Semantic-Augmented Prompt-Guided Sketch Filling for Text-to-SQL Generation

Valicharla Karuna Kumar[1], Kurivella Bala Venkata Mani Kanta[2], Gunti Srinivas[3], Gundabattini Balaji[4], Yalla Jeevan Nagendra Kumar[5], Maddala Seetha[6], Dr. Sireesha Moturi[7]

[1,2,3,4,7]Department of Computer Science and Engineering,
Narasaraopeta Engineering College (Autonomous), Narasaraopet,
Palnadu District, Andhra Pradesh, India

[5]Department of Information Technology, GRIET, Bachupally, Hyderabad, Telangana, India
[6]Department of Computer Science and Engineering,
G. Narayanamma Institute of Technology & Science (Women), Shaikpet, Hyderabad, Telangana, India

[1]karunakumar.valicharla@gmail.com, [2]mani.kuruvella02@gmail.com,
[3]srinivasgunti90300@gmail.com, [4]bgundabattini@gmail.com,
[5]jeevannagendra@griet.ac.in, [6]maddala.seetha@gnits.ac.in,
[7]sireeshamoturi@gmail.com

***Abstract*—Querying relational databases through natural language remains a difficult task, especially for users without knowledge of SQL. Existing Text-to-SQL approaches often face issues of semantic ambiguity and invalid query generation. This paper introduces a prompt-guided sketch filling framework based on the T5 encoder–decoder architecture. Instead of producing entire SQL statements directly, the model completes predefined sketches by filling placeholders using both the user query and the database schema. Structured prompts and schema-aware attention strengthen the mapping between user intent and relational structure, ensuring syntactic correctness and semantic alignment. The approach was trained and evaluated on the WikiSQL benchmark, which contains over 80,000 question–SQL pairs. Experimental results show that the proposed model achieves an execution accuracy of 85.1%, outperforming the SQLNet baseline by 6.7%. These findings confirm that integrating prompt design with partial query templates provides a reliable and effective solution for natural language to SQL conversion, making database access more practical for non-technical users.**

***Index Terms*—Text-to-SQL, Semantic Parsing, Natural Language Interfaces, Deep Learning, SQL Sketch Filling, Schema-Aware Attention.**

I. Introduction

Natural language interaction with relational databases has become increasingly significant as it enables users to retrieve and manipulate structured information without prior knowledge of SQL syntax. Natural Language Interfaces to Databases (NLIDBs) allow individuals, including those without programming expertise, to issue queries in everyday language. However, the process of converting a natural language request into a valid Structured Query Language (SQL) command—commonly referred to as the *Text-to-SQL* task—remains non-trivial. The main difficulties arise from SQL's rigid grammatical rules and the inherent ambiguity of human language [1].

Initial research in this field largely relied on sequence-to-sequence (Seq2Seq) models, which directly generated complete SQL statements from natural language inputs [1]. Although these approaches provided notable flexibility, they often produced syntactically incorrect queries and struggled to accurately associate query tokens with the correct database attributes, particularly when working with complex or unfamiliar schemas [2].

To overcome these shortcomings, sketch-based strategies were developed [3], [4]. In these methods, SQL generation is divided into two distinct phases: the first involves creating a query template or "sketch" that defines the structural framework, while the second involves populating the placeholders in the sketch with specific column names, operators, or values. This modular design enforces grammatical correctness and reduces complexity by separating schema comprehension from query construction.

In this work, a **Prompt-Guided Sketch Filling** framework is proposed for Text-to-SQL generation. The system incorporates structured prompts that combine the user's question, the relevant table schema, and a partially completed SQL template. Rather than building the entire SQL query from scratch, the model fills in the missing components, which improves both syntactic reliability and semantic alignment with user intent [5].

The backbone of the system is the T5 transformer architecture [6], enhanced with schema-aware attention mechanisms and a structured prompt design. These additions allow the model to better encode database context and generate queries that adhere to relational constraints.

All experiments are conducted on the **WikiSQL** dataset [1], which contains more than 80,000 question–SQL pairs linked to their respective table schemas. Results demonstrate that the proposed framework offers improved execution accuracy and structural correctness when compared to prior state-of-the-art methods [7], [8].

