Track-SQL: Enhancing Generative Language Models with Dual-Extractive Modules for Schema and Context Tracking in Multi-turn Text-to-SQL

Bingfeng Chen[1,2], **Shaobin Shi**[1], **Yongqi Luo**[1], **Boyan Xu**[1*], **Ruichu Cai**[1,3], **Zhifeng Hao**[1,4]
[1]School of Computer Science, Guangdong University of Technology
[2]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
[3]Peng Cheng Laboratory
[4]College of Science, Shantou University
chenbf@gdut.edu.cn
{d7inshi, lyongqi001, hpakyim, cairuichu}@gmail.com
haozhifeng@stu.edu.cn

Abstract

Generative language models have shown significant potential in single-turn Text-to-SQL. However, their performance does not extend equivalently to multi-turn Text-to-SQL. This is primarily due to generative language models' inadequacy in handling the complexities of context information and dynamic schema linking in multi-turn interactions. In this paper, we propose a framework named Track-SQL, which enhances generative language models with dual-extractive modules designed to track schema and contextual changes in multi-turn Text-to-SQL. Specifically, Track-SQL incorporates a *Semantic-enhanced Schema Extractor* and a *Schema-aware Context Extractor*. Experimental results demonstrate that Track-SQL achieves state-of-the-art performance on the SparC and CoSQL datasets. Furthermore, detailed ablation studies reveal that Track-SQL significantly improves execution accuracy in multi-turn interactions by 7.1% and 9.55% on these datasets, respectively. Our implementation will be open-sourced at https://github.com/DMIRLAB-Group/Track-SQL.

1 Introduction

Text-to-SQL (Zhong et al., 2017) is a critical semantic parsing task that converts natural language queries into corresponding SQL statements based on a given database schema. While generative language models have demonstrated significant potential in single-turn Text-to-SQL tasks, their performance diminishes in multi-turn scenarios. The challenges in these settings primarily stem from the models' difficulties in handling the complexities of context information and dynamic schema linking across multiple turns of interaction. However, extractive model to solving these two challenges—schema linking and context utilization—have limitations when directly applied to the generative language model paradigm.

The primary limitation in multi-turn Text-to-SQL is the ability to maintain effective schema linking as the dialogue progresses. Although many studies have confirmed that proper schema linking significantly enhances SQL generation, existing approaches struggle to handle the increasing complexity in multi-turn settings. In recent research, RASAT (Qi et al., 2022) leverages relational self-attention mechanisms to capture the relationships between text and schemas. However, in multi-turn dialogue scenarios, as the number of interactions between the user and the system increases along with the expansion of the database schemas, the scale of the schema linking graph grows, inevitably leading to the issue of redundant links. CQR-SQL (Xiao et al., 2022) rewrites multi-turn dialogues and simplifies schema linking information, which might also result in the loss of critical schema linking details due to over-simplification. TP-Link (Liu et al., 2024) integrates schema linking prediction into multi-task pre-training to reduce redundant relationships between questions and schemas, but overlooks the problem of semantic inconsistencies between questions and schemas. Moreover, existing schema linking methods are predominantly static, lacking mechanisms to incorporate linking results from prior turns. The key idea to overcome this limitation is to introduce a dynamic updating mechanism that adapts to the evolving dialogue contexts in multi-turn interactions.

The secondary limitation in multi-turn Text-to-SQL is their difficulty in managing continuous interactions where users reference or omit prior information, relying on the system to track the evolving context. In recent research, EditSQL (Zhang et al., 2019) uses prior turn SQL queries for predicting current turn queries, but loses effectiveness when dialogue lacks coherence. STaR (Cai et al., 2022) and CoE-SQL (Zhang et al., 2024a) track depen-

*Corresponding author, hpakyim@gmail.com dencies via SQL similarity and changes, but lack source record verification, which can lead to error buildup. The key idea to overcome this limitation is to design effective retrieval and verification mechanisms for improving the accuracy and reliability of multi-turn Text-to-SQL.

To address these issues, we propose a Track-SQL framework, aimed at solving the problems of dynamic schema linking and context information filtering in multi-turn Text-to-SQL dialogues. First, in the schema extraction phase, we developed a *Semantic-enhanced Schema Extractor* (SESE) that identifies the user's current focus schemas by combining changes in user interests with previously extracted signals. At the same time, we introduce a semantic enhancement module to reduce the semantic gap between user questions and schemas, thereby improving the precision of schema linking. Second, in the SQL generation phase, we design a *Schema-aware Context Extractor* (SACE) module to identify key SQLs from historical records. Then, we combine these with past questions and extracted schemas as input, and fine-tune the text-to-SQL generation model in a supervised manner, with the target output being normalized SQL queries. This approach reduces the difficulty for the generation model to learn schema linking and context information filtering, thus enhancing the accuracy of SQL generation. Additionally, the results from the aforementioned extractors make the basis for the model's SQL generation more transparent, increasing the explainability of the system.

Our contributions are summarized as follows:

- We propose the Track-SQL framework, specifically designed for Multi-turn Text-to-SQL tasks. This framework utilizes a Semantic-enhanced Schema Extractor to ensure that the SQL generation model acquires accurate schema information in each dialogue turn.

- We have devised a Schema-aware Context Extractor to obtain the most relevant historical SQL queries that fit the current conversational context, thereby enhancing the dialogue history understanding capability and SQL generation accuracy of the generative language model.

- We conducted extensive performance evaluations and detailed ablation studies to verify the effectiveness of each component. The Track-SQL framework achieved leading results on the validation sets of two authoritative benchmark datasets, SParC and CoSQL, demonstrating its superior performance and broad applicability.

2 Methodology

In the multi-turn Text-to-SQL task, the goal is to address the problem of mapping a sequence of multi-turn questions $\mathcal{Q}_{\leq m}$ and database schemas $\mathcal{S} = (t_i, c_{i,n_i})$ to the target SQL query s_m, where t_i represents the i^{th} table in the database, and c_{i,n_i} denotes the n_i^{th} column within t_i. This section will provide an overview of the framework designed to solve this problem and delve into its design details.

2.1 Model Overview

In multi-turn Text-to-SQL tasks, we decompose two preparatory tasks: Dynamic Schema Linking and Context Information Filtering. Enhancing the performance of the generative language model is achieved through utilizing dual-extractive modules to identify key schema and contextual information. Figure 1 illustrates the overall architecture of the proposed Track-SQL framework, which includes two history information repositories and their corresponding extraction modules, along with a supervised fine-tuned SQL generator. Specifically, during the m^{th} interaction, we input the first m questions and all schema information into Schema Extractor to obtain the probabilities of all schemas for the current question $\mathcal{Q}_m$. Based on these probabilities and using fixed threshold s, we filter and rank the schemas (Section 2.2). Additionally, by utilizing the schema probabilities stored in the History Schema Store, we can resolve the coreference relationships between $\mathcal{Q}_m$ and $\mathcal{Q}_{<m}$ and select the base SQL from History Question&SQL Store which applicable to current question (Section 2.3). These filtered schemas and base SQL serve together as prompt information and constraints to facilitate the generation of SQL at the m^{th} turn.

2.2 Semantic-enhanced Schema Extractor

In the context of multi-turn dialogues involving databases, redundant schema item information can significantly interfere with the generation of SQL queries. To address this issue, we designed SESE to filter out redundant table column information. The extractor consists of three intertwined sub-elements: Historical Extraction Item Tagging, Schema Semantic Enhancement and All-Column

