ADEPT-SQL: A High-performance Text-to-SQL Application for Real-World Enterprise-Level Databases

Yongnan Chen [1,2], Zhuo Chang[1], Shijia Gu[2], Yuanhang Zong[2], Mei Zhang[2], Shiyu Wang[2], Zixiang He[2], HongZhi Chen [2], Wei Jin[2], Bin Cui[1]

[1]Peking University, [2]Kunlun Digital Technology, CNPC

Correspondence: yongnanchen@pku.edu.cn, chenyongnan@cnpc.com.cn

Abstract

This paper presents ADEPT-SQL, a domain-adapted Text2SQL system that addresses critical deployment challenges in professional fields. While modern LLM-based solutions excel on academic benchmarks, we identify three persistent limitations in industrial application: domain-specific knowledge barriers, the schemas complexity in real-world, and the prohibitive computational costs of large LLMs. Our framework introduces two key innovations: a three-stage grounding mechanism combining dynamic terminology expansion, focused schema alignment, and historical query retrieval; coupled with a hybrid prompting architecture that decomposes SQL generation into schema-aware hinting, term disambiguation, and few-shot example incorporation phases. This approach enables efficient execution using smaller open-source LLMs while maintaining semantic precision. Deployed in petroleum engineering domains, our system achieves 97% execution accuracy on real-world databases, demonstrating 49% absolute improvement over SOTA baselines. We release implementation code to advance research in professional Text2SQL systems.

1 Introduction

The democratization of data access remains a fundamental challenge in modern database systems. While structured query languages like SQL provide precise data manipulation capabilities, their technical complexity creates a substantial barrier for non-expert users. The Natural Language to SQL (Text2SQL) task (Gao et al., 2024; Li et al., 2023b) has emerged as a promising solution, bridging this gap through intuitive natural language interfaces. While early systems employed rule-based approaches (Xu et al., 2020; Yaghmazadeh et al., 2017), the advent of large language models (LLMs) (Achiam et al., 2023; Ouyang et al., 2022; Guo et al., 2025) has revolutionized the field through their superior code-generation capabilities. Contemporary LLM-based solutions (Pourreza and Rafiei, 2023; Dong et al., 2023; Li et al., 2023a; Lyu et al., 2025; Fan et al., 2024) have developed sophisticated multi-stage paradigms incorporating schema linking, few-shot in-context learning, and automatic prompt generation, achieving state-of-the-art performance on standard benchmarks like Spider (Yu et al., 2018b) and BIRD (Li et al., 2023b).

Nevertheless, significant gaps persist when deploying these systems in real-world professional domains (Pi et al., 2022). Our empirical analysis reveals a significant performance drop for leading LLM-based methods (Gao et al., 2023; Pourreza and Rafiei, 2023; Gorti et al., 2025; Fan et al., 2024) on industrial databases. Three fundamental challenges undermine practical deployment:

Domain Knowledge Barriers. Professional domains exhibit unique semantic characteristics that challenge conventional Text2SQL paradigms due to: (1) *Domain-specific terminology* (e.g., "CDU" denoting atmospheric and vacuum distillation unit in petroleum engineering) often falls outside LLMs' general vocabulary; (2) *Complex formulations of professional metrics* (e.g., "production ratios" may vary across subdomains and require explicit contextualization) (Guo et al., 2019).

Semantic Schema Complexity. Real-world database schemas violate the *clean* structural assumptions of academic benchmarks. Our study on industrial databases uncovered two prevalent issues: (1) *Opaque column naming practices* (e.g., "PDO_23A" representing production daily output) requiring expert interpretation (Lin et al., 2020; Yu et al., 2018a), and (2) *Versioned tables with overlapping semantics* (e.g., "prod_2023v2" vs. "rpt_refinery_23" storing equivalent metrics under divergent schemas).

