

A Case-Based Reasoning Framework for Adaptive Prompting in Cross-Domain Text-to-SQL

Chunxi Guo, Zhiliang Tian^(✉), Jintao Tang^(✉), Pancheng Wang, Zhihua Wen, Kang Yang and Ting Wang^(✉)

College of Computer, National University of Defense Technology, Changsha, China
 {chunxi, tianzhiliang, tangjintao, wangpancheng13, zhwen, yangkang, tingwang}@nudt.edu.cn

Abstract. Recent advancements in Large Language Models (LLMs), such as Codex, ChatGPT and GPT-4 have significantly impacted the AI community, including Text-to-SQL tasks. Some evaluations and analyses on LLMs show their potential to generate SQL queries but they point out poorly designed prompts (e.g. simplistic construction or random sampling) limit LLMs’ performance and may cause unnecessary or irrelevant outputs. To address these issues, we propose CBR-ApSQL, a Case-Based Reasoning (CBR)-based framework combined with GPT-3.5 for precise control over case-relevant and case-irrelevant knowledge in Text-to-SQL tasks. We design adaptive prompts for flexibly adjusting inputs for GPT-3.5, which involves (1) adaptively retrieving cases according to the question intention by de-semanticizing the input question, and (2) an adaptive fallback mechanism to ensure the informativeness of the prompt, as well as the relevance between cases and the prompt. In the de-semanticization phase, we designed Semantic Domain Relevance Evaluator (SDRE), combined with Poincaré detector (mining implicit semantics in hyperbolic space), TextAlign (discovering explicit matches), and Positector (part-of-speech detector). SDRE semantically and syntactically generates in-context exemplar annotations for the new case. On the three cross-domain datasets, our framework outperforms the state-of-the-art (SOTA) model in execution accuracy by 3.7%, 2.5%, and 8.2%, respectively.

Keywords: Large language model, Case-based reasoning, Cross-domain Text-to-SQL

1 Introduction

Text-to-SQL task aims to convert Natural Language Question (NLQ) to Structured Query Language (SQL), allowing non-expert users to obtain required information from a database [1, 2]. Under a cross-domain setting, Text-to-SQL aims to create a more general model with various domain semantic information (school, sports, restaurant, etc.) in the training set, and then generalize to

unseen domains at the inference stage. To solve cross-domain generalization concerns [3, 4], researchers are turning to encoder-decoder architecture [5, 6], which reduces the requirement for specific domain knowledge via end-to-end training. These full-data fine-tuned models still demand diverse and extensive training data to guarantee the proper training of all modules.

Large pre-trained language models (LLMs) (e.g., GPT-3 [7] and Codex [8]) encompass significantly larger data and parameters than pre-trained language models (e.g., BERT [9], RoBERTa [10], BART [11] and T5 [12]) and exhibit superior performance on a variety of tasks, including Text-to-SQL. Rajkumar et al. [13] and Liu et al. [14] evaluate LLMs’ performance in Text-to-SQL and conduct experiments in zero- and few-shot settings. Cheng et al. [15] present a neural-symbolic framework that maps the task input to a program, which incorporates symbolic components into LLMs. However, poorly designed prompts (e.g. simplistic construction [13, 14] or random sampling [15]) can hinder the performance of LLM-based models [16]. As a result, these models are unable to achieve the same effects as those fine-tuned on full datasets. Moreover, LLMs unconsciously utilize knowledge beyond the task’s scope [16], leading to the generation of unnecessary and irrelevant components [13] [14].

We argue that incorporating more case-relevant knowledge into the prompt can help Text-to-SQL tasks. By providing similar question patterns and intentions to LLMs, we can generate corresponding SQL queries more accurately. Case-Based Reasoning (CBR) is an analogical reasoning method that adapts and reuses solutions from previous experiences [17]. Instead of merely memorizing patterns within its parameters, models can reuse the reasoning patterns of similar queries during the inference process.

We propose CBR-ApSQL, a CBR-based framework that adaptively prompts LLMs to accurately control case-relevant and case-irrelevant knowledge in cross-domain Text-to-SQL tasks. Concretely, case-relevant knowledge refers to schema items required for the question, as well as valid grammar for SQL generation. While case-irrelevant knowledge is the opposite. On one hand, we design a de-semanticization mechanism to maintain consistency with case-relevant knowledge. On the other hand, we design an adaptive fallback mechanism to adjust the range of schema items (i.e., tables and columns), which ensures the relevance between cases and the prompt, as well as the informativeness of the prompt.

In particular, we design the Semantic Domain Relevance Evaluator (SDRE) to retrieve semantic and syntactic cases, which integrates three key components: the Poincaré detector, TextAlign, and Positector. (1) Poincaré detector mines implicit semantics in hyperbolic space, which captures the hierarchy of semantic information more effectively. (2) TextAlign discovers explicit matches between NLQ tokens and schema items, overcoming the limitations of the vanilla self-attention mechanism in capturing explicit lexical cues. (3) Positector, a part-of-speech detector that identifies and analyzes the grammatical roles of words within a sentence, enhancing question skeleton generation accuracy.

