RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL

Jiexing Qi[1], **Jingyao Tang**[1], **Ziwei He**[1], **Xiangpeng Wan**[2],
Chenghu Zhou[3], **Xinbing Wang**[1], **Quanshi Zhang**[1], **Zhouhan Lin**[1*]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]NetMind.AI and Protagolab, Virginia, USA
[3]IGSNRR, Chinese Academy of Sciences, Beijing, China
{qi_jiexing, monstar, ziwei.he, zqs1022, xwang8}@sjtu.edu.cn
lin.zhouhan@gmail.com

Abstract

Relational structures such as schema linking and schema encoding have been validated as a key component to qualitatively translating natural language into SQL queries. However, introducing these structural relations comes with prices: they often result in a specialized model structure, which largely prohibits the use of large pretrained models in text-to-SQL. To address this problem, we propose RASAT: a Transformer seq2seq architecture augmented with relation-aware self-attention that could leverage a variety of relational structures while at the meantime being able to effectively inherit the pretrained parameters from the T5 model. Our model is able to incorporate almost all types of existing relations in the literature, and in addition, we propose to introduce co-reference relations for the multi-turn scenario. Experimental results on three widely used text-to-SQL datasets, covering both single-turn and multi-turn scenarios, have shown that RASAT could achieve competitive results in all three benchmarks, achieving state-of-the-art performance in execution accuracy (80.5% EX on Spider, 53.1% IEX on SParC, and 37.5% IEX on CoSQL). [1]

1 Introduction

The task of text-to-SQL aims at translating natural language questions into SQL queries. Since it could significantly break down barriers for non-expert users to interact with databases, it is among the most important semantic parsing tasks that are of practical importance (Kamath and Das, 2018; Deng et al., 2021).

Various types of relations have been introduced for this task since Zhong et al. (2017) collected the first large-scale text-to-SQL dataset, which has resulted in significant boosts in the performance through recent years. For example, Bogin et al. (2019b) introduced schema encoding to represent the schema structure of the database, and the resulting augmented LSTM encoder-decoder architecture was able to generalize better towards unseen database schema. Lin et al. (2020a) introduced relations between the entity mentioned in the question and the matched entries in the database to effectively utilize database content. Their BERT-based encoder is followed by an LSTM-based pointer network as the decoder generalizes better between natural language variations and captures corresponding schema columns more precisely. RAT-SQL (Wang et al., 2020a) introduced schema linking, which aligns mentions of entity names in the question to the corresponding schema columns or tables. Their augmented Transformer encoder is coupled with a specific tree-decoder. SADGA (Cai et al., 2021) introduced the dependency structure of the natural language question and designed a graph neural network-based encoder with a tree-decoder. On the other hand, a tree-decoder that is able to generate grammatically correct SQL queries is usually needed to better decode the encoder output, among which Yin and Neubig (2017) is one of the most widely used.

Although integrating various relational structures as well as using a tree-decoder have been shown to be vital to generating qualitative SQL queries and generalizing better towards unseen database schema, the development of various specifically designed model architectures significantly deviate from the general sequential form, which has made it hard if one considers leveraging large pre-trained models for this task. Existing methods either use BERT output as the input embedding of the specifically designed model (Cao et al., 2021; Choi et al., 2021; Wang et al., 2020a; Guo et al., 2019), or stack a specific decoder on top of BERT (Lin et al., 2020a).

In another thread, pretrained seq2seq models just

[*] Zhouhan Lin is the corresponding author.
[1]Our implementation will be available at https://github.com/LUMIA-group/rasat.

have unveiled their powerful potential for this task. Recent attempts by Shaw et al. (2021) show that directly fine-tuning a T5 model (Raffel et al., 2020) on this task without presenting any relational structures could achieve satisfying results. Moreover, PICARD (Scholak et al., 2021) presents a way to prune invalid beam search results during inference time, thus drastically improving the grammatical correctness of the SQL queries generated by the autoregressive decoder that comes with T5.

In this work, different from the more common approach of fine-tuning the original pretrained model or using prompt tuning, we propose to augment the self-attention modules in the encoder and introduce new parameters to the model while still being able to leverage the pre-trained weights. We call the proposed model RASAT[2]. Our model is able to incorporate almost all existing types of relations in the literature, including schema encoding, schema linking, syntactic dependency of the question, etc., into a unified relation representation. In addition to that, we also introduce coreference relations to our model for multi-turn text-to-SQL tasks. Experimental results show that RASAT could effectively leverage the advantage of T5. It achieves the state-of-art performance in question execution accuracy (EX/IEX) on both multi-turn (SParC and CoSQL) and single-turn (Spider) text-to-SQL benchmarks. On SParC, RASAT surpasses all previous methods in interaction execution accuracy (IEX) and improves the state-of-art performance from 46.4% to 53.1%, 6.7% absolute improvements. On CoSQL, we improve the state-of-art IEX performance from 26.2% to 37.5%, achieving 11.3% absolute improvements. Moreover, on Spider, we improve the state-of-art execution accuracy from 79.3% to 80.5%, achieving 1.2% absolute improvements.

2 Related Work

Early works usually exploit a sketch-based slot-filling method that use different modules to predict the corresponding part of SQL. These method decomposes the SQL generation task into several independent sketch and use different classifier to predict corresponding part, such as SQLNet (Xu et al., 2017), TypeSQL (Yu et al., 2018a), SQLOVA (Hwang et al., 2019), X-SQL (He et al., 2019), RYANSQL (Choi et al., 2021) et.al,. However, most of these methods only handle simple queries while failing to generate correct SQL in a complex setting such as on Spider.

