

SQL-to-Text Generation with Weighted-AST Few-Shot Prompting

Sriom Chakrabarti¹, Chuangtao Ma¹, Arijit Khan^{2,1}, and Sebastian Link³

¹ Aalborg University, Denmark
{scha, chuma}@cs.aau.dk

² Bowling Green State University, USA
arijitk@bgsu.edu

³ University of Auckland, New Zealand
s.link@auckland.ac.nz

Abstract. SQL-to-Text generation aims at translating structured SQL queries into natural language descriptions, thereby facilitating comprehension of complex database operations for non-technical users. Although large language models (LLMs) have recently demonstrated promising results, current methods often fail to maintain the exact semantics of SQL queries, particularly when there are multiple possible correct phrasings. To address this problem, our work proposes Weighted-AST retrieval with prompting, an architecture that integrates structural query representations and LLM prompting. This method retrieves semantically relevant examples as few-shot prompts using a similarity metric based on an Abstract Syntax Tree (AST) with learned feature weights. Our structure-aware prompting technique ensures that generated descriptions are both fluent and faithful to the original query logic. Numerous experiments on three benchmark datasets – Spider, SPaRC, and CoSQL show that our method outperforms the current baselines by up to +17.24% in Execution Accuracy (EX) performs superior in Exact Match (EM) and provides more consistent semantic fidelity when evaluated by humans, all while preserving competitive runtime performance. These results demonstrate that Weighted-AST prompting is a scalable and effective method for deriving natural language explanations from structured database queries. **We have made our datasets and code available at [6].**

Keywords: SQL-to-Text · Semantic Captioning · Human Evaluation

1 Introduction

Relational databases and SQL form the backbone of modern data management by enabling scalable and reliable storage and querying of structured data. These capabilities support reproducible analysis and robust system design. However, writing effective SQL requires detailed schema knowledge, set-oriented reasoning across multi-table joins, and an understanding of syntax variation, which are barriers for many users. Recent progress in large language models (LLMs) has demonstrated exceptional performance in semantic parsing (i.e., Text-to-SQL

conversion), enabling non-technical users to query databases seamlessly [10]. Though there has been considerable research on Text-to-SQL, its counterpart, SQL-to-Text generation, has not received much attention, despite its significance.

SQL-to-Text translates SQL queries into natural language, in contrast to Text-to-SQL, which generates executable queries from natural language. Several real-world applications, such as code development, integration, debugging, data analysis, and education, depend on SQL-to-text functionality [1,3,5,20,13]. With the growing popularity of LLMs, business intelligence tools, and collaborative analytics workflows, an increasing number of SQL queries are generated automatically rather than manually. Describing queries in natural language facilitates intent verification, troubleshooting complex logic, and communicating results to non-technical stakeholders. Transparency, interpretability, and confidence in data-driven systems is all made possible via SQL-to-Text, as users must understand the semantics and intent of these queries before they can trust the results. In addition, SQL-to-Text can also be viewed as a specific instance of automated code summarization, producing concise descriptions of code artifacts in natural language [9,25]. Recent research shows that SQL-to-Text is useful for testing Text-to-SQL’s robustness under schema-aligned paraphrasing, where drops in execution accuracy uncover the need for precise, transparent query descriptions [23]. Thus, SQL-to-Text serves both as a resource for users and as a vital component of a thorough evaluation.

Despite its importance, semantic captioning (SQL-to-Text) has been under-explored compared to its inverse task, Text-to-SQL. Previous work [27] translates SQL to text by using sequence-to-sequence models utilizing graph neural network encoders, demonstrating the importance of understanding query structure. Recently, researchers have investigated large language model (LLM) based methods for SQL-to-Text translation. The growing body of work on using LLMs for translating natural language to SQL highlights the mutual benefits of bidirectional translation between SQL and text. To improve the performance of small LLMs on the semantic captioning task, the prompting techniques and a graph-based few-shot method (AST-ICL) has been explored [1]. The standard automated metrics, i.e., BLEU-4, BERTScore, and AlignScore, were selected as the metrics to evaluate the generated translations on the SQL-to-Text captioning benchmark. However, although these metrics are widely used in machine translation and summarization, they do not adequately capture semantic correctness in SQL-to-Text translation.

Problem Statement: It is a common phenomenon that a single SQL query in SQL-to-Text can provide numerous linguistically different but conceptually comparable natural language descriptions. For instance, a query can be phrased in many different ways that all share the same meaning. However, existing metrics fail to handle one-to-many mappings effectively. In some cases, these metrics give a high score to an incorrect translation if there is unintentional lexical overlap, even when the meaning is incorrect. In other cases, the commonly used embedding-based metrics, such as BERTScore and AlignScore, can improve semantic matching to some extent but still fail to capture logical correctness.

Accordingly, there are two major challenges and limitations of the existing work on SQL-to-Text.