Computational Constraints. While current sys-

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 275–283

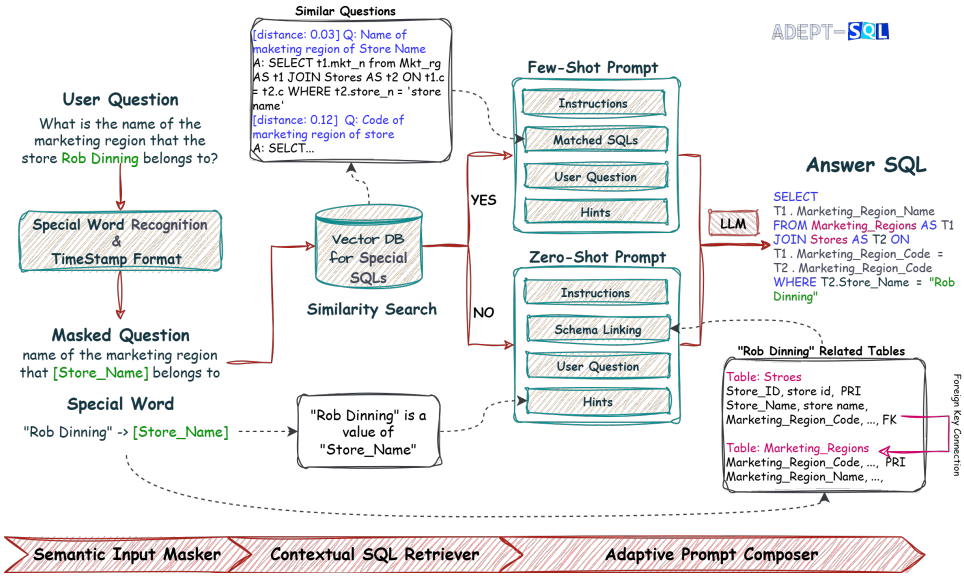

Figure 1: ADEPT-SQL framework architecture with three core modules: (a) Semantic Input Masker handles domain terminology alignment, (b) Contextual SQL Retriever resolves hidden business rules through vector-based QS pair matching, (c) Adaptive Prompt Composer optimizes prompt strategies based on the retrieval results. The dashed arrow indicates conditional execution flow.

tems rely on large-scale LLM APIs (175B+ parameters) for optimal performance (Fan et al., 2024; Lyu et al., 2025; Wang et al., 2024), their operational costs and latency become prohibitive for enterprise deployment. Smaller open-source models (<13B parameters) remain inadequate due to their limited capacity for complex, end-to-end Text2SQL generation. It necessitates reducing the LLM's accuracy burden, enabling smaller LLMs to perform efficiently.

To address these challenges, we propose **ADEPT-SQL**, a novel framework for domain-adapted Text2SQL generation that integrates two complementary innovations. First, our three-stage *grounding* mechanism systematically enhances domain understanding through: (1) *dynamic terminology expansion* using domain-specific corpora to capture specialized vocabulary (Zhao et al., 2022), (2) *context-aware schema alignment* mapping opaque schema elements to their conceptual equivalents, and (3) *historical query retrieval* that minimizes ambiguous references by leveraging past successful queries. Second, we introduce a hybrid prompting architecture (Tan et al., 2024; Shi et al., 2024; Tai et al., 2023) that strategically decomposes the SQL generation process into three coordinated phases: *schema-aware hinting* for structural guidance, *term disambiguation* for precise concept mapping, and *few-shot example incorporation* for syntactically valid output. The combined approach specifically targets the identified limitations of existing systems in professional deployment scenarios. Also, it enables smaller open-source models to achieve performance comparable to large-scale LLMs while maintaining execution efficiency.

Our system has been successfully deployed in petroleum domains, demonstrating a 97% execution accuracy on production databases - a 49% absolute improvement over existing baselines, showcasing operational reliability and usability in real-world deployment scenarios.

2 Architecture of ADEPT-SQL

Figure 1 shows system overview of ADEPT-SQL (Adaptive Dynamic Enhanced Prompt Text-to-SQL), with three core modules: (1) *Semantic Input Masker* handles domain terminology alignment, (2) *Contextual SQL Retriever* resolves hidden business rules through vector-based QS pair matching, and (3) *Adaptive Prompt Composer* optimizes prompt strategies based on the retrieval results.

2.1 Semantic Input Masker

The **Semantic Input Masker (SIM)** identifies the domain-specific terminologies and masks the terminologies with database metadata.

