generated knowledge and the human expert-annotated one on all difficulty levels. The human-annotated knowledge surpasses 8.67% and 8.73% to DELLM on execution accuracy, leaving a great gap for improvement. 2) The generated knowledge mainly works on simple and moderate questions. One reason is that LLMs have difficulty understanding the challenging question and schema, making it hard to utilize corresponding generated knowledge. Another reason is that challenging questions only occupy a small portion of the annotations (around 9%), leading to a data imbalance during the training process.

4.5 Performance on Partial Training Data

We select partial annotated knowledge in the BIRD training set to train the DELLM, then discuss the performance of that scenario. We conducted the experiments using GPT-4, and the results are shown as the improvement with the generated knowledge by DELLM on different ratios of training data.

As shown in Figure 3, where the x-axis represents the ratio of training data, the y-axis is the value of the metric's improvement. In practice, with a limited budget, we can hire human experts



(a) Execution Accuracy (b) Valid Efficiency Score

Figure 3: Improvement to GPT-4 on different metrics with DELLM on different ratios of training data.

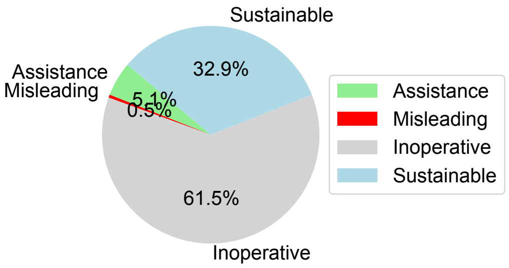


Figure 4: Different influences of DELLM bring on GPT-4's performance on the BIRD dev set.

to annotate about 60% of training data to obtain around 77% performance for expert knowledge generation of DELLM.

4.6 Statistical Analysis

We identify four types of influences that the DELLM's knowledge may bring: 1) Assistance, where the original predicted SQL is not correct but becomes correct with the help of the knowledge. 2) Misleading, where the original predicted SQL is correct, but becomes incorrect due to the influence of the knowledge. 3) Inoperative, where the original incorrect SQL remains incorrect despite the knowledge. 4) Sustainable, where the originally predicted SQL is correct and remains correct with the knowledge. The statistics are obtained based on the experiment result using GPT-4 on the BIRD dev set. From Figure 4, the DELLM brings around 5% assistance, and there is potentially 0.5% misleading of its generated knowledge. The inoperative knowledge leaves about 60% in DELLM and also around 53% in the expert annotation. One potential optimization to text-to-SQL is improving the effectiveness of incorporating expert knowledge.

5 Conclusion

In conclusion, our study highlights the significance of knowledge generation in enhancing the performance of LLMs for text-to-SQL tasks. By proposing a novel framework that leverages database content, execution feedback, and comparisons with ground-truth SQL, we address the challenges of bridging the knowledge gap between user questions and database schema. Through extensive experiments on BIRD and Spider datasets, our approach brings substantial improvements in execution accuracy and valid efficiency scores for models like GPT-4, underscoring its efficacy in advancing text-to-SQL research and fostering innovations in natural language processing and data mining.

6 Limitations

Two primary limitations are acknowledged in this study. Firstly, the performance of DELLM is mostly evaluated using proprietary models. However, the in-context learning and natural language understanding capability of open-source medium-scale local LLMs still have a large gap compared to proprietary LLMs. Although with the assistance of generated knowledge, their performance significantly improved, the overall performance is not good enough for practical use. This may limit the application of DELLM in offline deployment situations. Secondly, the generalizability of the proposed framework to real-world scenarios remains to be determined, as it has only been evaluated on standard benchmarks. The complexity, diversity, and potential noise in real-world database environments calls for further verification to confirm the framework's effectiveness in practical applications.

Ethics Statement

We confirm that we have fully complied with the ACL Ethics Policy in this study. All the datasets are publicly available and have been extensively used in research related to text-to-SQL.

Acknowledgements

This work was supported in part by the Innovation and Technology Commission of HKSAR and the Ministry of Science and Technology of China under the Innovation and Technology Fund - Mainland-Hong Kong Joint Funding Scheme (MHP/012/21), and in part by the Innovation and Technology Commission of HKSAR under the Research Talent Hub for ITF projects (Pih/054/23).

References

Anthropic. 2023. Introducing Claude.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt LLMs for text-to-SQL: A study in zero-shot, single-domain, and cross-domain settings. In *NeurIPS 2023 Second Table Representation Learning Workshop (NeurIPS)*.

