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

**Jiexing Qi[1], Jingyao Tang[1], Ziwei He[1], Xiangpeng Wan[2], Yu Cheng[3]**
**Chenghu Zhou[4], Xinbing Wang[1], Quanshi Zhang[1], Zhouhan Lin[1]***
[1]Shanghai Jiao Tong University, Shanghai, China
[2]NetMind.AI and ProtagoLabs, Virginia, USA
[3]Microsoft Research, Redmond, Washington, USA
[4]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 using 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 inheriting the pretrained parameters from the T5 model effectively. Our model can incorporate almost all types of existing relations in the literature, and in addition, we propose introducing 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 state-of-the-art results across all three benchmarks (75.5% EX on Spider, 52.6% IEX on SParC, and 37.4% IEX on CoSQL). [1]

## 1 Introduction

Text-to-SQL is the task that 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 utilize database content effectively. Their BERT-based encoder is followed by an LSTM-based pointer network as the decoder, which 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 can 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 dev 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.,

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



*Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3215–3229
December 7-11, 2022 

2020a).

In another thread, pretrained seq2seq models just 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 can 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 state-of-the-art performance from 21.6% to 52.6%, 31% absolute improvements. On CoSQL, we improve state-of-the-art IEX performance from 8.4% to 37.4%, achieving 29% absolute improvements. Moreover, on Spider, we improve state-of-the-art execution accuracy from 75.1% to 75.5%, achieving 0.4% absolute improvements.

## 2 Related Work

Early works usually exploit a sketch-based slot-filling method that uses different modules to predict the corresponding part of SQL. These methods decompose the SQL generation task into several independent sketches and use different classifiers to predict corresponding part, such as SQLNet (Xu et al., 2017), 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 various 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) introduces schema encoding and linking and assigns every two input items a relation, LGESQL (Cao et al., 2021) further distinguishes local and non-local relations by exploiting a line graph enhanced hidden module, SADGA (Cai et al., 2021) uses contextual structure and dependency structure to encode question-graph while database schema relations are used in schema graph, $S^2$SQL (Hui et al., 2022) adds 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) edits the last turn's predicted SQL to generate the newly predicted SQL at the token level. IGSQL (Cai and Wan, 2020) uses 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 assigns a probability to reuse subtree of historical Tree-SQLs. IST-SQL (Wang et al., 2021a) proposes an interaction state tracking method to predict the SQL query. RAT-SQL-TC (Li et al., 2021)adds two auxiliary training tasks to explicitly model the semantic changes in both turn grain and conversation grain. $R^2$SQL (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 are available, i.e., $\{\mathcal{Q}_1, \cdots, \mathcal{Q}_i\}$.

### 3.2 Relation-aware Self-Attention

Relation-aware self-attention (Shaw et al., 2018) 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):

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

where $H$ is the number of heads, and $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}$ are learnable weights. The $\boldsymbol{r}_{ij}^K, \boldsymbol{r}_{ij}^V$ 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 $\mathcal{S} =< \mathcal{T}, \mathcal{C} >$ with the database name $S$, 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,

$$X = \overline{\mathcal{Q} \mid S \mid t_1 : c_{11} \left[ v \right], \cdots, c_{1|T_1|} \mid t_2 : c_{21}, \cdots} \quad (2)$$

where $t_i$ 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 $\mathcal{Q}$, $S$, 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 add the questions in the history at the start of the sequence and truncate the trailing tokens in the front of the sequence when the sequence length reaches 512. i.e.,

$$X = \overline{\mathcal{Q}_1 \mid \mathcal{Q}_2 \mid \cdots \mid \mathcal{Q}_t \mid S \mid t_1 : c_{11} \left[ v \right], \cdots} \quad (3)$$

where | 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 will elaborate on this in Subsection 4.2. Moreover, since 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 overall increase of parameters is less than 0.01% (c.f. Appendix A).