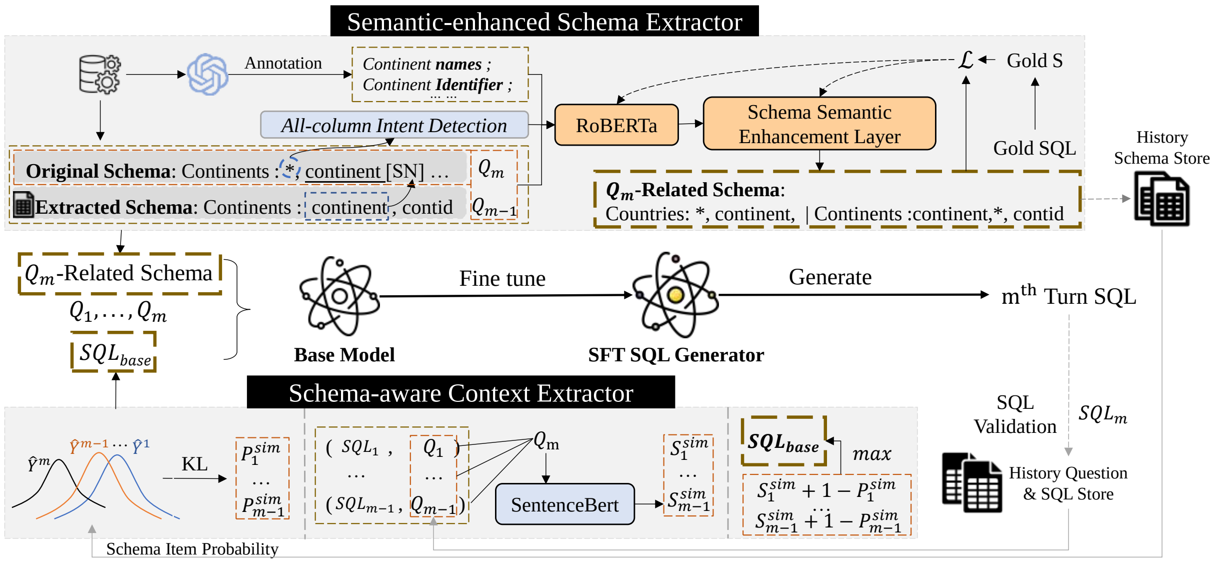

Figure 1: The overall framework of Dual-Extractive Modules for Schema and Context Tracking. The framework trains a schema item classification model and an SQL generator. Based on the former, we construct a *Semantic-enhanced Schema Extractor* and a *Schema-aware Context Extractor*. The extraction results from these two extractors are utilized for subsequent training of the SQL generation model. The core idea of Track-SQL is to reduce the gap between the input and the target SQL before entering the multi-turn SQL generation phase by means of dynamic schema linking and context information extraction.

Intent Detection. These elements respectively facilitate dynamic schema linking encoding, semantic alignment between question entities and schemas, and all-column intent encoding.

Regarding the specific implementation details, we first define the original sequence of concatenated multi-turn questions and schemas as: $\mathcal{X} = \mathcal{Q}_1 \ \& \ \ldots \ \& \ \mathcal{Q}_m \mid t_1 : c_{11}, \ldots, c_{1n_1} \mid \ldots \mid t_N : c_{N1}, \ldots, c_{Nn_N}$, where & connects multiple turns of questions, and | separates different schemas. To enhance the semantic expressiveness of the schemas, we introduce open-domain semantic knowledge, combining database content with large language models (LLMs) to enrich the semantic information of column names, and further utilize the enhanced column names to enrich the semantic information of table names. Specifically, we sample some values randomly from each column in each table, along with the type and name of the column, as inputs to prompt LLM to generate descriptive comments; then, based on the information of all columns, generate comments for the tables. Thus, we obtain a schema annotation sequence: $\hat{\mathcal{S}} = \hat{t}_1 : \hat{c}_{11}, \ldots, \hat{c}_{1n_1} \mid \ldots \mid \hat{t}_N : \hat{c}_{N1}, \ldots, \hat{c}_{Nn_N}$. In a multi-turn interactive environment, we retrieve the columns extracted from the previous turn from the history schema store and mark these columns using the symbol [**SN**] within the current turn's input $\mathcal{X}$. Subsequently, $\mathcal{X}$ and $\hat{\mathcal{S}}$ are sequentially input into RoBERTa (Liu et al., 2019). To integrate and classify the schemas along with their corresponding comments as a complete unit, we perform pooling operations on the output embeddings of each token after tokenization by RoBERTa for the schemas. To accomplish this, we used a pooling module composed of two-layer BiLSTM (Zhou et al., 2016) and a nonlinear fully connected layer. After pooling, the embedding of each table and its annotation can be represented as $T_i, \hat{T}_i \in \mathbb{R}^{1\times d}$ $(i \in \{1, \ldots, N\})$, and the embeddings of each column and its annotation can be represented as $C_{i,k}, \hat{C}_{i,k} \in \mathbb{R}^{1\times d}$ $(i \in \{1, \ldots, N\}, k \in \{1, \ldots, n_i\})$, where d denotes the size of the hidden layer.

Schema Semantic Enhancement Layer The naming of database schemas can sometimes be ambiguous, which may lead to a semantic discrepancy between the user's query intent and the actual data structure. As shown in Figure 1, parsing through LLM reveals that the "*continent*" column in both the *Continents* and *Countries* tables represents "*continent name*" and "*continent id*" respectively. Such abbreviated column names can lead to misinterpretations, thereby affecting the performance of the schema extractor. To address this issue of semantic inconsistency, we introduce an attention gating mechanism on top of the column-enhanced module (Li et al., 2023) to aggregate representations of schema names along with their associated annotations.

$$\mathbf{g_i^t} = \sigma(\mathbf{W}_g^t[\mathbf{W}_1^t T_i + \mathbf{b}_1^t; \mathbf{W}_2^t \hat{T}_i + \mathbf{b}_2^t] + \mathbf{b}_g^t) \quad (1)$$

$$T_i^G = Norm(T_i + (\mathbf{g_i^t} \circ T_i + (1 - \mathbf{g_i^t}) \circ \hat{T}_i)) \quad (2)$$

$$\mathrm{g}_{i,k}^c = \sigma(\mathbf{W}_g^c[\mathbf{W}_1^c C_{i,k} + \mathbf{b}_1^c; \mathbf{W}_2^c \hat{C}_{i,k} + \mathbf{b}_2^c] + \mathbf{b}_g^c) \quad (3)$$

$$\mathrm{C}_{i,k}^G = Norm(T_i + (\mathbf{g_{i,k}^c} \circ C_{i,k} + (1 - \mathbf{g_{i,k}^c}) \circ \hat{C}_{i,k})) \quad (4)$$

where $\mathbf{W}_j^t, \mathbf{W}_j^c \in \mathbb{R}^{d \times d}, \mathbf{b}_j^t, \mathbf{b}_j^c \in \mathbb{R}^d \, (j = 1, 2)$, and $\circ$ denotes the element-wise multiplication. Additionally, $\mathbf{b}_g^t, \mathbf{b}_g^c \in \mathbb{R}^d$, $\mathbf{W}_g^t, \mathbf{W}_g^c \in \mathbb{R}^{2d \times d}$, and $\sigma(\cdot)$ represents the sigmoid function, while $Norm(\cdot)$ is the row-wise L_2 normalization function. The gating vectors $\mathbf{g_i^t}$ and $\mathbf{g_{i,k}^c}$ represent the gating vectors for tables and columns, respectively. By using the probabilities from these gating vectors to perform a weighted average and normalization of the embeddings of schemas and their annotations, we obtain the enhanced table embeddings $T_i^G \in \mathbb{R}^{1 \times d}$ and column embeddings $C_{i,k}^G \in \mathbb{R}^{1 \times d}$.

After processing through the semantic enhancement layer, we obtain the classification probabilities of all schemas for the current turn. Next, based on a predefined threshold s, we extract the schemas that are most relevant to the question. This threshold must be set reasonably to avoid losing important schemas due to it being too high. We sort the selected schemas in descending order of their probability values and combine them with the serialized foreign key information, ultimately generating the serialized database schema representation required by the SQL generation model.

ALL-Column Intent Detection In question interactions, user intent is not always directly expressed as seeking information from specific columns but may implicitly involve retrieving information from all columns within a specified table. To effectively capture the implicit "all-columns intent" within user questions, we designed the schema extractor to specifically recognize the wildcard "*" in SQL as a special column identifier. For example, in the question "*Show ids and names of all continents!*", the user explicitly requests information from specific columns ("*ids*" and "*names*") in the "*continents*" table. However, when dealing with broader questions such as "*Which countries do they each have*", the user's intent implicitly demands retrieving all relevant data. In response to this, the schema extractor generates classification probabilities for "*" across all tables and determines whether and how to insert "*" into the input sequence based on these probabilities. This approach guides the SQL generator to more accurately understand and respond to the user's question intent.

2.3 Schema-aware Context Extractor

As the number of dialogue turns increases, the relationships between each turn become more intricate, with current questions often bearing inter-turn connections to historical questions. To address this, we have developed a Schema-aware Context Extractor that can identify the most relevant past SQL related to the current question and use the query as references for generating the current SQL. Since semantic associations between questions and the database schema have already been modeled during the schema extraction phase, in the subsequent context extraction process, we can directly utilize the pre-stored schema item encoding information in the history schema store, thereby avoiding redundant model training.

In the process of selecting historical question-SQL pairs, we mainly consider the following two factors. Firstly, the semantic relevance between the current question $\mathcal{Q}_m$ and historical questions $\mathcal{Q}_h$ $(h \in 1, ..., m-1)$. Higher semantic similarity typically indicates similar query intentions. For example, if $\mathcal{Q}_m$ is *"How many dog pets are raised by female students?"*, and the historical question $\mathcal{Q}_h$ is *"How many of those have dogs?"*, there is a clear semantic association between these questions. We employ the SentenceBERT (Reimers and Gurevych, 2019) to quantify this similarity, defined as:

$$\mathcal{S}_h^{sim} = SentenceBERT(\mathcal{Q}_h, \mathcal{Q}_m) \quad (5)$$

where $h \in [1, .., m-1]$. When $\mathcal{Q}_h$ and $\mathcal{Q}_m$ are semantically similar, their corresponding SQL queries likely share a similar structure.

