# G$^3$R: A Graph-Guided Generate-and-Rerank Framework for Complex and Cross-domain Text-to-SQL Generation

**Yanzheng Xiang[1], Qian-Wen Zhang[2], Xu Zhang[1], Zejie Liu[1], Yunbo Cao[2] and Deyu Zhou[1]***

[1]School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China

[2]Tencent Cloud Xiaowei, Beijing 100080, China

xyz1998seu@gmail.com, {d.zhou,xuzhang123,liuzejie}@seu.edu.cn, {cowenzhang, yunbocao}@tencent.com

## Abstract

We present a framework called G$^3$R for complex and cross-domain Text-to-SQL generation. G$^3$R aims to address two limitations of current approaches: (1) The structure of the abstract syntax tree (AST) is not fully explored during the decoding process which is crucial for complex SQL generation; (2) Domain knowledge is not incorporated to enhance their ability to generalise to unseen domains. G$^3$R consists of a graph-guided SQL generator and a knowledge-enhanced re-ranking mechanism. Firstly, during the decoding process, an AST-Grammar bipartite graph is constructed for joint modelling the AST and corresponding grammar rules of the generated partial SQL query. The graph-guided SQL generator captures its structural information and fuses heterogeneous information to predict the action sequence, which can uniquely construct the AST for the corresponding SQL query. Then, in the inference stage, a knowledge-enhanced re-ranking mechanism is proposed to introduce domain knowledge to re-rank candidate SQL queries from the beam output and choose the final answer. The SQL re-ranker is based on a pre-trained language model (PLM) and contrastive learning with hybrid prompt tuning is incorporated to stimulate the knowledge of the PLM and make it more discriminative. The proposed approach achieves state-of-the-art results on the Spider and Spider-DK benchmarks, which are challenging complex and cross-domain benchmarks for Text-to-SQL semantic analysis.

## 1 Introduction

Complex and cross-domain Text-to-SQL generation aims to translate natural language utterances into structurally complex SQL queries, where no database overlaps in training and testing. The challenges mainly lie in two aspects: (1) Zero-shot cross-domain setting, databases are not overlapped between training and test sets and belong to different domains. A domain represents a certain type of application scenario, e.g. medical and geographical. It is difficult for approaches to understand domain knowledge properly (Gan et al., 2021a) and generalise well to unseen databases. (2) Complexity, SQL queries involving multiple tables, containing nested queries and clauses such as GROUPBY and HAVING, are more complex in their structure.

| | |
|---|---|
| **NL:** | How many flights **arrive at** Luton Airport? |
| **SQL:** | ... WHERE flights.**TargetAirport** = "Luton Airport" |
| **NL:** | List the name of all players in the order of their **date of birth from the oldest to the youngest.** |
| **SQL:** | ... ORDER BY players.birth_date **ASC**. |

Table 1: Two samples that require domain knowledge to generate correct SQL queries, where the text reflecting domain knowledge is bolded.

Recently, various works have been proposed for complex and cross-domain Text-to-SQL generation based on the encoder-decoder paradigm. In order to improve generalizability to unseen database schemas, many approaches focus on the encoder part. Graph-based approaches (Cao et al., 2021; Wang et al., 2020a) have been proposed for jointly encoding the natural language utterance and relational structure of the database schema, thereby capturing the semantic relationships between them. For generating complex SQL queries, most decoding approaches (Cao et al., 2021; Guo et al., 2019; Wang et al., 2020a) pre-define abstract syntax description language (ASDL) rules and autoregressively generate the SQL query as an AST. Specifically, an AST is produced progressively in a pre-order depth-first traversal order based on a sequence of actions. In this way, the skeleton of complex SQL queries can be generated flexibly.

Despite much progress on complex and cross-domain Text-to-SQL generation, there are still two limitations. Firstly, The structure of AST is not fully explored during the decoding process which

*Corresponding author.



*Findings of the Association for Computational Linguistics: ACL 2023*, pages 338–352
July 9-14, 2023 

is crucial for generating complex SQL queries correctly. They predict the next action only based on the previous action sequence, without considering the structural information of the AST and how it changes dynamically. In addition, domain knowledge required by different domains is not incorporated, which results in the model not being well generalised to new domains. As shown in Table 1, for the first sample, "target airport" does not appear in the utterance and the model needs to understand that "Arriving at Luton Airport" means "Luton Airport" is the target airport for generating the correct SQL query. For the second sample, players should be listed in ascending order of birth, as the younger the birth date, the older the age. If the requirement is changed to "the ages of all players are listed", then it should be in descending order. (Gan et al., 2021b) has pointed out that understanding domain knowledge is crucial for cross-domain Text-to-SQL generalization.

To address the aforementioned limitations, this paper proposes a **G**raph-**G**uided **G**enerate-and-**R**erank Framework ($G^3R$) for complex and cross-domain Text-to-SQL generation. It consists of a graph-guided SQL generator and a knowledge-enhanced re-ranking mechanism. To make better use of the structural information of AST, we construct an AST-Grammar bipartite graph for both the AST and grammar rules of the generated partial SQL query. A graph-guided SQL generator is proposed to capture structural changes of the bipartite graph dynamically and fuse the heterogeneous information from the encoder to predict the action sequence.

Furthermore, we try to improve generalizability to unseen domains from a different perspective, by introducing a knowledge-enhanced re-ranking mechanism to choose the best SQL query from the beam output in the inference stage. PLM is adopted as an SQL re-ranker and hybrid prompt tuning is adopted to make the re-ranking task similar to language modelling, stimulating the knowledge in PLMs to bridge the gap between different domains. In addition, sometimes the differences between candidate SQL queries are subtle and contrastive learning is adopted to push away the distance of candidate queries' representations and thus make them more distinguishable.

