Graphix-T5: Mixing Pre-trained Transformers with Graph-Aware Layers for Text-to-SQL Parsing

Jinyang Li[1,2*], **Binyuan Hui**[2], **Reynold Cheng**[1,5†], **Bowen Qin**[3], **Chenhao Ma**[4], **Nan Huo**[1], **Fei Huang**[2], **Wenyu Du**[1], **Luo Si**[2], **Yongbin Li**[2 †]

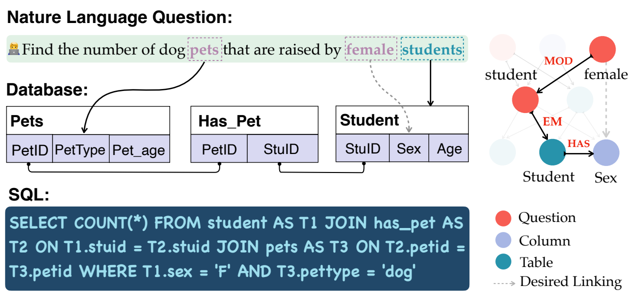
[1]The University of Hong Kong
[2]DAMO Academy, Alibaba Group
[3]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[4]The Chinese University of Hong Kong (Shenzhen)
[5]Guangdong–Hong Kong-Macau Joint Laboratory
{jl0725,huonan,wenyudu}@connect.hku.hk, ckcheng@cs.hku.hk,
bw.qin@siat.ac.cn, machenhao@cuhk.edu.cn,
{binyuan.hby,f.huang,luo.si,shuide.lyb}@alibaba-inc.com

Abstract

The task of text-to-SQL parsing, which aims at converting natural language questions into executable SQL queries, has garnered increasing attention in recent years. One of the major challenges in text-to-SQL parsing is domain generalization, *i.e.* , how to generalize well to unseen databases. Recently, the pre-trained text-to-text transformer model, namely T5, though not specialized for text-to-SQL parsing, has achieved state-of-the-art performance on standard benchmarks targeting domain generalization. In this work, we explore ways to further augment the pre-trained T5 model with specialized components for text-to-SQL parsing. Such components are expected to introduce structural inductive bias into text-to-SQL parsers thus improving model's capacity on (potentially multi-hop) reasoning, which is critical for generating structure-rich SQLs. To this end, we propose a new architecture **GRAPHIX-T5**, a mixed model with the standard pre-trained transformer model augmented by specially-designed graph-aware layers. Extensive experiments and analysis demonstrate the effectiveness of GRAPHIX-T5 across four text-to-SQL benchmarks: **SPIDER**, **SYN**, **REALISTIC** and **DK**. GRAPHIX-T5 surpass all other T5-based parsers with a significant margin, achieving new state-of-the-art performance. Notably, GRAPHIX-T5-large reaches performance superior to the original T5-large by 5.7% on exact match (EM) accuracy and 6.6% on execution accuracy (EX). This even outperforms the T5-3B by 1.2% on EM and 1.5% on EX.

1 Introduction

Relational database, serving as an important resource for users to make decision in many fields, such as health care, sports, and entertainment, has emerged frequently because of the big data era. It is efficient for data users to access the information from databases via structured query language,



Find the number of dog pets that are raised by female students

```
SELECT COUNT(*) FROM student AS T1 JOIN has_pet AS
T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid =
T3.petid WHERE T1.sex = 'F' AND T3.pettype = 'dog'
```

Figure 1: This is an illustration of cross-domain text-to-SQL challenge. The link between the target column `sex` and the token `female` is highly desired but extremely challenging for the model to capture. However, this dilemma can be mitigated by a multi-hop reasoning path (`female` $\xrightarrow{\text{MOD}}$ `student` $\xrightarrow{\text{EM}}$ `Student` $\xrightarrow{\text{HAS}}$ `Sex`).

e.g., SQL. Despite its effectiveness and efficiency, the complex nature of SQLs leads to extremely expensive learning efforts for non-technical users. Therefore, text-to-SQL (Cai et al. 2018; Zelle and Mooney 1996; Xu, Liu, and Song 2017; Yu et al. 2018a; Yaghmazadeh et al. 2017), aiming to convert natural language instructions or questions into SQL queries, has attracted remarkable attention.