Our contributions are as follows: (1) We develop a de-semanticization method for ensuring consistency with case-relevant knowledge and an adaptive fallback

mechanism to balance prompt relevance and informativeness. (2) We integrate case-based reasoning with large language models to precisely control the use of case-relevant and case-irrelevant knowledge. (3) The experimental outcomes on three cross-domain Text-to-SQL benchmarks show that our method surpasses the SOTA models.

2 Related Work

Text-to-SQL parsing tasks have achieved significant advancements through the utilization of encoder-decoder architectures [2]. The encoder is responsible for learning the representations of questions and schemas, while the decoder generates SQL queries based on the encoded information.

- **Encoder-based Text-to-SQL Parsing.** Guo et al. [18] proposed IR-NET, which uses attention-based Bi-LSTM for encoding and an intermediate representation-based decoder for SQL prediction. Later, [19, 20] introduced graph-based encoders to construct schema graphs and improve input representations. Works such as RATSQ [1], SDSQ [5], LGESQ [21], S2SQ [22], R2SQ [23], SCORE [24], and STAR [25] further improved structural reasoning by modeling relations between schemas and questions. GRAPHIX-T5 [6] overcomes the limitations of previous methods by incorporating graph representation learning in the encoder. Concurrently, RASAT [26] also provided T5 with structural information by adding edge embedding into multi-head self-attention.

- **Decoder-based Text-to-SQL Parsing.** We divide the methods into four categories: sequence-based methods (BRIDGE [27], PICARD [3]) directly translate NLQ into SQL query token by token, template-based methods (X-SQL [28], HydraNet [29]) employ predefined templates to regulate SQL generation and ensure structural coherence, stage-based methods (GAZP [30], RYANSQ [31]) first establish a coarse-grained SQL framework and then fills in the missing details in the frame which calls slot-filling methodologies, and hierarchical-based methods (IRNet [18], RAT-SQL [1]) generate SQL according to grammar rules in a top-down manner, resulting in a tree-like structure.

These full-data fine-tuned models require various and large training data to ensure that all modules are well-trained, whereas ours requires only a few unsupervised in-context exemplar annotations [16]. Yu et al. [32] divided highly similar solutions (SQL) into 30 categories and classified new cases according to the characteristics of questions in different categories of cases. However, our method obtains the question skeletons by de-semanticization and retrieves similar cases according to the skeleton.

3 Methodology

Our CBR-ApSQL framework utilizes LLMs’ abilities to control case-relevant and case-irrelevant knowledge in cross-domain Text-to-SQL tasks through adaptive prompting, as shown in Fig. 1. We first represent the case and de-semanticize the question, then we introduce three processes of our method. In the first stage, we

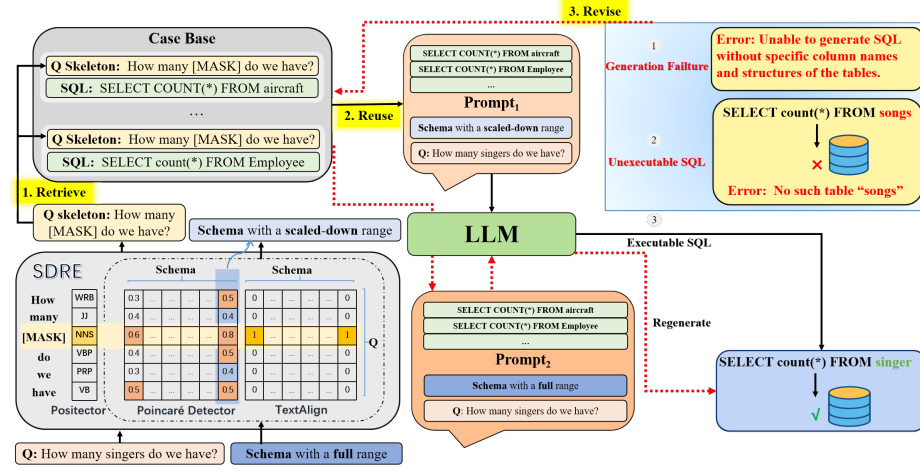


Fig. 1. The overview of the CBR-ApSQL framework. The process starts from the bottom left, and it always ends at the bottom right. Note that the schema is only processed by the matching parts on SDRE, resulting in a schema with a scaled-down range as part of *Prompt₁*, whereas the schema included in *Prompt₂* is with a full range. The second stage of revision is stimulated when the two types of errors (top right) occur, and the process follows the red dotted line.

convert NLQ into a de-semanticized format, called question skeletons, through our proposed Semantic Domain Relevance Evaluator (SDRE). In the retrieval stage, we adaptively retrieve k -nearest neighbor (k -NN) cases corresponding to the test case. In the reuse stage, we construct a prompt with retrieved cases as inputs for the GPT-3.5 to generate SQL queries in the new domain. In the revision stage, we propose a two-step algorithm based on the scaling prompt technique to resolve cases (top right) that are unable to generate executable SQL queries.

3.1 Case De-semanticization

To prepare the case base that suitable for our task, we define the case representation, then describe how to de-semanticize NLQ via SDRE.

