SQL-to-Text Generation with Database-Grounded Feedback via Preference Optimization

Anonymous ACL submission

Abstract

As AI systems increasingly handle complex data tasks, generating faithful natural-language explanations for structured queries, *i.e.*, SQL-to-text, is becoming vital for interpretability. This task, however, is challenging due to diverse professional requirements involving SQL faithfulness, database schema consistency, and textual quality. Rather than directly optimizing for these heterogeneous objectives, we revisit SQL-to-text from a data-centric perspective, casting it as a unified pairwise preference learning problem that jointly accounts for textual and semantic quality dimensions. To ensure data scalability, we collect AI-generated preference pairs and further enhance reliability by incorporating real database feedback, leveraging query execution and schema-consistency signals to construct verifiable preference examples. This feedback-based preference optimization (FPO) framework remains lightweight and data-efficient, requiring only minimal fine-tuning. Extensive experiments verify that FPO could consistently improve linguistic quality, perceived goodness, and execution fidelity across four widely used benchmarks, demonstrating the effectiveness of unifying SQL-to-text alignment under feedback-informed preference optimization.

1 Introduction

SQL-to-text aims to automatically translate a Structured Query Language (SQL) query into natural-language text. The generated description enables users to understand the intent and effect of a query without delving into its syntax or database schema, thereby supporting applications such as SQL comment generation. Moreover, SQL-to-text facilitates data augmentation for text-to-SQL tasks, effectively mitigating data scarcity in developing such systems (Yang et al., 2024; Cheng et al., 2025). In the era of generative AI, this task is becoming increasingly pivotal for bridging machine-executable logic and human-understandable reasoning, given the demonstrated capabilities of large language models (LLMs) in automating SQL-related operations (Salesforce, 2025; Microsoft, 2025). More generally speaking, SQL-to-text reflects a broader shift in which AI plays an active role in data annotation and synthesis. Considering the Web-rooted nature of modern LLMs, our study also connects to the pursuit of reliable and cost-efficient data curation grounded in large-scale web knowledge.

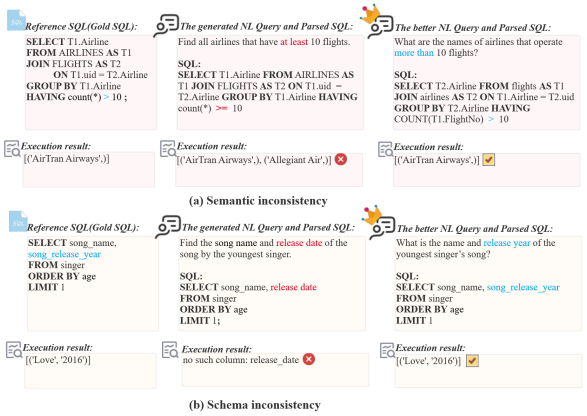

Figure 1: Inconsistency issues of SQL-to-text generation.

Building on its growing importance, SQL-to-text has been studied for a while under database engineering and natural language generation. Early approaches relied on manually crafted rules and templates to map SQL queries to natural-language descriptions (Koutrika et al., 2010), which required substantial human effort and deep domain expertise. Subsequent work adopted deep learning techniques, such as graph-to-sequence and medium-scale pre-trained language models, to produce more fluent and structurally coherent text (Ngomo et al., 2013; Shi et al., 2021; Bhardwaj et al., 2025). With the emergence of LLMs, recent studies have explored using LLMs to directly synthesize natural-language questions from SQL snippets (Shkapenyuk et al., 2025; Cheng et al., 2025). These approaches ben-

efit from broad textual exposure and open-source SQL corpora, providing a basic alignment between SQL logic and natural language.

Nevertheless, the synthesized SQL–text pairs often exhibit inconsistencies across multiple professional requirements, such as SQL faithfulness and database-schema consistency. First, **semantic inconsistency** arises when the generated description fails to accurately convey the logical intent of the query. For example, in Figure 1(a), the phrase "*at least 10 flights*" misinterprets the SQL condition "*count(*) > 10*", revealing a subtle yet critical semantic deviation that even advanced LLMs struggle to detect. Second, **schema inconsistency** occurs when the generated text references non-existent attributes or relations, leading to unfaithful or invalid translation (Xu et al., 2019; Shkapenyuk et al., 2025; Bhardwaj et al., 2025). As shown in Figure 1(b), the phrase "*release date*" refers to no column in the *singer* table, rendering the description mismatched with the original query.

These examples underscore a gap between surface-level fluency and semantic precision. A recent study (Al-Lawati et al., 2025) leveraged the Abstract Syntax Tree (AST) of the SQL query to retrieve structurally similar examples for In-Context Learning (ICL), guiding LLMs to generate text that better aligns with the given SQL. Although AST-guided ICL improves structural consistency, misalignment persists due to the domain-agnostic nature of general-purpose LLMs. Notably, prior studies (Al-Lawati et al., 2025; Bhardwaj et al., 2025) largely overlook direct interaction with databases, relying instead on surface-level metrics such as BLEU (Papineni et al., 2002) and BERTScore (Reimers and Gurevych, 2019). Such metrics cannot ensure execution correctness or detect subtle semantic mismatches. These limitations highlight the need for a unified and verifiable optimization framework capable of aligning generated text with both database-grounded semantics and human-readable quality.

To this end, we revisit the SQL-to-text task from a data-centric perspective and propose Feedback-involved Preference Optimization (FPO), a unified framework that aligns generated text with SQL faithfulness, schema consistency, and human-readable quality. Instead of directly optimizing these diverse objectives, we reformulate the task as a pairwise preference learning problem. Following preference optimization for LLMs (Ouyang et al., 2022), FPO first defines multi-dimensional preferences for SQL-to-text generation, encompassing semantic fidelity, execution success, and text quality. It then constructs cost-effective preference triplets based on existing SQL corpora. To ensure both scalability and verifiability, we adopt a hybrid preference construction strategy: (a) generating candidate pairs through automated LLM-based synthesis and rating, followed by (b) validating them via real database feedback. This database-grounded supervision allows the model to learn from verifiable signals rather than surface proxies, substantially improving the alignment with domain requirements.

We conduct experiments on four popular benchmarks (Spider (Yu et al., 2018), CoSQL (Yu et al., 2019a), SParC (Yu et al., 2019b) and BIRD (Li et al., 2023a)), and adopt both classical BLEU and BERTScore with additional Win Rate (WR), Exact Match (EM), and Execution Accuracy (EX) metrics to thoroughly assess the extrinsic and intrinsic quality of generated text. Extensive results demonstrate that FPO consistently outperforms prior approaches, achieving improved linguistic quality (+2.26% BLEU), perceived goodness (+9.3% WR), and execution fidelity (+4.47% EX) with the same Llama-3 backbone. Note that these gains are obtained by simply post-training an LLM with 1340 automatically-collected preference pairs. Our contributions are summarized as follows:

- Conceptually, we revisit SQL-to-text as preference optimization, and develop a lightweight, data-efficient framework involving data construction, preference alignment, and comprehensive evaluation to advance related studies.
- Technically, we introduce database-grounded feedback as the primary source for alignment—which enhances reliability by incorporating real query execution and schema-consistency signals—as well as systematically study and validate the quality of the generated synthetic data, offering insights for better utilizing augmented data in SQL-to-text tasks.
- Empirically, we conduct extensive tests using four datasets and five metrics, and results consistently confirm the effectiveness of unifying SQL-to-text alignment under FPO.