In this work, we explore the challenging cross-domain setting where a text-to-SQL parser needs to achieve *domain generalization*, *i.e.* , the ability to generalize to domains that are unseen during training. Achieving this goal would, in principle, contribute to a universal natural language interface that allows users to interact with data in arbitrary domains. The major challenge towards domain generalization (Wang et al. 2020a; Cao et al. 2021; Wang et al. 2022; Cai et al. 2021; Hui et al. 2022) is that generating structure-rich SQLs requires (potentially multi-hop) **reasoning**, *i.e.* the ability to properly contextualize a user question against a given database by considering many explicit relations (*e.g.*, table-

[*] Work done during an intern at Alibaba DAMO Academy.
[†] Corresponding authors are Reynold Cheng and Yongbin Li.

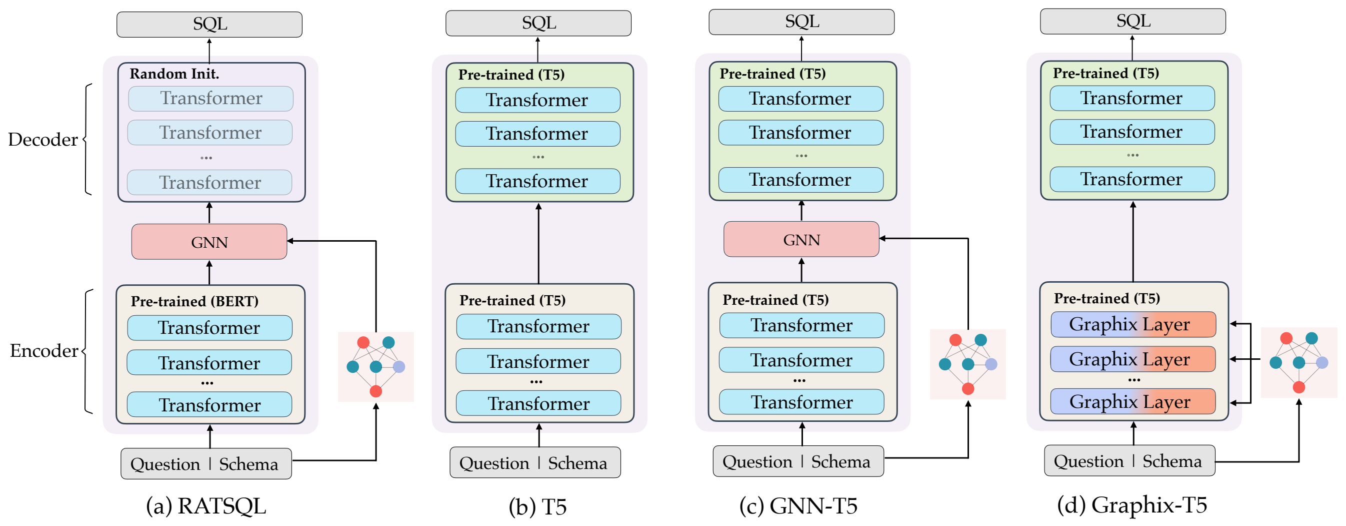


Figure 2: Graphical illustration of existing methods (a) RATSQL [pre-trained BERT-encoder → graph-based module → randomly initialized decoder]. (b) T5 [pre-trained T5-encoder → pre-trained T5-decoder] and the proposed variant (c) GNN-T5 [pre-trained T5-encoder → graph-based module → pre-trained T5-decoder] (d) GRAPHIX-T5 [semi-pre-trained graphix-module → pre-trained T5-decoder].

column relations specified by database schema) and implicit relations (*e.g.*, whether a phrase refers to a column or table). Figure 1 shows an introductory example of multi-hop reasoning in the text-to-SQL parsing and Figure 5 presents two more detailed cases.

From the modeling perspective, there are two critical dimensions along which we can differentiate current text-to-SQL parsers. The first is how to effectively imbue relational structures (both explicit and implicit) in the form of graphs into neural networks, and the second is how to take the most advantage of pre-trained models (*e.g.*T5 (Raffel et al. 2020)). These two dimensions are inter-connected and form a spectrum of methods. On one end of the spectrum, PICARD (Scholak, Schucher, and Bahdanau 2021) uses the original pre-trained T5 model by linearizing database schemas into sequences, hoping that T5 can successfully capture the underlying relational structures. On the other end of the spectrum, RAT-SQL (Wang et al. 2020a) only utilizes pre-trained encoders (*e.g.*, BERT (Devlin et al. 2019)) and explicitly captures desired relations via specialized relation-aware models. However, more powerful encoder-decoder based pre-trained models are not exploited in this framework, but relational structures are accommodated at most. In this work, we explore the cross zone where the encoder-decoder based pre-trained models (specifically T5) and relation-aware encodings are deeply coupled in favor of better domain generalization. We first observe that naively adding a relational graph-based module in the middle of T5, resulting in a 'T5-encoder → graph-based module → T5-decoder architecture' (see also Figure 2(c), namely GNN-T5), does not work very well on standard benchmarks. Presumably, the deficiency comes from the middle graph-based modules breaking the original information flow inside T5.