Case Representation. To represent cases in cross-domain Text-to-SQL tasks, we divide them into two parts: the problem and the solution. The problem part consists of the natural language question and its corresponding database schema, whereas the solution part includes the SQL query generated from the question and schema. We formalize the representation as follows:

We represent a natural language question sequence as $Q = (q_1, q_2, \dots, q_{|Q|})$. The corresponding database schema is denoted as $S = \langle T, C \rangle$, where $T =$

$(t_1, t_2, \dots, t_{|T|})$ is the sequence of tables and $C = (c_1, c_2, \dots, c_{|C|})$ is the sequence of columns. Noting that the database schemas in the case base differ from the one used for retrieval.

For the solution part of the case, we represent the generated SQL query that expresses the intention of the given natural language question as $SQL = (s_1, s_2, \dots, s_{|SQL|})$, which is a sequence of SQL tokens that can be executed against the corresponding database schema S .

For the de-semanticization process, we utilize the Electra-Large-Discriminator model (ELECTRA) [33] for token sequences encoding process. We concatenate the three parts Q , T and C into one sequence as:

$$[CLS], q_1, \dots, q_{|Q|}, [SEP], [CLS]t_1, \dots, t_{|T|}, [SEP], [CLS]c_1, \dots, c_{|C|}, [CLS],$$

where special tokens $[CLS]$ and $[SEP]$ are employed to structure the text, with the former representing classification and the latter indicating sentence separation.

De-semanticization via SDRE. We de-semanticize the question to alleviate the impact of the domain information, which is varied in different cases. To construct question skeletons, we propose Semantic Domain Relevance Evaluator (SDRE) to identify and mask case-relevant tokens in questions. SDRE leverages the Poincaré detector, TextAlign, and Positector to evaluate question tokens, all of which are performed simultaneously.

• **Poincaré Detector.** We identify the domain semantic information within the NLQ for schema linking of SQL generation, which captures underlying semantic correspondences from a pre-trained language model. Concretely, we elicit the correlation between NLQ tokens and schema items (i.e., columns and tables) from ELECTRA [33] based on Poincaré distance metric [34].

We choose the Poincaré distance metric rather than the typical Euclidean distance metric because hyperbolic space is more suited to model the hierarchy of semantic information than Euclidean space [34]. Our method is inspired by PROTON [35] which applies the capability of ELECTRA [33] model that detecting replaced token. Poincaré detector posits that a token’s semantic information is characterized if the masking of the token in question leads to a significant shift in the vector representation of the entire sequence beyond the pre-defined threshold.

We measure the implicit correlation between the question token q_i and the schema item s_j in the hyperbolic space [36]. By computing d_p on each pair of tokens (q_i, s_j) , we get the Proton metric $D_p \in \mathbb{R}^{|Q| \times |S|}$ as follows:

$$D_{p_{i,j}} = 2 \tanh^{-1} \left(\left\| -\tilde{h}_{j \setminus q_i}^s \oplus \tilde{h}_j^s \right\| \right), \quad (1)$$

where \oplus is the Möbius addition [36], \tilde{h}_j^s represents the embedding of the schema item s_j , and $\tilde{h}_{j \setminus q_i}^s$ represents the embedding if the question token q_i is masked out. Both \tilde{h}_j^s and $\tilde{h}_{j \setminus q_i}^s$ are hyperbolic representation [36].

• **TextAlign.** TextAlign discovers explicit matches between question tokens and schema items (i.e., table names, column names, and values). Although the Poincaré detector effectively captures the underlying semantic correspondences between NLQ tokens and schema items, it is insufficient to address all alignment challenges due to the presence of lexical variations, synonyms, or paraphrasing. This is because Proton [34] essentially mines matching knowledge based on self-attention, whereas vanilla self-attention models are not sensitive to text surface matching [37]. To overcome these limitations, we introduce TextAlign as an additional alignment strategy that considers both name-based and value-based matching for a more accurate NLQ-schema alignment.

Name-based matching identifies direct lexical matches between NLQ tokens and schema items. We define matrix $M_s \in \mathbb{R}^{|Q| \times |S|}$ to represent it:

$$M_{s_i,j} = \begin{cases} 1, & \text{if } q_{i\dots j} \subseteq s_j \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

Value-based matching detects possible value correspondences within the query. We define matrix $M_v \in \mathbb{R}^{|Q| \times |S|}$ to symbolize it:

$$M_{v_i,j} = \begin{cases} 1, & \text{if } q_i = v_j \\ 0, & \text{otherwise} \end{cases}, \quad (3)$$

where v_j represents the set of values in the j^{th} column of the corresponding table or column.

By combining the results of name-based matching and value-based matching, we can obtain the NLQ-schema textual matching score. So far, coupled with the previously calculated Proton metric D_p , we get the total NLQ-schema matching score, which measures the probability that the schema items will be used to compose the SQL query. We define the matching score matrix $M \in \mathbb{R}^{|Q| \times |S|}$ by element-wise summation:

$$M = D_p + M_s + M_v. \quad (4)$$

• **Positector.** Positector is a part-of-speech detector, which primary function is to detect and analyze parts of speech in a given text. POS information is essential for generating a question skeleton because it helps in understanding the structure and grammatical roles of words within a sentence. While the Poincaré detector captures the underlying semantic correspondences between NLQ tokens and schema items, and TextAlign addresses explicit knowledge, neither technique helps to identify key components of the question and formulate question structure. Consequently, we introduce Positector to enhance the accuracy of the question skeleton generation.