Faced with the multi-table and complex SQL setting, using graph structures to encode a variety of complex relationships is a major trend in the text-to-SQL task. For example, Global-GNN (Bogin et al., 2019a) represents the complex database schema as a graph, RAT-SQL (Wang et al., 2020a) introduce schema encoding and linking and assign every two input items a relation, LGESQL (Cao et al., 2021) further distinguish local and non-local relations by exploiting a line graph enhanced hidden module, SADGA (Cai et al., 2021) use contextual structure and dependency structure to encode question-graph while database schema relations are used in schema graph, S^2SQL (Hui et al., 2022) add syntactic dependency information in relational graph attention network(RGAT) (Wang et al., 2020b).

For the conversational context-dependent text-to-SQL task that includes multiple turns of interactions, such as SParC and CoSQL, the key challenge is how to take advantage of historical interaction context. Edit-SQL (Zhang et al., 2019) edit the last turn's predicted SQL according to the current turn's question to generate the newly predicted SQL at the token level. IGSQL (Cai and Wan, 2020) use cross-turn and intra-turn schema graph layers to model database schema items in a conversational scenario. Tree-SQL (Wang et al., 2021b) uses a tree-structured intermediate representation and proposes a reuse strategy that assigns a probability for each sub-tree of historical Tree-SQLs. IST-SQL (Wang et al., 2021a) proposes an interaction state tracking method to predict the SQL query in the current turn. RAT-SQL-TC (Li et al., 2021)add two auxiliary training tasks to explicit modeling the semantic changes in both turn grain and conversation grain. R^2SQL (Hui et al., 2021) and HIE-SQL (Zheng et al., 2022) introduce a dynamic schema-linking graph by adding the current utterance, interaction history utterances, database schema, and the last predicted SQL query.

Recently, Shaw et al. (2021) showed that fine-tuning a pre-trained T5-3B model could yield results competitive to the then-state-of-the-art. Based on this discovery, Scholak et al. (2021) proposed to constrain the autoregressive decoder through incremental parsing during inference time, effectively filtering out grammatically incorrect sequences on the fly during beam search, which significantly improved the qualities of the generated SQL.

[2]RASAT: Relation-Aware Self-Attention-augmented T5

3 Preliminaries

3.1 Task Formulation

Given a natural language question $\mathcal{Q}$ and database schema $\mathcal{S} = <\mathcal{T}, \mathcal{C}>$, our goal is to predict the SQL query $\mathcal{Y}$. Here $\mathcal{Q} = \{q_i\}_{i=1}^{|\mathcal{Q}|}$ is a sequence of natural language tokens, and the schema $\mathcal{S}$ consists of a series of tables $\mathcal{T} = \{t_i\}_{i=1}^{|\mathcal{T}|}$ with their corresponding columns $\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{|\mathcal{T}|}$. The content of database $\mathcal{S}$ is noted as $\mathcal{V}$. For each table t_i, the columns in this table is denoted as $\mathcal{C}_i = \{c_{ij}\}_{j=1}^{|\mathcal{C}_i|}$. For each table t_i, the table name contains $|t_i|$ tokens $t_i = t_{i,1}, \cdots, t_{i,|t_i|}$, and the same holds for column names. In this work, we present the predicted SQL query as a sequence of tokens, $\mathcal{Y} = \{y_i\}_{i=1}^{|\mathcal{Y}|}$.

In the multi-turn setting, our notations adapt correspondingly. i.e., $\mathcal{Q} = \{\mathcal{Q}_i\}_{i=1}^{|\mathcal{Q}|}$ denotes a sequence of questions in the interaction, with $\mathcal{Q}_i$ denoting each question. Also, the target to be predicted is a sequence of SQL queries, $\mathcal{Y} = \{\mathcal{Y}_i\}_{i=1}^{|\mathcal{Y}|}$, with each $\mathcal{Y}_i$ denoting the corresponding SQL query for the i-th question $\mathcal{Q}_i$. Generally, for each question, there is one corresponding SQL query, such that $|\mathcal{Q}| = |\mathcal{Y}|$. While predicting $\mathcal{Y}_i$, only the questions in the interaction history is available, i.e., $\{\mathcal{Q}_1, \cdots, \mathcal{Q}_i\}$.

3.2 Relation-aware Self-Attention

Relation-aware self-attention (Shaw et al., 2018; Wang et al., 2020a) augments the vanilla self-attention (Vaswani et al., 2017) by introducing relation embeddings into the key and value entries. Assume the input to the self attention is a sequence of n embeddings $X = \{x_i\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^{d_x}$, then it calculates its output z as ($||$ means concatenate operation):

$$\alpha_{ij}^{(h)} = \operatorname{softmax}\left(\frac{\boldsymbol{x_i} W_Q^{(h)} \left(\boldsymbol{x_j} W_K^{(h)} + \boldsymbol{r}_{ij}^K\right)^\top}{\sqrt{d_z/H}}\right)$$
$$\boldsymbol{z}_i = \Big\Vert_{h=1}^{H} \left[\sum_{j=1}^{n} \alpha_{ij}^{(h)} \left(\boldsymbol{x_j} W_V^{(h)} + \boldsymbol{r}_{ij}^V\right)\right] \tag{1}$$

where H is the number of heads, and $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}$ are learnable weights. The $\boldsymbol{r}_{ij}^K, \boldsymbol{r}_{ij}^K$ are two different relation embeddings used to represent the relation r between the i-th and j-th token.