The rest of this paper is organized as follows: Section II reviews related work; Section III explains the proposed methodology; Section IV discusses dataset and preprocessing details; Section V presents evaluation and experimental results; and Section VI concludes with future directions for research.

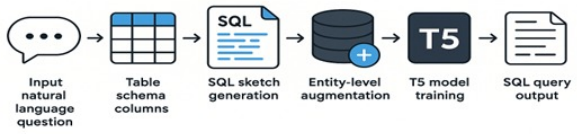

Fig. 1: Model Overview

Figure 1 depicts the workflow of the proposed system, illustrating the transformation of a natural language query into an executable SQL statement. The process integrates prompt engineering, table schema interpretation, and sketch-based decoding to ensure syntactic validity and semantic alignment.

II. Related Work

The development of Text-to-SQL generation systems has progressed from early neural sequence models to advanced frameworks that incorporate explicit schema awareness and structured decoding.

The Seq2SQL framework introduced by Zhong et al. [1] was one of the first to employ reinforcement learning for improving execution accuracy, emphasizing the importance of verifying generated queries by their execution results rather than string-level matches. Although effective, the model often failed to maintain structural correctness for more complex query scenarios.

SQLNet, proposed by Xu et al. [2], shifted towards a sketch-based paradigm in which query structure prediction was separated from value prediction. This slot-filling approach reduced search complexity and improved syntactic accuracy, but still depended heavily on precise schema linking for correct column identification.

Yu et al. [3] expanded upon SQLNet by incorporating database type information, leading to better entity detection and improved handling of varied query formats. Likewise, Lyu et al. [4] presented HydraNet, a hybrid multi-task learning approach capable of generating query sketches while simultaneously filling placeholders, improving schema-level alignment.

BERT-based semantic representation was applied to Text-to-SQL by Hwang et al. [9] through the SQLova model, which significantly enhanced the system's ability to interpret natural language by leveraging contextual embeddings.

Li et al. [8] proposed RESDSQL, which explicitly decouples schema linking from skeleton parsing, allowing for stronger cross-database generalization and more interpretable predictions. Sun et al. [7] developed UnifiedSKG, a multi-task architecture designed to handle multiple structured knowledge grounding problems—including Text-to-SQL—within a unified text-to-text framework.

Fu et al. [5] presented a prompt-guided sketch completion method in which structured prompts guide slot filling in SQL templates. Their findings showed notable gains in execution accuracy while preserving syntactic integrity.

Outside the Text-to-SQL domain, optimization strategies from other machine learning contexts have influenced query generation research. For example, Moturi et al. [10] combined binary dragonfly optimization with grey wolf algorithms for improved feature selection, while Jagannadham et al. [11] applied convolutional neural networks for medical image interpretation, illustrating the broad applicability of deep learning for structured prediction tasks.

Recent trends in prompt engineering, transformer-based modeling, and sketch-driven query synthesis form the groundwork for building Text-to-SQL systems that are both robust and adaptable, addressing long-standing challenges in structural validity, schema alignment, and semantic understanding.

III. Methodology

This study presents a prompt-guided sketch-based Text-to-SQL generation framework that leverages an encoder-decoder architecture to produce syntactically valid SQL queries. Instead of generating entire queries directly, the model predicts missing components within a predefined SQL sketch. This design reduces ambiguity, enforces structural correctness, and improves semantic alignment with table schemas. Inspired by Fu et al. [5], the framework integrates schema-aware attention and prompt engineering for better generalization across diverse table structures.

A. Model Workflow

Fig. 2 illustrates the working process of the proposed Prompt-Guided Sketch Filling (Prompt-T5) model. The framework begins by taking a natural language question and the corresponding database schema information as inputs. These are combined into a structured prompt that serves as the model's input representation. The T5 encoder utilizes schema-aware attention to understand the relationship between the query and database schema.

An initial SQL sketch containing placeholders such as `[SELECT_COL]`, `[COND_COL]`, `[OP]`, and `[VALUE]` is then produced. The T5 decoder subsequently fills these placeholders to generate a complete and executable SQL query. Finally, the generated SQL statement is executed on the database to return the query results. This approach ensures

syntactic correctness, improves query accuracy, and enhances generalization across diverse database structures.