In summary, the main contributions of this paper are listed as follows:

- A **G**raph-**G**uided **G**enerate-and-**R**erank Framework ($G^3R$) is proposed for complex and cross-domain Text-to-SQL generation. An AST-Grammar bipartite graph is constructed to model the AST and grammar rules of the generated partial SQL query jointly and a novel graph-guided SQL generator is proposed to capture the structural information and fuse heterogeneous information to generate the SQL query.

- A knowledge-enhanced re-ranking mechanism is proposed to introduce domain knowledge to choose the best SQL query from the beam output. PLM is adopted as an SQL re-ranker and contrastive learning with hybrid prompt tuning is incorporated to stimulate the knowledge of PLMs and make it more discriminative. As far as we know, we are the first to leverage the abundant knowledge of PLMs to re-rank SQL queries to mitigate performance degradation in unseen domains.

- Comprehensive experiments were conducted on Spider and Spider-DK. The results show that the proposed approach achieves superiority over some state-of-art approaches.

## 2 Related work

Our work is related to two lines of research, complex and cross-domain Text-to-SQL generation and prompt tuning.

### 2.1 Complex and cross-domain Text-to-SQL generation

In order to improve cross-domain generalizability, many approaches focus on the encoder part. (Bogin et al., 2019a) adopts a graph neural network (GNN) to deal with the graph structure of database schema. (Cao et al., 2021; Wang et al., 2020a) construct a heterogeneous graph for jointly encoding natural language utterances and relational structure in the database schema so that the network can generalise to unseen database schemas. In contrast, our proposed approach tackles this problem from a different perspective, by exploring the knowledge of PLMs and re-ranking candidate SQL queries.

Most decoding approaches for complex and cross-domain Text-to-SQL generation (Brunner and Stockinger, 2021; Wang et al., 2020a; Guo et al., 2019) consider the ASDL rules as the prior knowledge and adopt a syntax-based SQL generator (Yin and Neubig, 2017) to output a sequence

of actions based on a pre-defined fixed set of grammar rules. Then an abstract syntactic tree (AST) corresponding to the target SQL query is uniquely constructed in pre-order depth-first traversal order. However, they cast SQL generation into sequence-to-sequence translation and ignore the structural information of AST and how it dynamically changes during the decoding process.

### 2.2 Prompt Tuning

Fine-tuning PLMs with task-specific heads has been widely applied to natural language processing and achieved great success in many downstream tasks (Radford et al., 2019; Zhou et al., 2021). However, the abundant knowledge in PLMs is not fully exploited because there is a big gap between the fine-tuning objectives and pre-training objectives. Subsequently, GPT-3 (Brown et al., 2020) proposed prompt tuning for downstream tasks. It is a new paradigm that adopts natural language prompts to make downstream tasks similar to language modelling and does not require the addition of model parameters.

A number of works defined hard templates manually where each token is meaningful and understandable (Schick and Schütze, 2021; Gu et al., 2022) and some approaches generated hard templates automatically (Gao et al., 2021; Shin et al., 2020). However, there is no need to limit templates to be human-interpretable because the aim is to find a way to enable PLMs to perform downstream tasks effectively. Soft prompts (Wu and Shi, 2022; Lester et al., 2021) have been proposed and the tokens (i.e. virtual tokens) in the template are continuous vectors which can be learnt during the tuning stage. In addition, (Han et al., 2022) proposed to insert some tunable embeddings into a hard prompt template which is called "Hybrid Prompt Tuning".

## 3 Preliminaries

**Autoregressive top-down SQL generation** In this paper, we tackle the SQL generation problem by generating the AST through syntax-based autoregressive top-down decoding (Yin and Neubig, 2017; Krishnamurthy et al., 2017; Rabinovich et al., 2017; Yin and Neubig, 2018), which guarantees to decode of syntactically valid SQL queries. The generation process is considered to be the sequential application of actions. We define three actions including ApplyRule, SelectTable and SelectColumn. The ApplyRule action applies a grammar rule to expand a non-terminal node in the AST, while the SelectTable and SelectColumn actions populate a terminal node by appending a table or a column name from the database schema.

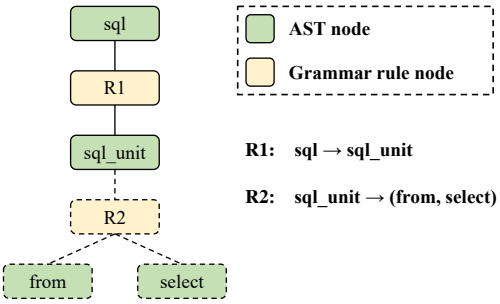


Figure 1: An example of the AST-Grammar Bipartite graph. The nodes in the dashed box represent the new nodes and edges generated after the "sql_unit" node is expanded using grammar rule 2.

In AST, non-terminal nodes sketch the general structure of the target SQL query, while terminal nodes correspond to operations, constants and variables. The decoding process does not end until all leaf nodes are terminal nodes and the SQL generation is regarded as a sequence generation problem.

**Problem Setting** Given a natural language question $Q = \{w_1, w_2, ..., w_n\}$ and a corresponding database schema $S = \langle C, T \rangle$, the goal is to generate an action sequence $\hat{a}$, which can construct an AST and deterministically be converted to a SQL query $y$. Here the database schema $S$ consists of tables $T = \{t_1, t_2, ..., t_{|T|}\}$, columns $C = \{c_1, c_2, ..., c_{|C|}\}$ and a set of foreign key, primary key column pairs describing relations between columns.