### 4.2 Interaction Graph

Equipped with relation-aware self-attention, we can incorporate various types of relations into the

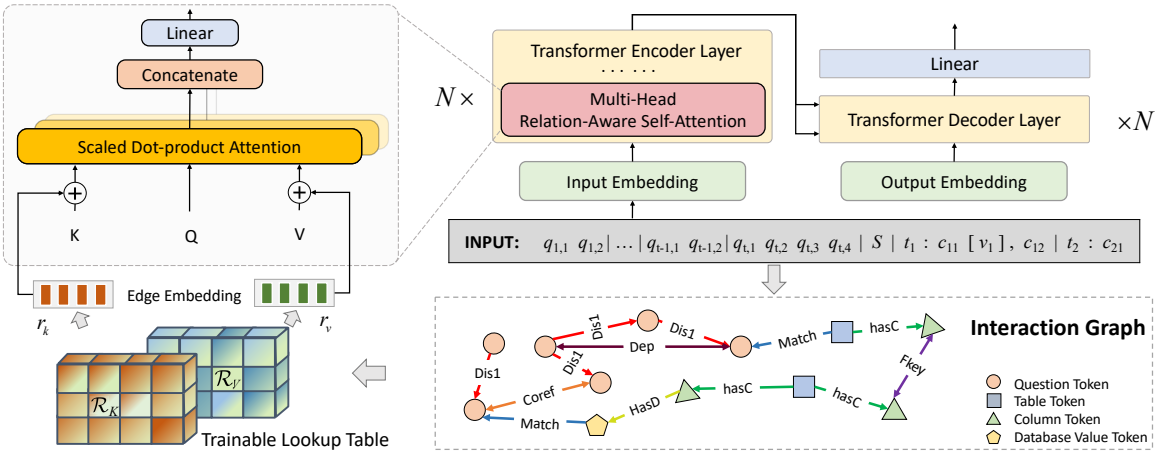


Figure 1: The overview of our model. Our model inherits the seq2seq architecture of T5, consisting of $N$ layers of encoders and decoders. The self-attention modules in the encoder are substituted with relation-aware self-attention, introducing two additional relation embedding lookup tables $\mathcal{R}_K$ 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 retrieve 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<br>BELONGS-TO | T is the primary-key for H<br>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<br>PARTIAL-MATCH | H is part of T, and T is a span of the entire question<br>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 D.

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*.

We assign two relation embeddings for each type of introduced relation. 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 $\mu$ 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 relation embedding between different heads and layers but untie them between key and value. During forward computation, for all the layers, $r^K_{ij}$ and $r^V_{ij}$ 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 51 different relations in the model (c.f. Appendix D). 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 will 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., $H, T \in \mathcal{Q}$. Unlike 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 figure out the referent of the pronouns correctly. 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., $H \in \mathcal{Q}_i, T \in \mathcal{Q}_j$. To our best knowledge, there are no works utilizing this relation in the text-to-SQL literature despite the importance of this relation. Although pre-trained models like T5 are believed to have the capability to handle this implicitly, we still find that explicitly adding these links could significantly improve the model's performance.

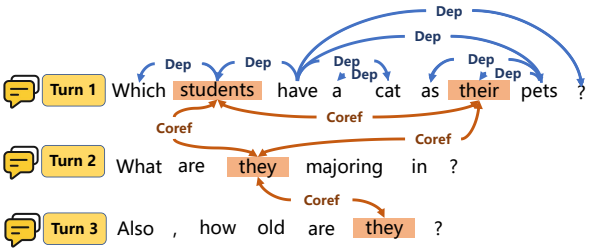


Figure 2: An example to show the coreference and syntactic dependency relations on user questions between different turns.