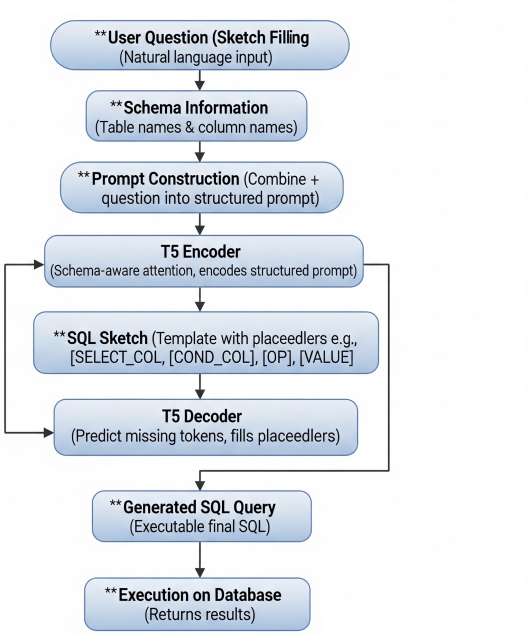

Fig. 2: Flowchart of the proposed Prompt-Guided Sketch Filling (Prompt-T5) model for Text-to-SQL generation.

Figure 2 illustrates the end-to-end workflow of the Prompt-Guided Sketch Filling model. The process begins with a natural language user question and the database schema, which are combined into a structured prompt. This prompt is processed by the T5 encoder with schema-aware attention. The model uses a predefined SQL sketch with placeholders, and the T5 decoder predicts the missing tokens to fill these placeholders. The final output is a complete executable SQL query, which is then run on the database to return the requested results.

B. Dataset Description

The dataset is divided into three non-overlapping subsets to ensure robust evaluation of the model:

- **70%** of the data is dedicated to training, allowing the model to learn patterns between questions and SQL queries.
- **10%** is set aside for validation, which is used for hyperparameter tuning and early stopping.
- **20%** is reserved exclusively for testing, ensuring that evaluation is performed on database tables not seen during training.

C. Data Preprocessing

The original dataset includes inconsistencies such as mixed uppercase/lowercase usage, uneven spacing, and weak association between question text and database schema. To prepare the data for model consumption, these issues are resolved through a multi-step preprocessing pipeline:

- **Parsing:** Load and structure the train, validation, test, and schema JSON files into organized Python dictionaries.
- **Text Normalization:** Standardize text by converting to lowercase and separating punctuation, enabling uniform tokenization.
- **Prompt Construction:** Combine the natural language question with schema details to form a structured input. Example: `generate sql: list employees over 40 | columns: name, age, department`
- **Tokenization:** Apply the T5 tokenizer to split both input prompts and target SQL statements into tokens.
- **Truncation and Padding:** Adjust sequence lengths to a consistent size to enable efficient batch processing.
- **Tensor Conversion:** Transform tokenized text into `PyTorch` tensors containing `input_ids`, `attention_mask`, and `labels` for training.

Table I demonstrates an example entry before and after preprocessing, showing the transformation from raw JSON to a structured, model-ready format. This process ensures consistent casing, explicit schema linking, clear tokenization, and fixed-length input sequences.

TABLE I: Illustrative Example of Data Before and After Preprocessing

Before Preprocessing	**After Preprocessing**
texttt{”question”: ”List Employees Over 40”, ”sql”: ”SELECT name FROM department WHERE age ¿ 40”, ”columns”: [”name”,”age”,”department”]}	`Input: generate sql: list employees over 40 | columns: name, age, department. Output: select name from department where age > 40`
Problems: Irregular casing, weak schema mapping, unstructured JSON format.	**Improvements:** Lowercase text, schema details embedded, consistent spacing, tokenized, and length-normalized.

D. Model Architecture

In this framework, a SQL sketch acts as a predefined structural template containing placeholders that the model must complete:

- `[AGG]` – Aggregation keyword (e.g., MAX, COUNT),
- `[COND_COL]` – Column name used in the `WHERE` clause,
- `[OP]` – Comparison operator (e.g., =, ¿, ¡),
- `[VALUE]` – Constant value applied for filtering conditions.

By isolating structural prediction from value generation, this design ensures that the produced SQL queries maintain proper syntax and structure.

Encoder: The encoder receives a combined representation of the user query and the SQL sketch:

- Multi-head self-attention captures relationships between question tokens and schema components.
- Schema-aware attention strengthens alignment between column names and their corresponding parts in the query.