**AST-Grammar Bipartite Graph** A bipartite graph is a simple graph where the graph vertices are decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. In order to represent the structure of the partial AST and grammar rules that have been generated, we design an AST-Grammar Bipartite graph $G = \langle \mathcal{V}^{ast}, \mathcal{E}, \mathcal{V}^{g} \rangle$. $\mathcal{V}^{g}$ is the set of grammar rule nodes and $\mathcal{V}^{ast}$ is the set of AST nodes, which refers to the head and child nodes of all grammar rule nodes. $\mathcal{E}$ is the set of edges. A grammar rule node is considered an intermediate node between its head and child AST nodes, so a grammar rule node is only adjacent to AST nodes and vice versa. As shown in Figure 1, the grammar

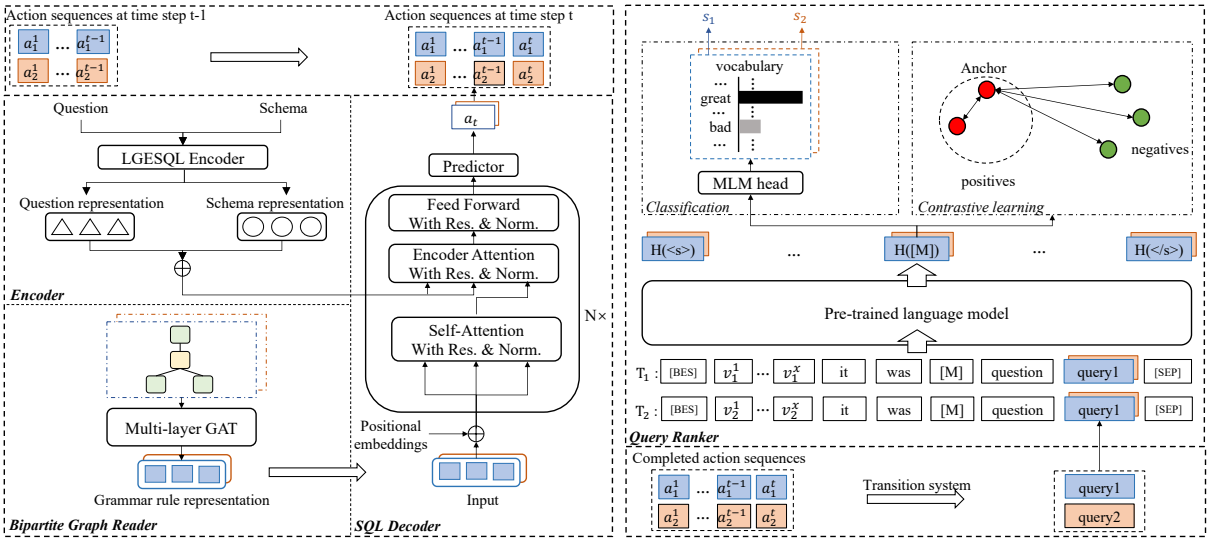


Figure 2: Overview of the architecture of the proposed G$^3$R framework and the process of generating action sequences and re-ranking SQL queries in the inference stage. The graph-guided SQL generator is in the left part of the figure, and the SQL re-ranker is in the right part of the figure. The blue and orange boxes represent two different action sequences in the beam and the maximum length of the sequence is $t$.

rule "2 : sql_unit $\rightarrow$ (from, select)" is composed of one head AST node "sql_unit" and two child AST nodes "from" and "select". When a grammar rule is generated, the graph will be expanded. The dashed box in Figure 1 represents the nodes and edges generated after the 'sql_unit' node is expanded using the grammar rule 2. In the beginning, the graph has only one root AST node "sql". As the actions are generated, this graph gradually expands until the SQL is completely generated.

The AST-grammar bipartite graph jointly models the generated partial AST and the corresponding grammar rules. After graph representation learning, the representation of a grammar rule node contains not only the structural information of the AST, but also more fine-grained information from its surrounding AST nodes. Each AST node represents a construct occurring in the SQL query and can guide future SQL generation. For example, the AST node "from" in Figure 1 means that the subtree with it as the root node is the from clause of the SQL query. Therefore, the next grammar rules generated should be used to create the from clause. For existing grammar rule embedding approaches in code generation, Grape (Zhu et al., 2022) creates a grammar relation graph that only includes grammar rule nodes, TreeGen (Sun et al., 2020a) enhances grammar rules by adding position and depth embedding and uses Tree Convolution to incorporate information from their ancestors, GrammarCNN (Sun et al., 2019) adopts three CNNs to capture the local feature of an AST node, features of AST sequences and tree paths. However, They considered either the structure of AST or the relationship between grammar rules rather than modelling them jointly.

## 4 Method

### 4.1 Graph-Guided SQL Generator

As mentioned above, existing SQL generators lack the ability to capture the structure of AST. We propose a graph-guided SQL generator based on the Encoder-Decoder framework to solve this problem. The architecture of the proposed graph-guided SQL generator is shown in the left part of Figure 2.

We leverage LGESQL (Cao et al., 2021) (Please refer to the Appendix A.1 for further details) encoder to embed the question and database schema items into joint representations $\boldsymbol{W} = [\boldsymbol{w_1}, ..., \boldsymbol{w_n}]$, $\boldsymbol{T} = [\boldsymbol{t_1}, ..., \boldsymbol{t_{|T|}}]$, $\boldsymbol{C} = [\boldsymbol{c_1}, ..., \boldsymbol{c_{|C|}}]$ for each question token $w_i \in Q$, table $t_i \in T$, and column $c_i \in C$ respectively. Considering a generated partial action sequence $\hat{a}' = \{a_1, .., a_{t-1}\}$ and the corresponding AST-Grammar Bipartite graph $\mathcal{G}^b_{t-1}$, the graph-guided SQL generator aims to learn the structural information of $\mathcal{G}^b_{t-1}$ and fuse multiple heterogeneous inputs to predict the next action $a_t$. It consists of two parts: (1) Bipartite Graph Reader, graph attention network (GAT) (Veličković et al., 2017) is adopted to learn the node representation of $\mathcal{G}^b_{t-1}$; (2) SQL Decoder, a neural network which is similar to the decoder of Transformer is proposed

to fuse heterogeneous input and predict the next action.