4 RASAT

4.1 Model Overview

The overall structure of our RASAT model is shown in Figure 1. Architecture-wise it is rather simple: the T5 model is taken as the base model, with its self-attention modules in the encoder substituted as relation-aware self-attentions.

The input to the encoder is a combination of question(s) $\mathcal{Q}$, database schema $\mathcal{S} = <\mathcal{T}, \mathcal{C}>$, as well as database content mentions and necessary delimiters. We mostly follow Shaw et al. (2021) and Scholak et al. (2021) to serialize the inputs. Formally,

$$X = \overline{\mathcal{Q} \,|\, \mathcal{S} \,|\, t_1 : c_{11}\, [v]\, , \cdots , c_{1|T_1|} \,|\, t_2 : c_{21}\, , \cdots} \tag{2}$$

where $\mathcal{S}$ is the database name, t_i is the table name, c_{ij} is the j-th column name of the i-th table. The $v \in \mathcal{V}$ showing after column c_{11} is the database content belonging to the column that has n-gram matches with the tokens in the question. As for delimiters, we use | to note the boundaries between $\mathcal{Q}$, $\mathcal{S}$, and different tables in the schema. Within each table, we use : to separate between table name and its columns. Between each column, , is used as the delimiter.

As for the multi-turn scenario, we append the questions in the history in the end of the sequence and truncate them when the sequence length limit of T5 is reached. i.e.,

$$X = \overline{\mathcal{Q}_t \,|\, \mathcal{S} \,|\, t_1 : c_{11}\, [v]\, , \cdots || \mathcal{Q}_{t-1} \,|\, \mathcal{Q}_{t-2} \,|\, \cdots} \tag{3}$$

where $||$ and | are the corresponding delimiters.

Next, we add various types of relations as triplets, linking between tokens in the serialized input, which naturally turns the input sequence into a graph (Figure 1). We'll elaborate on this in Subsection 4.2. Moreover, since for almost all relation triplets, its head and tail correspond to either a word or a phrase, while the T5 model is at subword level, we also introduce relation propagation to map these relations to subword level, which is detailed in Subsection 4.3.

To fine-tune this model, we inherit all the parameters from T5 and randomly initialize the extra relation embeddings introduced by relation-aware self-attention. The training follows the normal supervised fine-tuning scheme.

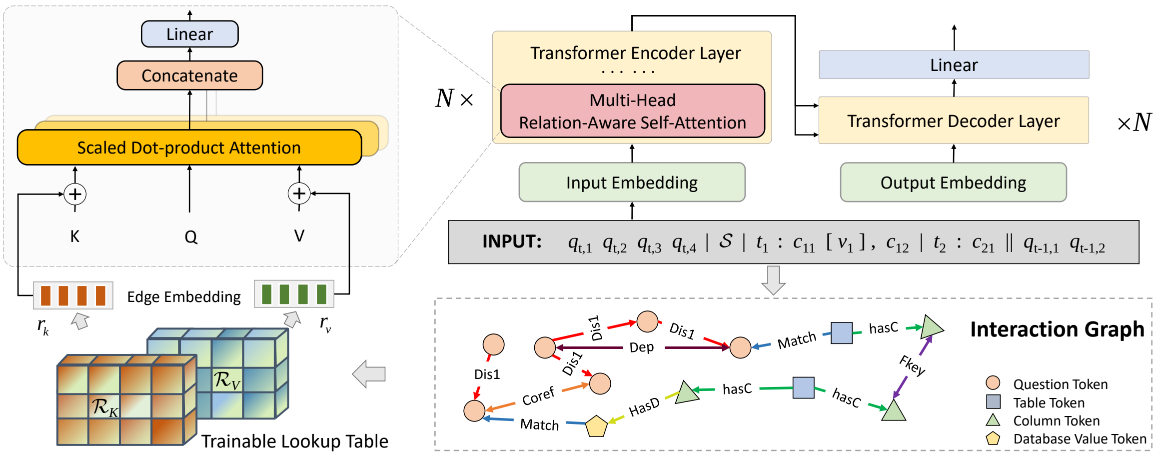

Figure 1: The overview of our model. Our model inherits the seq2seq architecture of T5, consisting of N layers of encoder and decoders. The self-attention modules in the encoder are substituted with relation-aware self-attention, introducing two additional relation embedding lookup tables $\mathcal{R}_K$ and $\mathcal{R}_V$. We convert the sequential input into an interaction graph by introducing various types of relations and adapting them to the subword level through relation propagation. During the forward process, the relation-aware self-attention modules read out the relations of each token through the interaction graph and retrieves the corresponding relations embeddings from the lookup tables $\mathcal{R}_K$ and $\mathcal{R}_V$.

Type	Head H	Tail T	Edge Label	Description
Schema Encoding	$\mathcal{T}$	$\mathcal{C}$	PRIMARY-KEY BELONGS-TO	T is the primary-key for H T is a column in H
	$\mathcal{C}$	$\mathcal{C}$	FOREIGN-KEY	H is the foreign key for T
Schema Linking	$\mathcal{Q}$	$\mathcal{T}/\mathcal{C}$	EXACT-MATCH PARTIAL-MATCH	H is part of T, and T is a span of the entire question H is part of T, but the entire question does not contain T
Question Dependency	$\mathcal{Q}$	$\mathcal{Q}$	DEPENDENCY	H has a forward syntactic dependencies on T
Question Coreference	$\mathcal{Q}$	$\mathcal{Q}$	COREFERENCE	H is the coreference of T
Database Content	$\mathcal{Q}$	$\mathcal{C}$	VALUE-MATCH	H is part of the candidate cell values of column T

Table 1: Description of some representatives for each relation type in the interaction graph. For a complete list of relations, please refer to Appendix A.