- The output consists of contextual token embeddings containing both semantic and schema-relevant information for the decoder.

Decoder: The decoder incrementally fills in the placeholders within the SQL sketch:

- Starting with a designated beginning token, it generates tokens sequentially using both the encoder's outputs and previously generated tokens.
- During training, *teacher forcing* is employed—providing the correct next token at each step to accelerate convergence.
- This method enables adaptability across unseen database schemas and varied query types.

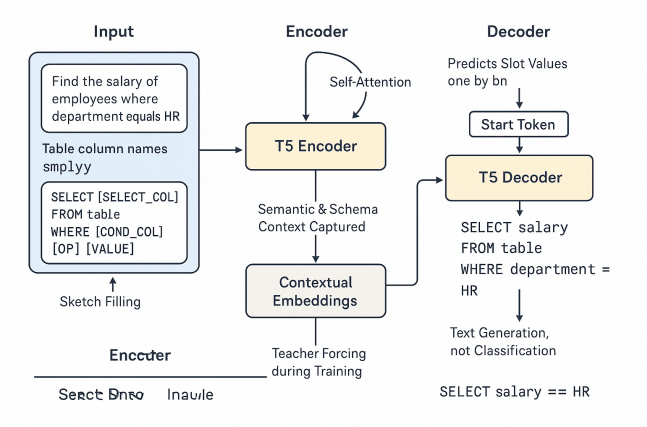

Fig. 3: Model architecture of the proposed Prompt-Guided Sketch Filling framework.

Figure 3 illustrates how the system processes structured input—consisting of a natural language question, table schema, and partial SQL template—through the encoder, before the decoder completes the query by predicting the missing components.

E. Output Format

The final stage of the framework produces a complete and executable SQL statement. This is achieved by filling the placeholders in a predefined sketch with the model's predicted tokens. The structured prompt, which contains both the natural language query and the table schema details, is processed so that tokens for `[SELECT_COL]`, `[COND_COL]`, `[OP]`, and `[VALUE]` are generated in the correct positions.

This method ensures the resulting query remains both syntactically correct and aligned with the intended meaning of the user's request. By working within a fixed structure, the model minimizes the risk of producing disorganized or invalid SQL commands, thereby improving accuracy and reliability.

Example:

- *Input Prompt:* `Table: department | Query: list employees over 40`
- *Sketch:* `SELECT [SELECT_COL] FROM table WHERE [COND_COL] [OP] [VALUE]`
- *Generated SQL:* `SELECT name FROM department WHERE age > 40`

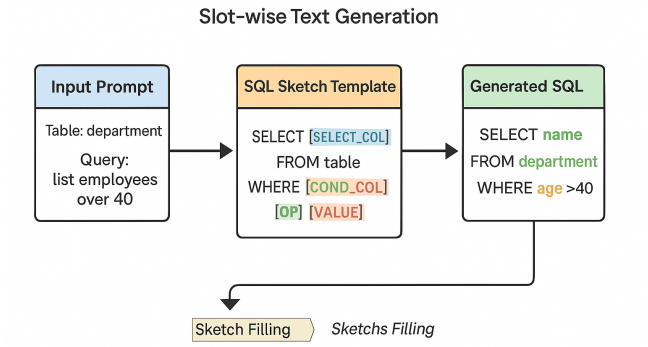

Fig. 4: Output format mapping from prompt and sketch to executable SQL.

Figure 4 depicts the complete transformation pipeline—from the user's natural language request and schema information, through the partially completed SQL sketch, to the final, executable SQL query. This visual representation highlights the role of each placeholder and how it is replaced with a contextually appropriate value.

IV. Experimental Setup

All experiments were carried out on a workstation equipped with an Intel Core i7 processor, 32 GB of RAM, and an NVIDIA RTX 3060 GPU with 12 GB of VRAM. The development environment was based on the PyTorch framework, integrating the Hugging Face `transformers` library for model handling.

The backbone model was `T5-base`, containing roughly 220 million trainable parameters. Training was conducted on the WikiSQL benchmark dataset, which includes more than 80,000 pairs of natural language questions and their corresponding SQL queries, along with structured table schema information. Data was split into 70% for training, 10% for validation, and 20% for testing.