**Bipartite Graph Reader** In the decoding process, the AST-Grammar Bipartite graph will be expanded gradually. As the graph structure will change dynamically, we apply GAT which employs multi-head attention for graph representation learning.

For the graph $\mathcal{G}_{t-1}^{b}$ that contains $m$ grammar rule nodes and $h$ AST nodes, it has a node list $N = \{n_1^g, ..., n_m^g, n_1^{ast}, ..., n_h^{ast}\}$. Grammar rules and AST elements are represented as trainable embedding matrices $\boldsymbol{X^g} \in \mathbb{R}^{|V^g| \times k}$, $\boldsymbol{X^{ast}} \in \mathbb{R}^{|V^{ast}| \times k}$ where $|V^g|$, $|V^{ast}|$ are the sizes of grammar rule dict and AST element dict respectively. $k$ is the dimension of the embedding vector. We initialize the representation of each node by looking up the table. For the special AST node 'tab_id' and 'col_id', which can generate tables and columns, we initialize it using the corresponding representation from the encoder. In this way, we have node initial embeddings $\boldsymbol{X} = [\boldsymbol{x_1^g}, ..., \boldsymbol{x_m^g}, \boldsymbol{x_1^{ast}}, ..., \boldsymbol{x_h^{ast}}] \in \mathbb{R}^{(m+h) \times k}$. Then we use the GAT network to obtain the final hidden vectors of grammar rule nodes $\boldsymbol{H^g} = [\boldsymbol{h_1^g}, ..., \boldsymbol{h_m^g}] \in \mathbb{R}^{m \times k}$ that incorporate the structural information of AST.

**SQL Decoder** The SQL decoder considers heterogeneous information, including the question $Q$, database schema $S$, generated actions $\hat{a}'$ and the structure of $\mathcal{G}_{t-1}^{b}$ to predict the next action $a_t$. The neural structure of the SQL decoder is similar to the decoder of Transformer (Vaswani et al., 2017) because it can help alleviate long-dependency problem (Sun et al., 2020b). It is composed of a stack of blocks (N blocks in total), and each block contains three sub-layers (namely, the self-attention sub-layer, the encoder attention sub-layer and the feed-forward sub-layer). The residual connection and layer normalization are incorporated into each sub-layer. The input of the SQL decoder is $\boldsymbol{H^g}$.

**Encoder Attention Sub-layer** The main difference with the decoder of Transformer is in the encoder attention sub-layer. The decoder should be aware of the question and the database schema when predicting an action. The encoder attention sub-layer adopts multi-head attention to incorporate the information from the encoding stage in a way similar to the decoder's attention to the encoder in Transformer (Vaswani et al., 2017). Concretely speaking, we concatenate the representations of the questions with the representations of the database schema items $\boldsymbol{E} = [\boldsymbol{W}, \boldsymbol{T}, \boldsymbol{C}]$ which are considered as key, value of the multi-head attention layer.

**Action Prediction** The final representation $\boldsymbol{H} \in \mathbb{R}^{m \times k}$ is obtained through the stack of the blocks. We apply self attention to construct a representation $\boldsymbol{H_r} \in \mathbb{R}^k$ for ApplyRule action:

$$A_r = \text{softmax}(\tanh(W_r \boldsymbol{H})) \tag{1}$$

$$\boldsymbol{H_r} = A_r \boldsymbol{H} \tag{2}$$

where $W_r$ is a parameter. $\boldsymbol{H_t} \in \mathbb{R}^k$ and $\boldsymbol{H_c} \in \mathbb{R}^k$ are constructed for SelectTable and SelectColumn action in the same way.

For ApplyRule Action, the probability distribution $P_r$ over the grammar rule space is calculated as follows:

$$P_r = \text{softmax}(\tanh(W_s H_r)) \tag{3}$$

Where $W_s$ is a weight parameter of the fully connected layer mapping $H_r$ to the grammar rule space.

For the SelectTable action (SelectColumn action), we apply multi-head attention to calculate the similarity matrix $A_t \in \mathbb{R}^{|T|}$ where the query is $\boldsymbol{H_t}$, key and value are $\boldsymbol{T}$. Then softmax activation function is applied to calculate the probability for all tables $P_t \in \mathbb{R}^{|T|}$:

$$P_t = \text{softmax}(A_t) \tag{4}$$

The decoding process of generating a SQL query is shown in Algorithm 1. The model is optimized by minimizing the negative log-likelihood $\mathcal{L}_g$ of the ground truth action sequence:

$$\mathcal{L}_g = -\sum_{i=0}^{N} log \prod_{j=1}^{N_i} p(a_j | a_{<j}, Q, S) \tag{5}$$

Where $N$ denotes the number of samples in the training set and $N_i$ represents the length of the action sequence for the i-th sample.

### 4.2 Knowledge-Enhanced Re-ranking Mechanism with contrastive soft prompt tuning

In the inference stage, we applied the beam search decoding method which keeps $M$ locally best candidates from the decoder output and generates the

$M$ most likely candidate action sequences. However, top-1 prediction does not necessarily correspond to the best SQL query and (Kelkar et al., 2020) first introduced a re-ranking algorithm on complex Text-to-SQL generation and the re-ranker is based on BERT (Devlin et al., 2019) with a linear classification layer. The improvement of their proposed re-ranking algorithm is unstable and highly dependent on a threshold. And at certain threshold settings, it even has a negative effect. The reason is they don't explore the knowledge of PLM sufficiently. (Hui et al., 2021) proposed a feature-enhanced re-ranker which is also based on PLM with a linear layer. In order to introduce domain knowledge, we propose a knowledge-enhanced re-ranking mechanism. The SQL re-ranker is based on a PLM and hybrid prompt tuning is adopted to explore the knowledge in PLMs to bridge the gap between different domains without introducing extra parameters. Contrastive learning is applied to push away distances of candidate queries' representations so that they are more distinguishable.

