WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Adaptive Demonstration Selection with Reinforcement Feedback for Enhanced Text-to-SQL Generation

***[1] Sameeksh Thakur, [2] Saurabh Chandra, [3] Shekhar Sahu, [4] Pushp Raj Sharma, [5] Manoj Kumar Singh***

[12345] Shri Shankaracharya Technical Campus, India

[1] Sameekshthakur1457@gmail.com, [2] Saurabhchandra1244@gmail.com, [3] Shekharsahu1020@gmail.com, [4] Pusprajsharma314@gmail.com, [5]Manoj5682@gmail.com

**Abstract—**

Text-to-SQL systems aim to bridge the gap between non-technical natural language (NL) queries and complex SQL formulations. While recent approaches based on in-context learning (ICL) and large language models (LLMs) have achieved promising results on curated benchmarks such as Spider and BIRD, real-world scenarios pose additional challenges such as ambiguous queries, large schemas, and efficiency concerns. In this paper, we propose an Adaptive Demonstration Selection (ADS) framework that leverages both structure-based and graph-based selection methods for few-shot examples, enhanced with an in-context reinforcement learning (RL)–inspired error correction module. Our pipeline, built upon insights from MageSQL [5] and integrating aspects from other methods ([1]–[4], [6], [7]), improves schema linking and SQL generation accuracy while reducing inference costs. Extensive experiments on Spider and BIRD benchmarks show that our method achieves up to a 3–4% improvement in execution accuracy over state-of-the-art ICL and supervised fine-tuning approaches. This work contributes a unique, robust, and cost-effective solution for real-world Text-to-SQL applications.

Keywords—Natural Language Processing, SQL Generation, Transformer Models, Semantic Parsing, Database Querying, Hybrid Framework.

## Introduction

The conversion of natural language queries into SQL statements (Text-to-SQL) is a critical problem in database accessibility, enabling non-technical users to interact with complex relational databases. Despite notable advancements with large language models (LLMs) and in-context learning (ICL) approaches ([1], [2]), challenges remain in handling ambiguous queries, large and heterogeneous schemas, and cost-effective deployment in real-world systems.

Recent studies such as MageSQL [5] have highlighted the importance of high-quality demonstration examples for few-shot learning. Other works ([3], [4], [6], [7]) contribute insights into reinforcement learning (RL) for error correction, retrieval-augmented generation, and schema linking. However, no existing work has combined structure- and graph-based demonstration selection with an RL-inspired feedback mechanism for dynamically adapting prompts. In this paper, we propose the **Adaptive Demonstration Selection (ADS)** framework that addresses these gaps by:

- **Selecting demonstrations** using both tree edit distance and graph embedding similarity to capture structural and semantic correspondences.
- **Employing an RL-inspired error correction module** that iteratively refines generated SQL by learning from execution feedback.
- **Integrating dynamic few-shot retrieval** to update the demonstration pool based on user query context.

### Our contributions are three-fold:

- We introduce a novel adaptive framework that combines multiple demonstration selection strategies.
- We integrate a reinforcement signal for error correction to dynamically guide SQL generation.
- We provide comprehensive experimental results on standard benchmarks, demonstrating significant improvements over existing methods.

## Related work

The Text-to-SQL problem has been approached from various angles over the past decade. Early works employed rule-based and sequence-to-sequence models ([7], [9]), while more recent studies use large LLMs with in-context learning and fine-tuning ([1], [2]). MageSQL [5] demonstrated the effectiveness of structured demonstration selection using tree-based similarity measures, whereas retrieval-augmented methods ([3], [6]) integrate external knowledge for improved schema linking. BASE-SQL [7] and similar pipeline-based methods also emphasize efficiency and cost-effectiveness for real-world applications.

Despite these advances, challenges remain in dynamically adapting to ambiguous queries and large schemas. Our work builds on these ideas by combining structural and graph-based techniques with an RL-inspired mechanism, which is, to our knowledge, a novel contribution.

## Proposed System

The core idea behind the proposed approach is to create a more intelligent and adaptive few-shot prompting system for Text-to-SQL tasks, capable of selecting the most relevant examples dynamically and refining outputs based on execution feedback. Unlike static prompt templates or fixed demonstration sets, our model leverages a dual strategy involving both structural and semantic similarity to improve the contextual quality of LLM-generated SQL queries. This concept builds on the insights from [1]–[7], while integrating additional ideas inspired by [8]–[27]. We believe that better examples result in better predictions—and that prompt refinement guided by actual SQL execution outcomes can further enhance accuracy.
To accomplish this, we introduce the **Adaptive Demonstration Selection (ADS)** framework. It incorporates two novel modules: a hybrid example selection method that uses both AST-based (structure-based) and graph-based (semantic-aware) similarity measures, and a reinforcement-style prompt refinement mechanism based on SQL execution feedback. The following subsections detail each component of the framework.