In order to address this problem, we present a novel architecture called **GRAPHIX-T5** that is capable of effectively modelling relational structure information while maintaining the powerful contextual encoding capability of the pretrained T5. **First**, we design a GRAPHIX layer that simultaneously encodes a mixture of semantic and structural information. Concretely, hidden states of inputs composed by questions and databases are modeled by contextualized semantic encoding, and the structural representation is injected in each transformer layer using a relational GNN block that enhances multi-hop reasoning through message passing (Fang et al. 2020; Velickovic et al. 2018) to capture explicit and implicit relations. **Second**, we construct a new encoder by stacking the GRAPHIX layers and replacing the original T5 encoder. In each GRAPHIX layer, the parameters of the semantic block are still initialized by T5, in an attempt to maintain the contextualized encoding power of the pretraining. In contrast to the severed GNN-T5 (Figure 2.(c)), the GRAPHIX-T5 (Figure 2.(d)) will allow intensive interaction between semantic and structure from the starting layers.

We empirically show the effectiveness of GRAPHIX-T5 on several cross-domain text-to-SQL benchmarks, *i.e.* , SPIDER, SYN, DK and REALISTIC. On these datasets, the proposed model achieves new state-of-the-art performance, substantially outperforming all existing models by large margins. Specifically, GRAPHIX-T5-large surprisingly beats the vanilla T5-3B. Furthermore, we verified that GRAPHIX-T5 can also achieve the significant improvement in the low-resource and compositional generalization obviously thanks to the introduction of structural bias.

2 Task Formulation and Notations

2.1 Task Definition

Given a natural language question $\mathcal{Q} = \{q_1, ..., q_{|\mathcal{Q}|}\}$ with its corresponding database schemas $\mathcal{D} = \langle \mathcal{C}, \mathcal{T} \rangle$, where $\mathcal{C} =$

$\{c_1, ..., c_{|\mathcal{C}|}\}$ and $\mathcal{T} = \{t_1, ..., t_{|\mathcal{T}|}\}$ represent columns and tables, $|\mathcal{C}|$ and $|\mathcal{T}|$ refer to the number of columns and tables in each database respectively. The goal of text-to-SQL is to generate the corresponding SQL query y.

2.2 Vanilla T5 Architecture

Model Inputs The most canonical and effective format of inputs to T5 performing text-to-SQL task is *PeteShaw* (Shaw et al. 2021), which unifies natural language questions $\mathcal{Q}$ and database schema $\mathcal{D}$ as a joint sequence as shown:

$$x = [q_1, ..., q_{|Q|} \,|\, \mathcal{D}_{name} \,|t_1 : c_1^{t_1}, ..., c_{|\mathcal{C}|}^{t_1}|...|t_{|\mathcal{T}|} : c_1^{t_{|\mathcal{T}|}}, ..., c_{|\mathcal{C}|}^{t_{|\mathcal{T}|}}|*], \tag{1}$$

where q_i is i^{th} token in the question, t_j represents j^{th} table in the $\mathcal{D}$, and $c_k^{t_j}$ refers to the k^{th} column in the j^{th} table. $*$ is the special column token in the database. $\mathcal{D}_{name}$ is the name of each database.

Encoder-Decoder Training Mechanism Following (Shaw et al. 2021), T5 (Raffel et al. 2020) adopt an encoder-decoder mechanism to generate SQLs. First, the bi-directional encoder learns the hidden state h of input x, then the decoder generates SQLs based on h as:

$$h = \mathbf{Enc}_{\Theta}(x)\,; y = \mathbf{Dec}_{\Upsilon}(h), \tag{2}$$

where Θ and Υ refers to parameters of the encoder and decoder, and h connects the encoder and decoder. The model is initialized with pretrained T5 parameters and optimized as the following objective.

$$\max_{\Theta,\Upsilon} \log p_{\Theta,\Upsilon}(y \mid x) = \sum_{i=1}^{|y|} \log p_{\Theta,\Upsilon}(y_i \mid y_{1:i-1}, x), \tag{3}$$

where x, y indicates the input and output tokens respectively and $|y|$ is the max length of generation SQL.

