

MSI-SQL: Unleashing Data Potential of Large Language Models for Text-to-SQL via Multi-Task Self-Iterative Generation

Li Gao^{1,†}, Zixuan Wang^{2,†}, Boci Peng³, Mingjun Nie⁴, Huan Luo⁴, Chun Gan^{4,*}

¹Tsinghua University ²Harbin Institute of Technology

³Peking University ⁴JD.com, Beijing, China

l-gao21@tsinghua.org.cn 24S103396@stu.hit.edu.cn ganchun1@jd.com

Abstract

Text-to-SQL parsing increasingly relies on high-quality Chain-of-Thought (CoT) data to train specialized models. However, eliciting accurate reasoning paths for complex queries remains a formidable challenge, causing existing methods to suffer from severe data underutilization and fragile free-form generation. To systematically resolve these issues, we propose **MSI-SQL**, a **Multi-task Self-Iterative** framework that enhances both data quantity and quality. To expand data scale, we introduce a verification-based self-iterative loop that progressively conquers difficult queries. To ensure reasoning stability, we design a masked SQL completion auxiliary task that serves as a structural anchor, explicitly constraining the CoT generation space. Coupled with a novel mask-augmented soft self-consistency mechanism during inference, evaluations on the BIRD and Spider benchmarks demonstrate that MSI-SQL establishes a new state-of-the-art for open-source Text-to-SQL parsers.

1 Introduction

Text-to-SQL (Cai et al., 2017; Zelle and Mooney, 1996; Yu et al., 2018a) has witnessed remarkable progress with the advent of Large Language Models (LLMs) (Gao et al., 2023; Mao et al., 2024; Yao et al., 2025; Papicchio et al., 2026). To circumvent the high inference costs and strict privacy risks associated with massive proprietary models, a prevailing paradigm distills their reasoning abilities into smaller, specialized open-source models using task-specific data (Yang et al., 2024; Zhu et al., 2025; Hao et al., 2025; Song et al., 2024). While this distillation fundamentally relies on high-quality CoT data (Wei et al., 2022; Zhai et al., 2025; Thaker and Bresler, 2025), eliciting accurate reasoning paths for complex queries with intricate schemas remains a formidable challenge (Li et al., 2025b; Hong et al., 2025; Liu et al., 2025).

Current data generation paradigms (Hu et al., 2023; Li et al., 2025a; Cai et al., 2025; Caferoglu et al., 2025), which predominantly rely on single-pass prompting, suffer from two critical bottlenecks. First, this approach frequently fails on complex queries, leading to severe data underutilization (Liu et al., 2026; Gulcehre et al., 2023). As a result, a large portion of valuable hard queries and intricate schemas are discarded, severely bottlenecking the overall *quantity* of effective training data. Second, the free-form CoT reasoning elicited by these methods is inherently fragile in structurally rigorous Text-to-SQL tasks (Tai et al., 2023; Zhai et al., 2025). Although recent efforts attempt to prompt models to fixed logical steps (Rajkumar et al., 2022; Yao et al., 2025; Pourreza and Rafiei, 2023; Qin et al., 2024; He et al., 2025), they still lack explicit structural constraints. Even a minor hallucination during the schema linking phase can completely derail the subsequent SQL generation. Ultimately, without a robust structural anchor, the model’s reasoning trajectory easily deviates, compromising the *quality* and logical consistency of the elicited CoT paths.

To systematically unleash the potential of Text-to-SQL training data and maximize model performance, we propose MSI-SQL, a novel framework centered on multi-task guided self-iterative generation. To overcome the data quantity bottleneck, our **Verification-Based Self-Iterative Generation** mechanism continuously samples and execution-verifies reasoning paths, progressively conquering hard queries to monotonically expand the valid training corpus. To guarantee data quality, our **Multi-Task Guided Generation** introduces an auxiliary masked SQL completion task. Serving as a robust structural anchor, it explicitly constrains the CoT search space to prevent reasoning deviation. During inference, we employ **Mask-Augmented Soft Self-Consistency**, an error-tolerant mechanism that expands the candi-

[†]Equal contribution. ^{*}Corresponding author.

date pool via test-time masked completion and resolves syntactic mismatches through soft consensus. Extensive experiments demonstrate that MSI-SQL significantly outperforms all existing baselines on the BIRD and Spider benchmarks.

Our contributions are summarized as follows:

- To tackle data underutilization and reasoning fragility in Text-to-SQL parsing, we propose the MSI-SQL framework to unleash data potential across both training and inference.
- We introduce a Verification-Based Self-Iterative Generation mechanism that progressively conquers complex queries to expand the coverage of high-quality training data.
- We design a Multi-Task Guided Generation mechanism where an auxiliary masked SQL completion task acts as a structural anchor, explicitly constraining the generation space to guarantee CoT reasoning stability.
- We propose Mask-Augmented Soft Self-Consistency, an execution-aware inference mechanism that rescues semantically equivalent queries, propelling MSI-SQL to state-of-the-art performance on BIRD and Spider.