2 Related Work

2.1 SQL-to-text

Recent work increasingly leverages LLMs to synthesize high-quality (text, SQL) pairs, which are

essential for Supervised Fine-tuning (SFT) and ICL in SQL-related tasks. Early approaches relied on rule- or template-based generation (Koutrika et al., 2010; Ngomo et al., 2013), but suffered from limited diversity and unnatural language expression. Later work leveraged graph-to-sequence models (Xu et al., 2019) and pretrained language models such as Qwen (Qwen et al., 2025), further improving generation quality (Shi et al., 2021; Wang et al., 2021; Liu et al., 2026; Baek et al., 2025; Chen et al., 2025). Al-Lawati et al. (2025) improved semantic alignment by selecting ICL examples based on structured SQL representations. Methods like Sense(Yang et al., 2024) and OMINISQL (Li et al., 2025) use ICL/CoT (Wei et al., 2022) to generate (database, question, SQL) triples, boosting Text-to-SQL performance via efficient data synthesis. More recent works integrate reasoning with preference alignment, leveraging GPT-generated CoT samples and SQL for DPO training, as well as stepwise execution-aware reinforcement learning to guide SQL generation (Liu et al., 2025; Zhai et al., 2025). Nevertheless, these methods overlook data quality and preference alignment in SQL-to-text, which we address by introducing a feedback-involved preference optimization framework that jointly considers text quality, semantic fidelity, and execution feedback.

2.2 LLM Preference Alignment

In recent years, numerous methods have leveraged human feedback to align LLMs with human preferences. Reinforcement Learning from Human Feedback (RLHF) is a key approach (Ouyang et al., 2022), typically minimizing human preference loss to train a reward model (Bradley, 1964), followed by policy optimization using Proximal Policy Optimization (PPO) (Schulman et al., 2017). Despite its effectiveness, PPO is resource-intensive. To reduce costs and stabilize training, Rafailov et al. (2023) propose DPO, which directly optimizes LLMs using preference pairs. Building upon DPO, several improvements such as ORPO (Hong et al., 2024), KTO (Ethayarajh et al., 2024), and SimPO (Meng et al., 2024) have since been developed. In addition, preference alignment has also been widely applied to downstream tasks (Yu et al., 2025). For example, Xu et al. (2024) use RLHF to enhance translation quality. Other studies focus on improving alignment through better data quality. Deng et al. (2025) propose a marginal maximization method to filter preference pairs, leading to improved performance. Compared with the above methods, our method introduces the preference alignment to SQL-to-text, where the preference is especially identified with the consideration of three perspectives.

3 Preliminary

3.1 Problem Statement

We formally give a definition to SQL-to-text. Given an SQL query X associated with a database $\mathcal{D}$, a system should automatically convert the query into a natural language text $\hat{Y}$ that explicates the semantics of X as follows:

$$f : \mathcal{X} \rightarrow \mathcal{Y}, \quad \text{where } \hat{Y} = f(X).$$

Noting that it is not trivial to evaluate the quality of $\hat{Y}$. Considering the real-world text-to-SQL scenarios (Shkapenyuk et al., 2025; Cheng et al., 2025), we exhibit distinct preferences in SQL-to-text with the following three perspectives:

- **Semantic Fidelity**: The generated texts $\hat{Y}$ should have the aligned semantics with SQL X, including the classes (tables), attributes (columns), and the relationships.
- **Execution Success**: The generated texts $\hat{Y}$ could be successfully verified via the database execution. Its intention should be clear enough such that it can be easily parsed to X in reverse.
- **Text Quality**: Similar to the text generation in general cases, the generated question $\hat{Y}$ should be fluent, coherent, and preferred by humans, which showcases the readability of the text.

According to previous studies of preference alignment (Ouyang et al., 2022; Rafailov et al., 2023; Meng et al., 2024), text quality is a general evaluation aspect for natural languages, while semantic fidelity and execution consistency are distinct preferences we should consider elaborately for SQL-to-text.

4 Our Method

We propose a novel framework with feedback-involved preference optimization, a preference learning framework designed for SQL-to-text, as shown in Figure 2. Starting from the reverse of existing text-to-SQL pairs, we construct preference prompts to guide GPT-4o in generating preferred data, and then optimize the model using a feedback-aware objective to further improve the generated texts. The overall pipeline is summarized in Algorithm A.10, with an example in Appendix A.11.

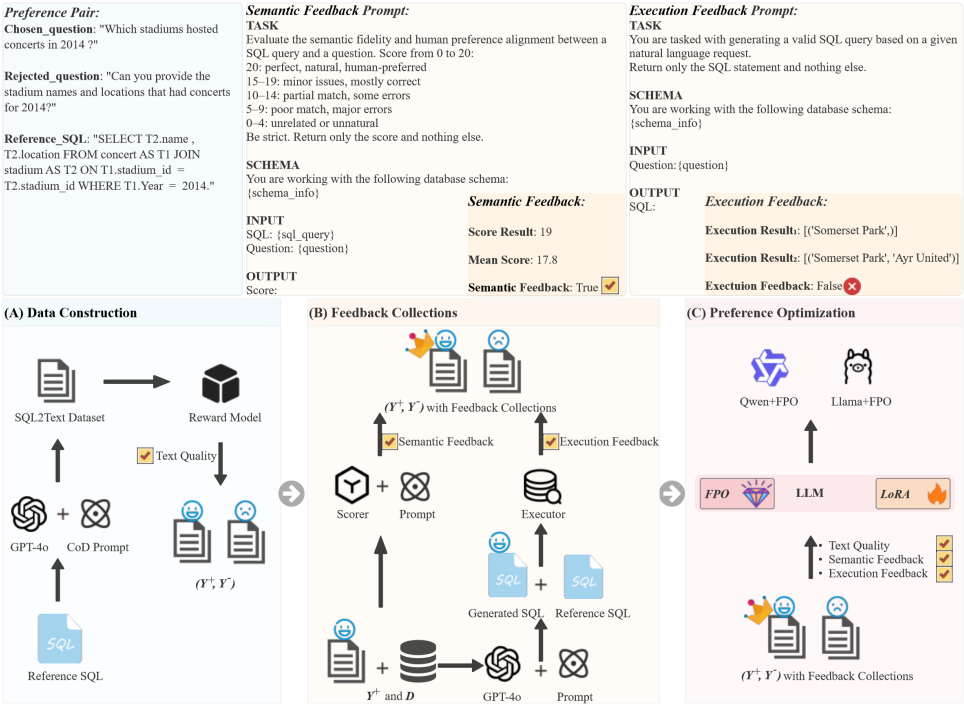

Figure 2: Overview of our framework

4.1 Data Construction

Therefore, we utilize the identified perspectives as prior knowledge to guide the LLMs to synthesize preferred texts. Following the prior studies (Al-Lawati et al., 2025), we reverse the text-to-SQL process to obtain a set of texts based on SQLs. Specifically, for each SQL query, we design Chain-of-Density (CoD) (Adams et al., 2023) prompts, whereby an initial question is generated and made increasingly schema dense in an iterative manner.

With the above procedure, we obtain n pieces of different texts via GPT-4o given an SQL query, which are denoted as $\{Y_1, ..., Y_i, ..., Y_n\}$. We consider the text in the original text-to-SQL dataset as a reference, which is denoted as Y_{ref}. Then, we leverage Skywork-Reward-Llama-3.1-8B-v0.2 (Liu et al., 2024), which is commonly used for labeling preference data to score both the original text and the synthesized texts. We form a preference triplet in the following two cases:

1. If the annotated score of Y_{ref} is higher than that of the synthesized text with the lowest score, we combine SQL query, Y_{ref} as well as the synthesized text with the lowest score in a triplet, denoted as $(\text{SQL}, Y_{\text{ref}}, Y_i)$, where Y_{ref} and Y_i play the roles of preferred Y^+, and unpreferred Y^-, respectively.

2. We also form a preference triplet using SQL query, synthesized texts with the highest and lowest scores, denoted as (SQL, Y_j, Y_i), where Y_j and Y_i play the roles of preferred Y^+, and unpreferred Y^-, respectively.