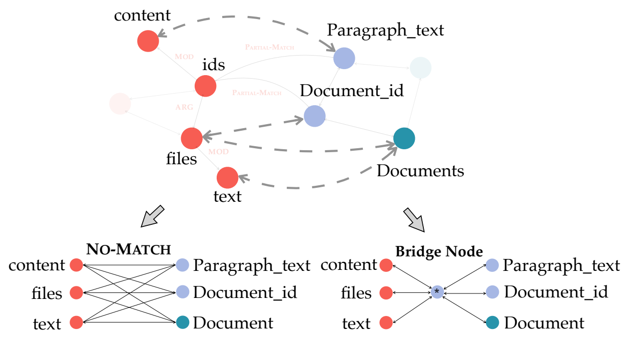
3 Proposed Approach: GRAPHIX-T5

3.1 Model Inputs

Contextual Encoding We continue to take both questions and database schemas as depicted in Eq. (1) to encode the contextual information through the original T5.

Graph Construction The joint input questions and schemas can be displayed as a heterogeneous graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{R} \rangle$ consisting of three types of nodes $\mathcal{V} = \mathcal{Q} \cup \mathcal{C} \cup \mathcal{T}$ and multiple types of relations $\mathcal{R} = r_1, ..., r_{|\mathcal{R}|}$, where each r_i refers to a one-hop relation between nodes and a multi-hop relation r^k is defined as a composition of one-hop relations: $r^k = r_1 \circ r_2 \cdots \circ r_I$ as shown in the Figure 1, where I refers to the length of each r^k. Inspired by (Wang et al. 2020a; Cao et al. 2021; Qin et al. 2022b; Hui et al. 2022), we enumerated a list of pre-defined relations to connect nodes. The relation sets can be divided into three main categories:

- Schema relations: FOREIGN-KEY, PRIMARY-KEY, and SAME-TABLE pertain to the particular explicit schema relations that the original T5 cannot obtain from linear inputs.



(a) No-Match Mode (b) Bridge Node Mode

Figure 3: Figure shows the circumstances when entities in the question are hard to string-match the schema items. (a) is the strategy to solve this case by NO-MATCH Mode, which fully connects schema nodes with all token nodes. (b) is our solution to add a bridge node to link the question and schema nodes via less number of edges.

- Schema linking relations: EXACT-MATCH, PARTIAL-MATCH, and VALUE-MATCH are implicit linking relations between question and schema nodes. A new type of relation BRIDGE is introduced.
- Question relations: MODIFIER and ARGUMENT are implicit dependency relations between tokens in a question.

NO-MATCH Mode vs. BRIDGE Mode Previous works (Cao et al. 2021; Hui et al. 2022) through adding the dummy edges called NO-MATCH indicate that the there are question tokens and the schema tokens, which should be correlated but cannot be linked due to existing string-matched rules. However, as shown in Figure 3, NO-MATCH may lead to an over-smoothing problem (Chen et al. 2020a) since they bring out too many noisy neighbors to compute the attention score. Suppose there exists A tokens for the question and B schema items that are semantically relevant but not linked by the rule, the number of edges need to be linked as NO-MATCH is $A \times B$. In contrast, we leverage the special token $\star$ as a bridge node, allowing all schema nodes to be reached from the question token nodes by decreasing the number of edges drastically from $A \times B$ to $A + B$.

3.2 Graphix-Layer

The GRAPHIX layer is designed to integrate semantic information obtained from each transformer block with structural information of a relational graph neural network (GNN) block.

Semantic Representation The semantic representations of hidden states are firstly encoded by a Transformer (Vaswani et al. 2017) block, which contains two important components, including Multi-head Self-attention Network (**MHA**) and Fully-connected Forward Network (**FFN**). In the l^{th} GRAPHIX Layer, the hidden states represent $\mathcal{H}_S^l = \left\{h_1^{(l)}, \ldots, h_N^{(l)}\right\}$, N is the max length of the inputs. MHA at first maps query matrix $\mathbf{Q} \in \mathbb{R}^{m \times d_k}$, key and value matrix $\mathbf{K} \in \mathbb{R}^{n \times d_k}$, $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ into an attention vector via self-attention mechanism as Eq. (4)