We perform POS tagging on Q using Stanza [38] to obtain a set of POS tags t_1, t_2, \dots, t_n , where t_i is the POS tag of a token q_i . Then we generate the Lexical metric $P \in \mathbb{R}^{|Q|}$ for each token q_i based on its POS tag t_i :

$$P_i = \begin{cases} \alpha, & \text{if } t_i \text{ is a noun or a number} \\ 0, & \text{otherwise} \end{cases}. \quad (5)$$

Incorporating the three strategies mentioned above, the SDRE obtains a score for each question q_i . We calculate $Q_{\text{sco}} = (q_{\text{sco}1}, q_{\text{sco}2}, \dots, q_{\text{sco}|Q|})$ using the following equation:

$$Q_{\text{sco}i} = \frac{1}{2} \left(\frac{1}{n} \sum_{j=1}^{|S|} M_{ij} + P_i \right), \quad \forall i \in 1, \dots, |Q|. \quad (6)$$

We generate the question skeleton based on Q_{sco} and τ , where τ is a hyper-parameter that controls the minimum similarity score required for a token to be considered relevant via SDRE.¹

In this way, we remove tokens that are relevant to the case-relevant semantics of the question and generate the de-semanticized question skeletons, which allows for better identification of cases with similar question patterns and intentions.

3.2 Case Retrieval and Reuse

K-NN Retrieval. We project de-semanticized question skeletons of the case base into a vector space and retrieve k -NN cases corresponding to the new question skeleton. The new question skeleton serves as the key for the retrieval process, and the returned value consists of the k -NN cases from the case base.

Our approach is based on the idea that understanding the intent behind a question and structuring a logical response forms the basis of a reusable SQL experience. Questions with similar intents typically exhibit similar sentence structures, leading to similar SQL query generation.

Reuse via Prompt. We reuse the retrieved cases to construct a prompt for the LLM to generate the new SQL query.

In the pre-experiments, we attempt to provide NLQ-SQL pairs and Schema-NLQ-SQL formats. However, both approaches suffer from information overload, making it challenging to concentrate on the current case for generating SQL. Consequently, we choose SQL queries in k cases as a demonstration, aiming to allow the LLMs to learn to generate more reasonable and efficient SQL queries in a consistent SQL style.

We design a three-part prompt comprising the retrieved cases, the database schema, and hint words. Specifically, the first part of the prompt is a list of SQL queries based on retrieval; the second part is the database schema (i.e., tables, columns), and foreign keys; the third part is hint words, including the leading words for the above two parts as well as the NLQ to be answered.

3.3 Case Revision

We propose a two-stage algorithm to revise and regenerate SQL queries, addressing issues like SELECT Extra Columns [14] and redundant Foreign Keys Join.

¹ If q_{sco} is below the threshold of τ , we retain the original question token; otherwise, we replace it with the pre-defined [MASK] token.

The algorithm as shown in Algorithm 1 adaptively adjusts the schema range. Initially, we use a scaled-down range based on SDRE. If LLMs do not generate SQL but an error, or the SQL query fails to execute, we fallback to the complete database schema range to generate a revised SQL query.

Algorithm 1 Case Revision

Require: Q, M, S, θ
Ensure: S_α or S_β

```

1: Initialize  $S' \leftarrow \{*\}$ 
2: for  $j = 1$  to  $|S|$  do
3:    $Sr_j \leftarrow \max_{0 < i < |Q|} (M_{i,j})$ 
4:   if  $Sr_j > \theta$  then
5:      $S' \leftarrow S' \cup x \in S$ 
6:   end if
7: end for
8:  $S_\alpha \leftarrow \text{GenerateSQL}(Q, S')$ 
9: if  $\text{IsExecutable}(S_\alpha)$  then
10:  return  $S_\alpha$ 
11: else
12:   $S_\beta \leftarrow \text{GenerateSQL}(Q, S)$ 
13:  return  $S_\beta$ 
14: end if

```

Scaled-Down Database Schema Range. In the first stage of revision, we reduce the schema range by considering only the most relevant schema items.

We compute the relevance scores of the database schema items based on the matrix M defined in Eq. (4). For each schema item j , we define Sr_j to represent the relevance score with the test case. We decide the highest matching score with question tokens for each schema item j , as shown in Line 3.

Subsequently, we apply a threshold θ to filter out less relevant schema items. Specifically, we only consider columns and tables with scores higher than the θ . We construct a list of relevant tables and columns by looking up the original names in the database schema and appending an asterisk (*) to the list of relevant columns to include all columns in the SQL query.

The full database schema range equals S , which was introduced in Sect. 3.1. We define a scaled-down schema range S' , which is calculated in lines 1-7. Line 8 describes that we reconstructed a prompt for LLM with the scaled-down schema range S' to generate an initial SQL query, which is denoted as S_α . If we generate executable SQL, this algorithm ends in the first stage.

Fallback to Full Database Schema Range. If the initial SQL query S_α fails to execute or the LLM outputs an error message, the algorithm proceeds to the second stage, which employs a fallback mechanism to generate a revised

SQL query S_β . This stage takes into account the full database schema range S , providing a broader context for the SQL generation process, and we produce a revised SQL query S_β .