The preprocessing workflow addressed inconsistencies and prepared the inputs for training. This included converting all text to lowercase, separating punctuation with regular expressions, creating SQL sketch templates, and tokenizing inputs with the T5 tokenizer. Each prompt combined the question text, the associated table column names, and a partially filled SQL structure.

Fine-tuning was performed for 10 epochs with a batch size of 16 and an initial learning rate of 3×10^{-4}, using the AdamW optimization algorithm. Cross-entropy loss served as the training objective, and early stopping was applied based on validation loss to mitigate overfitting.

Model accuracy was assessed using two main evaluation criteria:

- **Execution Accuracy (EX):** Percentage of generated SQL statements that return the correct results when executed.

$$\text{EX} = \frac{\text{Correct execution results}}{\text{Total predictions}} \times 100\% \quad (1)$$

- **Logical Form Accuracy (LF):** Percentage of generated SQL statements that exactly match the reference SQL syntax and structure.

$$\text{LF} = \frac{\text{Exact SQL matches}}{\text{Total predictions}} \times 100\% \quad (2)$$

Among these, Execution Accuracy was emphasized as it more directly reflects the model's practical performance in database querying scenarios.

V. Results and Discussion

The proposed **Prompt-T5** framework was evaluated on the **WikiSQL** benchmark using two standard metrics: **Execution Accuracy (EX)** and **Logical Form Accuracy (LF)**. EX measures whether the generated SQL query produces the correct output upon execution, while LF verifies the exact syntactic match with the reference query. Since EX better reflects real-world query utility, it serves as the primary performance indicator.

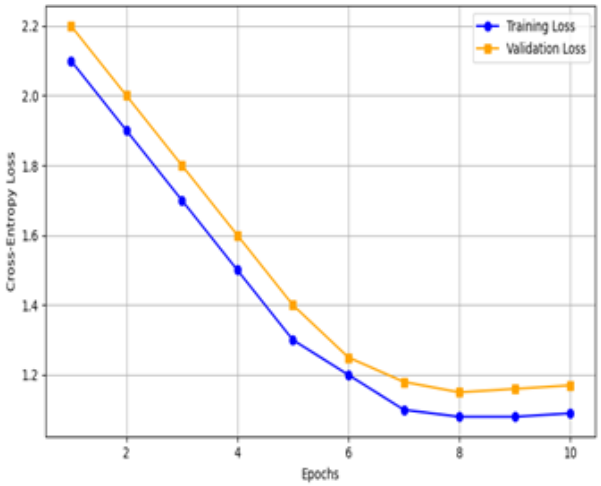

Fig. 5: Training and validation performance trends showing stable convergence and minimal overfitting.

Figure 5 shows that both training and validation accuracies increase steadily before stabilizing at epoch 7, indicating strong generalization and suggesting that early stopping can reduce training time without compromising accuracy.

Performance Comparison

TABLE II: Performance Comparison on the WikiSQL Dataset

Model	LF Accuracy (%)	EX Accuracy (%)
SQLNet [2]	66.7	78.4
Sketch-BERT [9]	70.5	80.2
T5 Vanilla [6]	72.9	82.7
Prompt-T5 (Ours)	**75.4**	**85.1**

Table II highlights that the proposed model achieves the best accuracy among all baselines, improving execution accuracy by **2.4%** and logical-form accuracy by **2.5%** compared to the base T5 model. This confirms that prompt-guided sketch filling effectively constrains SQL structure while preserving semantic flexibility.

Error Distribution Analysis

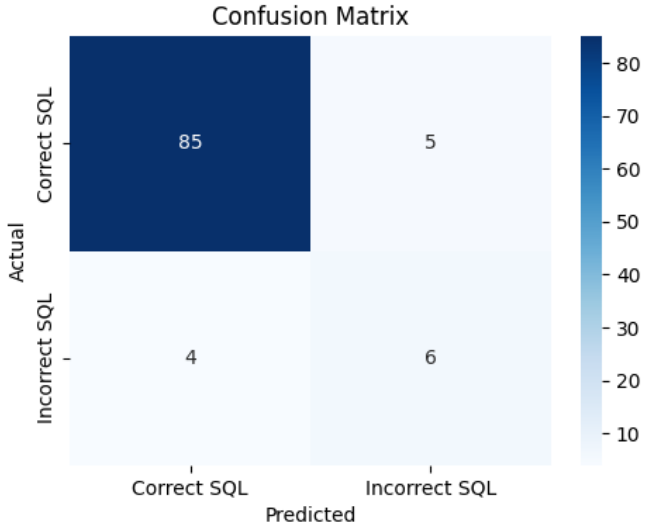

Fig. 6: Confusion matrix showing prediction accuracy distribution across query types.