**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 relations 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., 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: we propagate the relations into the subword level by cre-

| Approach | Dev | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|
| | QEM | IEM | QEX | IEX | QEM | IEM | QEX | IEX |
| RAT-SQL + SCoRe (Yu et al., 2020) | 62.2 | 42.5 | - | - | 62.4 | 38.1 | - | - |
| HIE-SQL + GraPPa (Zheng et al., 2022) | 64.7 | 45.0 | - | - | 64.6 | 42.9 | - | - |
| RAT-SQL-TC + GAP (Li et al., 2021) | 64.1 | 44.1 | - | - | 65.7 | 43.2 | - | - |
| GAZP+BERT (Zhong et al., 2020) | 48.9 | 29.7 | 47.8 | - | 45.9 | 23.5 | 44.6 | 19.7 |
| TreeSQL v2+BERT (Wang et al., 2021b) | 52.6 | 34.4 | 50.4 | 29.4 | 48.1 | 25.0 | 48.5 | 21.6 |
| UNIFIEDSKG (Xie et al., 2022) | 61.5 | 41.9 | 67.3 | 46.4 | - | - | - | - |
| RASAT | 65.0 | 45.7 | 69.9 | 50.7 | - | - | - | - |
| RASAT+PICARD | **67.7** | **49.1** | **73.3** | **54.0** | **67.7** | **45.2** | **74.0** | **52.6** |

Table 2: Results on SParC dataset. Models in the upper block do not predict SQL values, while the ones in the middle block do.

| | *Spider* | *Realistic* | *SParC* | *CoSQL* |
|---|---|---|---|---|
| Train | 7000 | - | 3034/9025 | 2164/7485 |
| Dev | 1034 | 508 | 422/1203 | 292/1008 |
| Test | 2147 | - | 842/2498 | 551/2546 |

Table 3: Dataset statistics for Spider, Spider-Realistic (*Realistic* in table), SParC and CoSQL. For Spider and Spider-Realistic, 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".

ating 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 will show our model's performance on three common text-to-SQL datasets: Spider (Yu et al., 2018), SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). Besides, we experiment on a more realistic setting of the Spider dataset: Spider-Realistic (Deng et al., 2021) to test the generalizability of our model. The statistics of these datasets are shown in Table 3. 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. In addition, we put 2 case studies in Appendix C.

### 5.1 Experiment Setup

**Datasets** 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. All test data is hidden to ensure fairness, and we submit our model to the organizer of the challenge for evaluation.

**Evaluation Metrics** We use the official evaluation metrics: Exact Match accuracy (EM) and EXecution accuracy (EX). EM measures whether the whole predicted sequence is equivalent to the ground truth SQL (without values), 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 involve a multi-turn scenario, both EM and EX can be measured at the question and interaction levels. 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). We align most of the hyperparameter settings with Shaw et al. (2021) to provide a fair comparison. For coreference resolution, we use coreferee[3] to yield

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

| Approach | Dev | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|
| | QEM | IEM | QEX | IEX | QEM | IEM | QEX | IEX |
| RAT-SQL + SCoRe (Yu et al., 2020) | 52.1 | 22.0 | - | - | 51.6 | 21.2 | - | - |
| HIE-SQL + GraPPa (Zheng et al., 2022) | 56.4 | **28.7** | - | - | 53.9 | 24.6 | - | - |
| GAZP+BERT (Zhong et al., 2020) | 42.0 | 12.3 | 38.8 | - | 39.7 | 12.8 | 35.9 | 8.4 |
| UNIFIEDSKG (Xie et al., 2022) | 54.1 | 22.8 | 62.2 | 26.2 | - | - | - | - |
| T5-3B (Scholak et al., 2021) | 53.8 | 21.8 | - | - | 51.4 | 21.7 | - | - |
| T5-3B+PICARD (Scholak et al., 2021) | 56.9 | 24.2 | - | - | 54.6 | 23.7 | - | - |
| RASAT | 56.2 | 25.9 | 63.8 | 34.8 | - | - | - | - |
| RASAT+PICARD | **58.8** | 27.0 | **67.0** | **39.6** | **55.7** | **26.5** | **66.3** | **37.4** |

Table 4: Results on CoSQL dataset. Models in the upper block do not predict SQL values, while the ones in the middle block do.