One of the benefits of revision is its ability to detect and fix execution errors as they occur. It’s important to note that this is not a comparison against the gold SQL query. Instead, it’s an attempt to run the SQL query in the target database to gather error information. On the other hand, by employing a two-stage algorithm, we balance the relevance of the (case, prompt) pair against the informativeness of the prompt. This method allows the model to initially focus more on the current information by reducing the schema range, while still ensuring the successful execution of the complex SQL query requiring more information through the fallback mechanism.

4 Experiment

4.1 Experimental Setup

Datasets. We conduct experiments on three public benchmark datasets as follows: (1) **Spider** [39] is a large-scale benchmark of cross-domain Text-to-SQL across 138 different domain databases. (2) **Spider-Syn** [40] is a challenging variant based on Spider that eliminates explicit alignment between questions and database schema by synonym substitutions. (3) **Spider-DK** [41] is also a variant dataset based on Spider with artificially added domain knowledge.

Evaluation. In our evaluation process, we prioritize three key metrics: valid SQL (VA), execution accuracy (EX), and test-suite accuracy (TS) [42]. VA measures the percentage of SQL queries that are executed without any errors. EX measures the accuracy of the execution results by comparing them with the standard SQL query. TS measures the effectiveness of the distilled test suite in achieving high code coverage for the database through execution, which can serve as a better proxy for semantic accuracy.

Noting that we do not rely on the mainstream exact match accuracy metric (EM), as SQL queries that serve the same purpose may be expressed in different ways. EM is tailored to a limited style of the dataset and serves as an intermediate solution evaluation metric for Text-to-SQL tasks.

Baselines. We choose the most advanced full-data fine-tuned models in terms of these assessment indicators: (1) **PICARD** [3], a technique that constrains auto-regressive decoders in language models through incremental parsing; (2) **RASAT** [26], which incorporates relation-aware self-attention into transformer models while also utilizing constrained auto-regressive decoders; and (3) **RES-DSQL** [5], which introduces a ranking-enhanced encoding and skeleton-aware decoding framework to effectively separate schema linking and skeleton parsing. Notably, all three models are built upon the T5-3B architecture.

In the selection of LLM-based models, We use the ChatGPT baseline model [14] and Codex baseline model [13], both of which are currently the best results for evaluating Text-to-SQL capability using LLMs. Additionally, our method relies on GPT-3.5, which offers a balance between capability and availability. The rapid updates and continuous improvements of LLMs make selecting the most effective model for evaluation a challenging task.

Setting. We apply FAISS [43] for storing the de-semanticized question skeletons and efficient retrieval in the case base. For hyperparameter settings, we assign $\alpha=0.9$, $\tau=0.6$, and $\theta=0.4$.

Our approach expands the database content beyond the limitations of the Text-to-SQL prompt used in the OpenAI demo website², which only contains table and column names. We re-formatted the prompt to achieve better results, even though it differs from the format used in the official data training.

Table 1. Comparison of the performance of our model and other baseline models on Spider, Spider-SYN, and Spider-DK datasets. The null data were due to the absence of corresponding assessment scores in the papers.

Models\Datasets		SPIDER			SPIDER-SYN			SPIDER-DK		
		VA	EX	TS	VA	EX	TS	VA	EX	TS
Full-data	PICARD	98.4	79.3	69.4	98.2	69.8	61.8	97.8	62.5	-
Fine-tuned	RASAT	98.8	80.5	70.3	98.3	70.7	62.4	98.5	63.9	-
Models	RESDSQL-3B	99.1	84.1	73.5	98.8	76.9	66.8	98.8	66.0	-
LLM-based	GPT-3.5	87.0	57.2	56.7	83.1	39.3	39.2	88.6	48.8	48.0
	Codex	91.6	67.0	55.1	-	-	-	-	-	-
	ChatGPT	97.7	70.1	60.1	96.2	58.6	48.5	96.4	62.6	-
	CBR-ApSQL	99.0	87.8	84.8	99.0	79.4	75.9	99.4	74.2	69.7

4.2 Main Results

We present a comparison between LLM-based models and full-data fine-tuned models which are SOTA as shown in Table 1. Our CBR-ApSQL framework outperforms all models in almost all evaluation metrics, except for the Spider dataset where the VA metric is only 0.1 worse than the next best model (RESDSQL-3B). LLM-based models may face difficulty generating SQL queries that conform to strict syntactical and semantic rules, resulting in lower VA scores. Overall, it indicates a significant improvement in semantic and syntactic accuracy.

² <https://platform.openai.com/examples/default-sqltranslate>

4.3 Strategies Analysis

To further investigate the performance of different strategies for retrieving cases, we perform a fine-grained analysis.

These experimental results in Table 2 indicate that our strategy outperforms the other three strategies across all datasets and metrics. The adaptive retrieval mechanism in the CBR-ApSQL method ensures that the retrieved cases are semantically closer to the test case. This results in more relevant SQL grammar being used for the new case, ultimately leading to better performance.