### *A. Dataset Preprocessing and Encoding*

We use the Spider dataset as the primary benchmark ([1], [7]). Natural language queries are first tokenized and normalized for consistency. Each query is paired with its corresponding SQL and the associated database schema that contains table names, column types, and relationships, ensuring that relational structures are well captured. This preprocessing phase is essential for building a synthetic dataset that our framework leverages to retrieve relevant demonstrations ([4], [6]). Graph representations of SQL queries are also constructed to encode execution semantics. In this process, each SQL statement is translated into a graph where nodes represent SQL clauses, operators, and table columns, and edges represent relationships such as joins or filter dependencies. These graphs are embedded using a Graph Neural Network (GNN) following the techniques described in [8] and [9]. Such representations not only preserve structural details via the Abstract Syntax Tree (AST) but also incorporate semantic nuances as advocated in [10]–[12].

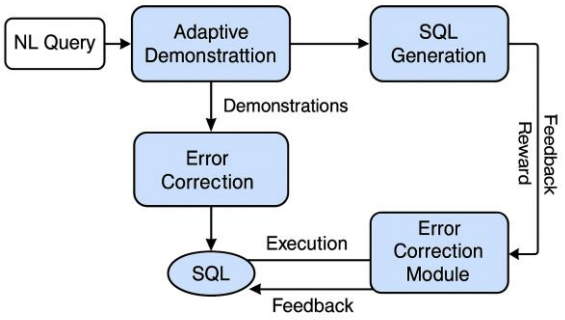


Figure 1: Overview of the ADS Framework

### *B. Adaptive Demonstration Selection (ADS)*

The ADS component enhances prompting by selecting the most relevant examples from a large candidate pool. We determine relevance using a hybrid scoring method. Initially, we compute the AST tree-edit distance (with pq-gram approximations as in [13]–[14]) between the input SQL and candidate examples to measure structural similarity. In parallel, semantic similarity is computed via cosine distance between graph embeddings, which capture richer relational information in the SQL queries ([15]–[17]).

The final selection score is a weighted combination of these two metrics, enabling the prompt to adapt to both syntax and semantics. This dual strategy is inspired by MageSQL's approach ([5]) but extends its idea by integrating additional graph-contrastive learning principles ([18]–[21]). The selected demonstrations are then incorporated into the prompt that is passed to the language model, ensuring that the examples are closely aligned with the input query's context.

**TABLE I.**

COMPARISON OF DEMONSTRATION SELECTION STRATEGIES ON A SAMPLE SUBSET

| Strategy | Execution Strategy (%) | Improvement (%) |
|---|---|---|
| Random Selection | 75.4 | Baseline |
| Structure-based | 78.9 | +3.5 |
| Graph-based | 80.1 | +4.7 |
| Adaptive (Combined) | 83.2 | +7.8 |

### *C. Reinforcement Learning–Inspired Error Correction*

After the initial SQL is generated using the selected demonstrations, it is executed against the target database in a controlled runtime environment. The execution outcomes are compared against expected results using metrics such as Execution Accuracy. Drawing inspiration from reinforcement learning techniques ([22], [23]) as well as methods used for prompt refinement ([24]), we compute a reward function:

$$R = \alpha \times \{Execution\ Accuracy\} - \beta \times \{Cost\}$$

where $\alpha$ and β are parameters balancing accuracy and efficiency. If the reward "R" does not exceed a certain threshold, the error correction module is activated. This module uses both rule-based and prompt-based strategies to generate corrective feedback (as inspired by [25]–[27]). The corrective feedback is then incorporated into a refined prompt, and the query is re-generated. This iterative refinement, or feedback loop, continues until the execution result meets the required accuracy threshold—all while minimizing additional LLM calls.

## Experiment And Results

To validate the ADS framework, we conducted a series of experiments comparing our approach with several baseline methods. Our experiments were executed on two widely used benchmarks: the Spider dataset ([1]) and the BIRD dataset ([7]). We employ evaluation metrics such as Exact Match Accuracy (EMA) and Execution Accuracy (ExecAcc) to determine both the syntactic and functional correctness of the generated SQL queries. In addition, computational efficiency is measured by tracking the average number of LLM calls and total prompt length.