The SIM module clarifies user domain-specific questions using two knowledge repositories: the **Metadata repository** and the **Terminology repos-**

itory. It first masks the disambiguated terminologies in the user query to the database field names using the Terminology repository. The Terminology repository stores domain-specific nominal words (e.g., "CDU-I" and "hydrocracking") and their corresponding field names in user database. These words are continuously updated with the database.

Next, the SIM module aligns the schema with the user's question by utilizing the Metadata repository. This repository contains table and field information, including names, comments and data-type, from the enterprise database. It filters versioned tables and fields, and keeps their comments clean and clear for better understanding.

Original Question:
The production of CDU-I on 3 Mar?
With Terminology Repository:
The production of [UNIT_ALIAS] on 3 Mar?
With Metadata Repository:
The production of [refinery unit name] on 3 Mar?

In the above example, SIM module maps "CDU-I" to the field "UNIT_ALIAS" using the Terminology repository, then maps "UNIT_ALIAS" to its description "refinery unit" from the Metadata repository. The finalized **Masked Question** preserves the user's intent while bypassing the domain knowledge barriers.

Original Question:
The production of CDU-I on 3 Mar?
Hint Sentence:
Word [CDU-I] is a value of field [UNIT_ALIAS].

During this process, SIM module produces **Hint** sentences for domain-knowledge augmentation. The hint sentences consist of the detected terminologies and the their field names. It will be incorporated in **Adaptive Prompt Composer** to ensures value bindings for SQL generation.

2.2 Contextual SQL Retriever

The **Contextual SQL Retriever (CSR)** retrieves in-context learning (ICL) materials by matching the masked user query to pre-stored Question-SQL (QS) pairs in the **QS Repository** via vector similarity search.

We observe from operational traces of practical database queries that a limited number of high-frequency SQL queries cover the majority of usage scenarios. For example, in manufacturing fields, high-frequency QS pairs like *"Retrieve Line A's production of today → SELECT..."* cover 50% of daily reporting needs. Based on this, we build the QS repository by extracting these high-frequency queries from the database's query history log, capturing query semantics and business logic.

Inside the QS repository, questions, masked question, and its answer SQL query are maintained, and the masked question is vectorized using Bge-m3 (Chen et al., 2024a). The CSR module calculates the similarity between the masked question vector (v_q) and vectors stored in QS repository ($v_*, * \in 1...n$) with L2-norm distance (Bektaş and Şişman, 2010):

$$d(v_q, v_*) = \sqrt{\sum_{i=1}^{n}(v_{q,i} - v_{*,i})^2}, \quad \forall * \in 1...n \tag{1}$$

The QS pairs with $d(v_q, v_*)$ larger than the user set threshold would be used as the ICL materials in the downstream SQL generation module.

Input Question
Q1: Yesterday production of Line B?
In Repository
Q2: Retrieve Line A's production of today.
SQL: SELECT ... WHERE unit = "Line A"
Embedding Similarity Score = 0.8, for:
Q1: Yesterday production of [unit name]
Q2: Retrieve [unit name]'s production of today

As above, CSR identifies the similar questions of user input question from the repository.

Further, this module enables LLMs to "acquit" the implicit business logic behind the user question, as the solutions for complex operations like metrics calculation and multi-table joins are implied in the answer SQLs stored in QS repository.

2.3 Adaptive Prompt Composer

The **Adaptive Prompt Composer (APC)** combines the relevant information gathered from previous modules, including domain-specific terminologies identified by the SIM and the contextual QS pairs retrieved by the CSR. These information is adaptively incorporated into two distinct prompt templates for SQL generation: the Few-shot prompt and the Zero-shot prompt, which are determined based on the availability of matching QS pairs in the CSR module (Figure 1).

Both prompt templates share common components, Instructions, User Question, and Hints sentences; while differ in contextual components.