**Hybrid prompt tuning** PLM is adopted as the SQL re-ranker and the template $T$ is designed by inserting some tunable embeddings into a hard prompt:

$$\{E([\text{BES}]), V, E(\text{“it”}), E(\text{“is”}), E([\text{mask}]), E(Q), E(y), E([\text{SEP}])\} \quad (6)$$

Where $V = [v_1, ..., v_x]$ is a learnable virtual template word and $x$ is the length. $Q$ and $y$ represent the question and query respectively. The scoring task can be formalized as a binary classification task and transformed into a cloze-style objective form, by predicting the masked position and mapping the predicted words to the corresponding class labels. The label word set is defined as {“bad”, “great”} and we can determine whether the candidate query is the selected answer or not based on the PLM predicting “great” or “bad” at the mask position. The probability distribution $P \in \mathbb{R}^2$ over label space is calculated as follows:

$$H_v = \text{MLMhead}(T, Q, y) \quad (7)$$

$$P = \text{softmax}(H_v(\text{“bad”}), H_v(\text{“great”})) \quad (8)$$

**Contrastive Learning** The differences between the candidate queries sometimes are subtle, resulting in undifferentiated representations of the mask position and similar scores for different queries. We introduce contrastive learning to push away the representation of each candidate query, thus making them more distinguishable.

Given a question $Q$ and all its candidate queries $Y = \{y_1, ..., y_M\}$, we construct inputs $I = \{i_1^{y_1}, i_2^{y_1}, ..., i_1^{y_M}, i_2^{y_M}\}$ for SQL re-ranker by using two different prompt template $T_1$ and $T_2$ where $V_1$ and $V_2$ are distinct and randomly initialised. For a query $y_i$, the inputs $i_1^{y_i}$ and $i_2^{y_i}$ constructed from two templates form a positive pair while negative pairs are formed with other $2(M-1)$ inputs. All inputs $I$ for $Q$ are in a mini-batch and the contrastive learning loss (Oord et al., 2018; Chen et al., 2020) $\mathcal{L}_{cl}$ is calculated as follows :

$$\mathcal{L}_{cl} = -log \sum_{i=1}^{2M} \frac{\exp(\text{sim}(z_i, z_+)/\tau)}{\exp(\sum_{j=1}^{2M} \mathbb{1}_{[j \neq i]} \text{sim}(z_i, z_j)/\tau)} \quad (9)$$

where $z_i$, $z_+$ represent the hidden states of the PLM output at the mask position of the i-th sample and its corresponding positive sample respectively. $\mathbb{1}_{[j \neq i]} \in \{0, 1\}$ is a indicator function evaluating to 1 iff $i \neq j$. sim$(\cdot)$ is the cosine similarity function and $\tau$ is a temperature hyper-parameter that controls the sensitivity of the product. Further more, we introduce a margin loss $\mathcal{L}_d$ to constrain the two templates to be different:

$$\mathcal{L}_d = \max(0, \cos(V_1, V_2) - d) \quad (10)$$

where cos$(\cdot)$ is the cosine similarity and $d$ is a hyperparameter. Cross-entropy loss $\mathcal{L}_{ce}$ is adopted as the loss function for SQL re-ranker following the same way as (Nam et al., 2014):

$$\mathcal{L}_{ce} = -\sum_{i=1}^{N_Q \times 2M} \sum_{j=1}^{2} (y_{ij} \log(P_i^j)) + (1 - y_{ij})(1 - \log(P_i^j)) \quad (11)$$

where $N_Q$ represents the number of questions. The total loss function $\mathcal{L}_r$ for SQL re-ranker is the sum of the three loss functions:

$$\mathcal{L}_r = \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{cl} + \lambda_2 \mathcal{L}_d \quad (12)$$

where $\lambda_1$ and $\lambda_2$ are hyperparameters. In order to fine-tune the SQL re-ranker, we construct a training (dev) set containing abundant positive and negative samples. For the training (dev) set, positive samples are the ground truth, while negative samples are the wrong samples generated by the SQL generator on the training (dev) set.

**Knowledge-Enhanced Re-Ranking Mechanism** For each query $y_i \in Y$, the log probability of a positive label output by the SQL re-ranker is considered to be the semantic score. Two scores $s_i^{T_1}$, $s_i^{T_2}$ can be obtained by using two templates. We take into account the log-likelihood $s_i^g$ of its corresponding action sequence produced by the SQL generator and get the final score $s_i$:

$$s_i = s_i^g + \frac{s_i^{T_1} + s_i^{T_2}}{2} \tag{13}$$

The candidate query with the highest score is considered the final result. The proposed SQL re-ranker and knowledge-enhanced re-ranking mechanism are general and model agnostic which can be applied to any Text-to-SQL approach.

## 5 Experiments

### 5.1 Dataset

In order to evaluate the proposed Graph-Guided Generate-and-Rerank framework, we conduct experiments on Spider (Yu et al., 2018) and Spider-DK (Gan et al., 2021b).

**Spider** It is a large-scale complex and cross-domain Text-to-SQL generation dataset[1]. There are 8659 samples in the training set across 146 databases and 1034 samples in the development set across 20 databases distinct from those in the training set. The test set is not public and can only be accessed through an evaluation server, we evaluate the proposed approach mainly on the development set.

**Spider-DK** It is a human-curated dataset based on the dev set of Spider [2]. The questions of the dataset are selected from Spider with some domain knowledge added. It focuses on evaluating the ability to understand domain knowledge.

### 5.2 Implementations

For the Graph-Guided SQL generator, we use GLOVE (Pennington et al., 2014) word embeddings of size 300 and ELECTRA (Clark et al., 2020) to encode the input question and the database schema, followed by 8 LGESQL encoder layers. The number of GAT layers is 2 and the number of parameters is about 385M (With ELECTRA). The whole model is trained by the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 5e-5 and 2e-4 for with and without a pre-trained language model (PLM) respectively. The mini-batch size of the input is set to 10. In the inference stage, the beam size is set to 5.