Table 2. Comparison of SQL generation performance using different strategies to generate prompts for LLM. The methods include Random Sample (randomly retrieving cases from the case base), Cosine Similarity (utilizing cosine similarity to match questions), Euclidean SDRE (using Euclidean distance instead of Poincaré distance), and Hyperbolic SDRE (our proposed method).

Methods / Datasets	SPIDER			SPIDER-SYN			SPIDER-DK		
	VA	EX	TS	VA	EX	TS	VA	EX	TS
Random Sample	94.3	66.9	65.9	92.9	56.8	55.3	93.3	59.4	57.9
Cosine Similarity	93.5	71.6	69.8	94.7	59.2	57.7	86.5	56.8	55.0
Euclidean SDRE	96.6	84.6	79.6	97.1	77.2	74.3	99.1	71.3	68.2
Hyperbolic SDRE	99.0	87.8	84.8	99.0	79.4	75.9	99.4	74.2	69.7

Noting that the introduction of domain knowledge in the SPIDER-DK dataset increases the complexity of the sentence components. This complexity negatively impacts the performance of the cosine similarity matching strategy. However, our strategy is able to mitigate this impact through its de-semantic approach, which reduces the effect of case-irrelevant knowledge while still enabling a better understanding of domain info via LLMs’ extensive knowledge.

To further specify the effectiveness of the CBR-ApSQL approach in generating adaptive prompts, we present a case study later.

4.4 Ablation Study

We investigate the individual contributions of various components of our proposed model to the overall performance of three different datasets as shown in Fig. 2. The full model surpasses all the ablated variants across all datasets and performance metrics.

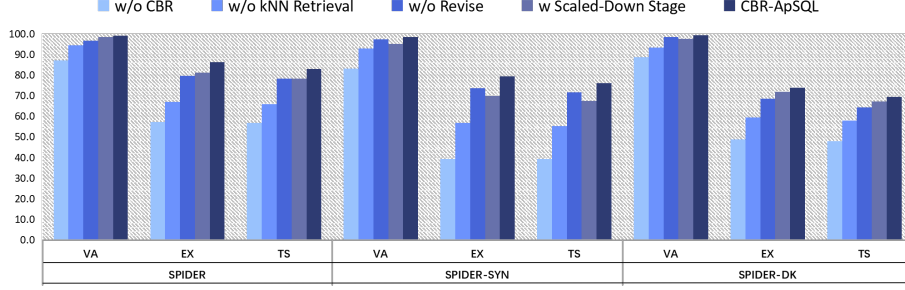


Fig. 2. Ablation study of model components for performance on SPIDER, SPIDER-SYN, and SPIDER-DK Datasets. We compare our full model with four ablated variants, each with a specific component removed or modified: (1) w/o CBR: removing the CBR component, only relying on the LLM. (2) w/o k NN Retrieval: replacing the k similar cases via retrieval with k random samples. (3) w/o Revise: using k -NN retrieved cases with a full database schema range without the revision process. (4) w Scaled-Down Stage: using k -NN retrieved cases with a scaled-down schema range without the revision process.

Role of Case-Based Reasoning: The performance drop when the CBR component is removed (w/o CBR). This highlights the critical role of CBR in the model’s ability to generate accurate SQL queries. By leveraging past experiences and similar cases, CBR allows the model to make informed decisions in generating SQL queries.

Significance of k NN Retrieval: Replacing the k NN retrieval with random samples (w/o k NN Retrieval) leads to a substantial decrease in performance. This emphasizes the importance of the SDRE-based k NN retrieval process in finding relevant samples. Using k NN retrieval allows the model to identify and use cases that share similar intent, which often exhibit similar sentence structures. Consequently, this similarity leads to identical SQL logic generation.

Effect of Revision Process: The revision process helps the model to further refine the SQL query by incorporating more accurate and relevant information from the database schema. This refinement leads to an improvement in the model’s ability to generate semantically correct SQL queries. Comparing the last two settings, the results indicate that scaling down the schema range (using a filtered range) improves the model’s performance, even without the revision process. This suggests that narrowing down the scope of the schema range allows the model to focus on a smaller set of relevant information, thereby enhancing its accuracy and overall efficiency.

4.5 Case Study

To illustrate our method more intuitively, We use the first example to show our de-semantic match method can lead to more case-relevant and adaptive samples to construct a prompt. In the second example, we demonstrate the impact of

using full and scaled-down database schema ranges on the generation of SQL queries, striking a balance between focusing on case-relevant info and handling more complex cases, in an adaptive way.

As shown in Fig. 3, the random sample strategy does not consider the semantic or structural similarity between the cases and the test case. Therefore, even though some demonstration samples are provided for the new case, it can have a negative impact by showing case-irrelevant SQL grammar. Cosine similarity, while focusing on token-level similarity, is not well-suited for Text-to-SQL tasks because it can lead to high literal similarity without capturing the true intent of the questions. Actually, the same question can be asked in a variety of ways.