(i) **One-to-many ambiguity:** A fixed reference set is ultimately insufficient because there are several accurate translations for a single SQL query. Lexical metrics may underestimate an accurate description produced by a model that only uses different terminology or synonyms since it does not match the reference phrase (false negatives). On the other hand, an inaccurate description may accidentally overlap with reference words, leading to an inflated score (false positives). To put it another way, n -gram overlap and a lack of references make it difficult to fully capture the range of acceptable responses, which results in an inaccurate evaluation of a model’s actual capabilities or techniques.

(ii) **Semantic insensitivity:** Lexical overlap (BLEU) and embedding-based similarity (BERTScore) does not guarantee that the generated text preserves the semantic meaning of the SQL query. These metrics focus on syntax rather than semantics. For example, consider an SQL condition WHERE amount > 100. A correct English translation might say “retrieve all records where the amount is greater than 100,” but a generated translation could mistakenly say “amount is less than 100.” This subtle one-word difference flips the query’s meaning. Yet, because the rest of the sentence overlaps with the reference, a metric like BLEU may still assign a high score, and even BERTScore might not fully penalize the inversion. These instances demonstrate how minor omissions or logical mistakes that significantly alter the query’s intent may be overlooked by current metrics. According to Renggli et al. [21], using aggregated metrics for the SQL-to-Text task can overestimate or underestimate actual performance.

Motivation & Contribution: To address the above challenges and limitations, we propose a retrieval-based prompting technique that leverages AST (abstract syntax tree¹)-weighted similarity to generate high-quality few-shot examples combined with a prompt to generate natural language descriptions of SQL queries. Accordingly, our main contributions are summarized as follows:

(i) **Critical Evaluation of SQL-to-Text Metrics:** We provide an in-depth human examination of current SQL-to-Text evaluation procedures and highlight their limitations, demonstrating that high BLEU, BERTScore, and AlignScore frequently evaluate semantic accuracy incorrectly, resulting in a high number of false positives and negatives.

(ii) **Protocol for evaluating semantic reliability:** We propose a human-centered evaluation methodology for SQL-to-Text that emphasizes semantic correctness, utilizing human judgment with a round-trip consistency test using a popular Text-to-SQL model to compare accuracy and execution.

(iii) **Weighted AST-based prompting for SQL-to-Text:** We develop a novel few-shot prompting technique combined with the Weighted-AST approach, which finds similar examples using AST feature importance, resulting in a more accurate and semantically meaningful SQL-to-Text translations.

To the best of our knowledge, our work is the first to apply round-trip consistency evaluation for an SQL-to-Text task. Human evaluators found that every

¹ https://en.wikipedia.org/wiki/Abstract_syntax_tree

time they attempted to reconstruct the original SQL, it was associated with a description that was incomplete or erroneous. In contrast, we offer a semantically meaningful methodology for assessing SQL-to-Text, integrating human experts in the loop. Our methodology raises the bar further, based on the existing benchmark dataset, by specifically evaluating whether the generated description retains all of the original SQL’s semantics, providing a more accurate and meaningful evaluation of model performance.

2 Related Work

2.1 Semantic Parsing (Text-to-SQL)

The process of translating natural language questions into SQL queries has a long history in natural language processing (NLP), with numerous benchmarks (such as CoSQL, Sparc, and Spider) and solutions ranging from neural sequence models and grammar-based techniques to the most recent LLM-based methods [15,8,35]. Modern Text-to-SQL models improve accuracy on complex databases using methods like execution-guided decoding, copy mechanisms, and schema encoding. Since equivalent SQL queries can look different, string matching can be misleading, so execution accuracy is the preferred evaluation metric. Latest works [21,19] highlight challenges in evaluating Text-to-SQL, such as ambiguous language and unreliable matching metrics.

CHASE-SQL [18] is a multi-agent, chain-of-thought framework that improves stability on complex queries, while SeLECT-SQL [24] utilizes self-correction and ensemble chain-of-thought prompting. Open-source projects such as CodeS [16] train models from 1B to 15B parameters specifically designed for Text-to-SQL, achieving robustness on BIRD and Spider datasets with outstanding performance and generalization. Other recent innovations MCTS-SQL [34] uses Monte-Carlo Tree Search with heuristic refinement to outperform the previous SOTA on Spider and BIRD, and Arctic-Text2SQL-R1 [30] use reinforcement learning rewards for top accuracy. Together, they advance candidate generation, self-consistency, and scalable open-source performance.