Secondly, relying solely on semantic similarity between questions might lead to misjudgment. For instance, the questions *"Who are the female students?"* and *"Of those, who has a pet?"* are sequential inquiries but appear quite different semantically. Therefore, we also incorporate the schema item probabilities provided by the History Schema Store to measure the overlap of entities involved in the two questions. Let the normalized schema item extraction probability vectors obtained from the m^{th} and h^{th} turns be denoted as $\hat{\mathcal{Y}}^m$ and $\hat{\mathcal{Y}}^h$, respectively. The normalized Jensen-Shannon divergence can then be used to assess the difference between these entities:

$$\mathrm{P}_h^{sim} = \frac{1}{2\ln 2}(D_{KL}(\hat{\mathcal{Y}}^m||\bar{\mathcal{Y}}) + D_{KL}(\hat{\mathcal{Y}}^h||\bar{\mathcal{Y}})) \quad (6)$$

where $\bar{\mathcal{Y}} = \frac{1}{2}(\hat{\mathcal{Y}}^m + \hat{\mathcal{Y}}^h)$ is the average distribution of $\hat{\mathcal{Y}}^m$ and $\hat{\mathcal{Y}}^h$, and D_{KL} represents the Kullback-Leibler divergence, which can be expressed as:

$$D_{KL}(\hat{\mathcal{Y}}^m||\hat{\mathcal{Y}}^h) = \sum_{i=1}^{N}\left(\hat{y}_i^h \log\frac{\hat{y}_i^h}{\hat{y}_i^m} + \sum_{k=1}^{n_i}\hat{y}_{i,k}^h \log\frac{\hat{y}_{i,k}^h}{\hat{y}_{i,k}^m}\right) \quad (7)$$

where $\hat{y}_i^h$ denotes the normalized predicted probability of the i^{th} table in the h^{th} turn, while $\hat{y}_{i,k}^h$ represents the normalized predicted probability of the k^{th} column within that table.

$\mathcal{S}_h^{sim}$ and $\mathcal{P}_h^{sim}$ are two key scoring metrics used for retrieving and filtering historical information, both of which are constrained within the interval [0,1]. Specifically, $\mathcal{S}_h^{sim}$ represents a maximization score metric, whereas P_h^{sim} is a minimization score metric. To unify the direction of the metrics, we adjust the form of $\mathcal{P}_h^{sim}$ to $1-\mathcal{P}_h^{sim}$, thereby allowing both metrics to be optimized towards maximization. Based on these definitions, we can calculate the comprehensive relevance score $\mathcal{R}_h$ between $\mathcal{Q}_h$ and $\mathcal{Q}_m$ as follows:

$$\mathcal{R}_h = \mathcal{S}_h^{sim} + 1 - \mathcal{P}_h^{sim} \quad (8)$$

By selecting historical SQL with the highest $\mathcal{R}_h$ values, we can utilize appropriate historical SQL as reference inputs during the supervised fine-tuning of an LLM. This facilitates generating accurate SQL outputs in response to Q_m, effectively reducing the discrepancy between inputs and outputs.

2.4 SQL Generation Fine-tuning

By screening the schemas in multiple turns of data sets and filtering out irrelevant historical information, we obtain a streamlined input sequence that is highly relevant to the target SQL. Based on this, we transform the multi-turn dataset into a single-turn Text-to-SQL corpus. Each entry's input sequence consists of the question $\mathcal{Q}_{\leq m}$, the extracted sequence of schemas $E(\mathcal{S})$, and SQL_{base}, while the actual SQL serves as the desired output sequence s_m. In practice, when converting multi-turn questions into single-turn questions, we found that directly rewriting the questions or other forms of filtering could lead to severe error propagation. Therefore, we define the problem $\mathcal{Q}_{\leq m} = \mathcal{Q}_1\&...\&\mathcal{Q}_m$, using symbols to concatenate the sequences to ensure the semantic integrity of the multi-turn question series. For the first question $\mathcal{Q}_1$, due to the lack of a previous SQL as a reference, we set SQL_{base} as an empty sequence. Therefore, the minimization loss function of a set of interaction samples can be expressed as:

$$\min_{\varepsilon, M^*}\sum_m \mathcal{L}(M^*\varepsilon(\mathcal{Q}_{\leq m}, E(\mathcal{S}), SQL_{base}, s_m)) \quad (9)$$

where $\varepsilon(\cdot)$ defines a sequence format and details can be found in Appendix B.4. Due to the Track-SQL framework enabling dynamic schema linking and effective filtering of historical information, the LLM can focus more on capturing the essential connections between key information and SQL queries during training, thereby reducing the interference caused by redundant information in the SQL generation task.

3 Experiments

3.1 Experimental Setup

Datasets We validated the effectiveness of the proposed method on the SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a) benchmark datasets. The SParC dataset contains 4,298 multi-turn dialogue sequences, covering over 12,000 individual questions and their corresponding SQL queries. The CoSQL dataset includes over 10,000 annotated SQL queries, with each dialogue sequence designed to simulate real-world scenarios where ordinary users explore databases and interact with them. In these scenarios, non-expert users ask questions in natural language, while experts use SQL to retrieve answers.

Evaluation Metrics To evaluate the performance of our method in the text-to-SQL task, we adopted two official metrics: Question Match (QM) and Interaction Match (IM), along with three widely recognized sub-metrics: Exact Match Accuracy (EM), Execution Accuracy (EX), and Test Suite Accuracy (TS). QM measures whether the predicted SQL is accurate for a single question, while IM evaluates whether all predicted SQL queries meet the QM standard across multiple turns of a conversation. Specifically, EM measures the structural accuracy of the predicted SQL; EX focuses on whether the execution result of the predicted SQL is correct; TS not only examines the accuracy of query execution but also requires correct results for each query executed over multiple database instances and schemas.

To provide a finer-grained assessment of the extraction accuracy of database schemas, we introduced a set of strict evaluation metrics: Table Redundancy Score ($TRS@s$) and Column Redundancy Score ($CRS@s$), where s represents the extraction threshold. A schema item is considered for extraction when its classification probability calculated by the schema extractor is greater than s. These metrics are formally defined as follows:

$$V^t = \{i' \mid y_{i'} = 1\},\ \hat{V}^t = \{i' \mid \hat{y}_{i'} \geq s\} \quad (10)$$

where V^t denotes the set of indices of tables included in the ground truth SQL, and $\hat{V}^t$ denotes the set of indices of tables extracted by the schema extractor. The score $score_j$ is calculated based on the ratio of redundant elements to total elements in $\hat{V}^t$, defined as:

$$score_j = \begin{cases} 0 & if\ V^t = \hat{V}^t \\ \frac{|\hat{V}^t - V^t|}{|\hat{V}^t|} & if\ V^t \subset \hat{V}^t \\ 1 & if\ V^t \not\subset \hat{V}^t \end{cases} \quad (11)$$

$$TRS@s = \frac{1}{D}\sum_{j=1}^{D} score_j \quad (12)$$

where $|\cdot|$ denotes the number of elements in the set. $TRS@s$ is obtained by averaging $score_j$, where D represents the total number of samples. For $CRS@s$, we use a similar approach, combining the set of indices of columns included in the ground truth SQL with the set of indices of columns extracted by the schema extractor to compute the score $score_{i,j}$. Here, $score_{i,j}$ represents the redundancy of the set of extracted columns in the i^{th} table referred to the j^{th} sample. By summing these scores, we can obtain the $CRS@s$ for column name extraction, where N_j denotes the number of tables contained in the j^{th} sample's database.

$$CRS@s = \frac{1}{\sum_{j=1}^{D} N_j}\sum_{j=1}^{D}\sum_{i=1}^{N_j} score_j^i \quad (13)$$

Implementation Details To generate descriptive annotations for the database schemas, we utilized the GPT-3.5-turbo model[1] and provided detailed input prompts in Appendix B.3. During the training of the schema extraction model, we employed the AdamW optimizer (Loshchilov and Hutter, 2019). For the training phase of the SQL generator, we used the transformers library for LoRA fine-tuning (Hu et al., 2021). The specific configurations were as follows: LoRA rank was set to 32, LoRA alpha was set to 64, and LoRA dropout was set to 0.1. The batch size for training was set to 6.