2 Methodology

We propose the MSI-SQL framework to systematically unleash the potential of Text-to-SQL training data. In this section, we first formalize the task, then elaborate on its two core mechanisms: verification-based self-iterative data generation and multi-task guided generation. Figure 1 shows the overview of our MSI-SQL.

2.1 Problem Formulation

We consider an initial Text-to-SQL dataset $\mathcal{D} = \{(q_i, s_i, d_i)\}_{i=1}^N$, consisting of natural language questions q_i , gold SQL queries s_i , and database schemas d_i . To effectively train an LLM for complex parsing, our primary objective is to elicit high-quality CoT reasoning paths c_i that connect the inputs (q_i, d_i) to the target s_i . Thus, we aim to construct an instruction-tuning dataset $\mathcal{D}_{CoT} = \{(q_i, d_i, c_i, s_i)\}$.

2.2 Verification-Based Self-Iterative Data Generation

Existing single-pass sampling methods frequently fail on complex queries, resulting in severe data

underutilization. To systematically extract valid CoT paths for these hard queries, we propose a verification-based self-iterative generation mechanism. Initializing with a base LLM $\mathcal{M}_0$, each iteration $t \geq 1$ proceeds as follows:

Data Elicitation. We use the model from the previous iteration, $\mathcal{M}_{t-1}$ (where $\mathcal{M}_0$ is used for $t = 1$), to sample multiple candidate reasoning paths and corresponding SQL queries for each instance in the training set.

Execution-based Filtering. To filter out invalid reasoning paths, all generated SQL candidates are executed against the corresponding databases. We exclusively retain the samples that yield execution results identical to the gold SQL s_i . These verified instances constitute the valid dataset $\mathcal{D}_{valid}^{(t)}$. As iterative refinement enhances model capability, the valid data pool monotonically expands: $|\mathcal{D}_{valid}^{(1)}| < |\mathcal{D}_{valid}^{(2)}| < \dots < |\mathcal{D}_{valid}^{(t)}|$. By continuously unlocking previously unsolvable queries, this growing corpus provides the essential foundation for the subsequent model update.

Base-Model Reset Fine-Tuning. Empirical evidence suggests that standard iterative self-training suffers from error accumulation and mode collapse when continuously fine-tuning on $\mathcal{M}_{t-1}$ (Details are provided in Appendix B.4). To circumvent this, our Base-Model Reset strategy explicitly abandons the intermediate checkpoint $\mathcal{M}_{t-1}$. Instead, we perform supervised fine-tuning (SFT) directly on the initial base model $\mathcal{M}_0$ using the newly expanded dataset $\mathcal{D}_{valid}^{(t)}$, producing the updated model $\mathcal{M}_t$. This prevents the inheritance of degenerative reasoning patterns, ensuring the model safely capitalizes on the increasing data volume to conquer previously unsolvable queries.

2.3 Auxiliary Masked SQL Completion Task

Although the self-iterative process increases the quantity of data, free-form CoT generation is highly susceptible to error accumulation. To ensure the quality and direction of the elicited reasoning paths, we introduce a masked SQL completion task as an auxiliary instruction tuning objective.

During the data construction phase, we construct a masked version of the gold SQL, denoted as s_i^{mask} , by replacing specific structural tokens (e.g., table names, JOIN conditions) with a [MASK] token. To augment the standard CoT