4.2 Interaction Graph

Equipped with relation-aware self-attention, we can incorporate various types of relations into the T5 model, as long as the relation can be presented as a triplet, with its head and tail being the tokens in the input sequence X. Formally, we present the triplet as

$$< H, r, T > \tag{4}$$

where H, T are the head and tail items in the triplet, and r represents the relation.

Given the input sequence X of length $|X|$, we assume that for each direction of a given pair of tokens, there only exists up to one relation. Thus, if we consider the tokens in X as vertices of a graph, it could have up to $|X|^2$ directed edges, with each edge corresponding to an entry in the adjacency matrix of the graph. In this paper, we call this graph, containing tokens from the whole input sequence as its vertices and the incorporated relations as its edges, as *interaction graph*.

For each type of introduced relation, we assign two relation embeddings for it. Thus the Transformer encoder comes with two trainable lookup tables storing relations embeddings to compute the key and value in the self-attention (c.f. Figure 1). Formally, we denote them as $\mathcal{R}_K, \mathcal{R}_V \in \mathbb{R}^{\mu\times d_{kv}}$ where μ is the kinds of relations and d_{kv} is the dimension of each attention head in the key and value states. Note that we share the same relation embedding between different heads and layers but untie them between key and value. During forward computation, for all the layers, r_{ij}^K and r_{ij}^V in Equation 1 are retrieved from the two trainable look-up tables.

We reserve a set of *generic* relations for serving as mock relations for token pairs that do not have a specific edge. In total, we have used 50 different relations in the model (c.f. Appendix A). Apart from the mock *generic* relations, there are generally 5 types of relations, which are: *schema encoding, schema linking, question dependency structure, coreference between questions,* and *database content mentions*. Please refer to Table 1 for some representative examples for each type. We'll describe each of them in the following paragraphs.

Schema Encoding. Schema encoding relations refer to the relation between schema items, i.e., $H, T \in \mathcal{S}$. These relations describe the structure information in a database schema. For example, PRIMARY-KEY indicates which column is the primary key of a table, BELONGS-TO shows which table a column belongs to, and FORIGN-KEY connects the foreign key in one table, and the primary key in another table.

Schema Linking. Schema linking relations refer to the relations between schema and question items, i.e., $H \in \mathcal{S}, T \in \mathcal{Q}$ or vice versa. We follow the settings in RAT-SQL (Wang et al., 2020a), which uses n-gram matches to indicate question mentions of the schema items. Detecting these relations is shown to be challenging in previous works (Guo et al., 2019; Deng et al., 2021), due to the common mismatch between natural language references and their actual names in the schema. Thus, we also discriminate between exact matches and partial matches to suppress the noise caused by imperfect matches.

Question Dependency Structure. This type of relation refers to the edges of a dependency tree of the question, i.e., $H, T \in \mathcal{Q}$. Different from the previous two relation types, it is less explored in the literature on text-to-SQL. Since it reflects the grammatical structure of the question, we believe it should also be beneficial for the task. In our work, to control the total number of relations and avoid unnecessary overfitting, we do not discriminate between different dependency relations. Figure 2 shows an example of dependency relations in one of its questions.

Coreference Between Questions. This type of relation is unique to the multi-turn scenario. In a dialog with multiple turns, it is important for the model to correctly figure out the referent of the pronouns. Figure 2 shows a typical case of coreference resolution. The question item "their" in Turn 1, "they" in Turn 2, and "they" in Turn 3 all refer to the question item "students" in Turn 1. i.e., $H \in \mathcal{Q}_i, T \in \mathcal{Q}_j$. We found that despite its importance of it, there are no works utilizing this relation in the text-to-SQL literature. Although pre-trained models like T5 are believed to have the capability to handle this implicitly, we still find that explicitly adding these links significantly improves the model's performance.

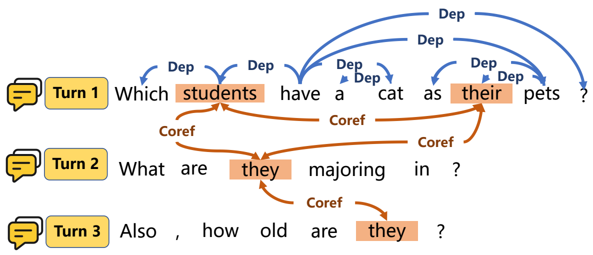

Figure 2: An example to show the coreference and syntactic dependency relations on user questions between different turns.

Database Content Mentions. Instead of mentioning the table or column names, the user could mention the values in a specific column. In this case, the informative mention could escape from the aforementioned schema linking. In this work, we follow the same procedures in BRIDGE (Lin et al., 2020b) to capture database content mentions. It first performs a fuzzy string match between the question tokens and the values of each column in the database. i.e., $H \in \mathcal{Q}, T \in \mathcal{V}$. Then the matched values are inserted after the corresponding column name in the input sequence. This relation is denoted as VALUE-MATCH in Table 1 and is also widely used in many graph-structured models (Wang et al., 2020a; Cao et al., 2021).