As a result, we denote the set of constructed preference data as $\mathcal{U}$. The constructed preference data judged by the reward model largely meets our preference criteria of text quality.

4.2 Feedback-involved Preference Optimization

DPO has demonstrated strong effectiveness in text generation, but it heavily relies on the assumption that both the training data and preference annotations are reliable. However, in SQL-to-text, this assumption often fails due to the semantic misalignment between the SQL query and the generated text. Moreover, without verification or interaction with databases, it is likely to generate texts having conflict with the databases.

On the one hand, the constructed preference triplets based on the highest and lowest scoring could ensure the text quality of the texts, but they remain susceptible to misalignment with the database. For example, as shown in Figure 2, the reference text is *"Find the name and location of the stadium that hosted concerts in 2014."* while the preferred text is *"Which stadiums hosted concerts in 2014?"*. Although the latter is more concise and

natural in phrasing, it only implicitly suggests the stadium name and fails to mention the location, which is misaligned with the original SQL query. This results in an incorrect result, demonstrating that higher preference scores cannot guarantee execution success or semantic fidelity. On the other hand, if the original texts from existing datasets contain errors — as frequently observed in Text-to-SQL corpora (Yu et al., 2018) — constructing pairs using reference and low-scoring candidates may introduce incorrect preferences as noisy signals.

To address this issue, we propose feedback-involved preference optimization where the semantic fidelity and execution success are further confirmed.

Feedback Collections. We first collect the feedback for optimization by interacting with a database $\mathcal{D}$. We take full advantage of either the schema information or all the data stored in the database.

Regarding the semantic fidelity, we expect there to be semantic overlap between the SQL query and the generated question, such as classes, attributes, and relationships. So we employ an LLM as the scorer to obtain the feedback. For Y^+ in each preference triplet (SQL, Y^+, Y^-) with the prompt in Figure 2. As a result, the LLM judges whether the semantic similarity between the synthesized texts and the SQL query in the context of the database is top-ranked. We denote it as:

$$Y^+ = \text{Top } k[\text{ Scorer}(Y^+, \text{SQL}, \mathcal{D})], \quad (1)$$

where $\mathcal{D}$ denotes the schema information of the database and k denotes the top-ranked synthesized texts we would like to retrieve corresponding to the same SQL.

Regarding execution success, LLMs are considered effective tools for parsing natural language into structured queries (Zhang et al., 2024). They leverage semantic understanding and database ontology to improve SQL generation accuracy, with the ontology also regarded as an important context to facilitate parsing. For Y^+ in each preference triplet (SQL, Y^+, Y^-), we employ GPT-4o as the SQL parser, prompting it to generate SQL queries given both the texts and relevant schema as shown in Figure 2. This results in an SQL query. Then we execute it against the database and retrieve the results. For example, we obtain the feedback "*Somerset Park*" through interacting with the database. Then, we execute the SQL query in the triplet and obtain feedback. We can compare the feedback with the SQL query in the triplet; if the execution feedback is the same, this indicates that the generated question has consistent execution results as the original SQL.

$$\hat{X} = \text{GPT-4o}(Y^+),$$
$$\text{Executor}(\hat{X}) = \text{Executor}(\text{SQL}). \quad (2)$$

Training Objective. Based on the collected feedback, we define an indicator function $z(\cdot)$ to indicate whether a preference triplet is valid. Then we integrate the indicator function into the DPO objective, and our training loss becomes:

$$\mathcal{L}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(\text{SQL}, Y^+, Y^-)}\Big[z(\text{SQL}, Y^+, Y^-) \cdot \log \sigma\Big(\beta\big(\log \tfrac{\pi_\theta(Y^+|\text{SQL})}{\pi_{\text{ref}}(Y^+|\text{SQL})} - \log \tfrac{\pi_\theta(Y^-|\text{SQL})}{\pi_{\text{ref}}(Y^-|\text{SQL})}\big)\Big)\Big]$$

where

$$z(\text{SQL}, Y^+, Y^-) = 1 \quad \text{iff} \quad \text{Eq. (1)} \wedge \text{Eq. (2)}$$

As we can see, $z(\text{SQL}, Y^+, Y^-)$ controls the semantic fidelity and execution success for the LLMs' annotation. Only when the annotation has strong semantic alignment to the SQL query and its back translation to SQL query can be successfully executed against the database, the preference triplet can be activated. Compared to the standard DPO training loss, our feedback-involved preference optimization includes an indicator to annotate the necessity of the preference data.

Please note that our proposed feedback-involved preference optimization can be deemed as a unified pairwise preference learning problem that can be extended to more general dimensions such as term quality and difficulty level control.

5 Experimental Setup

5.1 Dataset

We evaluate our method on four SQL-to-text datasets: CoSQL-S2T, SParC-S2T, Spider-S2T, and BIRD-S2T. For brevity, the -S2T suffix is omitted in the tables. The first three datasets are derived from the widely used text-to-SQL benchmarks CoSQL (Yu et al., 2019a), SParC (Yu et al., 2019b), and Spider (Yu et al., 2018), following prior SQL-to-text work (Al-Lawati et al., 2025). BIRD (Li et al., 2023a) is a recently released text-to-SQL dataset covering complex SQL queries across 95 domains, and we construct BIRD-S2T via reverse translation and LLM synthesis.

To evaluate generalization, we adopt a leave-one-dataset-out setup (Liu et al., 2023; Zhang et al., 2023). Specifically, the model is trained on three datasets and evaluated on the remaining one.

5.2 Evaluation Metrics

We evaluate generated texts with both intrinsic and extrinsic metrics. For intrinsic evaluation, we adopt BERTScore (Reimers and Gurevych, 2019) and BLEU (Papineni et al., 2002) to measure semantic similarity and textual quality. We also compute win rate (WR) in two forms: (1) Group WR using the open-source reward model *Skywork-Reward-Llama-3.1-8B-v0.2* (Liu et al., 2024), where a generated text wins only if its score surpasses all candidates in the group including Y_{ref} and $\{Y_1, \ldots, Y_k\}$; and (2) Pair-wise WR using GPT-4o as a preference judge following common LLM evaluation practices (Li et al., 2023b, 2024; Zheng et al., 2023). For extrinsic evaluation, we report Exact Match (EM) and Execution Accuracy (EX) by translating the generated texts back into SQL.

5.3 Comparable Methods

We select six widely used open-source and closed-source LLMs that are effective in SQL-to-text generation, including GPT-J-6B (Wang and Komatsuzaki, 2021), Mistral-7B (Jiang et al., 2023), CodeLlama-7 (Rozière et al., 2024), Llama3-8B-Instruct (Llama) (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Qwen) (Qwen et al., 2025), and GPT-4o (OpenAI et al., 2024). These models serve as our baselines. AST-ICL (Al-Lawati et al., 2025) is an advanced inference strategy designed for SQL-to-text, which extracts the Abstract Syntax Tree (AST) as the structure features from the SQL query as the promoting contexts. We reproduced the AST-ICL method with different backbone models and report the best experimental results on the test sets, measured by Win Rate.

5.4 Implementation Details

We perform FPO on both Llama and Qwen backbones with the preference temperature coefficient $\beta = 0.01$, using $n = 5$ initial triplet candidates and selecting the top $k = 3$. Models are trained with AdamW for 10 epochs. Additional training and implementation details are provided in Appendix A.9.

6 Results

We evaluate our method to answer the following research questions: RQ1) the quality of the text generated by our method; RQ2) how our method benefits the SQL-to-text task; RQ3) what aspects of text quality are improved by our method; RQ4) the effects of each component in our method.