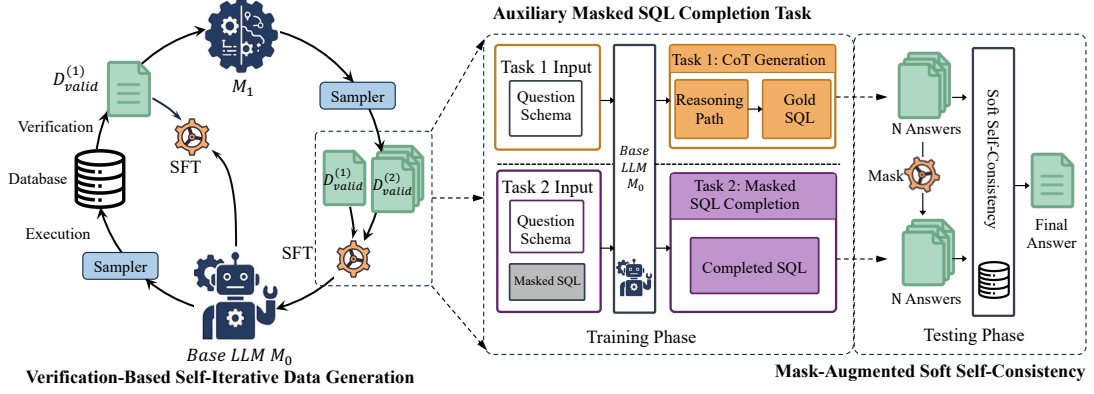


Figure 1: The overview of MSI-SQL framework, comprising three core components: (a) *Verification-Based Self-Iterative Data Generation*, a loop that progressively expands the high-quality training set and fine-tunes the base LLM to prevent mode collapse; (b) *Auxiliary Masked SQL Completion Task*, which stabilizes the CoT reasoning process during generation; and (c) *Mask-Augmented Soft Self-Consistency*, an inference-time selection mechanism that leverages execution-aware consensus to robustly derive the final SQL prediction.

generation task, which optimizes the baseline probability $p(c_i, s_i \mid q_i, d_i)$, our framework explicitly emphasizes the masked SQL completion task by optimizing $p(s_i \mid q_i, d_i, s_i^{mask})$. Both objectives are seamlessly mixed into a single training corpus and jointly trained via SFT. Furthermore, we incorporate this masked completion task into the Direct Preference Optimization (DPO) framework (Rafailov et al., 2023) to maintain structural rigor during contrastive learning. By acting as an implicit structural anchor, the partially visible gold SQL constrains the generation space. This prevents the reasoning trajectory from deviating, thereby significantly boosting the yield rate and robustness of valid CoT paths during self-iterative sampling. Note that this task is training-only; at inference, the model receives only the question and schema, with no access to any portion of the gold SQL.

2.4 Mask-Augmented Soft Self-Consistency

To maximize inference performance, we introduce Mask-Augmented Soft Self-Consistency, integrating our learned masked completion capability with an execution-aware selection mechanism. First, we generate N initial SQL candidates via temperature sampling. Applying test-time Masked SQL Completion to these candidates yields N additional repaired queries, forming a diverse pool of size $2N$.

Standard self-consistency (Wang et al., 2022) employs exact-match hard voting, failing to account for semantically equivalent queries with minor execution order variances (Liu et al., 2026). To address this, we execute all $2N$ queries and compute continuous pairwise similarity scores among their execution results. The final consensus score

for each candidate is the sum of its similarities with all other candidates. By selecting the query with the maximum consensus score, this error-tolerant mechanism effectively mitigates syntactic fragility and significantly improves execution accuracy.

3 Experiments

3.1 Experimental Settings

Benchmarks and Metrics. We evaluate MSI-SQL on **Spider** (Yu et al., 2018b) and **BIRD** (Li et al., 2023). We report Execution Accuracy (EX) and Test Suite Accuracy (TS) (Zhong et al., 2020) for Spider, and EX and Valid Efficiency Score (VES) (Li et al., 2023) for BIRD to assess both correctness and efficiency. Appendix B.1 provides more details about the benchmarks.

Baselines. We compare MSI-SQL against: (1) Prompting-based closed-source models: DIN-SQL (Pourreza and Rafiei, 2023), MAC-SQL (Wang et al., 2025), DAIL-SQL (Gao et al., 2023), and MCS-SQL (Lee et al., 2025); (2) Fine-tuned open-source models, including synthetic data generation (CodeS (Li et al., 2024), SENSE (Yang et al., 2024), OmniSQL (Li et al., 2025a)), multitask collaboration ROUTE (Qin et al., 2024), and verification-based self-play fine-tuning SPFT-SQL (Zhang et al., 2025).

Implementations. The details of implementation are listed in Appendix B.2.

3.2 Overall Performance

Table 1 summarizes the results on both datasets. MSI-SQL achieves state-of-the-art performance