Test Case:	
NLQ:	What are the names, countries, and ages for every singer in descending order of age?
SQL:	SELECT Name, Country, Age FROM singer ORDER BY Age DESC
Random Sample	
Sample ₁ :	NLQ: How many heads of the departments are older than 56 ?
	SQL: SELECT count(*) FROM head WHERE age > 56
Sample ₂ :	NLQ: Show the number of card types.
	SQL: SELECT count(DISTINCT card_type_code) FROM Customers_Cards
Cosine Similarity Match	
Sample ₁ :	NLQ: What are the names of everybody sorted by age in descending order?
	SQL: SELECT name FROM Person ORDER BY age DESC
Sample ₂ :	NLQ: What are the order ids and customer ids for orders that have been Cancelled, sorted by their order dates?
	SQL: SELECT order_id , customer_id FROM customer_orders WHERE order_status_code = "Cancelled" ORDER BY order_date
De-semantic Match	
Sample ₁ :	NLQ: What are the themes and years for exhibitions, sorted by ticket price descending?
	SQL: SELECT exhibition.Theme, exhibition.Year FROM exhibition ORDER BY exhibition.Ticket_Price DESC
Sample ₂ :	NLQ: What are the names of wrestlers and their teams in elimination, ordered descending by days held?
	SQL: SELECT T2.Name, T1.Team FROM Elimination AS T1 JOIN wrestler AS T2 ON T1.Wrestler_ID=T2.Wrestler_ID ORDER BY T2.Days_held DESC

Fig. 3. Comparison of samples in prompt generated by random samples, cosine similarity matching, and de-semantic matching. The questions or the question skeletons are in green font, the SQL components are in blue font, and the domain semantically relevant words are in red font.

Our de-semantic matching method removes the interference of case-irrelevant knowledge and retrieves the demonstration samples with similar questioning intention through the domain-irrelevant question skeleton. These similarities include the phrasing of the query, the order in which information is requested, and the general structure of the sentence. In the given example, both SQL queries share similar structures: (1) They request sorting or ordering of the data based on a particular attribute (e.g., ticket price, days held). (2) They use similar word-

ing for ordering (e.g., "ordered descending by"). Recognizing these similarities can be helpful for LLMs to identify patterns and generate the appropriate SQL queries or responses.

In Fig.4, we illustrate the need for adaptive prompts with different schema ranges. In the first example, a scaled-down schema range generates the correct SQL query (SQL_2), while the full range introduces unnecessary links, leading to a redundant Foreign Keys Join problem. It demonstrates the impact of the expanded schema range potentially introducing case-irrelevant knowledge. The second example shows that a limited schema range fails to provide enough information for the correct SQL query, resulting in an error message ($Output_1$). However, with a full schema range, the LLM generates the correct query ($Output_2$), showcasing the successful execution of complex SQL queries.

NLQ: What are the names, countries, and ages for every singer in descending order of age?	
The case with a full database schema range :	
SQL ₁ :	SELECT singer.Name, singer.Country, singer.Age FROM singer INNER JOIN singer_in_concert ON singer_in_concert.Singer_ID = singer.Singer_ID ORDER BY singer.Age DESC ✗

The case with a scaled-down schema range :	
SQL ₂ :	SELECT Name, Country, Age FROM singer ORDER BY Age DESC ✓
NLQ: Which division offers the most number of degrees? List division name and id.	
The case with a scaled-down schema range :	
Output ₁ :	"It is not clear from the provided table and column names which table(s) and column(s) contain information about divisions and degrees. Without knowing the specific structure of the database, it is not possible to provide the SQL query to answer this question."

The case with a full database schema range :	
Output ₂ :	SELECT Departments.department_id, Departments.department_name FROM Degree_Programs JOIN Departments ON Degree_Programs.department_id = Departments.department_id GROUP BY Departments.department_id, Departments.department_name ORDER BY COUNT(Degree_Programs.degree_program_id) DESC LIMIT 1 ✓

Fig. 4. Impact of using different schema ranges on SQL generation.

5 Conclusions and Future Work

Our CBR-ApSQL framework effectively combines CBR with LLMs to improve cross-domain Text-to-SQL performance. By employing adaptive prompts and a de-semanticization phase, our approach overcomes poorly designed prompt limitations, enabling more accurate control of case-relevant and case-irrelevant knowledge. Performance improvements across three datasets validate the framework's potential for unlocking LLMs' capabilities in Text-to-SQL tasks.

In future work, efforts can focus on utilizing more efficient cases for reasoning and designing complex iterative prompts in the revision phase. While our proposed framework generates adaptive prompts, alternative methods for prompt generation could be investigated to determine if further performance improvements can be achieved, such as using unsupervised learning or reinforcement learning approaches.