6.1 Results of Intrinsic Evaluation (RQ1)

Based on Table 1 and Figure 3, we have the following two observations:

(1) Our proposed FPO consistently outperforms the standard DPO and the vanilla AST-ICL (Al-Lawati et al., 2025) methods across CoSQL-S2T, SParC-S2T, Spider-S2T, and BIRD-S2T datasets. FPO achieves average absolute improvements in WR of 9.3% and 3.7% over the vanilla AST-ICL methods and the standard DPO on the Llama3 backbone, respectively, while also achieving superior performance on other intrinsic metrics such as BLEU and BERTScore. These results demonstrate the effectiveness and generalizability of our method for the SQL-to-text task, enhancing the preference alignment. We further evaluate the generalization ability of FPO under a pure zero-shot setting without in-context demonstrations. FPO consistently improves over both the vanilla zero-shot baseline and the DPO-enhanced variant. Detailed results are provided in Appendix A.1.

(2) To provide more explicit evaluation on the quality of the generated text, we employ GPT-4o as a judge to pairwisely compare our outputs and those from baselines, including dataset references, Llama+DPO, GPT-4o, as well as the Llama backbone. A unified prompt guides the evaluation, with position swaps and multiple rounds ensuring fairness and robustness (Li et al., 2023b). As illustrated in Figure 3, FPO is consistently preferred over all baselines, indicating clear advantages in preference alignment. Notably, FPO's preference over original references suggests that some dataset descriptions may be suboptimal in schema alignment or semantic fidelity.

In addition, a separate human evaluation in Appendix A.7 corroborates these findings. A further comparison with GPT-4o (Appendix A.2) shows that FPO enables smaller models to approach the performance of large proprietary ones.

6.2 Results of Extrinsic Evaluation (RQ2)

To showcase the effect of our proposed FPO method in boosting the text-to-SQL challenges with three types of data for training: (1) the references in datasets; (2) the generated text from Llama+AST-ICL; and (3) the generated text from

Model	Method	WR (%)				BERTScore				BLEU			
		CoSQL	SParC	Spider	BIRD	CoSQL	SParC	Spider	BIRD	CoSQL	SParC	Spider	BIRD
GPT-J-6B	-	6.5	10.4	13.9	11.7	0.7429	0.7311	0.7446	0.6848	0.1988	0.2143	0.2252	0.1489
Mistral-7B-v0.1	-	17.2	25.0	28.1	15.4	0.8216	0.8411	0.8473	0.7645	0.2944	0.3242	0.3255	0.2382
CodeLlama-7B	-	13.8	19.1	20.1	9.6	0.8042	0.8336	0.8321	0.7399	0.2610	0.3111	0.3178	0.2061
Llama	-	35.1	31.2	36.9	34.6	0.8415	0.8529	0.8498	0.7838	0.3305	0.3516	0.3551	0.2805
	DPO	40.7	41.0	42.3	36.2	0.8481	0.8597	0.8592	0.7829	0.3395	0.3668	0.3624	0.2809
	FPO	**45.2**	**43.1**	**46.7**	**39.9**	**0.8523**	**0.8672**	**0.8687**	**0.7871**	**0.3567**	**0.3786**	**0.3786**	**0.2943**
Qwen	-	40.3	38.5	41.6	37.8	0.8453	0.8524	0.8484	0.7813	0.3579	0.3692	0.3643	0.2518
	DPO	43.8	42.7	45.6	39.9	0.8520	0.8646	0.8646	0.7943	0.3648	0.3855	0.3977	0.2887
	FPO	**46.1**	**44.8**	**47.4**	**43.6**	**0.8566**	**0.8669**	**0.8694**	**0.8053**	**0.3724**	**0.3976**	**0.4104**	**0.3009**

Table 1: Performance comparison of different models and methods on four benchmarks. "-" denotes vanilla AST-ICL (Al-Lawati et al., 2025), the leading method for SQL-to-Text generation, without any post-training. WR denotes the Group Win Rate (Group WR).

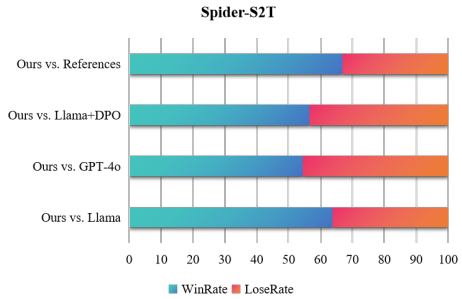

Figure 3: Pairwise preference win rates evaluated by GPT-4o on Spider-S2T dataset.

Parsing Model	Text Source	EX (%)	EM (%)
Qwen2.5-7B	Dataset Reference	63.81	29.52
	Llama+AST-ICL	61.43	24.76
	Llama+FPO	**66.67**	**30.00**
Llama3-8B	Dataset Reference	52.38	14.29
	Llama+AST-ICL	51.32	13.76
	Llama+FPO	**55.02**	**16.40**

Table 2: Evaluation results on EX and EM with different training data on Spider-S2T.

Llama+FPO. We employ Qwen2.5-7B-Instruct and Llama3-8B-Instruct to train a model for translating a natural text to its SQL query. Following the previous studies (Zhang et al., 2024; Cen et al., 2025), the relevant schema to a piece of text is considered as the context to form an instruction input. Regarding the synthesized text, annotated SQLs are given to generate natural text, and these texts are utilized as the training data to train a similar text-to-SQL model. Eventually, we evaluate EX and EM of text-to-SQL tasks to verify the quality of the training data. As shown in Table 2, FPO consistently outperforms the baselines in EX and EM, demonstrating that FPO effectively enhances consistency among the generated text, the corresponding SQL queries, and the database. This also suggests its potential for improving the overall quality of existing datasets as well as boosting the text-to-SQL performance in the shortage of training data.

We further analyze the performance across different SQL difficulty levels, and the detailed results are provided in Appendix A.3.

6.3 Visualization Analysis (RQ3)

To investigate the fine-grained factors behind performance improvements, we present a comparative scatter plot (Figure 4), where each point denotes a test case. Llama+FPO shows clear advantages in many cases. On the left of the figure, most blue arrows point from lower-left to upper-right, indicating that our method improves both text quality and semantic quality. This trend suggests that our model could better capture the intent of complex SQL queries while maintaining textual fluency. The consistent upward movement of the red arrow further confirms the overall improvement trend of FPO.

However, we also observe a small subset of cases where the Llama+AST-ICL model outperforms ours. As shown on the right side of Figure 4, for these cases, the outputs of the Llama+AST-ICL receive higher reward and semantic similarity scores compared to those of Llama+FPO. We investigate the cases and find that this may be caused by subtle semantic deviations introduced during the conversion from standard SQL to natural language and back, due to the inherent randomness in LLM sampling. Although the expressions are more fluent and natural, small shifts such as using adjectival nationality terms (e.g., "is American" instead of "is from America") can mislead the model and

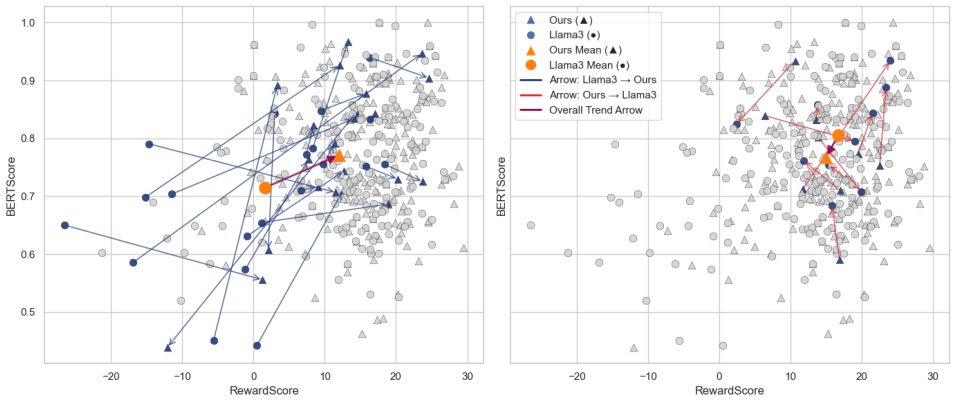

Figure 4: Visualizing EX shifts between the Llama+AST-ICL and Llama+FPO on Spider-S2T. Each dot denotes a piece of text in the test set generated via either Llama+AST-ICL or Llama+FPO. The x-axis denotes the reward derived from *Skywork-Reward-Llama-3.1-8B-v0.2*, and the y-axis reflects the BERTScore of each generated text. For each generated text, we translate it into an SQL query and execute it. If there are two texts derived from the same SQL query, the one generated via Llama+FPO is executable while the other one generated via Llama+AST-ICL is unexecutable, and we connect them through a blue arrow from Llama+AST-ICL to Llama+FPO. Otherwise, we draw a revised arrow in orange. The red arrows showcase the overall directional trend.