Method	Base Model	SPIDER			BIRD	
		Dev		Test	Dev	
		EX	TS	EX	EX	VES
Prompting with Closed-Source LLMs						
DIN-SQL	GPT-4	82.8	74.2	85.3	50.7	58.8
DAIL-SQL	GPT-4	83.5	76.2	86.6	54.8	56.1
MCS-SQL	GPT-4	89.5	-	89.6	63.4	64.8
Fine-Tuning Open-Source LLMs						
SFT	Qwen2.5-Coder-7B	82.9	79.0	83.7	54.4	56.1
SFT	Qwen2.5-Coder-14B	84.8	79.6	84.4	58.5	63.9
SFT	Qwen2.5-Coder-32B	85.2	79.5	86.4	61.2	66.6
CodeS-7B	StarCoder-7B	85.5	-	84.4	55.8	60.3
CodeS-15B	StarCoder-15B	84.9	79.4	85.0	58.5	56.7
SENSE-7B	Llama-2-7B	83.2	81.7	83.5	51.8	-
SENSE-13B	Llama-2-13B	84.1	83.5	86.6	55.5	-
OmniSQL-7B	Qwen2.5-Coder-7B-Instruct	81.2	-	87.9	63.9	-
OmniSQL-14B	Qwen2.5-Coder-14B-Instruct	81.4	-	88.3	64.2	-
OmniSQL-32B	Qwen2.5-Coder-32B-Instruct	80.0	-	87.6	64.5	-
ROUTE	Qwen2.5-7B	83.6	77.5	83.7	54.4	56.1
ROUTE	Qwen2.5-14B	87.3	80.9	87.1	60.9	65.2
SPFT-SQL	Qwen2.5-Coder-7B	87.2	81.3	87.4	61.0	67.0
SPFT-SQL	Qwen2.5-Coder-14B	87.7	81.9	89.0	63.6	68.9
SPFT-SQL	Qwen2.5-Coder-32B	87.8	83.2	89.1	65.2	70.5
Ours						
MSI-SQL	Qwen2.5-Coder-7B	86.6	72.6	87.4	66.2	112.3
MSI-SQL	Qwen2.5-Coder-14B	87.9	78.3	89.4	68.3	117.9
MSI-SQL	Qwen2.5-Coder-32B	87.3	80.4	88.8	71.1	117.2

Table 1: Main results on SPIDER and BIRD benchmarks. EX and TS denote Execution Accuracy and Test Suite Accuracy, while VES represents Valid Efficiency Score. Our methods utilize iterative data from both Spider and BIRD for SFT/DPO.

	SPIDER			BIRD-Dev	
	Dev-EX	Dev-TS	Test-EX	EX	VES
Our	87.3	80.4	88.8	71.1	117.2
w/o SI	78.9	75.4	84.5	68.4	95.9
w/o MSC	85.6	78.5	86.0	70.3	114.0
w/o MSSC	85.0	78.3	85.8	69.9	96.8

Table 2: Ablation study.

across most metrics, demonstrating that self-iterative generation and multi-task guided training, coupled with mask-augmented soft voting, effectively unleash data and model potential, establishing an effective and robust data utilization framework for Text-to-SQL parsing. Furthermore, MSI-SQL significantly outperforms existing methods in terms of execution efficiency. This suggests that our multi-task self-iterative generation approach enables the model to optimize query logic, producing SQL queries that exhibit superior execution speed compared to other methods, even surpassing the efficiency of the gold queries.

3.3 Ablation Study

We assess the contribution of each module using the 32B model via three variants (Table 2): **w/o Self-Iteration** removes the Verification-Based Self-Iteration Generation, **w/o MSC** discards the Masked SQL Completion task during training, and

w/o MSSC replaces our Mask-Augmented Soft Self-Consistency with standard majority voting. MSI-SQL consistently outperforms all variants, verifying that these components are fundamentally necessary and complementary. Notably, removing Self-Iteration causes the most severe performance degradation, underscoring its necessity in expanding the valid training scale by progressively unlocking complex queries.

3.4 Sensitivity Analysis

We study the impact of self-iterative data generation rounds and the mask token ratio. *Iteration Rounds*: Fewer rounds underutilize the data, whereas excessive iterations lead to overfitting (Table 3 and Table 4). *Mask Ratio*: An excessively high masking ratio fails to provide effective guidance for data generation, while a low ratio restricts CoT diversity (Appendix B.5).

Round	Num _{Valid}	Train EX	Dev EX
Round 0	7,587	80.47	64.60
Round 1	7,693	81.60	65.91
Round 2	7,712	81.80	66.36
Round 3	7,725	81.94	65.51

Table 3: Performance evolution over self-iteration rounds on the BIRD benchmark (SFT only). Train EX denotes the proportion of training instances that contain correct CoT rationales.

Round	Num _{Valid}	Train EX	Dev EX	Test EX
Round 0	6,636	76.64	82.81	85.28
Round 1	6,711	77.50	84.63	86.48
Round 2	6,726	77.68	84.94	86.43
Round 3	6,741	77.85	84.64	86.76

Table 4: Performance evolution over self-iteration rounds on the Spider benchmark (SFT only).

4 Conclusion

We propose MSI-SQL, a multi-task guided self-iterative framework that systematically unleashes the potential of Text-to-SQL parsing. By integrating a verification-based self-iterative loop with a Masked SQL auxiliary task, it expands high-quality training data while preventing reasoning deviation. Coupled with a novel mask-augmented soft self-consistency mechanism during inference, MSI-SQL achieves new state-of-the-art results on BIRD and Spider, establishing a robust post-training paradigm for open-source models.