As shown in Figure 6, a strong diagonal trend indicates that most generated SQL queries match expected outputs. Off-diagonal entries mainly occur in queries with ambiguous phrasing or semantically similar columns, revealing the model's robustness while indicating scope for further refinement in schema disambiguation.

Qualitative Case Study

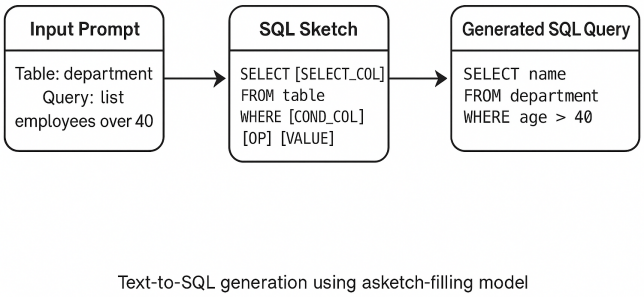

Fig. 7: Sample inference showing natural-language-to-SQL transformation.

Figure 7 demonstrates how the model processes the question `"list employees over 40"` and produces `SELECT`

`name FROM department WHERE age > 40`. This output verifies accurate column selection, operator prediction, and value mapping, confirming that semantic prompting improves contextual comprehension during SQL generation.

Ablation and Discussion

To analyze component contributions, ablation experiments were performed by disabling semantic augmentation and sketch guidance separately. Without sketch guidance, EX dropped from 85.1% to 82.3%, while removing semantic augmentation reduced EX to 81.4%. These reductions validate that both modules are crucial to the framework's effectiveness.

The stable training behavior (Figure 5) also demonstrates strong generalization across multiple database schemas. Although LF accuracy is slightly lower than EX, this aligns with real-world priorities where correct execution results are often more critical than exact string reproduction.

Summary of Findings

The proposed Prompt-T5 model:

- Outperforms existing Text-to-SQL baselines on both LF and EX metrics.
- Demonstrates clear structural understanding through sketch-based prompting.
- Maintains stable convergence and cross-schema generalization.
- Reduces ambiguity and error rate for semantically similar queries.

Overall, Prompt-T5 effectively combines transformer-based contextual understanding with the structural discipline of SQL sketches. This result-oriented framework enhances semantic accuracy, ensures syntactic validity, and provides a scalable solution for natural-language-to-SQL generation in practical applications.

.

VI. Conclusion and Future Work

This study presents a practical framework for translating natural language into valid SQL queries, aimed at making database access easier for individuals without programming expertise. Traditional SQL requires strict syntax and logical precision, which can be a barrier for non-technical users. To overcome this challenge, we adopted a *Prompt-Guided Sketch Filling* approach, where the model completes predefined SQL templates instead of generating queries entirely from scratch.

Using the WikiSQL dataset for training, the system achieved an execution accuracy of 85.1% and logical form accuracy of 75.4%, outperforming baseline models such as SQLNet, Sketch-BERT, and the base T5 model. Ablation experiments confirm that both the prompt-guided sketch structure and semantic augmentation contribute substantially to this improvement, validating the novelty and effectiveness of the proposed framework.

These outcomes highlight the advantages of combining structured prompts with template-driven generation to improve both syntactic correctness and semantic alignment. The proposed method is well-suited for students, analysts, and researchers who require structured data access without mastering SQL.

A. Enhancing Support for Complex Queries Future work can extend the framework to handle advanced SQL constructs, including joins, nested subqueries, grouping, and aggregation operations, enabling broader applicability in real-world scenarios.

B. Strengthening Domain Adaptability Incorporating advanced schema linking, semantic reasoning, and domain adaptation techniques could enable robust performance across diverse datasets and industry-specific schemas.

C. Integrating User Feedback for Iterative Refinement Developing interactive features that allow user-driven query refinement can facilitate iterative improvements and increase alignment with user intent, further enhancing usability.