Model	WR (%)	BERTScore	BLEU
Llama	36.9	0.8498	0.3551
Llama+FPO	**46.7**	**0.8687**	**0.3786**
w/o Max	41.6	0.8652	0.3696
w/o Ref	44.2	0.8662	0.3658
w/o Exe	43.1	0.8665	0.3680
w/o Sem	43.5	0.8647	0.3671

Table 3: Ablation Study on Spider-S2T. In this table, WR denotes the Group Win Rate (Group WR). "Max" denotes the triplets with synthesized text annotated with the highest and lowest scores, "Ref" denotes the triplets with reference text and synthesized text annotated with the lowest scores. "Exe" denotes the existence of execution feedback, and "Sem" denotes the existence of semantic feedback.

reduce SQL accuracy. This also suggests potential avenues for future improvement of our model on specific cases.

6.4 Ablation Study (RQ4)

To assess the impact of different components in our method, we conduct ablation studies using Llama+FPO. As shown in Table 3, removing the triplets with synthesized text annotated with the highest and lowest scores ("w/o Max") significantly reduces WR, highlighting their role in providing strong positive signals. Excluding the triplets with reference text and synthesized text annotated with the lowest scores ("w/o Ref") causes a smaller yet consistent performance drop, indicating that Y_{ref} also offers reliable supervision. These findings validate our design: "Max" pairs help discriminate output quality, while "Ref" triplets guide alignment with human references—jointly enhancing semantic fidelity and text quality.

We further observe that removing execution or semantic feedback leads to notable performance drops, highlighting their importance. Execution feedback guides the model toward generating texts that yield consistent SQL results, aligning with our strategy of verifying execution consistency via SQL parsing and database interaction. Semantic feedback enhances alignment by leveraging an LLM to assess semantic overlap between the text and SQL within the database context. Together, these signals provide complementary supervision, improving both the accuracy and fidelity of generation.

7 Conclusion

We revisit the SQL-to-text task and propose a unified framework that addresses key limitations of existing methods by aligning generation with human preferences. To improve semantic fidelity and execution consistency, we introduce Feedback-involved Preference Optimization (FPO), which incorporates database interaction to guide the optimization process. We evaluate our approach using BLEU, BERTScore, EM, EX, and WR, and experimental results show consistent improvements over existing SQL-to-text methods. Our framework provides a more accurate, executable, and preference-aligned solution for generating natural language from structured queries.

Limitations

Although the proposed FPO framework demonstrates consistent improvements across multiple datasets and models, several limitations remain.

First, our execution feedback mechanism relies on SQL back-translation, which may occasionally introduce noise. Although FPO mitigates this by combining execution feedback with text quality scoring and semantic feedback (see Appendix A.8.1), some back-translation errors may still persist, potentially affecting preference data quality in edge cases.

Second, our experimental setup assumes a static database state, whereas real-world databases are inherently dynamic. Consequently, one-time execution feedback may not generalize to evolving data environments. While FPO learns general SQL-to-text mapping rules rather than domain-specific schema knowledge, this suggests a natural foundation for transfer. Nevertheless, adapting the framework to handle dynamic database updates remains an important direction for future work.

References

Asma Ben Abacha, Wen wai Yim, Yujuan Fu, Zhaoyi Sun, Meliha Yetisgen, Fei Xia, and Thomas Lin. 2025. Medec: A benchmark for medical error detection and correction in clinical notes. *Preprint*, arXiv:2412.19260.

Griffin Adams, Alexander Fabbri, Faisal Ladhak, Eric Lehman, and Noémie Elhadad. 2023. From sparse to dense: Gpt-4 summarization with chain of density prompting. *Preprint*, arXiv:2309.04269.

Ali Al-Lawati, Jason Lucas, and Prasenjit Mitra. 2025. Semantic Captioning: Benchmark Dataset and Graph-Aware Few-Shot In-Context Learning for SQL2Text. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 8026–8042. Association for Computational Linguistics.

Jinheon Baek, Horst Samulowitz, Oktie Hassanzadeh, Dharmashankar Subramanian, Sola Shirai, Alfio Gliozzo, and Debarun Bhattacharjya. 2025. Knowledge base construction for knowledge-augmented text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26569–26583, Vienna, Austria. Association for Computational Linguistics.

Meher Bhardwaj, Ethari Hrishikesh, and Dennis Singh Moirangthem. 2025. Ezsql: An SQL intermediate representation for improving sql-to-text generation. *Expert Syst. Appl.*, 289:128293.

Ralph A. Bradley. 1964. 'the rank analysis of incomplete block designs. ii. additional tables for the method of paired comparisons.'. *Biometrika*, 51(1-2):288–288.

Jipeng Cen, Jiaxin Liu, Zhixu Li, and Jingjing Wang. 2025. Sqlfixagent: Towards semantic-accurate text-to-sql parsing via consistency-enhanced multi-agent collaboration. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 49–57. AAAI Press.

Bingfeng Chen, Shaobin Shi, Yongqi Luo, Boyan Xu, Ruichu Cai, and Zhifeng Hao. 2025. Track-SQL: Enhancing generative language models with dual-extractive modules for schema and context tracking in multi-turn text-to-SQL. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10690–10708, Albuquerque, New Mexico. Association for Computational Linguistics.

Song Cheng, Qiannan Cheng, Linbo Jin, Lei Yi, and Guannan Zhang. 2025. Sqlord: A robust enterprise text-to-sql solution via reverse data generation and workflow decomposition. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 919–923. ACM.

Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. 2025. Less is more: Improving llm alignment via preference data selection. *Preprint*, arXiv:2502.14560.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: model alignment as prospect theoretic optimization. *CoRR*, abs/2402.01306.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 11170–11189. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Georgia Koutrika, Alkis Simitsis, and Yannis E. Ioannidis. 2010. Explaining structured queries in natural

language. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 333–344. IEEE Computer Society.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. Omnisql: Synthesizing high-quality text-to-sql data at scale. *Preprint*, arXiv:2503.02240.

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 Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023a. 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 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *Preprint*, arXiv:2406.11939.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in llms. *Preprint*, arXiv:2410.18451.

Hanbing Liu, Haoyang Li, Xiaokang Zhang, Ruotong Chen, Haiyong Xu, Tian Tian, Qi Qi, and Jing Zhang. 2025. Uncovering the impact of chain-of-thought reasoning for direct preference optimization: Lessons from text-to-sql. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 21223–21261. Association for Computational Linguistics.

Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2023. Towards out-of-distribution generalization: A survey. *Preprint*, arXiv:2108.13624.

Yifu Liu, Yin Zhu, Yingqi Gao, Zhiling Luo, Xiaoxia Li, Xiaorong Shi, Yuntao Hong, Jinyang Gao, Yu Li, Bolin Ding, and 1 others. 2026. Xiyan-sql: A novel multi-generator framework for text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Microsoft. 2025. Power bi q&a: Natural language queries.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 977–988. International World Wide Web Conferences Steering Committee / ACM.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, and 1 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, and 1 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi,

Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Salesforce. 2025. Tableau gpt: Generative ai for data analysis.