2.2 Semantic Captioning (SQL-to-Text)

Early work address the SQL-to-Text problem with sequence-to-sequence (Seq2Seq) neural networks [11] and template mechanism [7,14]. Graph2seq [28] presents a graph-to-sequence encoder that outperforms a simple seq2seq model by integrating the SQL structure through graph neural networks. The lack of reliable and high-quality datasets are a major obstacle in SQL-to-Text research because SQL queries can naturally correspond to several valid natural language interpretations. To fill this gap, Al-Lawati et al. [1] created the first benchmark for SQL-to-Text and explicitly define the SQL semantic captioning challenge. They also make one of the first attempts to handle SQL-to-Text using a graph-aware in-context learning (ICL) technique. They initially parse SQL queries into abstract

syntax trees (ASTs) and then use a graph neural network (GNN) to create graph embeddings rather than training a conventional sequence-to-sequence model. By calculating the cosine similarity between these graph embeddings, they can recover the most similar SQL queries at inference time. This technique shows how to use SQL’s built-in structural features to improve the demonstration selection, which results in notable gains over text-based or random retrieval techniques like BM25. EzSQL [4] introduces an intermediate representation focused on SQL-to-Text generation. EzSQL restructures SQL queries by replacing operators and keywords with natural language-aligned tokens and removing the set operators, thereby assisting alignment between SQL and narrative text. This intermediate version provides state-of-the-art results on the WikiSQL and Spider datasets and integrates more efficiently with pre-trained generative models. Furthermore, the model produces synthetic Text-to-SQL pretraining data, which improves semantic parsing tasks downstream. SQLucid [26], instead of only producing descriptions, creates an interactive user interface that provides visual alignment, stepwise translation results, and editable natural language explanations of SQL queries. Two user studies show that compared to the current NLIDB interfaces, SQLucid significantly improves task accuracy and user confidence.

3 Methodology

To improve the quality of generated descriptions, we propose a weighted AST-based retrieval with a few-shot prompt for the SQL-to-Text generation. The overview of the proposed approach is demonstrated in Figure 1. Our approach consists of three main steps: (1) Feature extraction and weighting, (2) Top- k examples retrieval and (3) Chain-of-Thought (CoT) prompting strategy. We train on SQL-only data from the training splits of the original Spider [32], SParC [33], and CoSQL [31] benchmarks. Evaluation is conducted on the SQL-to-Text test sets—Spider-S2T, SParC-S2T, and CoSQL-S2T [1]. No S2T data (or natural-language references) is used during training; the multiple references in S2T are used only for scoring. We follow the official splits to ensure no train-test leakage.

3.1 Feature Extraction and Weighting

An Abstract Syntax Tree (AST) represents the syntactic structure of an SQL query by mapping nodes and establishing parent-child relationships in the form of a tree structure. By capturing nested subqueries, function calls, and identifier scopes, the AST preserves a query’s hierarchical structure, enabling direct structural comparisons between queries.

Feature Extraction. Our approach extracts two categories of features. Each query Q is represented as a bag of *features* with counts $c_Q(f)$.

(1) SQL features: Each SQL query in the training set is processed to produce a set of surface features. In order to find single- and multi-word SQL keywords, the extraction algorithm scans neighboring tokens and combines surface-level information such as keywords, aggregation functions, and table names.

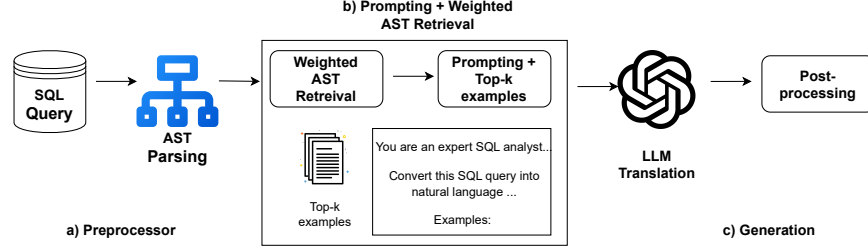


Fig. 1: Overall architecture of the proposed SQL-to-Text generation framework. **a) Preprocessing:** The input SQL query undergoes a series of preprocessing steps to normalize its structure and simplify complex constructs. **b) Weighted AST Retrieval with Few-Shot Prompting:** Our main contribution introduces a *Weighted Abstract Syntax Tree (AST) Retrieval* mechanism that retrieves the most semantically relevant examples based on a weighted similarity score. These top- k examples are seamlessly integrated into the LLM’s prompt, allowing the model to better capture the intent of the SQL query and produce more precise and fluent translations. **c) Generation:** After preprocessing and retrieval, the structured prompt is passed to the LLM, which generates a high-quality natural language description of the SQL query, ensuring improved semantic fidelity and readability.

(2) AST-based structural features: In addition to surface features, a token-level abstract syntax tree (AST) is constructed using `sqlparse` [2]. Traversing the tree produces a concise set of structural features: (i) node-type features (TYPE:Statement, TYPE:Identifier, TYPE:Comparison); (ii) keyword features within their syntactic context; (iii) function features (FUNCTION:AVG, FUNCTION:COUNT); (iv) identifier features (IDENTIFIER:singer, IDENTIFIER:age); and (v) depth features (e.g., node depth distribution, maximum tree depth, depth of specific node types) that capture the hierarchical level of each node in the tree structure. This feature set captures relationships and semantic constructs that are not apparent from the surface form.

Feature Weighting Mechanism. Not all SQL features contribute equally to semantic similarity between two queries. Both *direct SQL features* and *AST-derived structural features* are weighted using a unified mechanism that integrates two complementary measures of importance.