In the experimental configuration of the Track-SQL framework, systematic settings were implemented for three key aspects: schema linking stability, model input constraints, and context management. The schema item sequence perturbation mechanism applies random reshuffling and irrelevant item insertion to schema items with probability 0.15 during training, while employing dynamic filtering with an extraction threshold $s = 0.1$ during inference. To accommodate the 512-token input limit of the Roberta base model, a column-level schema segmentation strategy divides large databases into column-unit-based sub-schema sets that conform to length constraints. For context expansion management, we designed:

- A fixed-length sliding window ($L_w = 5$)
- A queue-based update rule: When context unit c_t arrives, append it if window capacity permits; otherwise remove the earliest unit c_{t-L_w} before inserting c_t

All experiments were conducted on a high-performance server configured with an NVIDIA A800 (80GB) GPU, a Hygon C86 7390 32-core processor, 2TB of memory, and the Ubuntu 22.04.3 LTS operating system.

3.2 Main Results

During the experimental process involving multiple turns of SQL generation, we selected three representative models of 7B scale: CodeLlama[2], DeepSeek[3], and Mistral[4]. We directly utilized the multi-turn questions and their database schemas from the SparC and CoSQL datasets as inputs to train and infer using these 7B-scale models, thereby obtaining baseline results. Subsequently, we applied the Track-SQL method to optimize the inputs and retrained and inferred with the models. By comparing the experimental results under the

[1]https://openai.com/

[2]https://huggingface.co/codellama/CodeLlama-7b-Instruct-hf

[3]https://huggingface.co/deepseek-ai/deepseek-coder-6.7b-instruct

[4]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

Model	SparC						CoSQL					
	QM			IM			QM			IM		
	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS
In-Context Learning Approach												
ACT-SQL (Zhang et al., 2023)	51.0	63.8	56.9	24.4	38.9	29.6	46.0	63.7	55.2	13.3	30.7	21.5
CoE-SQL (Zhang et al., 2024b)	56.0	70.3	63.3	36.5	50.5	41.9	52.4	69.6	60.6	23.9	39.6	30.4
Fine-tuned Model												
HIE-SQL + GraPPA (Zheng et al., 2022)	64.7	-	-	45.0	-	-	56.4	-	-	28.7	-	-
RASAT + PICARD (Qi et al., 2022)	67.7	73.3	-	49.1	54.0	-	**58.8**	67.0	-	27.0	39.6	-
QDA-SQL (Sun et al., 2024)	61.3	-	-	44.1	-	-	57.3	-	-	**30.0**	-	-
Ours												
SFT Codellama 7B	59.18	67.99	61.51	36.96	46.68	39.33	50.54	60.17	54.51	17.74	28.66	23.54
+ Track-SQL	61.26	70.07	63.75	43.36	52.13	45.73	53.82	66.73	59.18	24.57	37.88	29.01
SFT Deepseek 7B	64.33	71.40	65.08	43.36	50.71	43.36	54.71	66.03	58.88	23.20	34.12	26.96
+ Track-SQL	65.17	**75.39**	**69.16**	46.44	**57.81**	**50.71**	58.19	70.60	62.26	28.67	43.67	32.76
SFT Mistral 7B	64.17	70.82	65.58	43.60	52.13	45.49	56.20	64.94	59.68	24.57	34.81	29.01
+ Track-SQL	**65.41**	73.23	67.83	**46.91**	54.73	48.57	57.69	**71.10**	**65.54**	27.30	**45.05**	**36.17**

Table 1: Performance of Track-SQL and previous works on the SparC and CoSQL dev set.

Track-SQL framework with the baseline results, we obtained the detailed comparative data presented in Table 1.

In terms of single-turn and multi-turn evaluation metrics, the 7B-scale models under the Track-SQL framework significantly outperformed the baseline models, validating the effectiveness of our input optimization method. Specifically, under the DeepSeek 7B base model, Track-SQL improved the single-turn EX metric by 3.99% and the TS metric by 4.08% on the SparC development set; it also improved the multi-turn EX metric by 7.1% and the TS metric by 7.35%. Similar improvements were confirmed within the CoSQL dataset. These results indicate that the Track-SQL method is highly effective in handling multi-turn interactive question answering tasks.

We also categorized other multi-turn Text-to-SQL research efforts according to different inference methodologies into two classes: In-context learning methods and fine-tuning methods. Compared to these methods, Track-SQL achieved the best performance on the SparC and CoSQL development sets, surpassing previous In-context learning and fine-tuning techniques under both single-turn and multi-turn evaluation metrics. Specifically, compared to In-context learning methods, such as ACT-SQL and CoE-SQL which rely on GPT 3.5, the fine-tuning experimental results under the Track-SQL framework with 7B-scale models were superior. Unlike In-context learning methods which can be influenced by historical questions, fine-tuning methods are better at focusing on the core of the current turn's question. Moreover, even though RASAT+PICARD combines beam search strategies for SQL correction, Track-SQL achieved notably higher multi-turn inference accuracies on the SparC and CoSQL development sets without employing any SQL post-processing techniques.

	QM		IM	
	EX	**TS**	**EX**	**TS**
Track-SQL	**75.39**	**69.16**	**57.81**	**50.71**
w/o SESE	68.57(↓ 6.82)	62.42(↓ 6.74)	51.42(↓ 6.39)	45.26(↓ 5.45)
w/o ACID	74.56(↓ 0.83)	67.91(↓ 1.25)	56.39(↓ 1.42)	48.57(↓ 2.14)
w/o SACE	72.73(↓ 2.66)	67.16(↓ 2)	51.89(↓ 5.92)	44.78(↓ 5.93)
w/o SACE & SESE	71.40(↓ 3.99)	65.08(↓ 4.08)	50.71(↓ 7.10)	43.36(↓ 7.35)

Table 2: Ablation result on SparC dev set.

	QM		IM	
	EX	**TS**	**EX**	**TS**
Track-SQL	**70.60**	**62.26**	**43.67**	**32.76**
w/o SESE	64.94(↓ 5.66)	58.98(↓ 3.28)	36.86(↓ 6.81)	31.74(↓ 1.02)
w/o ACID	68.22(↓ 2.38)	61.37(↓ 0.89)	39.24(↓ 4.43)	32.42(↓ 0.34)
w/o SACE	68.81(↓ 1.79)	61.46(↓ 0.8)	37.88(↓ 5.79)	29.01(↓ 3.75)
w/o SACE & SESE	66.03(↓ 4.57)	58.88(↓ 3.38)	34.12(↓ 9.55)	26.96(↓ 5.8)

Table 3: Ablation result on CoSQL dev set.

3.3 Ablation Studies

To further validate the effectiveness of the proposed method, we conducted detailed ablation studies on the development sets of Sparc and CoSQL to evaluate the importance of each module. Tables 2 and 3 present the experimental results based on the Deepseek 7B model. Additionally, in Appendix

A.1.3, we provide the results of extra ablation experiments conducted using models of different 7B parameter scales, thereby confirming that the performance improvements attributed to each module are not due to the randomness associated with any particular 7B model. To further analyze how each module enhances model performance, we categorized the datasets according to the complexity of the SQL statements and the number of dialogue interactions, and conducted targeted ablation experiments accordingly. The detailed results are shown in the appendix A.1.1 and A.1.2. By labeling the table columns included in the ground truth SQLs of the multi-turn conversation datasets, we performed ablation experiments on the schema extractor. The results are shown in Table 4, confirming the effectiveness of this module in reducing redundancy during the schema item extraction process.

Effect of Semantic-enhanced Schema Extractor We employ a two-phase evaluation metric to assess the effectiveness of SESE. As shown in Table 4, the introduction of the semantic enhancement module in the extractor significantly reduces redundancy rates in table and column extraction. This improvement is particularly pronounced on the CoSQL dataset, which contains a higher proportion of multi-turn ambiguous dialogue samples compared to the SparC dataset, posing more stringent challenges for schema understanding mechanisms. Through LLM-based contextual semantic expansion and annotation embedding mechanisms, this module effectively mitigates schema comprehension biases in complex dialogue turns. Specifically, on CoSQL dev set, TRS@0.5 and CRS@0.5 decrease by approximately 0.88% and 4.03% respectively, demonstrating substantial progress in schema item localization accuracy. Additional ablation study results under alternative classification metrics are provided in Appendix A.2, further validating the critical role of the semantic enhancement module.

During SQL generation, when removing SESE module, QM-TS and IM-TS metrics decline by 6.74% and 5.45% respectively. This not only confirms the strong task coupling between schema extraction and SQL generation, but also verifies our core hypothesis: optimizing the precision-redundancy balance in schema extraction can effectively enhance multi-turn SQL generation performance, thereby establishing the theoretical feasibility of our proposed approach.