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

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cícero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13806–13814. AAAI Press.

Vladislav Shkapenyuk, Divesh Srivastava, Theodore Johnson, and Parisa Ghane. 2025. Automatic Metadata Extraction for Text-to-SQL. *CoRR*, abs/2505.19988. ArXiv: 2505.19988.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2760–2766. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2019. Sql-to-text generation with graph-to-sequence model. *Preprint*, arXiv:1809.05255.

Nuo Xu, Jun Zhao, Can Zu, Sixian Li, Lu Chen, Zhihao Zhang, Rui Zheng, Shihan Dou, Wenjuan Qin, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Advancing translation preference modeling with rlhf: A step towards cost-effective solution. *Preprint*, arXiv:2402.11525.

Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. Synthesizing text-to-SQL data from weak and strong LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7864–7875.

Donglei Yu, Yang Zhao, Jie Zhu, Yangyifan Xu, Yu Zhou, and Chengqing Zong. 2025. Simulpl: Aligning human preferences in simultaneous machine translation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, and 5 others. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1962–1979. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Bohan Zhai, Canwen Xu, Yuxiong He, and Zhewei Yao. 2025. Optimizing reasoning for text-to-sql with execution feedback. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 19206–19218. Association for Computational Linguistics.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. 2024. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *Preprint*, arXiv:2403.02951.

Xingxuan Zhang, Yue He, Renzhe Xu, Han Yu, Zheyan Shen, and Peng Cui. 2023. Nico++: Towards better benchmarking for domain generalization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16036–16047.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *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*.

A Appendix

A.1 Zero-Shot Evaluation

To assess the generalization ability of our method beyond in-context learning scenarios, we evaluate FPO under a pure zero-shot setting. Unlike the main experiments—where FPO operates together with AST-ICL demonstrations or task-specific exemplars are provided during inference. As shown in Table 4, FPO consistently improves performance over both the vanilla zero-shot baselines and the DPO-enhanced variants across all datasets. Specifically, when applied to Llama-3-8B-Instruct, FPO yields average absolute gains of 6.1% in WR over the vanilla zero-shot setting and also outperforms the DPO+Zero-shot counterpart. These results demonstrate that feedback-involved preference optimization can directly improve zero-shot generation quality without relying on contextual demonstrations.

A.2 Comparison with GPT-4o

We further investigate whether FPO can enhance lightweight LLMs to rival large proprietary models such as GPT-4o. According to prior studies (Abacha et al., 2025), the size of GPT-4o is estimated to be around 200 billion parameters. Table 5 compares GPT-4o with models trained using FPO on different backbones. Despite having significantly fewer parameters, Qwen+FPO achieves the highest BERTScore and BLEU while also obtaining the best Group Win Rate (WR). Llama+FPO also surpasses GPT-4o on BERTScore and WR. These results indicate that FPO can substantially improve smaller open-source models and narrow the performance gap with large proprietary models.

A.3 Performance Across SQL Difficulty Levels

We further conduct a fine-grained evaluation based on the difficulty levels of SQL queries. As shown in Table 6, FPO achieves 81.82% and 62.00% EX on the Easy and Medium levels, respectively, outperforming the baselines. On the Hard and Extra levels, FPO matches the performance of the dataset references and surpasses AST-ICL, demonstrating robustness when handling more complex queries.

A.4 FPO vs. DPO Comparison

To evaluate the effectiveness of FPO under limited data conditions, we compare FPO and DPO across multiple metrics and data efficiency using a radar chart. As shown in Figure 5, FPO achieves competitive results on BLEU, Group WR, BERTScore, and Pair WR while requiring substantially fewer training samples. With only $2/3$ of the data used by DPO for fine-tuning, FPO can generate better text for SQL snippets, highlighting its superior data efficiency. These findings demonstrate that FPO provides balanced improvements across multiple evaluation metrics while reducing data requirements and training cost, making it particularly suitable for low-resource scenarios.

A.5 Generalizability of FPO Across Different Preference Optimization Objectives

To evaluate the generality of our Feedback-involved Preference Optimization (FPO) method, we apply it on two popular existing preference alignment objectives: DPO and SimPO (Meng et al., 2024). As shown in Table 7, FPO consistently improves performance across all metrics including BERTScore, BLEU, and Group Win Rate, regardless of the underlying preference optimization baseline. These results suggest that FPO is a versatile optimization strategy that can enhance various preference alignment approaches in SQL-to-text generation tasks.

A.6 Effectiveness of FPO on Base Models

Our previous results demonstrated the effectiveness of the FPO method on instruction-tuned models. In this experiment, we extend the analysis to base models and observe consistent improvements. As shown in Table 8, applying FPO to Llama3-8B-base leads to significant gains across all evaluation metrics and benchmarks. These results indicate that FPO remains effective even in the absence of

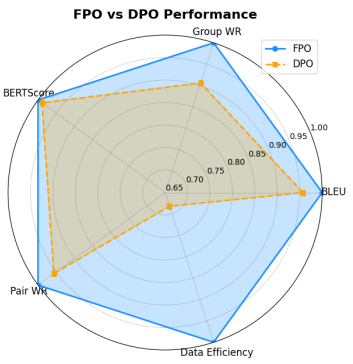

Figure 5: Performance comparison with normalized metrics on Spider-S2T. Data efficiency is defined as $\frac{1}{\text{Data Size}}$.

Model	Method	WR (%)				BERTScore				BLEU			
		CoSQL	SParC	Spider	BIRD	CoSQL	SParC	Spider	BIRD	CoSQL	SParC	Spider	BIRD
GPT-J-6B	Zero-shot	0.0	0.3	0.4	0.1	0.4891	0.5020	0.5027	0.5428	0.0510	0.0454	0.0501	0.0808
Mistral-7B-v0.1	Zero-shot	4.1	3.1	4.0	8.0	0.6279	0.6139	0.6314	0.6562	0.1660	0.1523	0.1539	0.1579
CodeLlama-7B	Zero-shot	1.0	1.4	1.8	2.1	0.6222	0.6164	0.6087	0.6765	0.1258	0.1227	0.1244	0.1423
Llama	Zero-shot	12.4	13.2	10.2	18.0	0.7693	0.7862	0.7888	0.7156	0.2229	0.2581	0.2548	0.1791
	DPO+Zero-shot	18.3	14.2	16.1	18.5	0.7685	0.7953	0.8070	0.7238	0.2282	0.2560	0.2813	0.1891
	FPO+Zero-shot	**20.0**	**17.4**	**20.8**	**20.1**	**0.8103**	**0.8144**	**0.8245**	**0.7330**	**0.2559**	**0.2765**	**0.2837**	**0.2171**

Table 4: Performance comparison of different models and methods under zero-shot settings on four benchmark datasets. WR denotes the Group Win Rate (Group WR).

Model	Size	BERTScore	BLEU	WR (%)
GPT-4o	~ 200 B	0.8682	0.4082	45.2
Llama+FPO	8B	0.8687	0.3786	46.7
Qwen+FPO	7B	**0.8694**	**0.4104**	**47.4**

Table 5: Performance comparison between GPT-4o and models trained with FPO on Spider-S2T. WR denotes the Group Win Rate.

Parser	Text Source	Easy	Medium	Hard&Extra
Qwen2.5-7B	Dataset Ref.	78.41	53.00	58.46
	Llama+AST-ICL	75.00	57.00	52.31
	Llama+FPO	**81.82**	**62.00**	**58.46**

Table 6: EX (%) across difficulty levels on Spider-S2T.

instruction tuning, suggesting its general applicability across model types.