First, **Inverse Document Frequency (IDF)** captures the global informativeness of a feature. It evaluates a feature’s overall informativeness throughout the corpus by down-weighting common elements such as `SELECT` and `FROM` while prioritizing rare, discriminative constructs like `BETWEEN` or domain-specific identifiers. Second, a **query-specific attention mechanism (Attn)** models contextual relevance by assigning higher weights to features that are particularly significant within a given query, such as nested subqueries in `WHERE` clauses or aggregates in `GROUP BY` contexts. The final feature weight combines these two complementary signals:

$$w(f | Q) = \alpha \cdot \text{IDF}(f) + (1 - \alpha) \cdot \text{Attn}(f | Q) \quad (1)$$

Here α balances global discriminativeness and query-specific contextual salience. This dual weighting strategy ensures that similarity computation captures both corpus-level feature informativeness and local structural relevance, enabling more accurate retrieval of semantically aligned examples for few-shot learning.

IDF-based Importance: The inverse document frequency (IDF) score for each feature is used to assess the discriminative power of SQL components.

$$\text{IDF}(f) = \log \left(\frac{N}{1 + \text{df}(f)} \right) \quad (2)$$

where N is the number of queries and $\text{df}(f)$ is the number of queries containing feature f . Within our similarity computation framework, the inverse document frequency (IDF) emphasizes rare and semantically relevant features, thereby reducing the influence of common tokens on similarity scores. This enables the retrieval of SQL queries that are structurally and semantically aligned with the target query, which improves the selection of few-shot demonstrations.

Attention-Based Importance. We estimate which SQL/AST features matter most using a self-supervised, light-weight model that learns directly from the training queries. Each query is parsed with `sqlparse` and converted into a multi-set of structured features (e.g., `KEYWORD:*`, `FUNCTION:*`, `IDENTIFIER:*`, `DEPTH:*`, and `PARENT_CHILD:*`). The model learns the feature sequence and is trained to reconstruct the set of features it observed via a multi-label (bag-of-features) prediction head. Reconstruction is only possible when the representation attends to the right tokens. The learned multi-head attention provides a usable salience signal. After training, we extract an attention score per feature by (i) computing the attention matrices, (ii) averaging across heads, and (iii) averaging for each feature occurrence.

Intuitively, query-specific attention captures contextual, query-local importance (e.g., the `COUNT` inside a `GROUP BY`), while IDF captures corpus-level distinctiveness (e.g., rare operators or schema identifiers). The resulting weights emphasize features that are both structurally pivotal and globally informative, and they are plugged into our weighted similarity (Equation 1) for retrieving few-shot examples.

3.2 Top- k Examples Retrieval

To improve few-shot performance, the retriever should select structurally and semantically relevant examples.

Given a target query Q_t , the retriever extracts SQL and AST-based features F_t from Q_t , and scores them via

$$S(Q_t, Q_i) = \frac{\sum_{f \in F_t} w(f|Q_t) \cdot \min(c_{Q_t}(f), c_{Q_i}(f))}{\sum_{f \in F_t} w(f|Q_t)} \quad (3)$$

You are an expert SQL analyst who translates SQL queries into natural language questions. Follow this detailed Chain of Thought process: Step 1: Analyze SQL Structure - Identify SELECT columns and their aliases - Identify FROM/JOIN tables and relationships - Identify WHERE conditions and filters - Identify GROUP BY, ORDER BY, LIMIT clauses - Understand the query's logical flow Step 2: Understand Business Intent - What business question does this query answer? - What entities are being queried? - What specific information is being requested? - What relationships are being explored? Step 3: Generate Natural Language Question - Use conversational, human-like language - Include ALL important details from the SQL - Mention specific columns being selected - Include table/entity names when relevant - Use appropriate verbs (show, list, find, count, etc.) - Make it specific and detailed, not generic Step 4: Quality Check - Does the question capture ALL the SQL logic? - Is it natural and conversational? - Does it include the right level of detail? - Would a human ask this question this way?	Guidelines for High-Quality Translation: - Be SPECIFIC: Include column names, table names, conditions - Be NATURAL: Use conversational language - Be COMPLETE: Don't omit important details - Be PRECISE: Match the exact intent of the SQL CRITICAL: Output ONLY the natural language question. Do NOT generate explanations or examples. Stop after the first translation. # Few-shot demonstrations (k=5) SQL: <SHOT_1_SQL> Natural Language: <SHOT_1_NL> SQL: <SHOT_2_SQL> Natural Language: <SHOT_2_NL> SQL: <SHOT_3_SQL> Natural Language: <SHOT_3_NL> SQL: <SHOT_4_SQL> Natural Language: <SHOT_4_NL> SQL: <SHOT_5_SQL> Natural Language: <SHOT_5_NL> # Target SQL: <TARGET_SQL> Natural Language:
--	--

Fig. 2: Prompt template for SQL-to-Text translation showing reasoning steps, quality checks, and few-shot demonstrations.

where $c_Q(f)$ counts occurrences of feature f in query Q (0 if the feature is not present in that query). We then select the top- k candidates with the highest similarity scores.