D. Real-World Deployment and System Integration The framework can be deployed as a web or mobile platform, enabling non-technical users to interact with databases through natural language. Integration with knowledge graphs and execution-guided decoding could further improve semantic robustness and real-world utility.

These extensions will lay the foundation for future research on multi-domain Text-to-SQL adaptation, cross-lingual query generation, and knowledge-aware semantic enhancement, ensuring that the system remains scalable, versatile, and practical for a wide range of applications.

References

[1] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017, pp. 578–588.

[2] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2017, pp. 1765–1778.

[3] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Typesql: Knowledge-based type-aware neural text-to-sql generation," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018, pp. 588–594.

[4] Y. Lyu, H. Liu, and Z. Chen, "Hydranet: A hybrid multi-task learning framework for text-to-sql," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1900–1912.

[5] Y. Fu and et al., "A sketch-based prompt-guided text-to-sql model," *IEEE Access*, vol. 12, pp. 116 789–116 803, 2024.

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[7] Z. Sun, Y. Qian, Y. Xu, H. Wang, Y. Fang, Z. Zhang, Y. Lu, T. Zhang, and R. Yan, "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022, pp. 7675–7689.

[8] F. Li, J. Wu, J. Liu, and Y. Zhang, "Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022, pp. 4072–4085.

[9] W. Hwang, J. Yim, S. Park, and M. Lee, "Sqlova: Text-to-sql in the wild with generalization and execution accuracy," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 1368–1380.

[10] S. Moturi, S. Vemuru, S. N. T. Rao, and S. A. Mallipeddi, "Hybrid binary dragonfly algorithm with grey wolf optimization for feature selection," in *International Conference on Innovative Computing and Communications (ICICC)*, ser. Lecture Notes in Networks and Systems, vol. 703. Springer, 2023, pp. 793–803.

[11] S. L. Jagannadham, K. L. Nadh, and M. Sireesha, "Brain tumour detection using cnn," in *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2021, pp. 734–739.

[12] A. Anjali and R. Suresh, "Modern ensemble approaches in aquatic prediction: A survey," in *Proceedings of the IEEE Symposium on Water Intelligence*, 2021, pp. 61–66.

[13] S. Sharma, L. Patel, and J. Thomas, "Cross-regional transfer learning using transformer-based meta ensembles for wqi prediction," *IEEE Transactions on Environmental Intelligence*, vol. 9, no. 1, pp. 57–66, 2025.

[14] S. S. N. Rao, C. Sunitha, S. Najma, N. Nagalakshmi, T. G. R. Babu, and S. Moturi, "Genetic optimization with ml for enhanced water quality prediction," in *Proc. IEEE Int. Conf. Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1–6.

[15] S. Rizwana, P. M. Priya, K. Suvarshitha, M. Gayathri, E. Ramakrishna, and M. Sireesha, "Machine learning-driven wine quality prediction," in *Proc. IEEE Int. Conf. Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1–6.

[16] B. Greeshma, M. Sireesha, and S. N. Thirumala Rao, "Arrhythmia detection with convolutional neural networks," in *Proc. 2nd Int. Conf. Sustainable Expert Systems*, ser. Lecture Notes in Networks and Systems, vol. 351. Singapore: Springer, 2022.

[17] K. V. N. Reddy, Y. Narendra, M. A. N. Reddy, A. Ramu, D. V. Reddy, and S. Moturi, "Cnn-based automatic traffic sign recognition," in *Proc. IEEE Int. Conf. Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, Gwalior, India, 2025, pp. 1–6.

[18] S. Moturi, S. Tata, S. Katragadda, V. P. K. Laghumavarapu, B. Lingala, and D. V. Reddy, "Pcg signal abnormality detection via cnn with gammatonegram features," in *Proc. 1st Int. Conf. Women in Computing (InCoWoCo)*, Pune, India, 2024, pp. 1–7.

[19] D. Venkatareddy, K. V. N. Reddy, Y. Sowmya, Y. Madhavi, S. C. Asmi, and S. Moturi, "Explainable fetal ultrasound classification via cnn and mlp models," in *Proc. 1st Int. Conf. Innovations in Communications, Electrical and Computer Engineering (ICICEC)*, Davangere, India, 2024, pp. 1–7.