A.7 Human Evaluation of Generated Text Quality

To evaluate the practical quality of generated texts, we conducted a human evaluation. We randomly sampled 270 examples, of which 241 were valid. The remaining 29 were excluded because the outputs from two or more methods were identical, preventing a meaningful comparison.

Annotators were presented with texts from three methods (FPO, DPO, Reference) without method labels to avoid bias. For each sample, they selected the best text based on three criteria: schema consistency, SQL semantic accuracy, and text readability/naturalness. If all three texts were indistinguishable, the sample was marked as a tie.

Three annotators (one PhD and two master's students in data science) independently evaluated all samples. Results were averaged across annotators. As shown in Table 9, FPO achieves the highest overall win rate 23.93% and the highest win rate excluding ties 40.61%, outperforming both DPO and the reference method.

Model	Method	BERTScore	BLEU	WR (%)
Llama	DPO	0.8592	0.3624	42.3
Llama	**FPO** (DPO)	**0.8687**	**0.3786**	**46.7**
Llama	SimPO	0.8682	0.3706	48.5
Llama	**FPO** (SimPO)	**0.8736**	**0.3874**	**50.7**

Table 7: Results on Spider-S2T showing that Feedback-involved Preference Optimization (FPO) improves upon different preference-based training objectives. All methods are based on the same base model (Llama). WR denotes Group Win Rate.

A.8 Efficiency Analysis

A.8.1 Back-Translation Reliability Analysis

Execution feedback in FPO relies on back-translating natural language text to SQL for execution comparison. A potential concern is whether back-translation inaccuracy could introduce noisy feedback signals. It is worth clarifying that the goal of execution feedback in FPO is not to achieve high text-to-SQL accuracy in itself, but to serve as a quality filtering signal that helps identify text descriptions semantically aligned with the original SQL. As long as back-translation can distinguish between semantically consistent and inconsistent samples, it provides useful guidance for preference data construction, without requiring perfect translation accuracy.

FPO's design further mitigates this risk through multi-signal cross-validation. Execution feedback, together with text quality and semantic feedback, forms three complementary quality signals that jointly establish a robust constraint for constructing high-quality preference data.

To directly measure the impact of back-translation reliability on sample filtering, we evaluate the execution accuracy of back-translated SQL on the Spider dataset, comparing unfiltered raw GPT-4o candidates against samples that pass our multi-signal filtering. As shown in Table 10, multi-signal filtering improves back-translation execution

Model	Method	CoSQL-S2T			SParC-S2T			Spider-S2T		
		BERTScore	BLEU	WR (%)	BERTScore	BLEU	WR (%)	BERTScore	BLEU	WR (%)
Llama3-8B-base	-	0.8075	0.2733	18.6	0.8389	0.3201	25.3	0.8382	0.3240	26.3
Llama3-8B-base	**FPO**	**0.8172**	**0.2790**	**22.4**	**0.8436**	**0.3368**	**29.5**	**0.8448**	**0.3348**	**32.1**

Table 8: Performance comparison of base models and methods on three benchmarks. In this table, "-" denotes vanilla AST-ICL methods without any post-training. WR denotes the Group Win Rate (Group WR).

Method	Overall Win Rate	Win Rate w/o Ties
Reference	21.02%	35.68%
Qwen+FPO	**23.93%**	**40.61%**
Qwen+DPO	13.97%	23.71%
Tie Rate	41.08%	–

Table 9: Human evaluation results on generated text quality.

Filtering Stage	EX
Raw GPT-4o candidates (unfiltered)	59.71%
After quality + semantic filtering	67.97%

Table 10: Back-translation EX before and after multi-signal filtering.

accuracy from 59.71% to 67.97%, demonstrating that our mechanism effectively filters out noisy samples and retains more reliable preference pairs.

Samples with conflicting signals across the three evaluators are directly filtered out during preference construction, further preventing biased optimization gradients that could arise from imprecise back-translation.

A.8.2 Computational Overhead of Execution Feedback

Compared with pure DPO training, FPO introduces an additional database execution feedback step during preference pair construction. This step includes database connection, query execution, and result comparison. We measure the additional computational cost on our experimental setup. The average additional latency per sample is approximately 1.89 ms. This cost mainly consists of the following components: database connection, SQL query execution, result comparison and Python processing. Because execution feedback is performed independently for each sample, the overall computational cost scales approximately linearly with the dataset size. In practice, this overhead is relatively small compared with the overall LLM inference time during training and data generation.

Stage	Tokens (Input/Output)	Cost (USD)
SQL Generation	∼440 / ∼100	\$0.55–\$0.73
Text Generation	∼1,200 / ∼350	\$2.00–\$2.16

Table 11: Average token consumption and cost per 1,000 samples. Cost ranges reflect dataset complexity: lower bound for Spider-series (simpler SQL), upper bound for BIRD (complex SQL with longer schema contexts).

A.8.3 Tokens Cost Analysis

We quantitatively analyze the token consumption and cost incurred by calling GPT-4o during the data construction phase. Based on GPT-4o official pricing (\$1.25/M input tokens, \$5.00/M output tokens), the average token consumption and cost per 1,000 samples are shown in Table 11.

Only ∼1,340 samples are needed for optimal performance, keeping the one-time construction cost low. Unlike methods that continuously rely on LLMs for inference, FPO requires no external LLM calls during inference (zero token cost), has a one-time low-cost data construction with small sample requirement, and achieves consistent improvements over baselines in WR, BERTScore, and BLEU.

A.9 Additional Implementation Details

Training Configuration. During training, the batch size is set to 1 with gradient accumulation over 8 steps. Models are trained for 10 epochs using the AdamW optimizer with an initial learning rate of $5e^{-6}$, where the first 10% of steps are used for warmup. We enable bf16 mixed-precision training.

Model and Framework. All experiments are implemented based on the open-source LLaMA-Factory framework. We perform experiments on both Llama and Qwen backbones. For parameter-efficient fine-tuning, LoRA with a rank of 8 is applied.

Baselines. For baseline comparisons, we adopt AST-ICL-TOP from the AST-ICL family. The number of in-context examples is set to 2 and the graph neural network component is implemented using a GCN.

Hardware. All experiments are conducted on a server equipped with eight NVIDIA A800 GPUs.

Evaluation. For semantic similarity evaluation, we compute BERTScore using the model *sentence-transformers/paraphrase-distilroberta-base-v1*. During inference, the temperature is fixed at 0.2 for fair comparison across all models.

A.10 Algorithm 1: FPO (Feedback-involved Preference Optimization)

Algorithm 1 FPO (Feedback-involved Preference Optimization)

Require: SQL queries Q, databases DB, generator G, reward model R, semantic scorer S, policy π_θ, reference policy π_{ref}, candidate number n, threshold k
Ensure: Optimized policy π_θ

// Stage 1: Preference Pair Construction
$U \leftarrow \emptyset$
for each $(X, DB) \in Q$ **do**
  $Y_{\text{ref}} \leftarrow \text{GetRef}(X)$
  $\{Y_1, \ldots, Y_n\} \leftarrow G(X, DB)$
  $C \leftarrow \{Y_1, \ldots, Y_n, Y_{\text{ref}}\}$
  for each $Y \in C$ **do**
    $\text{score}(Y) \leftarrow R(Y, X)$
  end for
  $Y_{\text{high}} \leftarrow \arg\max_Y \text{score}(Y)$
  $Y_{\text{low}} \leftarrow \arg\min_Y \text{score}(Y)$
  $U \leftarrow U \cup \{(X, Y_{\text{high}}, Y_{\text{low}})\}$
  if $\text{score}(Y_{\text{ref}}) > \text{score}(Y_{\text{low}})$ **then**
    $U \leftarrow U \cup \{(X, Y_{\text{ref}}, Y_{\text{low}})\}$
  end if
end for