3.3 Chain-of-Thought (CoT) Prompting

To provide the LLM both explicit instructions and top- k retrieved demonstrations, we carefully craft a few-shot prompt for each target SQL query, as shown in Figure 2. The prompt integrates four key design features: a step-by-step reasoning process, quality-oriented translation guidelines, example demonstrations, and constrained output instructions to ensure focused, accurate responses.

Expert Role and Task Instruction. We instruct the LLM to act as a skilled SQL analyst through a structured four-step process: first, parse the query to identify tables, joins, filters, aggregations, and selected columns; second, infer the underlying business question to ensure full semantic coverage; third, translate the SQL into clear, human-readable language that accurately reflects its logic,

constraints, and aggregations; and fourth, verify accuracy, completeness, and natural phrasing so the final description aligns with a coherent user query.

Few-Shot Examples. The prompt appends the top- k retrieved SQL queries with their ground-truth natural-language descriptions as examples. These illustrate how structurally similar queries map to descriptions, implicitly guiding the LLM and boosting both translation accuracy and contextual understanding.

Target Query. After demonstrations, the target SQL query is given, instructing the LLM to provide the natural language description for the target query.

Output Constraints. The prompt states, “Output only the natural language translation. Do not provide explanations, reasoning steps, or additional examples”. This constraint yields concise, contextually accurate translations by minimizing prompt leakage and hiding the LLM’s chain-of-thought.

4 Experimental Results

4.1 Experimental Setup

Datasets. We perform experiments on the Spider-S2T, SParC-S2T, and CoSQL-S2T datasets derived from a popular benchmarks Spider [32], SParC [33], and CoSQL [31]. While the original datasets were designed for the Text-to-SQL task, these S2T variants were constructed for the SQL-to-Text task by providing each SQL query with multiple natural language translations [1]. The statistics of our dataset are given in Table 1. Few-shot examples are exclusively retrieved from the training splits of the original Spider, SParC, and CoSQL Text-to-SQL datasets. This method ensures that during the retrieval process, the target SQL query’s translation remains inaccessible.

Table 1: Dataset statistics.

Controlled Decoding. We enforce controlled decoding conditions throughout the LLM’s generation to improve factual correctness, fluency, and consistency. An overview of the decoding parameters can be found in Table 2.

We fix the number of few-shot examples to $k = 5$ and set the balance parameter to $\alpha = 0.5$ (Equation 1). This configuration ensures a consistent prompt template and assigns equal weight to both global inverse document frequency (IDF)-based importance and query-specific attention.

Baseline Methods. We evaluate our weighted AST-based retrieval prompting with the Graph-AST ICL using 2 examples as the baseline [1], which uses GNN-based embedding similarity to identify the top- k similar SQLs for few-shot demonstrations. Simpler baselines, e.g., random and BM25 are also tested but perform worse and are omitted.

LLMs. We employ three open-source LLMs of different sizes: Mistral (7B) [12], Code Llama (7B) [22], and GPT-J (6B parameters) [17], using the prompt in Figure 2 to generate text in a few-shot setting, limited by a 2048-token context.

Dataset	#SQL
CoSQL-S2T	290
SParC-S2T	289
Spider-S2T	274

Table 2: Controlled decoding parameters used during generation.

Parameter	Value	Purpose
Temperature	0.4	Reduces unpredictability, resulting in terminology that is consistent and predictable.
Top- p	0.9	Eliminates low-likelihood tokens by limiting sampling to the top 90% probability mass.
Top- k	50	Ensures that only the tokens with the highest probability of 0.5 are taken into account, increasing reliability.
Max new tokens	250	Prevents overly verbose responses and guarantees concise, single-sentence outputs.

Table 3: Human evaluation results: #Q = total evaluated queries. Values show #Correct SQL-to-Text conversions with percentage accuracy in parentheses.

Dataset	#Q	LLM	Graph-AST ICL [1]	Weighted-AST [ours]
CoSQL-S2T	290	Code Llama 7B	108 (37.24%)	126 (43.44%)
		GPT-J 6B	118 (40.68%)	163 (56.2%)
		Mistral 7B	196 (67.58%)	237 (81.72%)
SParC-S2T	289	Code Llama 7B	191 (72.31%)	209 (72.32%)
		GPT-J 6B	123 (42.56%)	163 (56.4%)
		Mistral 7B	230 (79.58%)	234 (81.0%)
Spider-S2T	274	Code Llama 7B	183 (66.78%)	242 (88.32%)
		GPT-J 6B	125 (45.62%)	199 (72.62%)
		Mistral 7B	202 (73.72%)	252 (91.97%)

Evaluation Metrics. Currently used metrics for SQL-to-Text [1] such as BLEU, BERTScore, Exact Match (EM), and Execution Accuracy (EX) emphasize surface-level similarity and miss semantic accuracy. To address this limitation, we conduct (1) human evaluation by experts to judge whether generated descriptions accurately represent the underlying SQL logic; and (2) round-trip evaluation in which the generated text is converted back into SQL using a state-of-the-art semantic parser, and the resulting SQL is compared to the original query using the EM and EX metrics, as detailed below.