4.3 Relation Propagation

The various aforementioned types of relations are between types of items, with their H and T being either words or phrases. However, almost all pre-trained models take input tokens at the subword level, resulting in a difference in the granularity between the relation and the input tokens. Previous works use an extra step to aggregate multiple subword tokens to obtain a single embedding for each item in the interaction graph, such as mean pooling, attentive pooling, or with BiLSTMs (Wang et al.,

	Spider	SParC	CoSQL
Train	7000	3034/9025	2164/7485
Dev	1034	422/1203	292/1008
Test	2147	842/2498	551/-

Table 2: Dataset statistics for Spider, SParC and CoSQL. For Spider, the table shows the number of question-SQL pairs in the train-dev-test splits. For SParC and CoSQL, we list both the number of interactions and questions in the form of "#interactions/#questions". The number of question-SQL pairs in the test split of CoSQL is not publicly available.

2020a; Cao et al., 2021). However, these aggregation methods are detrimental to inheriting the pre-trained knowledge in the pretrained models.

In this work, we adopt the other way around: we propagate the relations into the subword level by creating a dense connection of the same type of relations between the tokens in H and T. For example, column `amenid` is a foreign key in table `has_amenity` and the corresponding primary key is column `amenid` in table `dorm_amenity`. Such that there is a directed relation FOREIGN-KEY between the two column names. At subword level, `amenid` consists of two tokens `amen` and `id`. Accordingly, we propagate the FOREIGN-KEY relation into 4 replicas, pointing from tokens in the source `amenid` to that of the target one, forming a dense connection between subword tokens on both sides.

With relation propagating, we could conveniently adapt word or phrase level relations to our RASAT model while keeping the pretrained weights learned at the subword level intact.

5 Experiments

In this section, we'll show our model's performance on three common Text-to-SQL datasets: Spider (Yu et al., 2018b), SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). We also present a set of ablation studies to show the effect of our method on different sized models, as well as the relative contribution of different relations. Lastly, we present several case studies to show the effect of introducing relations.

5.1 Experiment Setup

Datasets and Evaluation Metrics Spider is a large-scale, multi-domain, and cross-database benchmark. SparC and CoSQL are multi-turn versions of Spider on which the dialogue state tracking is required. Table 2 shows the statistics of the three datasets. For Spider, we list the number of question-SQL pairs.

For evaluation metrics, we use Exact Match accuracy (EM) and EXecution accuracy (EX). EM measures whether the whole predicted sequence is equivalent to the ground truth SQL, while in EX, it measures if the predicted executable SQLs (with values) can produce the same result as the corresponding gold SQLs. As for SParC and CoSQL, which involves a multi-turn scenario, both EM and EX can be measured at the question level and interaction level. Thus we have four evaluation metrics for these two datasets, namely Question-level Exact Match (QEM), Interaction-level Exact Match (IEM), question-level EXecution accuracy (QEX), and interaction-level EXecution accuracy (IEX).

Implementation Our code is based on Huggingface transformers (Wolf et al., 2020). For coreference resolution, we use coreferee[3] to yield coreference links. In total, 51 types of relations are used (c.f. Appendix A for a detailed list). For dependency parsing, stanza (Qi et al., 2020) is used. We align most of the hyperparameter settings with Shaw et al. (2021) to provide a fair comparison. Results are on the development set.

5.2 Results on SParC

The results on SParC are shown in Table 3, numbers are evaluated on its development set since the test set is private. Our proposed RASAT model combined with PICARD achieves state-of-the-art results on all four evaluation metrics.

Compared with the previous state-of-the-art HIE-SQL + GraPPa (Zheng et al., 2022), RASAT + PICARD brings the QEM from 64.7% to 66.7% and IEM from 45.0% to 47.2%. In addition, our model is able to produce SQL with values, whereas many of the models listed in the table don't provide value predictions.

Among the models that are able to predict with values, the fine-tuned T5-3B model from UNIFIEDSKG (Xie et al., 2022) is currently state-of-the-art. Our proposed RASAT without PICARD brings the IEX from 46.4% to 50.0%, with 3.6% absolute improvements. Moreover, RASAT + PICARD surpasses all previous methods in EX and improves the state-of-art QEX and IEX from 67.3%

[3]https://github.com/msg-systems/coreferee

and 46.4% to 72.5% and 53.1%, with 5.2% and 6.7% absolute improvements, respectively.

Approach	QEM	IEM	QEX	IEX
RAT-SQL + SCoRe	62.2	42.5	-	-
HIE-SQL + GraPPa	64.7	45.0	-	-
RAT-SQL-TC + GAP	64.1	44.1	-	-
GAZP + BERT	48.9	29.7	47.8	-
TreeSQL v2+BERT	52.6	34.4	50.4	29.4
UNIFIEDSKG	61.5	41.9	67.3	46.4
RASAT	64.2	43.8	68.7	50.0
RASAT + PICARD	**66.7**	**47.2**	**72.5**	**53.1**

Table 3: Results on SParC development set. Models in the upper block do not predict SQL values, while the ones in the middle block do. We compare RASAT with GAZP + BERT (Zhong et al., 2020), TreeSQL v2+BERT (Wang et al., 2021b), RAT-SQL + SCoRe (Yu et al., 2020), HIE-SQL + GraPPa (Zheng et al., 2022), RAT-SQL-TC + GAP (Li et al., 2021), UNIFIEDSKG (Xie et al., 2022).