References

1. Wang, B., Shin, R., Liu, X., Polozov, O., Richardson, M.: Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *ACL* (2020)
2. Cai, R., Xu, B., Zhang, Z., Yang, X., Li, Z., Liang, Z.: An encoder-decoder framework translating natural language to database queries. In: *IJCAI* (Jul 2018)
3. Scholak, T., Schucher, N., Bahdanau, D.: Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *EMNLP* (2021)
4. Cai, R., Yuan, J., Xu, B., Hao, Z.: Sadga: Structure-aware dual graph aggregation network for text-to-sql. *NIPS* (Dec 2021)
5. Li, H., Zhang, J., Li, C., Chen, H.: Decoupling the skeleton parsing and schema linking for text-to-sql. *arXiv:2302.05965* (2023)
6. Li, J., Hui, B., et al.: Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *arXiv:2301.07507* (2023)
7. Brown, T., Mann, B., Ryder, N., et al.: Language models are few-shot learners. *NIPS* 33, 1877–1901 (2020)
8. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., et al.: Evaluating large language models trained on code. *arXiv:2107.03374* (2021)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL* (2018)
10. Zhuang, L., Wayne, L., Ya, S., Jun, Z.: A robustly optimized bert pre-training approach with post-training. In: *CCL*. pp. 1218–1227 (2021)
11. Lewis, M., Liu, Y., et al.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: *ACL* (Jul 2020)
12. Raffel, C., Shazeer, N., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR* 21, 5485–5551 (2020)
13. Rajkumar, N., Li, R., Bahdanau, D.: Evaluating the text-to-sql capabilities of large language models. *arXiv:2204.00498* (2022)
14. Liu, A., Hu, X., Wen, L., Yu, P.S.: A comprehensive evaluation of chatgpt’s zero-shot text-to-sql capability. *arXiv:2303.13547* (2023)
15. Cheng, Z., Xie, T., Shi, P., et al.: Binding language models in symbolic languages. In: *ICLR* (2023)
16. Zhao, W.X., Zhou, K., Li, J., et al.: A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023)
17. Darias, J.M., et al.: Using case-based reasoning for capturing expert knowledge on explanation methods. In: *ICCB23*. pp. 3–17. Springer (2022)
18. Guo, J., et al.: Towards complex text-to-sql in cross-domain database with intermediate representation. In: *ACL*. pp. 4524–4535 (2019)
19. Bogin, B., Berant, J., Gardner, M.: Representing schema structure with graph neural networks for text-to-sql parsing. In: *ACL* (Sep 2019)
20. Chen, Z., Chen, L., Zhao, Y., Cao, R., Xu, Z., Zhu, S., Yu, K.: Shadowgnn: Graph projection neural network for text-to-sql parser. In: *NAACL* (Jun 2021)
21. Cao, R., Chen, L., et al.: Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations. In: *ACL* (Jul 2021)
22. Hui, B., Geng, R., Wang, L., et al.: S2sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers. In: *ACL*. pp. 1254–1262 (2022)
23. Hui, B., Geng, R., Ren, Q., Li, B., Li, Y., Sun, J., Huang, F., Si, L., Zhu, P., Zhu, X.: Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. *AAAI* (May 2021)

24. Yu, T., Zhang, R., Polozov, A., Meek, C., Awadallah, A.H.: {SC}ore: Pre-training for context representation in conversational semantic parsing. In: ICLP (2021)
25. Cai, Z., Li, X., Hui, B., Yang, M., Li, B., Li, B., Cao, Z., Li, W., Huang, F., Si, L., Li, Y.: Star: Sql guided pre-training for context-dependent text-to-sql parsing. EMNLP (Oct 2022)
26. Qi, J., Tang, J., He, Z., et al.: RASAT: integrating relational structures into pre-trained seq2seq model for text-to-sql. In: EMNLP. pp. 3215–3229 (2022)
27. Lin, X.V., Socher, R., Xiong, C.: Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In: EMNLP. pp. 4870–4888 (2020)
28. He, P., Mao, Y., Chakrabarti, K., Chen, W.: X-sql: reinforce schema representation with context. arXiv:1908.08113 (2019)
29. Lyu, Q., Chakrabarti, K., Hathi, S., Kundu, S., Zhang, J., Chen, Z.: Hybrid ranking network for text-to-sql. arXiv preprint arXiv:2008.04759 (2020)
30. Zhong, V., Lewis, M., Wang, S.I., Zettlemoyer, L.: Grounded adaptation for zero-shot executable semantic parsing. In: EMNLP. pp. 6869–6882 (2020)
31. Choi, D., Shin, M.C., Kim, E., Shin, D.R.: Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. CL (2021)
32. Yu, W., et al.: Similar questions correspond to similar sql queries: A case-based reasoning approach for text-to-sql translation. In: ICCBR. pp. 294–308. Springer (2021)
33. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. arXiv:2003.10555 (2020)
34. Chen, B., et al.: Probing bert in hyperbolic spaces. arXiv:2104.03869 (2021)
35. Wang, L., Qin, B., Hui, B., et al.: Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. SIGKDD (2022)
36. Ganea, O., Bécigneul, G., Hofmann, T.: Hyperbolic neural networks. NIPS (2018)
37. Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., Wattenhofer, R.: On identifiability in transformers. arXiv preprint arXiv:1908.04211 (2019)
38. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A python natural language processing toolkit for many human languages. In: ACL (Jul 2020)
39. Yu, T., Zhang, R., et al.: Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In: EMNLP (Jun 2019)
40. Gan, Y., Chen, X., Huang, Q., Purver, M., et al: Towards robustness of text-to-sql models against synonym substitution. In: ACL (Jul 2021)
41. Gan, Y., Chen, X., Purver, M.: Exploring underexplored limitations of cross-domain text-to-sql generalization. In: EMNLP (Dec 2021)
42. Zhong, R., Yu, T., Klein, D.: Semantic evaluation for text-to-sql with distilled test suites. In: EMNLP. pp. 396–411 (2020)
43. Johnson, J., Douze, M., Jegou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data p. 535–547 (Jun 2019)