In particular, following recent research [10] [1] [21], we assess the quality of generated SQL-to-text translations using both automatic metrics and human judgments. The exact match (EM) [36] measures syntactic equivalence between generated and reference SQL, while execution accuracy (EX) [29] verifies whether both queries yield identical results upon execution over the same dataset. In contrast, during human evaluation, an expert judge compares each generated description to its SQL query and assigns a binary correct/incorrect score.

4.2 Human Evaluation

To evaluate semantic correctness and readability, the first author assessed each description’s correctness, fluency, and completeness, and the second author partially verified these assessments for consistency.

Table 3 depicts that based on expert human review, our method significantly outperforms the SOTA approach across all datasets and LLMs, showcasing its adaptability. In all evaluation, Mistral-7B demonstrates the highest accuracy, achieving up to 91.97% on the Spider-S2T benchmark compared to

Table 4: Round-trip Execution Accuracy (EX) comparison. #Q = total evaluated queries. Columns report EX and EX% separately for each method.

Dataset	Q	LLM	Graph-AST ICL [1]		Weighted-AST [ours]	
			EX	EX%	EX	EX%
CoSQL-S2T	290	GPT-J 6B	31	10.69%	54	18.62%
		Code Llama 7B	0	0.00%	50	17.24%
		Mistral-7B	55	18.97%	91	31.38%
SParC-S2T	289	GPT-J 6B	26	9.00%	29	10.03%
		Code Llama 7B	43	14.88%	54	18.69%
		Mistral-7B	51	17.65%	58	20.07%
Spider-S2T	274	GPT-J 6B	7	2.55%	6	2.19%
		Code Llama 7B	7	2.55%	9	3.28%
		Mistral-7B	13	4.74%	10	3.65%

73.72% for the baseline, while GPT-J 6B records the lowest baseline accuracy. However, even for GPT-J 6B, our method significantly enhances performance, increasing correctness from 40.7–45.6% with Graph-AST ICL to 56.2–72.6% with Weighted-AST. These results highlight the benefits of our approach: it continuously improves translation quality regardless of the underlying model, especially enhancing the performance of more powerful LLMs like Mistral-7B. The reasons behind this is that Graph-AST ICL relies on graph embeddings, which capture only coarse structural similarity, often retrieving examples that are syntactically related but not semantically aligned with the target query. In contrast, our approach combines systematic prompting and weighted AST-based feature extraction to select extremely relevant few-shot samples. This approach significantly increases the accuracy of the generated translations by producing translations that are both semantically informative and structurally relevant.

4.3 Round-trip SQL-based Evaluation

To further evaluate the quality of our SQL-to-Text method, we convert the Text back to SQL using a recent GPT-4 based parser [8]. Their method reports an execution accuracy of about 86% on the Spider dataset. Given the description and schema, the parser reconstructs SQL, and we assess performance with Exact Match (EM) [36] and Execution Accuracy (EX) [29]. Here, EM ignores formatting, case, alias names, and other non-semantic variations to avoid penalizing syntactically different but semantically identical queries. EX, by contrast, directly assesses whether the generated queries and the reference gold queries yield identical results upon execution.

Execution Accuracy (EX) results given in Table 4 compare our Weighted AST prompting with the Graph-AST ICL baseline on CoSQL-S2T, Spider-S2T, and SParC-S2T datasets. On CoSQL-S2T with Mistral-7B, Weighted-AST achieves 91 correct executions (31.38%) compared to 55 (18.97%) with Graph-AST ICL. Code Llama-7B, which fails to produce any correct executions under the baseline (0.00%), reaches 50 correct executions (17.24%) with Weighted-AST. On SParC-S2T, improvements are also evident: Code Llama-7B increases from 43 (14.88%) to 54 (18.69%), and Mistral-7B from 51 (17.65%) to 58 (20.07%). For Spider-S2T, the improvements are modest: Code Llama-7B improves from 7 (2.55%)

Table 5: Round-trip Exact Match (EM) comparison. #Q = total evaluated queries. Columns report EM and EM% separately for each method.

Dataset	Q	LLM	Graph-AST ICL [1]		Weighted-AST [ours]	
			EM	EM%	EM	EM%
CoSQL-S2T	290	GPT-J 6B	5	1.72%	12	4.14%
		Code Llama 7B	0	0.00%	10	3.44%
		Mistral-7B	12	4.14%	20	6.90%
SParC-S2T	289	GPT-J 6B	11	3.81%	19	6.57%
		Code Llama 7B	14	4.84%	24	8.30%
		Mistral-7B	22	7.61%	27	9.34%
Spider-S2T	274	GPT-J 6B	3	1.09%	6	2.19%
		Code Llama 7B	4	1.46%	9	3.28%
		Mistral-7B	11	4.01%	9	3.28%

to 9 (3.28%), while GPT-J 6B and Mistral-7B show slight declines. Overall, our results imply that our approach improves execution fidelity, especially when applied to conversational datasets (CoSQL and SParC).