	SparC		CoSQL	
	TRS@0.5	CRS@0.5	TRS@0.5	CRS@0.5
Track-SQL	**8.50**	21.46	**12.74**	**25.90**
w/o com-enh	9.57	**21.05**	13.62	29.93
w/o com&col-enh	9.44	28.18	13.38	27.42

Table 4: Ablation results of SESE. com-enh represents comment-enhanced layer, and col-enh represents column-enhanced layer used in RESDSQL.

Effect of All-Column Intent Detection When the All-Column Intent Detection (ACID) strategy is eliminated, all metrics for Tables 2 and 3 exhibit slight performance degradation. Combined with the case study in Appendix A.3, this verifies that the ACID strategy possesses limited yet discernible capability in enhancing the model's recognition of user all-column intentions. In the inference experiments on the COSQL dev set, removing this module resulted in decreases of 0.89% and 0.34% in QM-TS and IM-TS respectively, while more significant drops of 2.38% and 4.43% were observed in QM-EX and IM-EX. This phenomenon suggests that the EX evaluation metrics contain a higher proportion of false positive samples during their calculation process. The optimization effect of this strategy on the overall performance of the Track-SQL framework demonstrates metric sensitivity characteristics. Final experimental data confirm that this module yields only limited performance gains in real-world application scenarios.

Effect of Schema-aware Context Extractor The strategy of employing historical SQL as prompt generation exhibits inherent limitations: when structural errors exist in historically generated statements, erroneous prompts may trigger cascading error propagation. To mitigate this, SACE incorporates syntactic error detection in historical SQL generation to reduce error transmission. Ablation experiments demonstrate that removing SACE module resulted in performance degradation of 5.92% and 5.93% on IM-EX and IM-TS metrics respectively in the SparC validation set, exhibiting more pronounced degradation compared to QM metrics. This phenomenon indicates that under relatively high model accuracy conditions, the SACE module significantly enhances the completeness of multi-turn SQL generation. Notably, the marginal effects revealed in Appendix A.1.1 demonstrate that when query difficulty escalates to EXTRA levels, SACE inadvertently induces error propagation. This revelation exposes the module's limitations, thereby

	SESE		SQL Generator	
	Train	Inference	Train	Inference
SparC	30.9±2.8(h)	0.20(s)	1.5±0.2(h)	1.15(s)
CoSQL	27.5±2.4(h)	0.21(s)	1.6±0.3(h)	1.15(s)

Table 5: Time performance of the Track-SQL framework, **Train** employs the number of hours required to achieve the optimal model as the performance metric, while **Inference** bases its evaluation on the time consumed to perform inference on a single batch.

suggesting that achieving more robust multi-turn SQL generation systems requires the implementation of systematic validation strategies.

3.4 Execution Time Analysis

To evaluate the practical performance of the Track-SQL framework, we conducted latency analysis and training efficiency validation on its core components. As shown in Table 5, experimental results indicate that under typical multi-turn database query scenarios, the system achieves an end-to-end response time of 1.35 seconds (0.20 seconds for SESE and 1.15 seconds for SQL Generator), satisfying the latency requirements for real-time interactive systems.Regarding training efficiency, under standard experimental configurations on the SparC dataset, the schema extractor achieves optimal performance after 30.9 hours of training, while the SQL generator converges to its optimal state in approximately 1.5 hours. This demonstrates the framework's significant efficiency advantages in computational resource utilization.

4 Related Work

Schema Linking Schema linking effectively reduces errors caused by schema misinterpretation, thereby enhancing the performance of text-to-SQL conversion. Currently, various methods focus on improving schema linking. For instance, DIN-SQL (Li et al., 2024b) utilizes GPT to identify relevant database elements, while DTS-SQL (Pourreza and Rafiei, 2024) employs a specialized model for efficient schema extraction. RESDSQL (Li et al., 2023) and CodeS (Li et al., 2024a) adopt strategies that assess the relevance between schemas and queries for sorting and filtering. Other methods, such as C3 (Dong et al., 2023), CHESS (Talaei et al., 2024), and MCS-SQL (Lee et al., 2024), adopt a step-by-step linking approach, first filtering out relevant tables and then selecting matching columns. Existing methods perform well in single-turn Text-to-SQL tasks, but they are inadequate in handling multi-turn dialogues and schema name ambiguities. This paper focuses on dynamic schema linking in multi-turn Text-to-SQL, with a particular emphasis on resolving schema name ambiguities. Our proposed method reduces redundant links and improves system performance.

Multi-turn Text-to-SQL Multi-turn text-to-SQL tasks more closely resemble real-world applications, allowing users to progressively refine their questions and adjust their requirements through dialogue. Research in this field includes ISTSQL (Wang et al., 2021), which improves accuracy by tracking the states of database schemas and SQL keywords; MIGA (Fu et al., 2023) utilizes multi-task learning to integrate information on reference relationships and schema links; CoE-SQL (Zhang et al., 2024b) tracks user intent by serializing changes in SQL queries. Inspired by copying mechanisms, methods like EditSQL (Zhang et al., 2019), refer to previous-turn SQL information. R2SQL (Hui et al., 2021) introduces a memory decay mechanism to simulate changes in the database schema. TP-Link (Liu et al., 2024) models word-level coreference to resolve complex coreference and ellipsis issues. These studies aim to support the Track-SQL framework by capturing dependencies between multi-turn text-to-SQL interactions. The framework improves the handling of multi-turn co-reference issues and enhances system performance and adaptability by integrating dynamic schema element awareness with question semantic information.

5 Conclusion

In this paper, we proposed the Track-SQL framework to address the challenges of multi-turn Text-to-SQL tasks, focusing on dynamic schema linking and effective utilization of historical context. Track-SQL integrates a *Semantic-enhanced Schema Extractor* and a *Schema-aware Context Extractor* to precisely capture schema and contextual changes, improving the system's ability to adapt to evolving user interactions. The core idea lies in establishing the association model between user intent, schemas, and contextual information in advance, thereby reducing the difficulty of recognizing changing intents during the SQL generation process. Experimental results indicate that the Track-SQL possesses significant advantages and effectiveness.

Limitations

In the schema extraction phase, using RoBERTa as the base model limits the maximum window length to 512, which increases the training time for handling large volumes of text data; even with an A800 (80G) GPU-equipped machine, training on the SparC dataset takes two days. We attempted to replace RoBERTa with a decoder-only model that supports a larger window size for classifier training, but the results were unsatisfactory. While the Track-SOL framework has advanced dynamic schema linking and context information extraction, its efficacy in extremely complex multi-turn dialogues and highly dynamic database schemas remains to be validated. Future efforts will focus on enhancing the framework's robustness to ensure high-performance and stability across diverse application scenarios.

Acknowledgments

This research was financially supported by the Natural Science Foundation of China (62406078), the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) under Grant No. GML-KF-24-23, the National Science and Technology Major Project (2021ZD0111501), the National Science Fund for Excellent Young Scholars (62122022), the Natural Science Foundation of China (U24A20233, 62476163, 62206064, 62206061), the major key project of PCL (PCL2021A12), the Guangdong Basic and Applied Basic Research Foundation (2023B1515120020), and the Collaborative Education Project of the Ministry of Education (202407). This research was enabled by the computational resources and support of the High Performance Computing Platform at the School of Computer Science, Guangdong University of Technology.

References

Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zheng Cao, Weijie Li, Fei Huang, Luo Si, and Yongbin Li. 2022. STAR: SQL guided pre-training for context-dependent text-to-SQL parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1235–1247, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, lu Chen, Jinshu Lin, and Dongfang Lou. 2023. C3: Zero-shot text-to-sql with chatgpt.

Yingwen Fu, Wenjie Ou, Zhou Yu, and Yue Lin. 2023. Miga: a unified multi-task generation framework for conversational text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12790–12798.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 13116–13124.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024a. Codes: Towards building open-source language models for text-to-sql.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ziqiang Liu, Shujie Li, Zefeng Cai, Xiangyu Li, Yunshui Li, Chengming Li, Xiping Hu, Ruifeng Xu, and Min Yang. 2024. TP-link: Fine-grained pre-training for text-to-SQL parsing with linking information. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16686–16697, Torino, Italia. ELRA and ICCL.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Mohammadreza Pourreza and Davood Rafiei. 2024. Dts-sql: Decomposed text-to-sql with small large language models.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3215–3229, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

T-YLPG Ross and GKHP Dollár. 2017. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988.

Yinggang Sun, Ziming Guo, Haining Yu, Chuanyi Liu, Xiang Li, Bingxuan Wang, Xiangzhan Yu, and Tiancheng Zhao. 2024. Qda-sql: Questions enhanced dialogue augmentation for multi-turn text-to-sql.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis.