// Stage 2: Multi-Signal Feedback Collection
for each $(X, Y^+, Y^-) \in U$ **do**
  $T_k \leftarrow \text{Top}_k\big(C,\ \{S(Y, X, DB) \mid Y \in C\},\ k\big)$
  $ok_{\text{sem}} \leftarrow (Y^+ \in T_k)$
  $ok_{\text{exe}} \leftarrow \big(\text{Exec}(Y^+, DB) = \text{Exec}(X)\big)$
  $z \leftarrow \begin{cases} 1, & \text{if } ok_{\text{sem}} \wedge ok_{\text{exe}} \\ 0, & \text{otherwise} \end{cases}$
end for

// Stage 3: Preference Optimization
for each $(X, Y^+, Y^-) \in U$ **do**
  $L \leftarrow -z \log \sigma\Bigg(\beta\Big[\log \dfrac{\pi_\theta(Y^+|X)}{\pi_{\text{ref}}(Y^+|X)} - \log \dfrac{\pi_\theta(Y^-|X)}{\pi_{\text{ref}}(Y^-|X)}\Big]\Bigg)$
  $\theta \leftarrow \theta - \nabla_\theta L$
end for
return π_θ

A.11 Illustrative Example of Preference Pair Construction Flow

Below, we present one representative case (spider-57) to illustrate the complete construction flow. This example demonstrates how RM prefers fluent but semantically imprecise texts and how multi-signal feedback corrects this bias.

DB: pets_1
SQL:

```
SELECT DISTINCT T1.Fname FROM student AS T1
JOIN has_pet AS T2 ON T1.stuid = T2.stuid
JOIN pets AS T3 ON T3.petid = T2.petid
WHERE T3.pettype = 'cat' OR T3.pettype = 'dog'
```

Schema: Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city_code), Has_Pet(StuID, PetID), Pets(PetID, PetType, pet_age, weight)

Key: SQL queries only `Fname` (first name), not `LName` (last name).

Stage 1: Preference Pair Construction

Step 1a: Candidate Generation

Five candidates Y_1–Y_5 are generated by GPT-4o, and Y_{ref} is the ground-truth text from dataset:

- Y_1: "Show me the names of students who own either a cat or a dog."
- Y_2: "Show me the names of students who are pet owners of either a cat or a dog."
- Y_3: "List the names of students who have either a cat or a dog as a pet."
- Y_4: "Show me the names of students who are pet owners of either a cat or a dog." (duplicate of Y_2)
- Y_5: "What are the unique first names of students who own either cats or dogs?"
- Y_{ref}: "Find the first name of students who have cat or dog pet."

Y_1–Y_3 use "names" (implying first + last name). Y_{ref} and Y_5 use "first name" which exactly matches `Fname`.

Step 1b: RM Scoring & Triplet Formation

RM scores: Y_3 = 24.125, Y_4 = 24.125 (tied rank 1), Y_1 = 21.875 (rank 3), Y_{ref} = 17.250 (rank 4), Y_5 = 17.000 (rank 5), Y_2 = 16.875 (rank 6). Thus $Y_{\text{high}} = Y_3$ and $Y_{\text{low}} = Y_2$.

Two triplets are formed:

- T_1: ($Y^+ = Y_{\text{ref}}$, $Y^- = Y_2$), because $\text{score}(Y_{\text{ref}}) > \text{score}(Y_{\text{low}})$.
- T_2: ($Y^+ = Y_3$, $Y^- = Y_2$), always formed from Y_{high} and Y_{low}.

Stage 2: Multi-Signal Feedback Validation

Step 2a: Semantic Feedback

Semantic scores (0–20) with $k = 3$: Y_1 = 13, Y_2 = 12, Y_3 = 13, Y_4 = 12, Y_5 = 18, Y_{ref} = 16. Average is 14. Top-3 are Y_5 (18), Y_{ref} (16), Y_1 (13). Therefore:

- T_1 ($Y^+ = Y_{\text{ref}}$): $Y^+ \in$ Top-3 $\rightarrow \text{ok}_{\text{sem}} = 1$.
- T_2 ($Y^+ = Y_3$): $Y^+ \notin$ Top-3 $\rightarrow \text{ok}_{\text{sem}} = 0$.

Step 2b: Execution Feedback

Back-translating each Y^+ to SQL and executing:

- Y_{ref}: "Find the first name..." $\rightarrow$ SELECT DISTINCT S.Fname FROM Student S JOIN ... $\rightarrow$ Execution matches the original SQL $\rightarrow \text{ok}_{\text{exe}} = 1$.
- Y_3: "List the names..." $\rightarrow$ SELECT DISTINCT S.LName, S.Fname FROM Student S JOIN ... $\rightarrow$ Execution returns both first and last names, exceeding original SQL scope $\rightarrow \text{ok}_{\text{exe}} = 0$.

Y_3 uses "names", which back-translation resolves to querying both `Fname` and `LName`. This is a typical case of fluency-driven semantic expansion.

Step 2c: z Computation

For T_1: $ok_{sem} = 1$, $ok_{exe} = 1 \rightarrow z = 1$ (Keep). For T_2: $ok_{sem} = 0$, $ok_{exe} = 0 \rightarrow z = 0$ (Filter).

Stage 3: Preference Optimization

Only triplets with $z = 1$ participate in FPO training. Thus, only T_1 is retained, where Y^+ = "Find the first name of students who have cat or dog pet." T_2 is filtered out due to semantic imprecision ("names" instead of "first name") causing execution mismatch.

A.12 Prompt Templates

Here are the prompt templates we used in Data Construction(Figure 6). During data construction, the temperature is set to 0.7 for CoD-based candidate generation:

A.13 AI Assistance Statement

AI writing assistants were used during the preparation of this work. Their usage was limited to grammar checking, wording optimization, and minor text polishing.

Chain-of-Density (CoD) Prompt Specification

You are given an original SQL query from a SQL-to-text generation task. Your goal is to iteratively generate and refine natural language questions that could lead to this SQL, using the Chain-of-Density (CoD) method.
Specifically, you will perform a 5-step iterative process where you:

Step 1: Initial Utterance Generate an initial natural language utterance that vaguely reflects the overall intent of the SQL query. This utterance should:

- Be approximately 80 characters in length ;
- Contain vague and generic language;
- Avoid specific schema details such as table names, column names, filter values, or SQL operations;
- Include filler phrases like "This query is about..." or "Find some information related to..." to pad length.

Steps 2 to 5: Schema Density Refinement For each of the next four steps (Step 2 to Step 5), perform the following:

- **Step A: Identify Missing Entities.** Identify 1–3 schema entities (e.g., table names, columns, values, operators) that are present in the SQL query but **have not yet been mentioned**. Each entity must:
  - Appear in the SQL query;
  - Be specific (e.g., "winner_name", "ORDER BY minutes");
  - Be no longer than 5 words.
- **Step B: Rewrite the Utterance.** Rewrite the utterance to:
  - Maintain approximately the same length;
  - Include **all previously added** schema entities plus the new ones;
  - Preserve fluency, naturalness, and logical alignment with the SQL;
  - Never remove or paraphrase any existing entity from earlier versions.

Special Requirement for Step 5 The final utterance (Step 5) must be rewritten as a clear, fluent natural language **question** that:

- Directly maps to the intent of the original SQL query;
- Contains all schema entities introduced in Steps 2 to 4;
- Is well-formed and unambiguous.

Output Format Return your response in the following `JSON` format:

```
[
  {
    "Missing_Entities": ["..."],
    "Denser_Summary": "..."
  },
  ...
  {
    "Missing_Entities": ["..."],
    "Denser_Summary": "Natural language question here."
  }
]
```

Constraints

- Do not invent schema entities. Use only those present in the SQL.
- Never remove any schema entity once introduced.
- Do not repeat filler phrases across steps.
- Ensure utterance length and meaning consistency across steps.
- The final output must be a natural language question.

Figure 6: CoD Prompt templates