| Approach | Dev | | Test | |
|---|---|---|---|---|
| | EM | EX | EM | EX |
| RAT-SQL+BERT | 69.7 | - | 65.6 | - |
| LGESQL+ELECTRA | 75.1 | - | 72.0 | - |
| S²SQL + ELECTRA | **76.4** | - | **72.1** | - |
| BRIDGE v2+BERT | 71.1 | 70.3 | 67.5 | 68.3 |
| NatSQL+GAP | 73.7 | 75.0 | 68.7 | 73.3 |
| SmBoP + GraPPa | 74.7 | 75.0 | 69.5 | 71.1 |
| T5-3B | 71.5 | 74.4 | 68.0 | 70.1 |
| T5-3B + PICARD | 75.5 | 79.3 | 71.9 | 75.1 |
| RASAT | 72.6 | 76.6 | - | - |
| RASAT+PICARD | 75.3 | **80.5** | 70.9 | **75.5** |

Table 5: Results on Spider dataset. 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), $S^2$SQL (Hui et al., 2022), T5 and PICARD (Scholak et al., 2021).

| Approach | EM | EX |
|---|---|---|
| RAT-SQL+STRUG (Deng et al., 2021) | 62.2 | 65.7 |
| T5-3B (Scholak et al., 2021) | 62.0 | 64.1 |
| T5-3B+PICARD (Scholak et al., 2021) | 68.7 | 71.4 |
| RASAT | 65.2 | 65.8 |
| RASAT+PICARD | **69.7** | **71.9** |

Table 6: Results on Spider-Realistic dataset. We reproduce Scholak et al. (2021) 's method to get the performance of T5-3B (+PICARD) and the performance of RAT-SQL+STRUG are from Deng et al. (2021) reported.

coreference links. In total, 51 types of relations are used (c.f. Appendix D for a detailed list). For dependency parsing, stanza (Qi et al., 2020) is used. The batch size we used is 2048. We use Adafactor (Shazeer and Stern, 2018) as optimizer and the learning rate is 1e-4. We set "parse with guards" mode for PICARD and beam size is set to 8. The max tokens to check for PICARD is 2. Experiments are run on NVIDIA A100-SXM4-80GB GPUs.

### 5.2 Results on SParC

The results on SParC are shown in Table 2. 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 RAT-SQL-TC + GAP (Li et al., 2021), RASAT + PICARD brings the QEM from 65.7% to 67.7% and IEM from 43.2% to 45.2% on the test set. In addition, our model can produce executable SQLs (with values), whereas many of the models listed in the table do not provide value predictions.

Among the models that can predict with values, the fine-tuned T5-3B model from UNIFIEDSKG (Xie et al., 2022) is currently the state-of-the-art. We did comparison of QEX/IEX on the dev set since they did not report their performance on the test set. RASAT + PICARD surpasses all previous methods and improves the state-of-art QEX and IEX from 67.3% and 46.4% to 73.3% and 54.0%, with 6% and 7.6% absolute improvements, respectively.

Furthermore, on the official leaderboard of SParc which reports over test set, our proposed RASAT + PICARD brings the IEX from 21.6% to 52.6%, achieving 31% absolute improvements.

### 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 still achieve state-of-the-art results (Table 4) on all four evaluation metrics.

By comparing to the previous state-of-the-art HIE-SQL + GraPPa (Zheng et al., 2022) and T5-3B+PICARD (Scholak et al., 2021), RASAT + PICARD brings the QEM from 54.6% to 55.7% and IEM from 24.6% to 26.5% on the test set.

For the same reason as on SParC, we mainly compare QEX/IEX performance on the dev set, and RASAT + PICARD surpasses all models that can predict executable SQLs (with values). Especially for IEX, our model surpasses the previous state-of-the-art from 26.2% to 39.6%, with 13.4% absolute improvement. Moreover, on the official leaderboard of CoSQL which reports over test set, RASAT + PICARD brings the IEX from 8.4% to 37.4%, with 29% absolute improvements.

### 5.4 Results on Spider and Spider-Realistic

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. On the dev set, compared with T5-3B, which also does not use the PICARD during beam search, our model's EX increases from 74.4% to 76.6%, achieving 2.2% absolute improvement. When augmented with PICARD, RASAT+PICARD brings the EX even higher to 80.5%, with 1.2% absolute improvement compared to T5-3B + PICARD. Furthermore, on the official leaderboard of Spider, our proposed RASAT + PICARD brings the EX from 75.1% to 75.5%, achieving new state-of-the-art.