Hao Chen, Yuanchen Bei, Qijie Shen, Yue Xu, Sheng Zhou, Wenbing Huang, Feiran Huang, Senzhang Wang, and Xiao Huang. 2024. Macro graph neural networks for online billion-scale recommender systems. In *International World Wide Web Conference (WWW)*.

Hao Chen, Yue Xu, Feiran Huang, Zengde Deng, Wenbing Huang, Senzhang Wang, Peng He, and Zhoujun Li. 2020. Label-aware graph convolutional networks. In *International Conference on Information and Knowledge Management (CIKM)*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *International Conference on Learning Representations (ICLR)*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Junnan Dong, Qinggang Zhang, Xiao Huang, Keyu Duan, Qiaoyu Tan, and Zhimeng Jiang. 2023a. Hierarchy-aware multi-hop question answering over knowledge graphs. In *International World Wide Web Conference (WWW)*.

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

Longxu Dou, Yan Gao, Xuqi Liu, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, Min-Yen Kan, and Jian-Guang Lou. 2022. Towards knowledge-intensive text-to-SQL semantic parsing with formulaic knowledge. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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. In *International Conference on Very Large Data Bases (VLDB)*.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W. Bradley Knox, and Dorsa Sadigh. 2024. Contrastive preference learning: Learning from human feedback without reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Zijin Hong and Jian Liu. 2024. Towards better question generation in qa-based event extraction. *arXiv preprint arXiv:2405.10517*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023a. Graphix-t5: mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Conference on Artificial Intelligence (AAAI)*.

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 Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. 2023c. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. *arXiv preprint arXiv:2305.11700*.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S Yu. 2023a. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research (JMLR)*.

Daking Rai, Bailin Wang, Yilun Zhou, and Ziyu Yao. 2023. Improving generalization in language model-based text-to-SQL semantic parsing: Two simple semantic boundary-based techniques. In *Association for Computational Linguistics (ACL)*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Empirical Methods in Natural Language Processing (EMNLP)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. Mac-sql: Multi-agent collaboration for text-to-sql. *arXiv preprint arXiv:2312.11242*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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 *Empirical Methods in Natural Language Processing (EMNLP)*.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023a. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Qinggang Zhang, Junnan Dong, Hao Chen, Xiao Huang, Daochen Zha, and Zailiang Yu. 2023b. Knowgpt: Black-box knowledge injection for large language models. *arXiv preprint arXiv:2312.06185*.

Qinggang Zhang, Junnan Dong, Qiaoyu Tan, and Xiao Huang. 2023c. Integrating entity attributes for error-aware knowledge graph embedding. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A Implementations Details.

We discuss the implementation details of our proposed framework. We used parameter-efficient fine-tuning (PEFT) in the training stage to train our models. Specifically, in each stage (supervised fine-tuning and DPO preference learning), we utilize low-rank adaptation (LoRA) (Hu et al., 2021) as PEFT method, the trainable parameters occupy 0.0622% of full parameters. All the experiments are conducted on four NVIDIA A800-SXM4-80GB GPUs, and the transformers package version is 4.36.2. The details of training and hyper-parameters are listed as follows.

A.1 Supervised Fine-tuning

As previously introduced, the backbone model of DELLM is selected as LLaMA-2-13b with official configuration[4]; the training details are given in Table 6. The model's architecture is identical to the official provided in Huggingface.

Hyper-parameters	Value
data type	fp16
learning rate	5e-05
number of epochs	3
number of batch size	32
gradient accumulation steps	4

Table 6: Hyper-parameters of SFT.

A.2 DPO Training

The hyper-parameters of DPO are similar to the previous stage, the details are shown in Table 7.

Hyper-parameters	Value
data Type	fp16
learning rate	1e-05
number of epochs	1
number of batch size	16
gradient accumulation steps	4

Table 7: Hyper-parameters of the DPO.

A.3 Generation Configuration

In each generation of parts of our framework during training and testing, the configuration is identical. The details are listed in Table 8.

Configuration	Value
top p	0.9
do sample	True
temperature	0.6
max token length	4096
predict with generate	True

Table 8: Generation configuration

A.4 Versions of Proprietary LLMs

The version of ChatGPT in this work is GPT-3.5-Turbo-1106, and the version of GPT-4 is GPT-4-0613.

[4] https://huggingface.co/meta-llama