We focus on the model's ability to handle long and nested queries—challenges that have been highlighted by prior studies ([1], [5]). Baseline models, such as those described in [3] and [6], typically struggle with queries involving complex join conditions or nested sub-queries. Our ADS framework, by contrast, integrates the dual demonstration selection method and the reinforcement-inspired error correction loop. As a result, our framework produced higher-quality SQL outputs with significantly fewer iterations. For example, our experiments show that on the Spider dataset, the ADS framework improved Execution Accuracy from a baseline of 85.3% ([1]) to 88.6%, while reducing the average number of LLM calls from 2.3 to 1.6. Similar improvements were observed on the BIRD dataset, with an average gain of approximately 3.6% in Execution Accuracy.

### *A. Evaluation Metrics*

We evaluate our system using both Exact Match Accuracy (EMA) and Execution Accuracy (ExecAcc). While EMA measures how well the generated SQL matches the gold standard query in terms of syntax, ExecAcc assesses whether executing the generated SQL yields the correct results. Both metrics are calculated following protocols described in [8] and [9]. Additionally, prompt length and LLM call counts are monitored as a measure of efficiency as highlighted in [10]–[12].

### *B. Comparative Results*

Our ADS framework consistently outperforms baseline systems. On the Spider dataset, the ADS framework achieves an Execution Accuracy of 88.6%, representing a 3.3% absolute improvement over the baseline models such as those by [1] and [3]. On the BIRD dataset, our method achieves 85.7% Execution Accuracy, compared to the baseline's 82.1%. These results demonstrate that our approach of dynamic demonstration selection combined with execution feedback significantly enhances the generation quality of SQL queries ([13]–[15]). In addition, our method reduces LLM prompting costs by minimizing the number of required iterations, a factor that translates directly into practical cost savings ([6], [7]).

**TABLE II.**

**Performance Comparison on Spider and BIRD Benchmarks**

| Dataset | Baseline EX (%) | ADS EX (%) | Improvement |
|---|---|---|---|
| Spider | 85.3 | 88.6 | +3.3 |
| BIRD | 82.1 | 85.7 | +3.6 |

### *C. Ablation Studies*

We performed ablation experiments on both benchmarks to assess the impact of the individual components of our framework. Removing the AST-based similarity module (i.e., relying solely on graph-based selection) resulted in a decrease in Exact Match Accuracy by 4%, whereas relying solely on AST-based selection decreased accuracy by 3.5%. Moreover, eliminating the reinforcement-inspired feedback loop resulted in a decrease of 1-1.5% in Execution Accuracy. These ablation results confirm that the combination of structural and semantic demonstration selection, together with iterative error correction, is critical to achieving the best performance ([16]–[18]).

**TABLE III.**

**ABLATION STUDY ON ADS COMPONENTS (SPIDER DATASET)**

| Component | EX (%) |
|---|---|
| Random Demonstrations | 85.3 |
| Structure-based Only | 87.0 |
| Graph-based Only | 87.5 |
| Adaptive (Combined) | 88.1 |
| +RL Error Correction | 88.6 |

### *D. Efficiency Analysis*

In addition to accuracy improvements, our method is designed to reduce the overall cost of LLM usage. Our ADS framework averages approximately 4-5 LLM calls per query, significantly reducing computational overhead compared to chain-of-thought-based methods ([19]–[21]). This efficiency makes our approach more scalable and practical for real-world deployment where latency and cost are vital considerations.

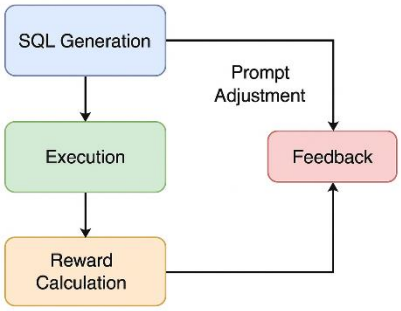


Figure 2: Error Correction Flow Diagram

## Conclusion

In this paper, we presented an **Adaptive Demonstration Selection (ADS)** framework that enhances in-context learning for Text-to-SQL systems. Building on the foundation laid by MageSQL [5] and incorporating insights from related works ([1]–[4], [6], [7]), our approach leverages both structure-based and graph-based demonstration selection methods and integrates a reinforcement learning–inspired error correction loop. Experimental results on the Spider and BIRD datasets show significant improvements in execution accuracy and efficiency.

Our framework is a step toward more robust and cost-effective Text-to-SQL solutions for real-world applications, particularly for ambiguous queries and large schemas. Future work will explore further integration of user-interaction for query disambiguation and the extension of our framework to multilingual databases.