Approach	QEM	IEM	QEX	IEX
RAT-SQL + SCoRe	52.1	22.0	-	-
HIE-SQL + GraPPa	56.4	**28.7**	-	-
GAZP + BERT	42.0	12.3	38.8	-
UNIFIEDSKG	54.1	22.8	62.2	26.2
T5-3B + PICARD	56.9	24.2	-	-
RASAT	56.2	24.9	63.8	34.8
RASAT + PICARD	**58.8**	26.3	**66.7**	**37.5**

Table 4: Results on CoSQL development set. Models in the upper block do not predict SQL values, while the ones in the middle block do. We compare RASAT with GAZP + BERT (Zhong et al., 2020), RAT-SQL + SCoRe (Yu et al., 2020), HIE-SQL + GraPPa (Zheng et al., 2022), RAT-SQL-TC + GAP (Li et al., 2021), UNIFIEDSKG (Xie et al., 2022), T5-3B + PICARD (Scholak et al., 2021).

5.3 Results on CoSQL

Compared with SParC, CoSQL is labeled in a Wizard-of-Oz fashion, forming a more realistic and challenging testbed. Nevertheless, our proposed model could also achieve a competitive performance on this dataset (Table 4), yielding state-of-the-art results on three out of the four metrics.

For exact match accuracy, RASAT + PICARD surpasses all models that are able to predict values. For those who don't predict values, its performance is comparable to state-of-the-art. i.e., for HIE-SQL + GraPPa, it reaches a QEM of 56.4% and IEM of 28.7%. For our model, the performance on QEM is 58.8%, and IEM is 26.3%.

Among models that predict SQLs with values, RASAT + PICARD achieves state-of-the-art performance on execution accuracy. Especially for IEX, our model surpasses the previous state-of-art from 26.2% to 37.5%, an 11.3% absolute improvement. Note that even when PICARD is not used, the RASAT model alone also beats all the previous approaches, bringing IEX from 26.2% to 34.8%.

Approach	EM	EX
RAT-SQL + BERT	69.7	-
LGESQL + ELECTRA	75.1	-
S^2SQL + ELECTRA	**76.4**	-
SmBoP + GraPPa	69.5	71.1
BRIDGE v2 + BERT	71.1	68.3
RATSQL + GAP+NatSQL	73.7	75.0
T5-3B	71.5	74.4
T5-3B + PICARD	75.5	79.3
RASAT	72.6	76.6
RASAT + PICARD	75.3	**80.5**

Table 5: Results on Spider development set. Models in the upper block do not predict SQL values, while the ones in the middle block do. We compare RASAT with some important baseline methods, such as RAT-SQL (Wang et al., 2020a), Bridge (Lin et al., 2020b), GAZP (Zhong et al., 2020), NatSQL (Gan et al., 2021), SmBoP (Rubin and Berant, 2021), LGESQL (Cao et al., 2021), S^2SQL (Hui et al., 2022), T5 (Shaw et al., 2021) and PICARD (Scholak et al., 2021).

5.4 Results on Spider

The results on the Spider is provided in Table 5. Our proposed RASAT model achieves state-of-the-art performance in EX and competitive results in EM. Compared with T5-3B, which also doesn't use the PICARD during beam search, our model's EX increases from 74.4% to 76.6%, a 2.2% absolute improvement. When augmented with PICARD, RASAT+PICARD brings the EX even higher to 80.5%, a 1.2% absolute improvement compared to T5-3B + PICARD.

5.5 Ablation Study

In this subsection, we conduct a set of ablation studies to examine various aspects of the proposed model. Due to the limited availability of the test sets, all numbers in this subsection are reported on

Approach	easy	medium	hard	extra
T5-3B + PICARD	95.2	85.4	67.2	50.6
RASAT + PICARD	**96.0**	**86.5**	**67.8**	**53.6**

Table 6: EX accuracy of RASAT+PICARD and T5-3B+PICARD on the examples with different level of difficulty.

the development set.

Effect on SQL difficulty. One might conjecture that the introduced relations are only effective for more difficult, longer SQL query predictions, while for predicting short SQL queries, the original T5 model could handle equally well. Thus, we evaluate our model according to the difficulty of the examples, where the question/SQL pairs in the development set are categorized into 4 subsets, i.e., easy, medium, hard, and extra hard, according to their level of difficulty. In Table 6 we provide a comparison between T5-3B + PICARD (Scholak et al., 2021) and RASAT + PICARD on the EX metric on the 4 subsets. RASAT + PICARD surpasses T5-3B + PICARD across all subsets, validating the effectiveness of the introduced relational structures for all SQL sequences.

Model Size Impact. To test the effectiveness of the introduced relational structures on pretrained models with different sizes, we implant RASAT into 4 T5 models of different sizes (T5-small, T5-base, T5-large, T5-3B) and test it on Spider (Table 7). Interestingly, for smaller pretrained models, our RASAT model could bring even larger performance gaps between its T5-3B counterpart. This suggests that the larger T5 model might have learned some of the relational structures implicitly. We believe this is consistent with the findings on other fine-tuning tasks, where larger pretrained models are more capable of capturing the abundant implicit dependencies in the raw text.

Relation Types. We conducted additional experiments to analyze the relative contribution of different relation types. The experimental results on Spider is shown in Table 8 while result on SParC is shown in Table 9 (since CoSQL has similar conversational modality with SParC, the experiments are only conducted on SParC). We find that both T5 and RASAT models can benefit from leveraging additional database content. Another important finding is that the performance has increased obviously by adding dependency relationship to RASAT(-small) on Spider. As for SParC, the database content plays a more important role by looking at EX results; from what we can see, IEX will decrease by 2.6% after removing database content from the input.