The exact match results and their percentages for our method compared to the existing approach, across all datasets and LLMs, is summarized in Table 5. Our approach consistently outperforms the SOTA method. Collectively, the round-trip results demonstrate that Weighted-AST consistently improves execution robustness and exact match fidelity. Both EX and human evaluation outcomes provide evidence that this method more effectively preserves semantic intent and yields more accurate SQL reconstructions compared to the baseline.

5 Conclusion

In this paper, we proposed Weighted-AST retrieval with a prompting system that integrates structurally aware signals into few-shot prompting to address the problem of generating accurate natural-language explanations of SQL queries. Our method learns feature importances over SQL/AST tokens, retrieves semantically aligned demonstrations, and constrains generation with a task-specific prompt. Our approach demonstrates superior performance compared to the Graph-AST ICL baseline across three benchmarks (CoSQL-S2T, SParC-S2T, Spider-S2T) and three large language models (Mistral-7B, Code Llama-7B, GPT-J-6B). Results from the human evaluation and improvements in round-trip Execution Accuracy and Exact Match in the experiments demonstrate the superiority of our method. Our experiments are conducted using a smaller SQL-to-Text datasets derived from the original Text-to-SQL dataset. Future research can focus on SQL-to-Text over multilingual settings and creating a large-scale dataset.

Acknowledgments

Sriom Chakrabarti, Chuangtao Ma, and Arijit Khan acknowledge support from the Novo Nordisk Foundation grant NNF22OC0072415.

References

1. Al Lawati, A., Lucas, J., Mitra, P.: Semantic Captioning: Benchmark Dataset and Graph-Aware Few-Shot In-Context Learning for SQL2Text. In: International Conference on Computational Linguistics (COLING). pp. 8026–8042 (2025)
2. Albrecht, A., contributors: python-sqlparse: A non-validating sql parser for python. <https://github.com/andialbrecht/sqlparse> (2025), version 0.5.0, Accessed: 2025-09-08
3. Amer-Yahia, S., Bonifati, A., Chen, L., Li, G., Shim, K., Xu, J., Yang, X.: From Large Language Models to Databases and Back: A discussion on research and education. *SIGMOD Rec.* **52**(3), 49–56 (2023)
4. Bhardwaj, M., Ethari, H., Moirangthem, D.S.: EzSQL: An SQL intermediate representation for improving SQL-to-text Generation. *Expert Systems with Applications* **289**, 128293 (2025)
5. Carr, N., Shawon, F.R., Jamil, H.M.: An Experiment on Leveraging ChatGPT for Online Teaching and Assessment of Database Students. In: IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE). pp. 1–8 (2023)
6. Chakrabarti, S., Ma, C., Khan, A., Link, S.: Code and datasets. <https://github.com/iamsriom/SQL-to-Text-Generation-with-Weighted-AST-Few-Shot-Prompting> (2025)
7. Eleftherakis, S., Gkini, O., Koutrika, G.: Let the Database Talk Back: Natural Language Explanations for SQL. In: Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data) co-located with International Conference on Very Large Data Bases (VLDB). CEUR Workshop Proceedings, vol. 2929, pp. 14–19 (2021)
8. Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., Zhou, J.: Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.* **17**(5), 1132–1145 (2024)
9. Geng, M., Wang, S., Dong, D., Wang, H., Li, G., Jin, Z., Mao, X., Liao, X.: Large Language Models are Few-Shot Summarizers: Multi-intent Comment Generation via In-Context Learning. In: IEEE/ACM International Conference on Software Engineering. pp. 1–13 (2024)
10. Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., Huang, X.: Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. *arXiv preprint arXiv:2406.08426* (2024)
11. Iyer, S., Konstas, I., Cheung, A., Zettlemoyer, L.: Summarizing Source Code using a Neural Attention Model. In: Annual Meeting of the Association for Computational Linguistics (ACL). pp. 2073–2083 (2016)
12. Jiang, D., Liu, Y., Liu, S., Zhao, J., Zhang, H., Gao, Z., Zhang, X., Li, J., Xiong, H.: From CLIP to DINO: Visual Encoders Shout in Multi-modal Large Language Models. *arXiv preprint arXiv:2310.08825* (2023)
13. Katsogiannis-Meimarakis, G., Xydias, M., Koutrika, G.: Data Democratisation with Deep Learning: The Anatomy of a Natural Language Data Interface. In: ACM International Conference on Web Search and Data Mining (WSDM). pp. 1260–1263 (2023)
14. Koutrika, G., Simitsis, A., Ioannidis, Y.E.: Explaining Structured Queries in Natural Language. In: International Conference on Data Engineering (ICDE). pp. 333–344 (2010)
15. Li, F., Jagadish, H.V.: Constructing an Interactive Natural Language Interface for Relational Databases. *Proc. VLDB Endow.* **8**(1), 73–84 (2014)