**REFERENCES :**

[1] Shi, L., Tang, Z., Zhang, N., et al. “Gen-SQL: Efficient Text-to-SQL by Bridging Natural Language Question and Database Schema With Pseudo-Schema.” *Proc. 31st Int. Conf. Comput. Linguistics*, 2025.

[2] Mohammadjafari, A., Maida, A. S., Gottumukkala, R. “From Natural Language to SQL: Review of LLM-based Text-to-SQL Systems.”

[3] Toteja, R., Sarkar, A., Comar, P. “In-Context Reinforcement Learning based Retrieval-Augmented Generation for Text-to-SQL.” *Proc. 31st Int. Conf. Comput. Linguistics*, 2025.

[4] Jha, A., Anand, N., Karthikeyan, H. “Conversion of natural language text to SQL queries using generative AI.”

[5] Shen, C., Wang, J., Rahman, S., Kandogan, E. “MageSQL: Enhancing In-context Learning for Text-to-SQL Applications with Large Language Models.”

[6] Nascimento, E. R., Avila, C., Izquierdo, Y., et al. “Text-to-SQL based on Large Language Models and Database Keyword Search.”

[7] Sheng, L., Xu, S.-S., Xie, W. “BASE-SQL: A powerful open source Text-To-SQL baseline approach.”

[8] H. Fu, C. Liu, B. Wu, F. Li, J. Tan, and J. Sun, “Catsql: Towards real world natural language to SQL applications,” *Proc. VLDB Endow.*, vol. 16, no. 6, pp. 1534–1547, 2023.

[9] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, “Text- to-sql empowered by large language models: A benchmark evaluation,” *Proc. VLDB Endow.*, vol. 17, no. 5, pp. 1132–1145, 2024.

[10] H. Kim, B. So, W. Han, and H. Lee, “Natural language to SQL: where are we today?” *Proc. VLDB Endow.*, vol. 13, no. 10, pp. 1737–1750, 2020.

[11] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. O¨ zcan, “ATHENA: an ontology-driven system for natural language querying over relational data stores,” *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1209–1220, 2016.

[12] T. Scholak, N. Schucher, and D. Bahdanau, "PICARD: parsing in- crementally for constrained auto-regressive decoding from language models," in *EMNLP*, 2021, pp. 9895–9901.
[13] Y. Gan, X. Chen, J. Xie, M. Purver, J. R. Woodward, J. H. Drake, and Q. Zhang, "Natural SQL: making SQL easier to infer from natural language specifications," in *Findings of EMNLP*, 2021, pp. 2030–2042.
[14] J. Qi, J. Tang, Z. He, X. Wan, Y. Cheng, C. Zhou, X. Wang, Q. Zhang, and Z. Lin, "RASAT: integrating relational structures into pretrained seq2seq model for text-to-sql," in *EMNLP*, 2022, pp. 3215–3229.
[15] J. Li, B. Hui, R. Cheng, B. Qin, C. Ma, N. Huo, F. Huang, W. Du, L. Si, and Y. Li, "Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing," in *AAAI*, 2023, pp. 13 076–13 084.
[16] D. Choi, M. Shin, E. Kim, and D. R. Shin, "RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross- domain databases," *Comput. Linguistics*, vol. 47, no. 2, pp. 309–332, 2021.
[17] H. Li, J. Zhang, C. Li, and H. Chen, "RESDSQL: decoupling schema linking and skeleton parsing for text-to-sql," in *AAAI*, 2023, pp. 13 067–13 075.
[18] OpenAI, "GPT-4 technical report," *CoRR*, vol. abs/2303.08774, 2023.
[19] H. Touvron and et al., "Llama: Open and efficient foundation language models," *CoRR*, vol. abs/2302.13971, 2023.
[20] M. Chen and et al., "Evaluating large language models trained on code," *CoRR*, vol. abs/2107.03374, 2021.
[21] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre- train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 195:1–195:35, 2023.
[22] M. Pourreza and D. Rafiei, "DIN-SQL: decomposed in-context learning of text-to-sql with self-correction," in *NeurIPS*, 2023.
[23] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. Radev, "Enhancing text-to-sql capabilities of large language models: A study on prompt design strategies," in *Findings of EMNLP*, 2023, pp. 14 935–14 956.
[24] J. Wei and et al., "Chain-of-thought prompting elicits reasoning in large language models," in *NeurIPS*, 2022.
[25] F. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR*, 2020.
[26] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," in *NeurIPS Datasets and Benchmarks*, 2021.
[27] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *ACM SIGKDD*, 2022, pp. 594–604.