Approach	EM	EX
T5-small	47.2	47.8
RASAT(-small)	**53.0(+5.8)**	**53.7(+5.9)**
T5-base	57.2	57.9
RASAT(-base)	**60.4(+3.2)**	**61.3(+3.4)**
T5-large	65.3	67.2
RASAT(-large)	**66.7(+1.4)**	**68.5(+1.3)**
T5-3B	71.5	74.4
RASAT(-3B)	**72.6(+1.1)**	**76.6(+2.2)**

Table 7: Result for different T5 model sizes on Spider development set. The performance of T5 baselines are from Scholak et al. (2021).

Approach	EM	EX
T5-small	47.2	47.8
w/o db_content	45.8(-1.4)	46.9(-0.9)
RASAT(-small)	53.0	53.7
w/o db_content	52.6(-0.4)	52.9(-0.8)
w/o dependency	51.3(-1.7)	51.7(-2.0)

Table 8: Ablation study on the relative contribution of different relation types. Experiment are conducted using RASAT(-small) on the Spider dataset.

5.6 Case Study

In Table 10, we demonstrate how the introduced relation could help the model predict SQL structures more accurately by demonstrating 2 examples of question-SQL pairs sampled from the SParC development set. We compare the predictions from T5-3B and our model, and both the two examples have three turns in the interaction. For the first case, the

Approach	QEM	IEM	QEX	IEX
RASAT(-3B)	63.6	43.6	**68.7**	**49.5**
w/o dep	**64.2(+0.6)**	**43.8(+0.2)**	68.2(-0.5)	48.3(-1.2)
w/o crf	63.2(-0.4)	**43.8(+0.2)**	**68.7**	**50.0(+0.5)**
w/o db	62.2(-1.4)	42.7(-0.9)	67.3(-1.4)	46.9(-2.6)

Table 9: Ablation study on the relative contribution of different relation types. Experiment are conducted using RASAT(-3B) on the SParC dataset. “dep” is short for dependency relation, and “crf” for coreference relation. “db” means database content.

Description	A database about employee hiring and evaluation.
Goal	**Find cities which more than one employee under age 30 come from.**
Question #1	Find all employees who are under age 30.
T5-3B	SELECT * FROM employee WHERE age <30
RASAT	SELECT * FROM employee WHERE age <30
Question #2	Which cities did they come from?
T5-3B	SELECT city FROM employee WHERE age <30
RASAT	SELECT city FROM employee WHERE age <30
Question #3	Show the cities from which more than one employee originated.
T5-3B	SELECT city FROM employee GROUP BY city HAVING COUNT(*) >1
RASAT	SELECT city FROM employee WHERE age <30 GROUP BY city HAVING COUNT(*) >1
Description	A database about courses and teachers.
Goal	**Show names of teachers and the courses they are arranged to teach in ascending alphabetical order of the teacher's name.**
Question #1	Find all the course arrangements.
T5-3B	SELECT * FROM course_arrange
RASAT	SELECT * FROM course_arrange
Question #2	Show names of teachers and the courses they are arranged to teach.
T5-3B	SELECT T2.name, T1.course FROM course_arrange AS T1 JOIN teacher AS T2 ON T1.teacher_id = T2.teacher_id
RASAT	SELECT T2.name, T3.course FROM course_arrange AS T1 JOIN teacher AS T2 ON T1.teacher_id = T2.teacher_id JOIN course AS T3 ON T1.course_id = T3.course_id
Question #3	Sort the results by teacher's name
T5-3B	SELECT T2.name, T1.course FROM course_arrange AS T1 JOIN teacher AS T2 ON T1.teacher_id = T2.teacher_id ORDER BY T2.name
RASAT	SELECT T3.name, T2.course FROM course_arrange AS T1 JOIN course AS T2 ON T1.course_id = T2.course_id JOIN teacher AS T3 ON T1.teacher_id = T3.teacher_id ORDER BY T3.name

Table 10: Some examples in the SParC development set. RASAT gives all correct predictions in these cases while the original T5-3B model fails.

vanilla T5-3B model neglects the condition "employees who are under age 30" when answering Question #3, while RASAT-SQL predicts it correctly by exploiting the relations inside the contexts. For the second case, the database schema is more complex, and the table `course_arrange` has no such a column called `course`. If one would like to access column `course`, the foreign key must be used. RASAT gives the correct SQL since these types of relational structures are explicitly embedded in the RASAT model, while the vanilla T5-3B fails to do it.

6 Conclusion

In this work, we proposed RASAT, a Relation-Aware Self-Attention-augmented T5 model for text-to-SQL generation. Compared with previous work, RASAT is able to introduce various types of structural relations into the sequential T5 model. Different from the more common approach of fine-tuning the original model or using prompt tuning, we propose to augment the self-attention modules in the encoder and introduce new parameters to the model while still being able to leverage the pre-trained weights. RASAT is able to achieve state-of-art performances, especially on execution accuracy, in the three most common text-to-SQL benchmarks.

References

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019a. 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 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Representing schema structure with graph neural networks for text-to-sql parsing. *arXiv preprint arXiv:1905.06241*.

Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. Sadga: Structure-aware dual graph aggregation network for text-to-sql. In *Advances in Neural Information Processing Systems*, volume 34, pages 7664–7676. Curran Associates, Inc.