Few-shot Prompt utilizes a set of QS pair examples retrieved from the QS repository. With the

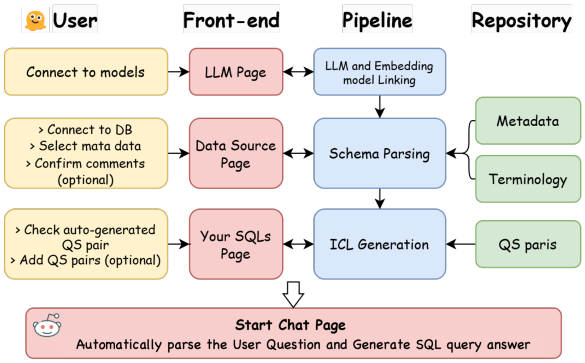

Figure 2: The interactions of User, Front-end, Backend Pipeline and Repositories of the ADEPT-SQL system.

augmentation of hints and examples, we exclude the large-scale databases schema in the prompt. This decision stems from our observations: (1) the schema information is already embedded in the example SQLs, and (2) redundancies in real-word database schema hinder SQL generation.

This approach enables the LLM to effectively imitate correct SQL patterns and reducing the likelihood of typographical errors.

Zero-shot Prompt offers target-oriented schema information related to user question when no QS pair is provided. The schema is identified by: (a) field names mentioned directly in the user query, (b) field names derived from the hint sentences, and (c) schema of the tables that contains these fields. Unlike semantic relevance-based schema linking methods (Gorti et al., 2025; Chen et al., 2024b; Gao et al., 2023), which might introduce redundancy and overlook the target columns, this approach ensures precise schema identification while maintaining system fluency.

For example, when user quires "*The production amount of CDU-I of today*", our method links the fields names to tables:

- pm_unit_t ← UNIT_ALIAS ← "CDU-I"
- rpt_daily_refinery_unit ← "amount"

While semantical methods returns the additionally unrelated table (*unit_maintain* with UNIT_ALIAS but no useful fields).

In the Appendix B, we show details for these two prompt branches.

3 Pipeline and Use Cases

As shown in Figure 2, the ADEPT-SQL system adopts a four-stage pipeline paradigm: (1) Environment Setting, (2) Schema Parsing, (3) ICL Selection, and (4) Prompt Generation. Users interact

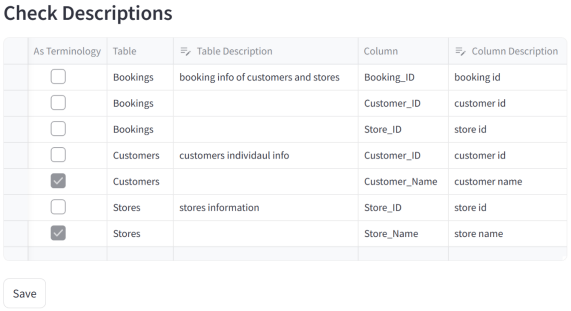

Figure 3: An example of Metadata Grounding and Terminology Grounding for the booking information inquiry assistant built on cre_db.

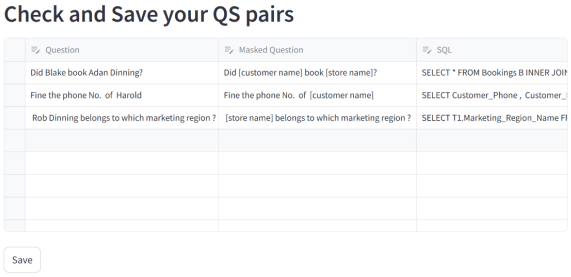

Figure 4: An example of QS candidate identified from the database log file of cre_db.

with the Front-end pages step by step, navigating through the Pipeline to complete the Repository maintaining.

Check the websites for online demo[1] and its video[2] and code[3]. A tutorial database from Spider dataset: *cre_Drama_Workshop_Groups.sqlite* (refer to as cre_db) is provided in the demo. Also, an use case, called Booking Information Inquiry Assistant, is illustrated throughout this section.

3.1 Environment Setting

In the **LLM and Data Source page**, users connect to their locally deployed LLMs, embedding models, and user database. We employ locally deployed open-source LLMs to meet confidentiality and security requirements for industrial deployments.

3.2 Schema Parsing

In the **Data Source page**, the system prepares the tables and fields that are related to the target topic of the assistant.