As for the SQL re-ranker, Grappa (Yu et al., 2021) is adopted for complex and cross-domain text-to-SQL generation because it is pre-trained for table semantic parsing and contains rich knowledge to bridge the gap between different domains. The number of parameters is about 355M. The length $x$ of the learnable virtual template word for the hybrid prompt is set to 10, and $\lambda_1$ and $\lambda_2$ are set to 0.1 and 1 respectively. It is optimized by Adam with a learning rate of 1e-5.

| Model | Dev | Test |
|---|---|---|
| **Without PLM** | | |
| EditSQL (Zhang et al., 2019) | 36.4 | 32.9 |
| Global-GNN (Bogin et al., 2019b) | 52.7 | 47.4 |
| IRNet (Guo et al., 2019) | 53.2 | 46.7 |
| RAT-SQL (Wang et al., 2020a) | 62.7 | 57.2 |
| LGESQL (Cao et al., 2021) | 67.6 | 62.8 |
| **G$^3$R** | **71.4** | **64.5** |
| **With PLM** | | |
| RAT-SQL+BERT-large | 69.7 | 65.6 |
| RAT-SQL+GAP (Shi et al., 2021a) | 71.8 | 69.7 |
| RAT-SQL+STRUG | 72.6 | 68.4 |
| RAT-SQL+GRAPPA | 73.4 | 69.6 |
| SmBoP (Rubin and Berant, 2021) | 74.7 | 69.5 |
| DT-Fixup SQL-up (Xu et al., 2020) | 75.0 | 70.9 |
| LGESQL+BERT | 74.1 | 68.3 |
| LGESQL+ELECTRA | 75.1 | 72.0 |
| **G$^3$R+ELECTRA** | **78.1** | **72.9** |

Table 2: Performance comparison with some state-of-art methods without and with PLM on the dev set and test set of Spider. G$^3$ is short for the graph-guided SQL generator and G$^3$R incorporates the re-ranking mechanism.

### 5.3 Evaluation Metrics

Following the previous work (Cao et al., 2021), we adopt exact match accuracy as the evaluation metric. It is calculated by decomposing SQL into several clauses and conducting a set comparison in each SQL clause.

[1]Spider: https://yale-lily.github.io//spider.
[2]Spider-DK: https://github.com/ygan/Spider-DK

| Model | Easy | Medium | Hard | Extra | ALL |
|---|---|---|---|---|---|
| **Without PLM** | | | | | |
| IRNet | 70.1 | 49.2 | 39.5 | 19.1 | 53.2 |
| RAT-SQL | 80.4 | 63.9 | 55.7 | 40.6 | 62.7 |
| LGESQL | 86.3 | 69.5 | 61.5 | 41.0 | 67.6 |
| **G$^3$** | 87.1 | 71.3 | 64.9 | 39.8 | 69.0 |
| **G$^3$R** | **89.1** | **74.1** | **67.8** | **41.8** | **71.4** |
| **With ELECTRA** | | | | | |
| LGESQL | **91.9** | 78.3 | 64.9 | 52.4 | 75.1 |
| **G$^3$** | 89.5 | 80.9 | 69.5 | 53.0 | 76.6 |
| **G$^3$R** | 89.5 | **82.7** | **73.0** | **54.9** | **78.1** |

Table 3: Exact match accuracy on the dev set of Spider according to the level of difficulty.

### 5.4 Main Results

Table 2 shows the experimental results of several Text-to-SQL methods without and with PLM. Regarding baseline models, LGESQL achieves the best performance and our proposed G$^3$R clearly outperforms it by a substantial margin which shows the superiority of our approach. With word vectors GLOVE, G$^3$R achieves an absolute improvement of 3.8%, 1.7% over LGESQL on exact match accuracy on the dev set and test set respectively. With PLM ELECTRA, G$^3$R surpasses all the SOTA baselines and achieves an absolute improvement of 3% and 0.9% over LGESQL on the dev set and test set respectively. It achieves an accuracy of 78.1%, and 72.9% on the dev and test set which is competitive compared to some newly released T5-3B based approaches (Li et al., 2023; Zhao et al., 2022). However, the size of our model is only one-fifth of theirs.

In addition, more fine-grained performances of some approaches ordered by the level of difficulty are shown in Table 3. We can observe that: (1) As the difficulty increases, the syntactic structure of SQL queries becomes more complex and the accuracy of all approaches decreases. (2) With word vectors GLOVE, compared with LGESQL, G$^3$R achieves improvements of 2.8%, 4.6%, 6.3% and 0.8% at the easy, medium, hard and extra hard levels respectively. With PLM ELECTRA, G$^3$R also achieves improvements of 4.4%, 8.1% and 2.5% over LGESQL at medium, hard, and extra hard levels respectively. It demonstrates the effectiveness of our proposed framework. (3) When the re-ranking mechanism is removed, the accuracy of G$^3$ (with ELECTRA) is decreased by 1.5%, but it still achieves an improvement of 1.5% compared with LGESQL. Notably, G$^3$ (with ELECTRA) has a significant improvement in complex SQL query generation, achieving improvements of 4.6% and 0.6% at the hard and extra hard levels. It demonstrates that our proposed graph-guided SQL generator can exploit the structural information of the AST and generate complex structured SQL queries more accurately.