Limitations

While effective, MSI-SQL has certain limitations. First, the self-iterative loop and execution verification introduce higher computational overhead during the offline data synthesis phase compared to single-pass prompting. Second, the verification mechanism relies on access to executable database environments, which might be restricted in highly sensitive scenarios. Finally, the bootstrapping process requires the base model to possess a minimum threshold of reasoning capability to successfully kickstart the self-iterative data generation loop.

Ethical Considerations

We strictly adhere to the ACL Ethics Policy. All experiments are conducted using publicly accessible datasets, publicly available API services, and open-source models. No private or sensitive user information is involved in the research process. We believe that this work is unlikely to cause any immediate harmful societal effects.

References

- Hasan Alp Caferoğlu, Mehmet Serhat Çelik, and Özgür Ulusoy. 2025. Sing-sql: A synthetic data generation framework for in-domain text-to-sql translation. *arXiv preprint arXiv:2509.25672*.
- Qifeng Cai, Hao Liang, Chang Xu, Tao Xie, Wentao Zhang, and Bin Cui. 2025. Text2sql-flow: A robust sql-aware data augmentation framework for text-to-sql. *arXiv preprint arXiv:2511.10192*.
- Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. 2017. An encoder-decoder framework translating natural language to database queries. *arXiv preprint arXiv:1711.06061*.
- Yiwen Ding, Zhiheng Xi, Wei He, Zhuoyuan Li, Yitao Zhai, Xiaowei Shi, Xunliang Cai, Tao Gui, Qi Zhang, and Xuan-Jing Huang. 2025. Mitigating tail narrowing in llm self-improvement via socratic-guided sampling. 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 10627–10646.
- Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.
- Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.
- Zhifeng Hao, Junqi Huang, Shaobin Shi, Ruichu Cai, and Boyan Xu. 2025. Genlink: Generation-driven schema-linking via multi-model learning for text-to-sql. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29880–29893.
- Mingqian He, Yongliang Shen, Wenqi Zhang, Qiuying Peng, Jun Wang, and Weiming Lu. 2025. Star-sql: Self-taught reasoner for text-to-sql. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24365–24375.
- Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2025. Next-generation database interfaces: A survey of llm-based text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*.
- Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.
- Yiqun Hu, Yiyun Zhao, Jiarong Jiang, Wuwei Lan, Henghui Zhu, Anuj Chauhan, Alexander Hanbo Li, Lin Pan, Jun Wang, Chung-Wei Hang, and 1 others. 2023. Importance of synthesizing high-quality data for text-to-sql parsing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1327–1343.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.
- Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2025. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 337–353.
- Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, and 1 others. 2025a. Omnisql: Synthesizing high-quality text-to-sql data at scale. *arXiv preprint arXiv:2503.02240*.
- Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data*, 2(3):1–28.
- Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Can llm already serve

- as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357.
- Jinyang Li, Xiaolong Li, Ge Qu, Per Jacobsson, Bowen Qin, Binyuan Hui, Shuzheng Si, Nan Huo, Xiaohan Xu, Yue Zhang, and 1 others. 2025b. Swe-sql: Illuminating llm pathways to solve user sql issues in real-world applications. *arXiv preprint arXiv:2506.18951*.
- Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A survey of text-to-sql in the era of llms: Where are we, and where are we going? *IEEE Transactions on Knowledge and Data Engineering*.
- Yifu Liu, Yin Zhu, Yingqi Gao, Zhiling Luo, Xiaoxia Li, Xiaorong Shi, Yuntao Hong, Jinyang Gao, Yu Li, Bolin Ding, and 1 others. 2026. Xiyang-sql: A novel multi-generator framework for text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*.
- Wenxin Mao, Ruiqi Wang, Jiyu Guo, Jichuan Zeng, Cuiyun Gao, Peiyi Han, and Chuanyi Liu. 2024. Enhancing text-to-sql parsing through question rewriting and execution-guided refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2009–2024.
- Simone Papicchio, Simone Rossi, Luca Cagliero, and Paolo Papotti. 2026. Think2sql: Blueprinting reward density and advantage scaling for effective text-to-sql reasoning. *Transactions on Machine Learning Research*.
- Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in neural information processing systems*, 36:36339–36348.
- Yang Qin, Chao Chen, Zhihang Fu, Ze Chen, Dezhong Peng, Peng Hu, and Jieping Ye. 2024. Route: Robust multitask tuning and collaboration for text-to-sql. *arXiv preprint arXiv:2412.10138*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.
- Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.
- Yewei Song, Saad Ezzini, Xunzhu Tang, Cedric Lothritz, Jacques Klein, Tegawendé Bissyandé, Andrey Boytsov, Ulrick Ble, and Anne Goujon. 2024. Enhancing text-to-sql translation for financial system design. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, pages 252–262.
- Chang-Yu Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain of thought style prompting for text-to-sql. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5376–5393.
- Khushboo Thaker and Yony Bresler. 2025. Knowledge distillation with structured chain-of-thought for text-to-sql. *arXiv preprint arXiv:2512.17053*.
- Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and 1 others. 2025. Mac-sql: A multi-agent collaborative framework for text-to-sql. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- 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.
- Zhewei Yao, Guoheng Sun, Lukasz Borchmann, Gaurav Nuti, Zheyu Shen, Minghang Deng, Bohan Zhai, Hao Zhang, Ang Li, and Yuxiong He. 2025. Arctic-text2sql-r1: Simple rewards, strong reasoning in text-to-sql. *arXiv preprint arXiv:2505.20315*.
- Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921.
- John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