$$\mathbf{Attn}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{4}$$

in which m is the number of query vectors and n is the number of key or value vectors. MHA executes the self-attention over h heads with each head i being independently parameterized by $\mathbf{W}_i^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_v}$ and mapping inputs into queries, key-value pairs. Usually $d_k = d_v = d_m/h$ in the transformer blocks of T5 and d_m denotes the dimension of T5. Then MHA calculates the attention outputs for each head and concatenate them as following:

$$\mathbf{head}_{\mathrm{i}} = \mathbf{Attn}(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V), \tag{5}$$

$$\mathbf{MHA}(\mathcal{H}_{\mathcal{S}}^{(l)}) = \mathbf{Concat}\left(\mathbf{head}_1, \cdots, \mathbf{head}_{\mathrm{h}}\right)\mathbf{W}^O, \tag{6}$$

$$\widehat{\mathcal{H}}_{\mathcal{S}}^{(l)} = \mathbf{MHA}(\mathcal{H}_{\mathcal{S}}^{(l)}), \tag{7}$$

where $\mathbf{W}^O \in \mathbb{R}^{d_m h \times d_m}$ is a trainable parameter matrix. Semantic hidden states need to be acquired through another component, *i.e.* , FFN, which is applied as Eq. (8).

$$\mathbf{FFN}(\widehat{\mathcal{H}}_{\mathcal{S}}^{(l)}) = \mathbf{max}\left(0, \widehat{\mathcal{H}}_{\mathcal{S}}^{(l)}\mathbf{W_1} + \mathbf{b_1}\right)\mathbf{W_2} + \mathbf{b_2}, \tag{8}$$

where linear weight matrices represent $\mathbf{W_1} \in \mathbb{R}^{d_m \times d_{ff}}$, $\mathbf{W_2} \in \mathbb{R}^{d_{ff} \times d_m}$ respectively. Experimentally, larger d_{ff} is preferred, which is usually set as $d_{ff} = 4d_m$. Eventually, the semantic hidden states are acquired after layer normalization and residual connection as

$$\tilde{\mathcal{H}}_{\mathcal{S}}^{(l)} = \mathbf{LayerNorm}(\widehat{\mathcal{H}}_{\mathcal{S}}^{(l)} + \mathbf{FFN}(\widehat{\mathcal{H}}_{\mathcal{S}}^{(l)})), \tag{9}$$

Structural Representation In each GRAPHIX Layer, structural representations are produced through the relational graph attention network (RGAT) (Wang et al. 2020b) over the pre-defined question-schema heterogeneous graph. Formally, given initial node embedding[1] e_i^{init} for i^{th} node and its j^{th} neighbor e_j^{init} linked by specific types of relations, it can be computed through:

$$\vec{\alpha}_{ij} = \frac{e_i^{init}\widetilde{\mathbf{W}}_Q\left(e_j^{init}\widetilde{\mathbf{W}}_K + \phi\left(r_{ij}\right)\right)^\top}{\sqrt{d_z}}, \tag{10}$$

$$\alpha_{ij} = \mathbf{softmax}_j\left(\vec{\alpha}_{ij}\right), \tag{11}$$

$$\hat{e}_i^{init} = \sum_{j \in \tilde{\mathcal{N}}_i} \alpha_{ij}\left(e_j^{init}\widetilde{\mathbf{W}}_V + \phi(r_{ij})\right), \tag{12}$$

$$\hat{e}_i^{(l)} = \mathbf{LayerNorm}(e_i^{init} + \hat{e}_i^{init}\widetilde{\mathbf{W}}_O), \tag{13}$$

$$\tilde{e}_i^{(l)} = \mathbf{LayerNorm}(\hat{e}_i^{(l)} + \mathbf{FFN}(\hat{e}_i^{(l)})), \tag{14}$$

Then the output node embeddings are collected as $\tilde{\mathcal{E}}_{\mathcal{G}}^{(l)} = \left\{\tilde{e}_1^{(l)}, \ldots, \tilde{e}_N^{(l)}\right\}$, where $\widetilde{\mathbf{W}}_Q$, $\widetilde{\mathbf{W}}_K$, $\widetilde{\mathbf{W}}_V$, $\widetilde{\mathbf{W}}_O \in \mathbb{R}^{d \times d}$ are trainable parameters in the RGAT. $\phi(r_{ij})$ is a mapping function that can produce a d-dim embedding representing for each relation between i^{th} node and j^{th} node. More importantly, $\tilde{\mathcal{N}}_i$ denotes the relational reception field, which is equal to the number of how many neighbors of i^{th} node that RGAT will consider when updating representation of each node via message passing.