With user database connected, the system lists the names and comments of all tables and fields

[1] https://adept-sql-demo-for-text-to-sql.streamlit.app/

[2] https://youtu.be/iW2O5j61QwM

[3] https://github.com/lilichennn/ADEPT-SQL-Demo/tree/main

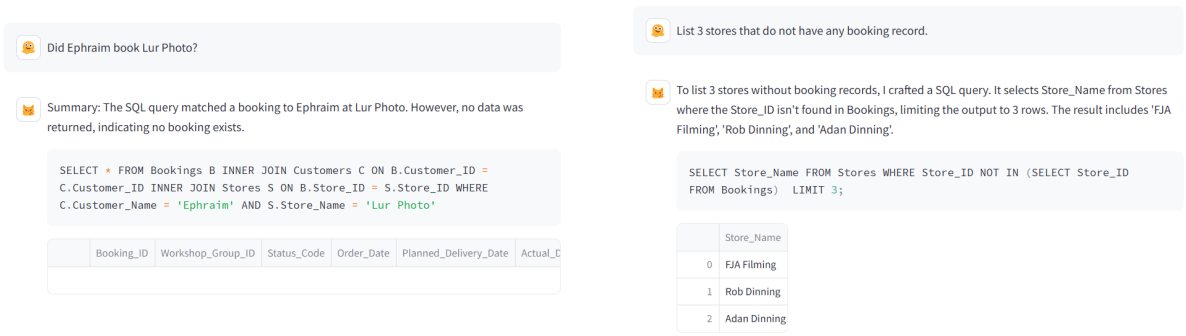

Figure 5: Chat examples of ADEPT-SQL using Few-shot (left) and Zero-shot (right) Promptings

in the database, where the comments are automatically extracted from the Data Definition Language (DDL). The tables and columns are ranked by the following rules: (a) tables/fields that are semantically similar with the assistant target are put top (Wang et al., 2020); (b) tables/fields with words like "copy, temp, v1" are put bottom.

User can go through the tables and field to select suitable fields according to the target topic of the assistant. In addition, considering the descriptions in the DDLs may not be very precise or too technical, the system allows the user to make changes to the descriptions. As shown in Figure 3, this step forms the *metadata repository*, and benefits the system to concentrate on assistant target.

In the meantime, the system decides the fields to be maintained in the *terminology grounding* and displays them in the "Terminology" column, as in Figure 3. Specifically, Fields named with "name", "alias" or "description" are selected, e.g. product_name or material_alias. It is highly probable that these fields contain the nominal vertical words of the user database. Also, users can check and change the field selections according to the assistant target.

Note that such terminology selection policy is more effective in real-world databases. Due to the scale and complexity of these databases, their metadata carries more property information.

3.3 ICL generation

In this stage, the system provides user with high-frequent SQLs in database query history and maintain them in *QS repository*. In detail, the SQL queries that were appeared and successfully executed for over three times were selected, and the corresponding questions generated with LLMs and confirmed by domain-specific users.

As shown in Figure 4, the system identified *SQL: SELECT ... WHERE T2.Store_Name = ...* a high-frequency query from the database log. Then, it adopted the connected LLM to generate a NL question *"Did Blake book Adan Dinning?"* for this query, and loaded the QS pair as candidate for QS repository.

During this process, the system also generates the masked question (i.e. "qmask" column) for each question leveraging the metadata repository, and the masked question is vectorized by the connected embedding model.

Similarly, the system allows users to upload new QS pair and modify or delete the QS candidates in **Your SQL page**.

3.4 Start Chat

The **Start Chat Page** provides an interface for users to interact with their dataset using natural language questions.

The user's input question goes through the SIM, CSR, and APC components to generate the corresponding SQL query. The SQL query is then executed on the user database, and the resulting table is sent back to the front-end. Additionally, the LLM is invoked again to summarize the entire task and provide natural language answers to the user's question.

Figure 5 shows two chat examples of the system. The left example adopts the Few-shot prompt template. The masked user input question in this example is semantically matched with the QS pair displayed in Figure 4. The resulting SQL correctly imitates the SQL and provides the natural language answer to the question.

The right example adopts the Zero-shot prompt. In this case, the system detects the word "Booking" and identifies tables containing this word from the metadata grounding, ultimately generating the correct SQL by itself.