### 5.5 Re-Ranking Results

| Spider | |
|---|---|
| **Model Name** | **Acc** |
| G$^3$ | 69.0 |
| + Bertrand-DR (Kelkar et al., 2020) | 70.7 |
| **+ KE-R** | **71.4** |
| Beam accuracy | 75.6 |
| G$^3$+ELECTRA | 76.6 |
| + Bertrand-DR | 77.2 |
| **+ KE-R** | **78.1** |
| Beam accuracy | 82.7 |

Table 4: Re-ranking results on the dev set of Spider. "KE-R" represents the proposed knowledge-enhanced re-ranking mechanism.

| Spider-DK | |
|---|---|
| **Model Name** | **Acc** |
| RAT-SQL + GAP (Shi et al., 2021b) | 44.1 |
| LGESQL+ELECTRA | 47.3 |
| G$^3$+ELECTRA | **49.7** |
| + Bertrand-DR (Kelkar et al., 2020) | 50.5 |
| **+ KE-R** | **51.6** |
| w/o CL | 50.8 |
| w/o ML | 51.0 |
| w/o CL&ML | 50.6 |
| Beam accuracy | 60.9 |

Table 5: Re-ranking results on Spider-DK. "CL" represents the contrastive learning and "ML" represents the margin loss.

We construct the training set and dev set of the SQL re-ranker in the manner mentioned in Section 4.2. Bertrand-DR (Kelkar et al., 2020) is adopted as a baseline for comparison with our knowledge-enhanced re-ranking mechanism. To be fair, it is also based on Grappa, with a linear layer

for binary classification. As shown in Table 4, the beam accuracy is on average about 6% higher than the top-1 accuracy. Our proposed KE-R can effectively filter out the correct answer from the beam. When it is applied, the exact match accuracy of $G^3$, $G^3$+ELECTRA and are improved by 2.4% and 1.5% respectively. Compared with Bertrand-DR, KE-R in combination with $G^3$ and $G^3$+ELECTRA achieves improvements of 0.7% and 0.9% which shows its superiority.

To further evaluate the ability of our proposed SQL re-ranker to introduce domain knowledge, we conduct experiments on Spider-DK and the results are shown in Table 5. It is obvious that: (1) The dataset is more challenging and the accuracy of all approaches has decreased. However, our proposed $G^3$ still outperforms the other two baselines and achieves an improvement of 2.4% compared with LGESQL+ELECTRA. (2) With KE-R, the accuracy of $G^3$+ELECTRA is improved by 1.9% compared to 0.8% with Bertrand-DR, indicating that our proposed SQL re-ranker can effectively incorporate domain knowledge. (3) When both contrastive learning and margin loss are removed, the accuracy drops to 50.6% and it still gains an improvement of 0.9% compared with $G^3$+ELECTRA which demonstrates the effectiveness of the hybrid prompt tuning. (4) When contrastive learning and margin loss are removed, the accuracy decreases by 0.8% and 0.6% respectively, suggesting that they both contribute to the significant improvement.

## 6 Conclusion

In this paper, a novel Graph-Guided Generated-and-Rerank framework is proposed for complex and cross-domain Text-to-SQL generation. In specific, we design the AST-Grammar Bipartite graph and propose a Graph-Guided SQL generator to capture the structural information of the generated complex SQL query. A knowledge-enhanced re-ranking mechanism is proposed to introduce domain knowledge to bridge the gap between different domains and re-rank candidate SQL queries generated from top-ranked action sequences. Experimental results on the Spider and Spider-DK benchmark datasets show that the proposed method outperforms other competitive Test-to-SQL baselines.

## 7 Limitation

Our proposed graph-guided SQL generators are superior in generating complex SQL queries. However, the model has a large number of parameters and requires more computational resources, which is a common problem with current methods of generating complex SQL queries.

In addition, the proposed knowledge-enhanced re-ranking mechanism is proposed to leverage the knowledge in PLM to choose the best SQL query from the beam output. However, it does not take into account the database schema which can be the source of domain knowledge.

In the future, we will design lighter models for complex and cross-domain text-to-SQL generation and explore some other re-ranking mechanisms to incorporate the prior knowledge of database schema.

## 8 Acknowledgement

We would like to thank anonymous reviewers for their valuable comments and helpful suggestions. We thank Tencent Cloud Xiaowei and the Big Data Computing Center of Southeast University for supporting this project. We thank Tao Yu, Yusen Zhang and Hongjin Su for their careful assistance with the evaluation. This work was funded by the National Natural Science Foundation of China (62176053).

## References

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-sql parsing. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4560–4565.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Global reasoning over database structures for text-to-sql parsing. 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 3657–3662.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ursin Brunner and Kurt Stockinger. 2021. Valuenet: A natural language-to-sql system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering, ICDE*, pages 2177–2182.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: line graph enhanced text-to-sql model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 2541–2555.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pages 1597–1607.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021a. Exploring underexplored limitations of cross-domain text-to-sql generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 8926–8931.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-sql generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 8926–8931.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 3816–3830.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 8410–8423.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, pages 4524–4535.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

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 *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, pages 13116–13124.

Amol Kelkar, Rohan Relan, Vaishali Bhardwaj, Saurabh Vaichal, Chandra Khatri, and Peter Relan. 2020. Bertrand-dr: Improving text-to-sql using a discriminative re-ranker. *arXiv preprint arXiv:2002.00557*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1516–1526.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 3045–3059.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *CoRR*, abs/2301.07507.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP*, pages 1532–1543.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1139–1149.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ohad Rubin and Jonathan Berant. 2021. Smbop: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 311–324.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 2339–2352.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cícero Nogueira dos Santos, and Bing Xiang. 2021a. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, pages 13806–13814.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. 2021b. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, volume 35, pages 13806–13814.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*.

Zeyu Sun, Qihao Zhu, Lili Mou, Yingfei Xiong, Ge Li, and Lu Zhang. 2019. A grammar-based structural CNN decoder for code generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, pages 7055–7062.

Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, and Lu Zhang. 2020a. Treegen: A tree-based transformer architecture for code generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pages 8984–8991.

Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, and Lu Zhang. 2020b. Treegen: A tree-based transformer architecture for code generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pages 8984–8991.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 7567–7578.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020b. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 3229–3238.

Hui Wu and Xiaodong Shi. 2022. Adversarial soft prompt tuning for cross-domain sentiment analysis. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 2438–2447.

Peng Xu, Wei Yang, Wenjie Zi, Keyi Tang, Chengyang Huang, Jackie Chi Kit Cheung, and Yanshuai Cao. 2020. Optimizing deeper transformers on small datasets: An application on text-to-sql semantic parsing. *CoRR*, abs/2012.15355.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 440–450.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 7–12.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 3911–3921.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. 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 5337–5348.

Yiyun Zhao, Jiarong Jiang, Yiqun Hu, Wuwei Lan, Henry Zhu, Anuj Chauhan, Alexander Hanbo Li, Lin Pan, Jun Wang, Chung-Wei Hang, Sheng Zhang, Marvin Dong, Joe Lilien, Patrick Ng, Zhiguo Wang, Vittorio Castelli, and Bing Xiang. 2022. Importance of synthesizing high-quality data for text-to-sql parsing. *CoRR*, abs/2212.08785.

Deyu Zhou, Yanzheng Xiang, Linhai Zhang, Chenchen Ye, Qian-Wen Zhang, and Yunbo Cao. 2021. A divide-and-conquer approach for multi-label multi-hop relation detection in knowledge base question answering. In *Findings of the Association for Computational Linguistics,EMNLP*, pages 4798–4808.

Qihao Zhu, Zeyu Sun, Wenjie Zhang, Yingfei Xiong, and Lu Zhang. 2022. Grape: Grammar-preserving rule embedding. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, pages 4545–4551.

**Algorithm 1** The process of generating a complex SQL query.

**Input:** Question $q$, Database schema $S$, Maximum decoding steps $M_d$.

**Output:** The action sequence $\hat{a}$ corresponding to the SQL query.

$\hat{a} \leftarrow$ ("[BES]" ), $\mathcal{G}_0^b \leftarrow$ "sql"
$\boldsymbol{W}, \boldsymbol{T}, \boldsymbol{C} \leftarrow \text{Encoder}(Q, S)$
**for** $i \leftarrow 1; i \leq M_d; i++$ **do**
  Bipartite Graph Reader captures the structure information of $\mathcal{G}_{i-1}^b$ and $\boldsymbol{H^g}$ is obtained (4.1).
  $a_i \leftarrow \text{Decoder}(\boldsymbol{H^g}, \boldsymbol{W}, \boldsymbol{T}, \boldsymbol{C})$
  **if** $a_i =$ "[EOS]" **then**
    $\hat{a} \leftarrow \hat{a} + a_i$
    Break
  **else**
    $\mathcal{G}_i^b \leftarrow$ Update the graph structure of $\mathcal{G}_{i-1}^b$ by applying the predicted action $a_i$.
    $\hat{a} \leftarrow \hat{a} + a_i$
  **end if**
**end for**
**return** $\hat{a}$

# A Appendix

## A.1 LGESQL Encoder

| Source x | Target y | Relation |
|---|---|---|
| Column | Table | x belongs to y.<br>x is the primary key of y. |
| Question | Question | x is the next token of y. |
| Column | Column | x is the foreign key of y. |
| Question | Column | x and y do not overlap.<br>x is part of y, y is (not) a span of the question (Exact/Partial Match). |
| Question | Table | x and y do not overlap.<br>x is part of y, y is (not) a span of the question (Exact/Partial Match). |

Table 6: Description of relations types in the heterogeneous graph $G^n$ between different types of nodes.

Unlike other code generation tasks, Text-to-SQL generation needs to consider not only the natural language question but also the database schema. It is a daunting problem how to jointly encode the question and database schema, as there exist various relations between these heterogeneous inputs. In order to address the problem, LGESQL constructs a node-centric heterogeneous graph $\mathcal{G}^n$. It consists of three kinds of nodes: tables, columns and question tokens. The relations between different types of nodes are shown in Table 6.

Besides, there exist many meta-paths between the nodes. Meta-path is defined as a composite relation linking two nodes, that can capture multi-hop semantics. For example, a meta-path <Question, "Question-ExactMatch-Column", Column, "Column-BelongsTo-Table", Table> means the question mentions a column in the table. As the length of the path increases, the number of possible meta-paths increases exponentially. In order to tackle this problem, LGESQL proposes to utilize a line graph to capture the topological structure of edges explicitly. An edge-centric graph $\mathcal{G}^e$ is constructed from the original node-centric heterogeneous graph $\mathcal{G}^n$ and a dual relational attention network (Dual RGAT) (Wang et al., 2020b) is adopted to capture the structure of the original graph and the line graph iteratively. Finally, the representations of the nodes are obtained through the LGESQL encoder.

### A.2 Decoding Process

The decoding process is shown in Algorithm 1.

## ACL 2023 Responsible NLP Checklist

### A For every submission:

☑ A1. Did you describe the limitations of your work?
*Section 7*

☑ A2. Did you discuss any potential risks of your work?
*Section 7*

☑ A3. Do the abstract and introduction summarize the paper's main claims?
*Section 1*

☒ A4. Have you used AI writing assistants when working on this paper?
*Left blank.*

### B ☑ Did you use or create scientific artifacts?

*Section 1, Section 5*

☑ B1. Did you cite the creators of artifacts you used?
*Section 1, Section 5*

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*Section 1, Section 5*

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
*Section 5*

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
*The dataset is open source and uses the setup from previous work.*

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*Section 5*

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
*Section 5*

### C ☑ Did you run computational experiments?

*Section 5*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*Section 5*

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
*Section 5*

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
*Section 5*

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
*Section 5*

**D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
*No response.*

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
*No response.*

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
*No response.*

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
*No response.*

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
*No response.*