Yitao Cai and Xiaojun Wan. 2020. IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6903–6912, Online. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 2541–2555, Online. Association for Computational Linguistics.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1337–1350, Online. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

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

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers. *arXiv preprint arXiv:2203.06958*.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978*.

Yuntao Li, Hanchu Zhang, Yutian Li, Sirui Wang, Wei Wu, and Yan Zhang. 2021. Pay more attention to history: A context modeling strategy for conversational text-to-sql.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020a. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020b. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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*, pages 311–324, Online. Association for Computational Linguistics.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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*, pages 7567–7578, Online. Association for Computational Linguistics.

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*, pages 3229–3238, Online. Association for Computational Linguistics.

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

Xiaxia Wang, Sai Wu, Lidan Shou, and Ke Chen. 2021b. An interactive nl2sql approach with reuse strategy. In *International Conference on Database Systems for Advanced Applications*, pages 280–288. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

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 (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

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

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2020. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. 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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

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

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

Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. Hie-sql: History information enhanced network for context-dependent text-to-sql semantic parsing. *arXiv preprint arXiv:2203.07376*.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6869–6882, Online. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Relations Used in Experiment

Table 11 shows all relations used in our experiment while most of these are consistant with RAT-SQL (Wang et al., 2020a) and (Cao et al., 2021). There are total 51 kinds relation used.

Head H	Tail T	Edge label	Description
$\mathcal{Q}$	$\mathcal{Q}$	Question-Question-Dist*	Question item H is at a distance of * before question item T in the input question
		Question-Question-Identity	Question item H is question item T itself
		Question-Question-Generic	Question item H and question item T has no pre-defined relation
		Forward-Syntax Backward-Syntax None-Syntax	Question item H has a forward/reverse/no syntactic dependencies on question item T
		Co_Relations	Question item H and question item T are considered as a whole in coreference relation
		Coref_Relations	Question item H is the coreference of question item T
$\mathcal{Q}$	$\mathcal{S}$	Question-*-Generic	Question item H and database item T has no pre-defined relation
$\mathcal{Q}$	$\mathcal{T}$	Question-Table-Exactmatch Question-Table-Partialmatch Question-Table-Nomatch	Question item H is spelled exactly/partially/not the same as table item T
$\mathcal{Q}$	C	Question-Column-Exactmatch Question-Column-Partialmatch Question-Column-Nomatch	Question item H is spelled exactly/partially/not the same as column item T
		Question-Column-Valuematch	Question item H is spelled exactly the same as a value in column item T
$\mathcal{S}$	$\mathcal{Q}$	*-Question-Generic	Database item H and question item T has no pre-defined relation
$\mathcal{S}$	$\mathcal{S}$	*-*-Identity	Database item H is database item T itself
$\mathcal{S}$	$\mathcal{T}$	*-Table-Generic	Database item H and table item T has no pre-defined relation
$\mathcal{S}$	$\mathcal{C}$	*-Column-Generic	Database item H and column item T has no pre-defined relation
$\mathcal{T}$	$\mathcal{Q}$	Table-Question-Exactmatch Table-Question-Partialmatch Table-Question-Nomatch	Table item H is spelled exactly/partially/not the same as question item T
$\mathcal{T}$	$\mathcal{S}$	Table-*-Generic	Table item H and database item T has no pre-defined relation
$\mathcal{T}$	$\mathcal{T}$	Table-Table-Generic	Table item H and table item T has no pre-defined relation
		Table-Table-Identity	Table item H is table item T itself
		Table-Table-Fk	At least one column in table item H is a foreign key for certain column in table item T
		Table-Table-Fkr	At least one column in table item T is a foreign key for certain column in table item H
		Table-Table-Fkb	Table item H and T satisfy both "Table-Table-Fk" and "Table-Table-Fkr" relations
$\mathcal{T}$	$\mathcal{C}$	Table-Column-Pk	Column item T is the primary key for table item H
		Table-Column-Has	Column item T belongs to table item H
		Table-Column-Generic	Table item H and column item T has no pre-defined relation
$\mathcal{C}$	$\mathcal{Q}$	Column-Question-Exactmatch Column-Question-Partialmatch Column-Question-Nomatch	Column item H is spelled exactly/partially/not the same as table item T
		Column-Question-Valuematch	Column item H is spelled exactly the same as a value in question item T
$\mathcal{C}$	$\mathcal{S}$	Column-*-Generic	Column item H and database item T has no pre-defined relation
$\mathcal{C}$	$\mathcal{T}$	Column-Table-Pk	Column item H is the primary key for table item T
		Column-Table-Has	Column item H belongs to table item T
		Column-Table-Generic	Column item H and table item T has no pre-defined relation
$\mathcal{C}$	$\mathcal{C}$	Column-Column-Identity	Column item H is column item T itself
		Column-Column-Sametable	Column item H and column item T are in the same table
		Column-Column-Fk Column-Column-Fkr	Column item H has a forward/reverse foreign key constraint relation with Column item T
		Column-Column-Generic	Column item H and column item T has no pre-defined relation
$\mathcal{C}$	$\mathcal{V}$	Has-Dbcontent	Db content item T belongs to column item H
$\mathcal{V}$	$\mathcal{C}$	Has-Dbcontent-R	Db content item H belongs to column item T
		No-Relation	Item H and item T has no relation (Used when item H or item T is a delimiter)

Table 11: All relations used in our experiment. $\mathcal{V}$ is the matched question item that extracted from $\mathcal{Q}$.