Furthermore, we also evaluate our model on a more challenging Spider variant, Spider-Realistic (Deng et al., 2021). It is a evaluation dataset that has modified the user questions by removing or paraphrasing explicit mentions of column names to present a realistic and challenging setting. Our model also achieves a new state-of-the-art performance (Table 6), which suggests strong ability of our model to generalize to unseen data.

### 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 the dev set.

| 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 7: EX accuracy of RASAT+PICARD and T5-3B+PICARD on the examples of Spider dev set with different levels of difficulty.

| 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)** | **69.2(+2.0)** |
| T5-3B | 71.5 | 74.4 |
| RASAT(-3B) | **72.6(+1.1)** | **76.6(+2.2)** |

Table 8: Result for different T5 model sizes on Spider dev set. The performance of T5 baselines are from Scholak et al. (2021).

**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 dev set are categorized into four subsets, i.e., easy, medium, hard, and extra hard, according to their level of difficulty. In Table 7 we provide a comparison between T5-3B + PICARD (Scholak et al., 2021) and RASAT + PICARD on the EX metric on the four subsets. RASAT + PICARD surpasses T5-3B + PICARD across all subsets, validating the effectiveness of the introduced relational structures for all SQL sequences.

| 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 9: Ablation study on the relative contribution of different relation types. Experiment are conducted using RASAT(-small) on the Spider dataset.

| Approach | QEM | IEM | QEX | IEX |
|---|---|---|---|---|
| RASAT | 64.5 | **45.7** | 69.2 | 50.4 |
| w/o Dp | **65.0(+0.5)** | 45.5(-0.2) | **69.9(+0.7)** | **50.7(+0.3)** |
| w/o Cf | **65.0(+0.5)** | 45.0(-0.7) | 69.4(+0.2) | 50.0(-0.4) |
| w/o Db | 64.1(-0.4) | 45.3(-0.4) | 67.9(-1.3) | 48.5(-1.9) |
| w/o SL | 64.5 | 45.5(-0.2) | 68.8(-0.4) | 49.4(-1.0) |
| w/o SE | 63.9(-0.6) | 44.6(-1.1) | 68.6(-0.6) | 48.9(-1.5) |

Table 10: Ablation study on the relative contribution of different relation types. Experiment are conducted using RASAT(-3B) on the SParC dataset. "Dp" is short for dependency relation, "Cf" for coreference relation, "SL" for schema linking relation, "SE" for schema encoding relation and "Db" means database content.

**Model Size Impact.** To test the effectiveness of the introduced relational structures on pretrained models with different sizes, we implant RASAT into four T5 models of different sizes (T5-small, T5-base, T5-large, T5-3B) and test it on Spider (Table 8). 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 9 while result on SParC is shown in Table 10 (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 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 1.9% after removing database content from the input.

## 6 Conclusion

In this work, we propose RASAT, a Relation-Aware Self-Attention-augmented T5 model for the text-to-SQL generation. Compared with previous work, RASAT can introduce various 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 had achieved state-of-the-art performances, especially on execution accuracy, in the three most common text-to-SQL benchmarks.

## Limitation

Our method consumes plenty of computational resources since we leverage the large T5-3B model. We train our models on 8 A100 GPUs (80G) for around 2 days. Our model truncates the source sequences to 512, this may lead to information loss when a sample has long input. We find that about 3% of training data in CoSQL will be affected. We only work with English since it has richer analytical tools and resources than other language.

## Acknowledgement

This work was sponsored by the National Natural Science Foundation of China (NSFC) grant (No. 62106143), and Shanghai Pujiang Program (No. 21PJ1405700). We would like to thank Tao Yu, Hongjin Su, and Yusen Zhang for running evaluations on our submitted models.