- Bohan Zhai, Canwen Xu, Yuxiong He, and Zhewei Yao. 2025. Excot: Optimizing reasoning for text-to-sql with execution feedback. *arXiv preprint arXiv:2503.19988*.
- Yuhao Zhang, Shaoming Duan, Jinhang Su, Chuanyi Liu, and Peiyi Han. 2025. Spft-sql: Enhancing large language model for text-to-sql parsing by self-play fine-tuning. *arXiv preprint arXiv:2509.03937*.
- Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411.
- Gaoyu Zhu, Wei Shao, Xichou Zhu, Lei Yu, Jiafeng Guo, and Xueqi Cheng. 2025. Text2sql: Pure fine-tuning and pure knowledge distillation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 54–61.

A Related Works

LLM-based Text-to-SQL Parsing. Recently, leveraging Large Language Models (LLMs) has become the dominant paradigm for Text-to-SQL parsing (Gao et al., 2023; Mao et al., 2024; Yao et al., 2025; Papicchio et al., 2026). To facilitate deployment in privacy-sensitive scenarios, extensive research has focused on instruction tuning to distill the reasoning capabilities of proprietary models into smaller, open-source counterparts (Yang et al., 2024; Zhu et al., 2025; Hao et al., 2025; Song et al., 2024). For instance, DAIL-SQL (Gao et al., 2023) and MAC-SQL (Wang et al., 2025) explore efficient prompting strategies and multi-agent frameworks. However, when constructing fine-tuning datasets, these methods typically rely on single-pass prompting. This approach frequently fails on hard queries with complex database schemas, leading to severe data underutilization. In contrast, MSI-SQL actively conquers these discarded hard queries through a self-iterative mechanism, maximizing the scale of valid training data.

Data Generation. To enhance the complex reasoning abilities of LLMs, self-iterative generation has garnered significant attention. Methods like STaR-SQL (He et al., 2025) and V-STaR (Hosseini et al., 2024) bootstrap reasoning by continuously fine-tuning the model on its own successfully generated rationales. However, applying these methods directly to structurally rigorous tasks like Text-to-SQL reveals two critical flaws: (1) continuously updating intermediate checkpoints makes the model highly susceptible to error accumulation and mode collapse (Appendix B.4); (2) free-form CoT generation lacks explicit constraints, easily leading to reasoning deviation (Tai et al., 2023; Zhai et al., 2025). MSI-SQL elegantly circumvents these issues by introducing a Base-Model Reset strategy and a Masked SQL multi-task structural anchor, ensuring both the quantity and quality of the elicited data.

B More Experiment Details

B.1 Benchmarks and Evaluation Metrics

To comprehensively evaluate our method, we conduct extensive experiments on two widely used text-to-SQL benchmarks, **SPIDER** (Yu et al., 2018b) and **BIRD** (Li et al., 2023). The data statistics of two benchmarks are demonstrated in Table 5.

Statistics	SPIDER	BIRD
Train Set	8,659	9,428
Dev Set	1,034	1,534
Test Set	2,147	—
Domains	138	37
Databases	206	95
Metrics	EX, TS	EX, VES

Table 5: Summary of the benchmarks and evaluation metrics used in our experiments.

B.2 Implementation Details

We employ Qwen2.5-Coder (7B/14B/32B) (Hui et al., 2024) as base models. For SFT, we train for 2 epochs using AdamW with a cosine scheduler (peak learning rate $1e-5$). For DPO, we use a constant learning rate of $1e-6$ for 2 epochs. The self-iterative data-generation process—including CoT candidate generation on thousands of training samples and SFT/DPO training in each iteration—was conducted on $8 \times$ Ascend 910B NPUs. All evaluation-time inference, including model deployment and Mask-Augmented Soft Self-Consistency (MSSC) on the development and test sets, was conducted on NVIDIA B200 GPUs. This includes the experiments reported in Table 1, the ablation studies, and the cross-dataset generalization experiments. During inference, we set the sampling temperature to $T = 1.0$.