16. Li, H., Zhang, J., Liu, H., Fan, J., Zhang, X., Zhu, J., Wei, R., Pan, H., Li, C., Chen, H.: Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data* **2**(3), 1–28 (2024)
17. Phang, J., Bradley, H., Gao, L., Castricato, L., Biderman, S.: EleutherAI: Going Beyond "Open Science" to "Science in the Open". *arXiv preprint arXiv:2210.06413* (2022)
18. Pourreza, M., Li, H., Sun, R., Chung, Y., Talaei, S., Kakkar, G.T., Gan, Y., Saberi, A., Ozcan, F., Arik, S.Ö.: CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. In: *International Conference on Learning Representations (ICLR)* (2025)
19. Pourreza, M., Rafiei, D.: Evaluating cross-domain text-to-sql models and benchmarks. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 1601–1611 (2023)
20. Prakash, K., Rao, S., Hamza, R., Lukich, J., Chaudhari, V., Nandi, A.: Integrating LLMs into Database Systems Education. In: *International Workshop on Data Systems Education: Bridging education practice with education research, DataEd* (2024)
21. Renggli, C., Ilyas, I.F., Rekatsinas, T.: Fundamental challenges in evaluating Text2SQL solutions and detecting their limitations. *arXiv preprint arXiv:2501.18197* (2025)
22. Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., et al.: Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023)
23. Safarzadeh, M., Oroojlooyjadid, A., Roth, D.: Evaluating NL2SQL via SQL2NL (2025), accepted to EMNLP 2025
24. Shen, K., Kejriwal, M.: SELECT-SQL: Self-correcting ensemble Chain-of-Thought for Text-to-SQL. *arXiv preprint arXiv:2409.10007* (2024)
25. Sun, W., Miao, Y., Li, Y., Zhang, H., Fang, C., Liu, Y., Deng, G., Liu, Y., Chen, Z.: Source Code Summarization in the Era of Large Language Models. In: *IEEE/ACM International Conference on Software Engineering (ICSE)*. pp. 1882–1894 (2025)
26. Tian, Y., Kummerfeld, J.K., Li, T.J., Zhang, T.: SQLucid: Grounding Natural Language Database Queries with Interactive Explanations. In: *ACM Symposium on User Interface Software and Technology (UIST)*. pp. 12:1–12:20 (2024)
27. Xu, K., Wu, L., Wang, Z., Feng, Y., Sheinin, V.: SQL-to-Text generation with graph-to-sequence model. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 931–936 (2018)
28. Xu, K., Wu, L., Wang, Z., Feng, Y., Witbrock, M., Sheinin, V.: Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks. *arXiv preprint arXiv:1804.00823* (2018)
29. Yang, Y., Wang, Z., Xia, Y., Wei, Z., Ding, H., Piskac, R., Chen, H., Li, J.: Automated Validating and Fixing of Text-to-SQL Translation with Execution Consistency. *Proceedings of the ACM on Management of Data* **3**(3), 1–28 (2025)
30. Yao, Z., Sun, G., Borchmann, L., Shen, Z., Deng, M., Zhai, B., Zhang, H., Li, A., He, Y.: Arctic-Text2SQL-R1: Simple Rewards, Strong Reasoning in Text-to-SQL. *arXiv preprint arXiv:2505.20315* (2025)
31. Yu, T., Zhang, R., Er, H., Li, S., Xue, E., Pang, B., Lin, X.V., Tan, Y.C., Shi, T., Li, Z., Jiang, Y., Yasunaga, M., Shim, S., Chen, T., Fabbri, A.R., Li, Z., Chen, L., Zhang, Y., Dixit, S., Zhang, V., Xiong, C., Socher, R., Lasecki, W.S., Radev, D.R.: Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In: *Conference on Empirical Methods*

- in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 1962–1979 (2019)
32. Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., Radev, D.R.: Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In: Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 3911–3921 (2018)
 33. Yu, T., Zhang, R., Yasunaga, M., Tan, Y.C., Lin, X.V., Li, S., Er, H., Li, I., Pang, B., Chen, T., Ji, E., Dixit, S., Proctor, D., Shim, S., Kraft, J., Zhang, V., Xiong, C., Socher, R., Radev, D.R.: Sparc: Cross-domain semantic parsing in context. In: Association for Computational Linguistics (ACL). pp. 4511–4523 (2019)
 34. Yuan, S., Chen, L., Yuan, M., Zhao, J., Peng, H., Guo, W.: MCTS-SQL: An effective framework for text-to-sql with monte carlo tree search. arXiv preprint arXiv:2501.16607 (2025)
 35. Zhang, Y., Deriu, J., Katsogiannis-Meimarakis, G., Kosten, C., Koutrika, G., Stockinger, K.: Sciencebenchmark: A complex real-world benchmark for evaluating natural language to SQL systems. *Proc. VLDB Endow.* **17**(4), 685–698 (2023)
 36. Zhong, R., Yu, T., Klein, D.: Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In: Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 396–411 (2020)