Method	Hard(4)	Extra Hard(62)
Stand-alone DeepSeek-V3	0.25	0.03
DIN-SQL + DeepSeek-V3	**1.00**	0.48
ADEPT-SQL + Qwen2.5-7b	**1.00**	0.90
ADEPT-SQL + DeepSeek-V3	**1.00**	**0.97**

Table 1: Execution Match on Industry dataset mes_db. Bold text indicates the highest score. Note the mes_db does not have Easy or Medium levels according to the difficulty levels of Spider.

4 Experiments

We deploy a relatively small-scale LLM **Qwen2.5-7b** (Team, 2024) and a SOTA LLM **DeepSeek-R1-Distill-Llama-70b** (refer to as DeepSeek-V3) (Guo et al., 2025) for the experiments. Bge-m3 (Chen et al., 2024a) is used as embedding model.

4.1 Industry Database

We collect a real-word Manufacturing Execution System database (refer to as "mes_db") from a petrochemical company of the PetroChina Co., Ltd. as the **Industry** dataset. This database contains data of materials and productions of production equipments. In total, the database has exceed 100 tables with on average 15 columns for each table. Also, 66 Questions-SQL pairs are collected from the daily usage scenarios. The detailed analysis of the database and ADPET-SQL settings are in A.

According to the SQL difficulty levels of Spider (Yu et al., 2018b), we divided the Question-SQL pairs into four levels according to the SQL token length, Easy (less than 10), Medium (10 to 20), Hard (20 to 30) and Extra Hard (over 30). To demonstrate the efficiency of ADPET-SQL, we compared it with DIN-SQL (Pourreza and Rafiei, 2023).

The Execution Match (EM) accuracies (Finegan-Dollak et al., 2018) are shown in Table 3. The results demonstrate significant improvements with ADEPT-SQL on mes_db. While DeepSeek-V3 struggles with hard and extra-hard tasks, DIN-SQL+DeepSeek-V3 performs well on hard tasks but fails on half of the extra-hard tasks. ADEPT-SQL maintains high performance even with the smaller Qwen2.5-7b LLM, highlighting its ability to overcome computational limitations in real-world scenarios.

4.2 Benchmark Databases

We used two Spider databases: 'cre_db' (Section 3) and 'products_gen_characteristics.sqlite' ('prod_db'). 'cre_db' contains 18 tables and 82 QS pairs, while 'prod_db' has 6 tables and 86 QS pairs. Detailed analysis is provided in A.

Database	Hardness (No. SQL)	ADEPT-SQL + Qwen2.5-7b	ADEPT-SQL + DeepSeek-V3
cre_db	Easy (20)	0.85	0.94
	Medium (18)	0.83	0.90
	Hard (24)	0.91	0.96
	Extra (24)	0.91	1.00
	Average	**0.88**	**0.95**
prod_db	Easy (22)	0.90	0.95
	Medium (40)	0.83	0.90
	Hard (18)	0.94	0.94
	Extra (2)	1.00	1.00
	Average	**0.92**	**0.95**

Table 2: The Execution Match of ADEPT-SQL on benchmark databases.

These values reflect ADEPT-SQL's robustness and its ability to adapt to different types of databases, as evidenced by the varying SQL hardness levels. In comparison to other top-performing models in the Spider Leaderboard, ADEPT-SQL's EM results are competitive, aligning closely with the other SOTA models. These results underscore the potential of ADEPT-SQL in handling diverse real-world Text2SQL tasks effectively.

5 Conclusion

In this paper, we introduced ADEPT-SQL, a Text-to-SQL framework designed for real-world enterprise databases. ADEPT-SQL addresses the challenges like domain-specific terminology, semantic mismatches, and redundant metadata in real-world with a novel architecture combining dynamic terminology expansion, contextual schema alignment, and historical SQL retrieval, along with hybrid prompting for efficient SQL generation.

The system balances accuracy, interpretability, and computational efficiency, making it ideal for enterprise applications. Our experiments on industrial and benchmark datasets show high performance, even with smaller open-source LLMs, proving its competitive accuracy.

Limitations

The limitations of ADEPT-SQL are:

- Better performance can be achieved by better assistant design, include narrowing the target of the assistant, making the comments fo tables and fields more clear and adding more

QS pairs. This situation adds burdens on the user side. Therefore, we recommend the specialists of the target application field to do the assistant settings.