B.3 Computational Overhead Analysis

We report the computational overhead of MSI-SQL for the 32B model. For training, SFT on 12,000 samples takes approximately 1 hour on $8 \times$ Ascend 910B NPUs; DPO training time varies depending on the data size of each iteration. For inference on NVIDIA B200 GPUs, batch processing handles approximately 7k samples in 7–10 minutes. For execution verification, we set a timeout of 30 seconds per query, with the vast majority of SQL queries completing within 20 seconds. Regarding the self-iteration process, MSI-SQL typically reaches optimal performance within 2–3 cycles, keeping the total training overhead practical.

B.4 Empirical Analysis of the Base-Model Reset Strategy

The *Base-Model Reset* (BMR) strategy introduced in Section 2.2 is driven by our empirical observations. Standard iterative self-training typically employs Continuous Fine-Tuning (CFT) (He et al., 2025; Ding et al., 2025), where the model M_t is

initialized with the intermediate checkpoint M_{t-1} . To substantiate our design choice, we compared CFT against our BMR strategy—which explicitly abandons M_{t-1} and initializes from the original base model M_0 —across multiple self-training iterations.

Our experiments reveal that CFT suffers from severe reasoning degradation (Table 6). As iterations progress, the M_{t-1} checkpoint overfits to the simpler syntactic patterns generated in early rounds. When confronted with newly introduced complex queries, the CFT model struggles to adapt, frequently generating chaotic and incoherent responses (i.e., mode collapse). In contrast, by resetting to M_0 , the BMR strategy preserves a highly generalized representation space. This enables the model to safely and effectively absorb new data without being constrained by the biased weights of intermediate checkpoints, ensuring stable and monotonic performance gains.

Round	CFT	BMR
1	64.69	64.69
2	63.23	65.91
3	60.63	66.36

Table 6: Performance comparison between Continuous Fine-Tuning (CFT) and Base-Model Reset (BMR) across self-training iterations on BIRD Dev Dataset (SFT only).

B.5 Sensitivity Analysis of Mask Ratio

As shown in Table 7, we evaluate the impact of varying the mask token ratio (20%, 40%, and 60%) on the BIRD-Dev execution accuracy. The results exhibit a clear inverted-U trend, with performance peaking at 71.06% when the mask ratio is set to 40%. This indicates that a moderate masking strategy provides the optimal “structural anchor” for the model.

Mask Ratio	BIRD-Dev (EX)
20%	69.03
40%	71.06
60%	70.27

Table 7: Sensitivity analysis of the mask token ratio on BIRD execution accuracy.

B.6 Ablation on Masking Granularity

To validate our choice of masking non-structural parts (table names and conditions) over structural SQL keywords (e.g., SELECT, JOIN), we compare both strategies under an SFT-only setting on the BIRD Dev set. Masking table names and conditions achieves 66.36% EX, outperforming masking SQL keywords by 2.15 points (64.21%). This gap demonstrates that masking semantically meaningful, query-specific tokens provides more effective structural anchors than masking generic SQL syntax, confirming that the gains stem from logical constraints rather than a simple regularization effect.

B.7 Quantitative Analysis of Data Underutilization

To substantiate the data underutilization claim in Section 2.2, we quantify the proportion of training samples that fail to produce valid SQL via single-pass prompting, and the proportion subsequently recovered by MSI-SQL’s self-iterative generation.

As shown in Table 8, single-pass prompting fails on 19.53% of BIRD and 23.36% of Spider training samples, causing these instances to be excluded from CoT fine-tuning. MSI-SQL recovers a substantial portion of these discarded samples through iterative generation, with gains concentrated in hard queries (those containing ≥ 2 JOINS, CTEs, or window functions): +3.55% on BIRD and +2.60% on Spider.

Dataset	Query Type	Total	Single-Pass	MSI-SQL
BIRD	Simple	7511	7223	7293 (+0.93%)
	Hard	1917	364	432 (+3.55%)
Spider	Simple	7080	6265	6314 (+0.82%)
	Hard	1579	371	412 (+2.60%)

Table 8: Number of training samples producing valid SQL under single-pass prompting vs. MSI-SQL. Hard queries contain ≥ 2 JOINS, CTEs, or window functions. Percentages indicate recovery relative to the total sample count.