Run-Ze Wang, Zhen-Hua Ling, Jingbo Zhou, and Yu Hu. 2021. Tracking interaction states for multi-turn text-to-sql semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13979–13987.

Dongling Xiao, Linzheng Chai, Qian-Wen Zhang, Zhao Yan, Zhoujun Li, and Yunbo Cao. 2022. Cqr-sql: Conversational question reformulation enhanced context-dependent text-to-sql parsers. *arXiv preprint arXiv:2205.07686*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Hanchong Zhang, Ruisheng Cao, Hongshen Xu, Lu Chen, and Kai Yu. 2024a. Coe-sql: In-context learning for multi-turn text-to-sql with chain-of-editions.

Hanchong Zhang, Ruisheng Cao, Hongshen Xu, Lu Chen, and Kai Yu. 2024b. CoE-SQL: In-context learning for multi-turn text-to-SQL with chain-of-editions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6487–6508, Mexico City, Mexico. Association for Computational Linguistics.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. HIE-SQL: History information enhanced network for context-dependent text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2997–3007, Dublin, Ireland. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th*

Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 207–212, Berlin, Germany. Association for Computational Linguistics.

A Appendix A

This section provides additional experimental data regarding the proposed Track-SQL framework, detailing the ablation study results categorized by different large language models, SQL complexity, and number of dialogue turns. These results further confirm the effectiveness and performance of the designed schema extractor and SQL generator. Specifically, in section A.1.1, we present the ablation study results of Track-SQL across different levels of SQL difficulty. Section A.1.2 evaluates the framework's performance under varying numbers of dialogue turns. In Section A.1.3, we selected three mainstream open-source LLM models and conducted ablation experiments on the SparC and CoSQL datasets. Section A.2 reports the performance of the *Schema Extractor* across multiple classification metrics. Section A.3, we conduct an in-depth analysis of specific ablation study cases. Finally, Section A.4 reports the performance of Track-SQL in terms of time and cost metrics.

A.1 Further Ablation Studies

A.1.1 Ablation by difficulty

In the Spider (Yu et al., 2018) dataset, SQL queries are categorized into four levels: Easy, Medium, Hard, and Extra. The difficulty level is determined by the number of SQL components, selections, and conditions. Queries containing more SQL keywords (such as *GROUP BY*, *ORDER BY*, *INTERSECT*, *nested subqueries*, *multi-column selections*, and *aggregation operations*) are considered to be of higher difficulty. Specifically, if a query includes more than two *SELECT* columns, more than two *WHERE* conditions, uses *GROUP BY* on two or more columns, or contains *EXCEPT* keywords or nested queries, it is classified as **Hard**. Queries that further increase complexity beyond this are classified as **Extra**. The SparC and CoSQL dev sets also follow this standard for categorizing sample difficulty.

We present the ablation results of Track-SQL on SparC and CoSQL dev sets in Tables 6 and 7, categorized by difficulty. The results indicate:

1. As the complexity of SQL reasoning increases, the SACE method—using historical base SQL to infer the SQL for the current problem—shows limitations, particularly when handling high-difficulty samples in the CoSQL dev set, where the base SQL might mislead large language models (LLMs). However, for medium-difficulty SQL, this method positively impacts SQL generation.

2. Overall, the ACID module provides a minor improvement in SQL generation performance across all difficulty levels. Combined with case analysis (A.3), this demonstrates that the ACID module successfully enhances the recognition of all columns involved in user intent, regardless of sample difficulty.

3. The SESE module significantly improves SQL generation performance across different difficulty levels in both datasets. This is attributed to the critical role of schema linking in various types of SQL queries. Accurate schema information is fundamental to achieving high-precision SQL generation.

A.1.2 Ablation by turn

Table 8 and Table 9 present the ablation study results of Track-SQL under different numbers of interaction turns. A comprehensive analysis of the experimental results on the SparC and CoSQL datasets indicates that the SACE module continues to perform well in multi-turn interaction scenarios. Specifically, in the CoSQL dataset, when the number of interaction turns exceeds four, the SACE module achieves a performance gain of 2.8%. In the SparC dataset, when the number of interaction turns reaches four, there is a 2.3% improvement. Additionally, the effectiveness of the ACID module and the SESE module is not affected by the number of interaction turns, and they consistently enhance the performance of the SQL generator across various samples.

A.1.3 Ablation experiment performance under diverse language models

Figures 2, 3 and 4 present the ablation study results of the Track-SQL framework's modules on the Codellama, Mistral, and DeepSeek models. The data on the left and right sides of the charts correspond to the test results from the SparC and CoSQL development sets, respectively. The evaluation metric used is the TS score under the IM environment. The results show that on various 7B benchmark

	Easy (483)			Medium (441)			Hard (145)			Extra (134)		
	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS
Track-SQL	**81.8**	**84.9**	**83.0**	**64.6**	**72.8**	**66.4**	**44.8**	**64.8**	**53.8**	**35.1**	**54.5**	**38.8**
w/o SESE	77.6	78.7	77.6	56.5	68.0	59.4	42.8	59.3	46.2	30.6	44.0	35.1
w/o ACID	79.3	85.5	82.8	62.8	72.8	65.8	44.8	63.4	49.7	39.6	53.0	41.0
w/o SACE	79.5	84.1	81.8	63.9	71.9	66.4	39.3	60.0	48.3	32.8	48.5	37.3
w/o SACE & SESE	79.1	80.7	78.5	63.3	71.7	64.2	43.4	59.3	46.9	37.3	50.0	39.6

Table 6: Ablation results of Track-SQL on SparC validation set by difficulty. The numbers in brackets () indicate the number of samples.

	Easy (417)			Medium (320)			Hard (163)			Extra (107)		
	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS
Track-SQL	**80.6**	**85.4**	**82.5**	**51.6**	**67.5**	**55.6**	**36.2**	**59.5**	**46.0**	**24.3**	**39.3**	**28.0**
w/o SESE	75.1	79.4	76.7	50.0	60.9	51.6	36.2	53.4	46.0	25.2	38.3	31.8
w/o ACID	78.2	83.5	81.1	52.8	65.0	56.2	31.3	55.8	44.2	22.4	37.4	26.2
w/o SACE	78.2	83.5	80.3	50.0	61.6	51.9	35.6	62.0	48.5	29.0	43.9	36.4
w/o SACE & SESE	75.8	77.7	75.8	48.4	60.9	51.6	38.0	66.3	52.8	16.8	35.5	24.3

Table 7: Ablation results of Track-SQL on CoSQL validation set by difficulty. The numbers in brackets () indicate the number of samples.

	Turn 1 (422)			Turn 2 (422)			Turn 3 (270)			Turn 4 (88)			Turn > 4 (1)		
	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS
Track-SQL	**74.2**	**79.6**	**76.5**	**65.9**	**75.6**	**67.8**	**57.8**	**67.4**	**59.6**	**51.1**	**68.2**	**60.2**	**0**	**100**	**100**
w/o SESE	70.1	75.6	71.8	60.9	69.2	61.4	51.1	60.4	53.0	40.9	58.0	52.3	0	0	0
w/o ACID	72.5	80.6	77.3	64.2	75.1	66.6	56.7	67.8	59.3	54.5	64.8	56.8	0	0	0
w/o SACE	72.3	79.1	75.8	63.0	72.3	64.9	55.9	65.9	59.6	51.1	65.9	60.2	0	0	0
w/o SACE & SESE	71.3	77.0	72.7	63.7	70.9	64.0	58.9	67.0	58.5	51.1	61.4	54.5	0	0	0

Table 8: Ablation results of Track-SQL on SparC validation set by turn. The numbers in brackets () indicate the number of samples.

	Turn 1(293)			Turn 2 (285)			Turn 3 (244)			Turn 4 (114)			Turn > 4 (71)		
	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS	EM	EX	TS
Track-SQL	**64.8**	**76.5**	**70.3**	**58.2**	**72.3**	**63.9**	**56.6**	**68.0**	**58.2**	**55.3**	**64.9**	**55.3**	**40.8**	**57.7**	**47.9**
w/o SESE	61.4	70.0	64.5	57.9	67.4	62.8	50.8	64.3	54.5	52.6	56.1	51.8	42.3	50.7	47.9
w/o ACID	65.2	73.4	68.6	55.1	69.1	62.8	54.5	66.8	58.6	51.8	62.3	53.5	42.3	57.7	47.9
w/o SACE	66.2	75.8	69.3	52.6	66.7	59.6	57.8	69.3	59.4	52.6	64.0	58.8	42.3	54.9	47.9
w/o SACE & SESE	62.8	71.0	65.2	55.1	68.1	62.5	52.5	65.2	55.7	50.0	58.8	50.0	35.2	52.1	43.7

Table 9: Ablation results of Track-SQL on CoSQL validation set by turn. The numbers in brackets () indicate the number of samples.

models, the components of the Track-SQL framework consistently demonstrate significant effectiveness, particularly the SESE and SACE modules, whose outstanding performance is especially noteworthy.