- Based on our experiments on three databases, we recommend users set the threshold in CSR as 0.85 to balance the semantical similarity of user queries and high-frequency queries in QS repository. The threshold is set as default in demo version, users can adjust it in the formal version.

- The current experiments are based on a relatively small number of datasets, and the evaluation of ADEPT-SQL's performance across a broader range of real-world scenarios remains limited. Future work should include testing on more diverse and larger-scale datasets to further validate the system's effectiveness. Also, the use of more novel methods in future iterations could potentially lead to even greater performance gains.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sebahattin Bektaş and Yasemin Şişman. 2010. The comparison of l1 and l2-norm minimization methods. *International Journal of the Physical Sciences*, 5(11):1721–1727.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Yongnan Chen, Shijia Gu, and Zixiang He. 2024b. FATO-SQL: a comprehensive framework for high-performance Text-to-SQL task. In *International Conference on Optics, Electronics, and Communication Engineering (OECE 2024)*, page 166, Wuhan, China. SPIE.

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

Yuankai Fan, Zhenying He, Tonghui Ren, Can Huang, Yinan Jing, Kai Zhang, and X. Sean Wang. 2024. Metasql: A Generate-then-Rank Framework for Natural Language to SQL Translation. *arXiv preprint*. ArXiv:2402.17144 [cs].

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

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 [cs].

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(5):1132–1145.

Satya Krishna Gorti, Ilan Gofman, Zhaoyan Liu, Jiapeng Wu, Noël Vouitsis, Guangwei Yu, Jesse C. Cresswell, and Rasa Hosseinzadeh. 2025. MSc-SQL: Multi-Sample Critiquing Small Language Models For Text-To-SQL Translation. *arXiv preprint*. ArXiv:2410.12916 [cs].

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. *arXiv preprint*. ArXiv:2302.05965 [cs].

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. 2023b. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *NeurIPS*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Shuai Lyu, Haoran Luo, Zhonghong Ou, Yifan Zhu, Xiaoran Shang, Yang Qin, and Meina Song. 2025. SQL-o1: A Self-Reward Heuristic Dynamic Search Method for Text-to-SQL. *arXiv preprint*. ArXiv:2502.11741 [cs].

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 *NeurIPS*.

Xinyu Pi, Bing Wang, Yan Gao, Jiaqi Guo, Zhoujun Li, and Jian-Guang Lou. 2022. Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2007–2022, Dublin, Ireland. Association for Computational Linguistics.

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.

Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. 2024. A Survey on Employing Large Language Models for Text-to-SQL Tasks. *arXiv preprint*. ArXiv:2407.15186 [cs].

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, Singapore. Association for Computational Linguistics.

Zhao Tan, Xiping Liu, Qing Shu, Xi Li, Changxuan Wan, Dexi Liu, Qizhi Wan, and Guoqiong Liao. 2024. Enhancing text-to-SQL capabilities of large language models through tailored promptings. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6091–6109, Torino, Italia. ELRA and ICCL.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

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

Silei Xu, Giovanni Campagna, Jian Li, and Monica S Lam. 2020. Schema2qa: High-quality and low-cost q&a agents for the structured web. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1685–1694.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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.

Chen Zhao, Yu Su, Adam Pauls, and Emmanouil Antonios Platanios. 2022. Bridging the generalization gap in text-to-SQL parsing with schema expansion. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5568–5578, Dublin, Ireland. Association for Computational Linguistics.

A Details of Experiments Setting

A.1 Metadata and Terminology Groundings

For the two well-annotated databases of Spider, **cre_db** and **prod_db**, we loaded all the tables and fields into the backend database. The information provided by the *table.json* is used as table and fields comments. Then, the fields that store nominal values are selected for terminology fields, e.g. *City_Town, Product_Name, etc*.

For the industry database **mes_db**, we design the metadata and terminology groundings according to the topics covered by the collected QS pairs, and try the best to make the repositories covers all the QS pairs. In total, metadata repository contains 19 tables and 87 fields, with 6 fields are set true in "As Terminology". Table 3 shows the settings for question *"What's the total planned production amount of CDU-I on Feb. 2024?"*

Note that the comments plays an important role in interpreting the original field names, e.g. *in-out_type -> "binary indicator of material feeding:0 and discharging:1"*, for the real-world databases. The original comments of "inout_type" is "type of in and out", which is nearly helpless for values binding in SQL generation.