B.8 Effectiveness of Self-Iteration Beyond Data Size

To demonstrate that the gains from MSI-SQL are not simply attributable to increased data size, we compare it against a baseline trained on data generated via 3-shot prompting with execution verification—a method that produces a comparable data volume without self-iteration. For a fair compar-

ison, both settings use SFT only and exclude the DPO stage. As shown in Table 9, MSI-SQL outperforms this baseline by 1.31, 1.74, and 1.35 EX points on BIRD Dev, Spider Dev, and Spider Test, respectively, demonstrating that the benefit of self-iteration stems from the progressive conquest of hard queries rather than mere data augmentation. Furthermore, replacing direct generation with fixed SQL templates leads to a 0.92% drop on BIRD, highlighting the importance of generation diversity.

Method	BIRD Dev EX	Spider Dev EX	Spider Test EX
3-shot prompting + execution verification	65.05	83.20	83.41
MSI-SQL w/o DPO	66.36	84.94	84.76

Table 9: Comparison of MSI-SQL (SFT only) against 3-shot prompting with execution verification under a matched data-size setting.

B.9 Effectiveness of Structure-Aware Masking over Random FIM

To validate the specificity of our Masked SQL Completion design over generic Fill-in-the-Middle (FIM) style completion, we compare against a baseline that pre-trains the base model with random FIM-style SQL splitting before standard SFT. Both settings exclude DPO for a fair comparison. As shown in Table 10, MSI-SQL w/o DPO achieves comparable performance on BIRD (−0.20 EX) while outperforming the FIM baseline by +1.77 and +1.02 EX points on Spider Dev and Spider Test, respectively. These results indicate that our structure-aware masking strategy, which targets semantically meaningful SQL tokens (e.g., table names, JOIN conditions), better captures SQL structural dependencies than random FIM-style completion.

Method	BIRD Dev EX	Spider Dev EX	Spider Test EX
FIM pretrain + SFT	66.56	83.17	83.74
MSI-SQL w/o DPO	66.36 (−0.20)	84.94 (+1.77)	84.76 (+1.02)

Table 10: Comparison between FIM-style pre-training and our Masked SQL Completion task under an SFT-only setting.

B.10 Effectiveness of Mask-Augmented Soft Self-Consistency

To isolate the effect of our Mask-Augmented Soft Self-Consistency (MSSC) mechanism from confounding factors such as model ensembling or DPO, we compare it against a single-model MSI-SQL baseline using greedy search under the same model

setting. As shown in Table 11, MSSC consistently outperforms greedy search by 2.40, 2.48, and 3.80 EX points on BIRD Dev, Spider Dev, and Spider Test, respectively, confirming that the performance gain is attributable to the proposed mechanism rather than increased sampling.

Method	BIRD Dev EX	Spider Dev EX	Spider Test EX
MSI-SQL w/ greedy search	68.77	84.82	85.00
MSI-SQL w/ MSSC	71.17	87.30	88.80

Table 11: Comparison between greedy search and Mask-Augmented Soft Self-Consistency (MSSC) under the same single-model setting.

B.11 Cross-Dataset Generalization

To evaluate the transferability of MSI-SQL beyond mixed-dataset training, we conduct zero-shot cross-dataset experiments where the model is trained on one dataset and directly evaluated on the other, without using any training data from the target dataset. As shown in Table 12, when trained solely on Spider and evaluated on BIRD Dev, our model achieves 67.7% EX. Conversely, training on BIRD alone yields 80.6% and 81.0% EX on Spider Dev and Spider Test, respectively. These results demonstrate that MSI-SQL generalizes well to unseen schemas and dataset-specific terminology, confirming that its strong performance is not solely attributable to mixed-dataset training.

Train	Test	EX
Spider	BIRD Dev	67.7%
BIRD	Spider Dev	80.6%
BIRD	Spider Test	81.0%

Table 12: Zero-shot cross-dataset generalization results.

C Prompt Templates for Masked SQL Completion

We present the prompt used for our Masked SQL Completion Task below.

Prompt Template for Masked SQL Completion Task.

System: You are a data science expert. Below, you are provided with a database schema and a partially masked SQL query. Your task is to understand the schema and the natural language question, then fill in the masked parts of the SQL query to complete a valid SQL statement that answers the question.

User:

Database Engine:
SQLite

Database Schema: {Database Schema}

This schema describes the database's structure, including tables, columns, primary keys, foreign keys, and any relevant relationships or constraints.

Question: {Question}

Instructions:

- You will be given a partially masked SQL query. Your task is to fill in the missing parts to complete a valid SQL query that answers the question.
- Make sure to only fill in the masked parts without adding extra or unrelated content.
- The completed SQL query should return all the information requested in the question, without missing anything.

The partially masked SQL query is: { Masked SQL }

Output Format:

In your answer, please enclose the completed SQL query in a code block:

```
``` sql
```

– Your completed SQL query here

```
```
```