## 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. $S^2$SQL: 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.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

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, 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. 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*, 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 Model Size

Compared with the original T5 model, only two embedding matrices are added to the encoder in our model, with $2 \times \mu \times d_{kv}$ parameters. These embedding matrices are shared in each encoder layer and each head. Here $\mu = 51$ is the total number of relations and $d_{kv}$ is the dimension of the key/value states in self-attention (64 in T5-small/base/large and 128 in T5-3B). The overall increase of parameters is less than 0.01%.

| Approach | #param |
|---|---|
| T5-small | 60,506,624 |
| RASAT(-small) | **60,512,768(+0.0107%)** |
| T5-base | 222,903,552 |
| RASAT(-base) | **222,909,696(+0.0029%)** |
| T5-large | 737,668,096 |
| RASAT(-large) | **737,674,240(+0.0009%)** |
| T5-3B | 2,851,598,336 |
| RASAT(-3B) | **2,851,610,624(+0.0005%)** |

Table 11: The number of paramerter comparation between RASAT and the same size T5 model.

## B Output Comparation between T5 and Tree-based Decoder Model

Here we show the output difference between T5 and most AST-based models. As it shown in Table 12, most of these models which exploited AST-tree-based decoder (such as RAT-SQL (Wang et al., 2020a), LGESQL (Cao et al., 2021)) usually use a place holder (i.e. "value") to represent the real value("France" in this example). These outputs can not be executed in a real database and they fail to evaluate in the EXecution Accuracy(EX/QEX/IEX) metric.

## C Case Study

In Table 13, 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 dev 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 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.

## D Relations Used in Experiment

Table 14 shows all relations used in our experiment while most of these are consistant with RAT-SQL (Wang et al., 2020a) and LGESQL (Cao et al., 2021). There are total 51 kinds relation used.

| Question | What is the average, minimum, and maximum age of all singers from France? |
|---|---|
| Tree-based model | **SELECT AVG**(singer.Age) , **MAX**(singer.Age) , **MIN**(singer.Age) **FROM** singer **WHERE** singer.Country = "*value*" |
| RASAT | **SELECT AVG**(singer.Age) , **MAX**(singer.Age) , **MIN**(singer.Age) **FROM** singer **WHERE** singer.Country = "*France*" |

Table 12: An example to show the difference between AST-based decoder model's output and T5's output.

| | |
|---|---|
| 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 13: Some examples in the SParC dev set. RASAT gives all correct predictions in these cases while the original T5-3B model fails.

| Head H | Tail T | Edge label | Description |
|---|---|---|---|
| $\mathcal{Q}$ | $\mathcal{Q}$ | Question-Question-Dist*<br>Question-Question-Identity<br>Question-Question-Generic | Question item H is at a distance of * before question item T in the input question<br>Question item H is question item T itself<br>Question item H and question item T has no pre-defined relation |
| | | Forward-Syntax<br>Backward-Syntax<br>None-Syntax | Question item H has a forward/reverse/no syntactic dependencies on question item T |
| | | Co_Relations<br>Coref_Relations | Question item H and question item T are considered as a whole in coreference relation<br>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<br>Question-Table-Partialmatch<br>Question-Table-Nomatch | Question item H is spelled exactly/partially/not the same as table item T |
| $\mathcal{Q}$ | C | Question-Column-Exactmatch<br>Question-Column-Partialmatch<br>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<br>Table-Question-Partialmatch<br>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<br>Table-Table-Fkr<br>Table-Table-Fkb | At least one column in table item H is a foreign key for certain column in table item T<br>At least one column in table item T is a foreign key for certain column in table item H<br>Table item H and T satisfy both "Table-Table-Fk" and "Table-Table-Fkr" relations |
| $\mathcal{T}$ | $\mathcal{C}$ | Table-Column-Pk<br>Table-Column-Has | Column item T is the primary key for table item H<br>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<br>Column-Question-Partialmatch<br>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<br>Column-Table-Has<br>Column-Table-Generic | Column item H is the primary key for table item T<br>Column item H belongs to table item T<br>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<br>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 14: All relations used in our experiment. $\mathcal{V}$ is the matched question item that extracted from $\mathcal{Q}$.