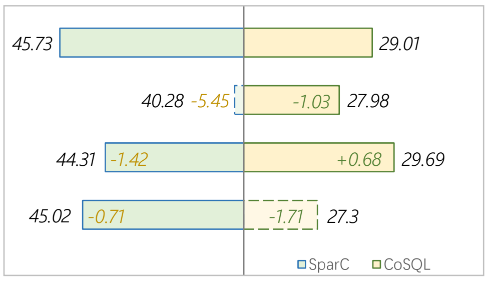

Figure 2: The results of the ablation study on the Codellama 7B+Track-SQL model on the SparC and CoSQL dev sets (calculated using the multi-turn TS metrics).

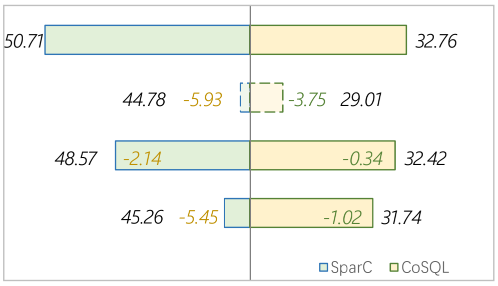

Figure 3: The results of the ablation study on the DeepSeek 7B+Track-SQL model on the SparC and CoSQL dev sets (calculated using the multi-turn TS metrics).

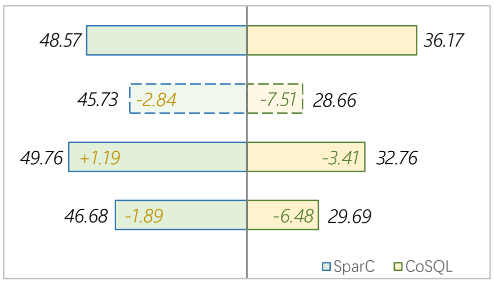

Figure 4: The results of the ablation study on the Mistral 7B+Track-SQL model on the SparC and CoSQL dev sets (calculated using the multi-turn TS metrics).

A.2 Detailed evaluation of the Semantic-Enhanced Schema Extractor

A more comprehensive evaluation of the extractor is shown in Table 10 and 11. Tables 10 and 11 present the ablation study results of our proposed semantic-enhanced schema extractor on the SparC and CoSQL datasets, using overall classification accuracy as the evaluation metric. This evaluation method treats the classification probabilities of all columns as a unified whole, without distinguishing columns based on the table structure, nor imposing penalties for missing or redundant schemas. Therefore, while this measure of classification accuracy is somewhat rough, it still holds some reference value. The data from both tables indicate that the classification accuracy decreases to varying degrees when the semantic enhancement module is removed; further removal of the column enhancement components used in RESDSQL leads to an additional drop in accuracy. This suggests that the semantic enhancement module has a significant improvement effect on the base classifier and supports dynamic schema linking functionality.

A.3 Case Study

In ablation experiments conducted on the SparC dataset, we randomly selected several results generated by the model and will provide a detailed analysis of these samples in this section. All selected samples are listed in Table 12, and all examples are based on the deepseek-coder-6.7b model.

w/o SACE Case Analysis In the first case, there is a notable correlation between question#2 and question#3; with the support of SQL_{base}, Track-SQL is able to generate valid SQL statements that include correct foreign key joins. Specifically, during the second turn of reasoning, the model successfully generates SQL_{base}, which simplifies the task of generating the foreign key join SQL statement in the third turn. However, when the SACE module is removed, this cumulative effect no longer exists, leading to generated SQL statements that fail to execute the correct foreign key join operations. In the second case, when Track-SQL enters the reasoning phase for the third question, it leverages the SQL_{base} generated from the first question to achieve precise predictions of schemas.

w/o ACID Case Analysis From the third and fourth examples, it can be observed that removing the ACID module diminishes the model's ability to identify whether the user needs to display all columns of a table, resulting in errors in the generation of SQL statements.

w/o SESE Case Analysis The fifth and sixth examples demonstrate that the removal of the SESE module leads to a decrease in accuracy when generating schemas. Specifically, in the fifth example, the model-generated result lacks the "*Continent*"

	Table-Weighted Avg			Table-Macro Avg			Column-Weighted Avg			Column-Macro Avg		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SESE	96.48	96.43	96.45	95.74	**96.42**	96.07	**97.09**	**96.93**	**96.99**	90.95	**94.34**	**92.55**
w/o com-enh	96.18	96.11	96.12	95.32	96.16	95.72	96.91	96.68	96.76	90.11	94.24	92.03
w/o com-enh & col-enh	**96.58**	**96.57**	**96.57**	**96.09**	96.31	**96.20**	96.70	96.67	96.69	**91.31**	92.05	91.68

Table 10: The general precision index score of the semantic-enhanced schema extractor under the SparC dataset

	Table-Weighted Avg			Table-Macro Avg			Column-Weighted Avg			Column-Macro Avg		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SESE	**95.33**	**95.28**	**95.30**	**94.29**	**94.87**	**94.57**	**96.85**	96.71	96.77	89.70	**92.65**	**91.11**
w/o com-enh	95.32	95.30	95.31	94.47	94.66	94.57	96.81	**96.79**	**96.80**	**90.90**	91.24	91.07
w/o com-enh & col-enh	94.65	94.39	94.45	92.83	94.64	93.64	96.40	95.98	96.13	86.97	92.78	89.59

Table 11: The general precision index score of the semantic-enhanced schema extractor under the CoSQL dataset

column; whereas in the sixth example, the model incorrectly generates the "*tv_channel.series_name*" column. In contrast, the SQL_{base} model reduces instances of erroneous generation and missing columns.

A.4 Resource overhead experiment

A.4.1 Quantitative analysis of time overhead

In this section, we conducted detailed experiments on Track-SQL in the time dimension. As shown in Table 13. This includes the average inference time and training time of the schema extractor and the SQL generator, and all experiments were carried out on the same hardware specified in the main text.

Specifically, the average inference time for processing each sample using the Track-SQL framework is 1.352 seconds, which is significantly faster than context learning-based methods like CoE-SQL, which requires an average of 3.544 seconds. This highlights the obvious efficiency advantage of Track-SQL inference. In addition, when the batch size is 6, the schema extractor and the SQL generator reach the best model state within a reasonable training duration, as shown in Table 6. It is worth noting that the inference time was measured on the validation set, while the training time was based on the SparC and CoSQL training sets, which include 9,025 and 7,343 entries respectively.

A.4.2 Memory Costs of Training and Inference

As shown in Table 14, the Track-SQL framework is fully capable of performing inference on a 24G graphics card, demonstrating its characteristic of maintaining low cost while maintaining high accuracy. In terms of model training, by reducing the batch size, Track-SQL can also perform training tasks on low-capacity graphics cards.

B Appendix B

In this section, we provide additional details about the methods described in the paper.
- In Section B.1, we described the strategy proposed by Track-SQL for the input window limitation of Schema Extractor.
- In Section B.2, we describe the loss function used for the Semantic-enhanced Schema Extractor. We adopted the focal loss used by REDSQL.
- In Section B.3, we describe the prompts used to generate annotations for database schemas, which serve as input to the GPT3.5-turbo model.
- In Section B.4, we detail the input and output formats used for supervised fine-tuning of the SQL expert model.

B.1 How does Track-SQL solve the window limitation problem of the schema item extractor?

Track-SQL incorporates two effective strategies to address this challenge:

1. **Schema Segmentation**: The schema extractor performs column-level segmentation, ensuring that even large-scale databases are processed efficiently within RoBERTa's token limit. This strategy allows our framework to handle schemas with up to 83 columns with only a marginal increase in inference time (see Table 15 for details).