Field	Comment	As Terminology
Table: rpt_t_daily_refinery_unit		
node_code	refinery unit code	Yes
inout_type	binary indicator of material feeding:"0" or discharging:"1"	No
r_date	production date	No
plan_total_amount	monthly planned production (T)	No
mtrl_alias_**show**	material alias	Yes
daily_amount	daily production (T)	No
Table: pm_unit_t		
UNIT_CODE	refinery unit code	Yes
UNIT_ALIAS	refinery unit alias	Yes

Table 3: A part of metadata and terminology groundings for mes_db. Bold field names indicate not matched semantic meanings to the real meanings.

SQL Type	No. in Dataset	No. in Repository
Database: cre_db		
Multiple aggregations	16	8
Sub-query	4	2
Table JOIN	16	8
Database: prod_db		
Sub-query	2	1
Table JOIN	8	4

Table 4: Medium-level SQL Summary of cre_db and prod_db.

A.2 Question-SQL Pairs Grounding

For **cre_db** and **prod_db**, Easy level QS pairs are left for the system to use Zero-shot prompts; all the QS pairs that are belong to Hard and Extra Hard levels are stored into the QS repository; and for Medium level pairs, the decision is made by the SQL structure complexity. To avoid putting the answer in the prompt, we made modifications on the QS pairs by replacing the value binding parts in the SQL queries, i.e. values on the right hand side of "=", and changed the Questions correspondingly. Table 4 summaries the medium level SQLs that are put into the repository.

Note that all the three datasets have repetitive SQL queries. For instance, in "cre_db," there are questions like *"Count the total number of bookings made"* and *"How many bookings do we have?"* with the same SQL answer. For such cases, only one QS pair is uploaded to the repository.

This mirrors real-world database interactions, where queries asking for the same information may vary in phrasing. In the "**mes_db**" database, such repentance also exist, leading to further pruning of the repository. Although there are 66 QS pairs in total, only 29 unique pairs are stored in the repository after eliminating the duplicates.

Also, the repository has a mechanism to avoid duplicates. This refined selection ensures that the system maintains efficiency without sacrificing the diversity of SQL queries that might arise in actual application scenarios.

B Prompt Templates

Here we exhibit the prompt templates that were used in ADPET-SQL Demo version. Note the places closed with { } should be filled with proper contents extracted form databases and repositories.

B.1 Few-Shot Prompt

```
#Character#
  You are an expert of SQL language and the best skill of you is mimic similar SQL statements to write new SQL statements. Also, you can replace the special terminologies and time points in SQL according to the user question.

  #Task#
  Write a SQL statement to answer the user question, modeled after the following Examples.

  #Limitations#
  1. Your SQL must use the terminologies given by "HINTS", DO NOT change the terminologies themselves.
  2. Your SQL must imitate the Examples to be grammarly correct.
  3. Your SQL should be careful on the dependency of fields you use.
  4. Make sure that the your SQL can be executed by pd.read_sql_query().

  #Examples#
  {examples}

  #Now Write SQL#
  Question: {user_question}

  HINTS: {hints}
```

B.2 Zero-Shot Prompt

```
#Character#
  You are an expert of SQL language and you serve in a world-class company. You are familiar with the tables and fields in the company's database. Therefore your job is answer the data retrieval queries from the staff using SQL.

  #Task#
  Now, you have a question to solve, the staff also told you the terminologies in this question are related to which tables and fields. You need to utilize the following table schema information to write a correct SQL.

  #Limitations#
  1. Your SQL must use the terminologies given by "HINTS", DO NOT change the terminologies themselves.
  2. Your SQL need to be readable, so enter line breaks where appropriate.
  3. Your SQL must be grammarly correct, so be careful on the dependency of fields you use.
  4. You can only write one SQL statement, NO ONE need extra explanations.
  5. Make sure that the your SQL can be executed by pd.read_sql_query().

  #Schema#
  {schema}

  #Now Write SQL#
  Question: {user_question}

  HINTS: {hints}
```