Case#1	Question#1 Question#2 Question#3	What is every student's id? Of those ids, which correspond to those who own cats as pets? List all the other ids.
	Gold	**SELECT stuid FROM student EXCEPT SELECT T1.stuid FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T3.petid = T2.petid WHERE T3.pettype = 'cat'**
	w/o SACE	SELECT stuid FROM has_pet except SELECT stuid FROM has_pet JOIN pets ON has_pet.petid = pets.petid WHERE pettype = 'cat'
	Track-SQL	SELECT stuid FROM student except SELECT student.stuid FROM student JOIN has_pet ON student.stuid = has_pet.stuid JOIN pets ON has_pet.petid = pets.petid WHERE pets.pettype = 'cat'
	SQL_{base}	SELECT student.stuid FROM student JOIN has_pet ON student.stuid = has_pet.stuid JOIN pets ON has_pet.petid = pets.petid WHERE pets.pettype = 'cat'
Case#2	Question#1 Question#2 Question#3	How many car models are produced in total? How many in Germany? How about in Japan?
	Gold	**SELECT count(*) FROM MODEL_LIST AS T1 JOIN CAR_MAKERS AS T2 ON T1.Maker = T2.Id JOIN COUNTRIES AS T3 ON T2.Country = T3.CountryId WHERE T3.CountryName = 'japan'**
	w/o SACE	SELECT COUNT(model_list.model) FROM model_list JOIN car_names ON model_list.model = car_names.model JOIN countries ON car_names.makeid = countries.countryid WHERE countries.countryname = 'japan'
	Track-SQL	SELECT COUNT(*) FROM model_list JOIN car_makers ON model_list.maker = car_makers.id JOIN countries ON car_makers.country = countries.countryid WHERE countries.countryname = 'japan'
	SQL_{base}	SELECT COUNT(*) FROM model_list
Case#3	Question	What are all the nations?
	Gold	**SELECT * FROM country**
	w/o ACID	SELECT **name** FROM country
	Track-SQL	SELECT * FROM country
Case#4	Question	Show all the available features.
	Gold	**SELECT * FROM Other_Available_Features**
	w/o ACID	SELECT **feature_name** FROM other_available_features
	Track-SQL	SELECT * FROM other_available_features
Case#5	Question	Which continents have an average life expectancy less than age 72?
	Gold	**SELECT sum(Population) , avg(LifeExpectancy), Continent FROM country GROUP BY Continent HAVING avg(LifeExpectancy) < 72**
	w/o SESE	SELECT **avg(lifeexpectancy), sum(population)** FROM country GROUP BY continent HAVING avg(lifeexpectancy) < 72
	Track-SQL	SELECT avg(lifeexpectancy), sum(population), continent FROM country GROUP BY continent HAVING avg(lifeexpectancy) < 72
Case#6	Question#1 Question#2	Tell me the director of the cartoon named "Day of the Dark Knight!". What is the channel of this cartoon?
	Gold	**SELECT Channel FROM Cartoon WHERE Title = "Day of the Dark Knight!"**
	w/o SESE	SELECT **tv_channel.series_name** FROM **cartoon JOIN tv_channel ON cartoon.channel = tv_channel.id** WHERE cartoon.title = 'Day of the Dark Knight!'
	Track-SQL	SELECT channel FROM cartoon WHERE Title = 'Day of the Dark Knight!'

Table 12: Case analysis of ablation experiments on the SParC dataset

	SESE(Inference)		SQL Generator(Inference)		SESE(Train)		SQL Generator(Train)	
	Total time(s)	Avg Time(s)	Total time(s)	Avg Time(s)	Total time(h)	Avg Time(s)	Total time(h)	Avg Time(s)
SparC	240.348±1.45	0.20±0.00013	1386.98±2.13	1.152±0.025	30.93h±2.78	3.88±0.43	1.45±0.21	11.23±2.31
CoSQL	214.456±2.56	0.21±0.0012	1170.41±1.56	1.15±0.012	27.49h±2.43	3.92±0.29	1.61±0.33	11.43±2.51

Table 13: Inference time performance of the Track-SQL framework. Avg Time(s) in terms of training represents the time consumption of a single batch, and Total time(h) represents the time required to obtain the best model. The SQL generator is based on Deepseek 7B

	SESE(Inference)	SQL Generator(Inference)	SESE(Train)	SQL Generator(Train)
Graphics Memory(GB)	2.235	16.477	20.997	62.194

Table 14: Memory Costs of Training and Inference in the Track-SQL Framework. In the training stage, the batch size is set to 6, and in the inference stage, the batch size is set to 1.The SQL generator is based on Deepseek 7B

2. **Sliding Window Context Management**: To address the expanding context in multi-turn interactions, we adopt a sliding window strategy with a length of 5. This approach effectively manages historical Question & SQL context, ensuring stable performance without compromising on multi-turn interaction accuracy.

These strategies demonstrate that our framework is specifically designed to handle the challenges posed by larger schemas and extensive multi-turn contexts, providing both scalability and efficiency.

B.2 Loss Function

Since an SQL query typically involves only a few tables and columns from the database, the label distribution in the training set is highly imbalanced. As a result, the number of negative samples is often several times greater than the number of positive samples, which can lead to significant training bias. To alleviate this issue, we adopt focal loss (Ross and Dollár, 2017) as our classification loss. Then, we formulate the loss function for the multi-turn schema extractor using a multi-task learning approach, where the loss function consists of table classification loss and column classification loss:

$$L_1 = \frac{1}{N}\sum_{i=1}^{N} FL(y_i, \hat{y}_i) + \frac{1}{M}\sum_{i=1}^{N}\sum_{k=1}^{n_i} FL(y_{i,k}, \hat{y}_{i,k}) \tag{14}$$

where the focal loss function is denoted by FL and y_i represents the true label of the i^{th} table. $y_i = 1$ indicates that the table is referenced by the SQL, otherwise, it is 0. $y_{i,k}$ represents the true label of the k^{th} column in the i^{th} table. Similarly, $y_{i,k} = 1$ indicates that the column is referenced by an SQL, otherwise it is 0. $\hat{y}_i$ and $\hat{y}_{i,k}$ are predicted probabilities, which are estimated based on the table embeddings and column embeddings T_i and $C_{i,k}$ through two different MLP modules:

$$\hat{y}_i = \sigma((\hat{T}_i U_1^t + b_1^t)U_2^t + b_2^t) \tag{15}$$

$$\hat{y}_{i,k} = \sigma((C_{i,k} U_i^c + b_1^c)U_2^c + b_2^c) \tag{16}$$

among them, $U_1^t, U_1^c \in \mathbb{R}^{d\times\omega}$, $b_1^t, b_1^c \in \mathbb{R}^{\omega}$, $U_2^t, U_2^c \in \mathbb{R}^{2\times\omega}$, $b_2^t, b_2^c \in \mathbb{R}^2$ are trainable parameters, and $\sigma(\cdot)$ denotes the Softmax function.

B.3 Database schema annotation generation

Table 16 above displays the LLM input sequences used for generating annotations for database columns. These sequences include prompt statements, corresponding table names, column names, column data types, and several example values randomly drawn from the database. Additionally, Table 16 below lists the LLM input format used for generating annotations for entire database tables, which includes prompt statements, target table names, and their respective column names.

B.4 Format for fine-tuning SQL generation

Table 17 shows the input and output formats used for supervised fine-tuning of the SQL generation model. Here, $E(\mathcal{S})$ represents the refined schema term sequence obtained through Semantic-enhanced Schema Extractor, SQL_{base} is the historical reasoning SQL selected by Schema-aware Context Extractor, and $\mathcal{Q}_{\leq m}$ is the combined se-

Number of Columns	13	21	30	47	57	67	83
Avg Time(s)	0.1504	0.2209	0.2441	0.3001	0.3885	0.3932	0.5201

Table 15: Inference Time of the Schema Extractor under Different Numbers of Database Columns

Input for generating annotated descriptions of database columns

You are a database schema designer who specializes in generating concise descriptions for table columns based on the provided column names, types, and sample values. The descriptions should be brief, adjective-noun phrases that reflect the nature of the data in the column.
 	
Table Name: {table_name}
Column: {column_name}
Type: {column_type}
Sample Values: {values}

Input for generating annotated descriptions of database tables
You are a database schema expert who is skilled at generating concise descriptions for database tables based on their names and column details. Your job is to create a brief, adjective-noun form description that captures the essence of the table. The description should be short and to the point, not exceeding a few words. You will be given the table name, column names, types, and sample values. Generate a descriptive phrase using this information. The table name is '{table_name}'. It has the following columns: {column_strs} Based on this information, please generate a concise description for the table.

Table 16: This table lists LLM input sequences used for generating annotations for database columns and tables

quence of historical questions and the current question. The output s_m is the target SQL.

Format for fine-tuning SQL generation
Input: You are a SQL query generator that converts multi-turn questions along with associated database schema information into corresponding SQL statements. The multi-turn questions will be concatenated using the '$\&$' symbol, and you should generate the SQL statement that answers the current turn of the question based on the provided database schema. Each database schema is provided in the following format: Table name : Column name1, Column name2 Different tables are separated by the '|' symbol, and the order of table names and column names is relevant to the current question; those appearing earlier are more closely related. Base SQL: $\{SQL_{base}\}$ database schema: $\{E(\mathcal{S})\}$ question: $\{\mathcal{Q}_{\leq m}\}$ **Output**: "' $\{s_m\}$; "' < | end_of_sentence | >

Table 17: Format for fine-tuning SQL generation