Jointly Representation After computing representations from both semantic and structural space, the l^{th} GRAPHIX Layer employs a mixture of semantic and structural information to enable information integration as following:

$$\tilde{\mathcal{H}}_{\mathcal{M}}^{(l)} = \tilde{\mathcal{H}}_{\mathcal{S}}^{(l)} + \tilde{\mathcal{E}}_{\mathcal{G}}^{(l)}, \tag{15}$$

3.3 Graphix-T5

Here we present our entire GRAPHIX-T5 model formally. The hidden states of the last layer of GRAPHIX-encoder can be represented as:

$$h = \mathbf{Enc}_{\Theta,\Psi}\left(x, \mathcal{G}\right), \tag{16}$$

where $\mathcal{G}$ is the question-schema heterogeneous graph, the Ψ are the additional parameters of the RGAT, which are initialized randomly. In order to preserve the pre-trained semantic knowledge, we migrate parameters Θ from original T5 encoder as the initial parameters of semantic transformer block of the GRAPHIX layer.

3.4 Training

Similar to original T5, we also follow a fine-tuning strategy. The whole training framework is to optimize the following log-likelihood.

$$\max_{\Theta,\Upsilon,\Psi} \log p_{\Theta,\Upsilon,\Psi}(y \mid x) = \sum_{i=1}^{|y|} \log p_{\Theta,\Upsilon,\Psi}\left(y_i \mid y_{1:i-1}, x, \mathcal{G}\right). \tag{17}$$

4 Experiment

4.1 Set Up

Datasets and Settings We conduct extensive experiments on four challenging benchmarks for cross-domain text-to-SQLs and two different training settings. **(1) SPIDER** (Yu et al. 2018b) is a large-scale cross-domain text-to-SQL benchmark. It contains 8659 training examples and 1034 development examples, which covers 200 complex databases across 138 domains. The testing set is not available for individual review. **(2) SYN** (Gan et al. 2021a) replaces the simple string-matched question tokens or schema names with their synonyms. **(3) DK** (Gan, Chen, and Purver 2021) requires the text-to-SQL parsers to equip with the capability of domain knowledge reasoning. **(4) REALISTIC** removes and switches the obvious mentions of schema items in questions, making it closer to the real scenarios. Furthermore, we also test the compositional generalization ability of our model on the **SPIDER-SSP** (Shaw et al. 2021) with three splits from SPIDER: Spider-Length (split dataset based on variant lengths); Spider-TMCD (Target Maximum Compound Divergence) and Spider-Template (split based on different

[1]Various initialization strategies could be implemented. In this work, we initialized the node embeddings with their semantic representations.

MODEL	EM	EX
RAT-SQL + BERT ♡	69.7	-
RAT-SQL + Grappa ♡	73.9	-
GAZP + BERT	59.1	59.2
BRIDGE v2 + BERT	70.0	68.3
NatSQL + GAP	73.7	75.0
SMBOP + GRAPPA	74.7	75.0
LGESQL + ELECTRA ♡	75.1	-
S^2SQL + ELECTRA ♡	76.4	-
T5-large	67.0	69.3
GRAPHIX-T5-large	72.7(↑ 5.7)	75.9(↑ 6.6)
T5-large + PICARD ♣	69.1	72.9
GRAPHIX-T5-large + PICARD ♣	76.6(↑ 7.5)	80.5(↑ 7.6)
T5-3B	71.5	74.4
GRAPHIX-T5-3B	75.6 (↑ 4.1)	78.2 (↑ 3.8)
T5-3B + PICARD ♣	75.5	79.3
GRAPHIX-T5-3B + PICARD ♣	**77.1**(↑ 1.6)	**81.0**(↑ 1.7)

Table 1: Exact match (EM) and execution (EX) accuracy (%) on SPIDER development set. ♡ means the model does not predict SQL values. ♣ means the model uses the constrained decoding PICARD. ↑ is an absolute improvement.

parsing templates). Finally, the performances of GRAPHIX-T5 on **LOW-RESOURCE** setting are evaluated on usage of 10%, 20%, 50% data separately.

Evaluation Metrics Following (Yu et al. 2018b), Exact Match (EM) and Execution Accuracy (EX) are the two standard metrics we use to measure performance of our model. EM can evaluate how much a generated SQL is comparable to the gold SQL.

Implementation Details We implement our codes [2] mainly based on hugging-face transformers library (Wolf et al. 2020) [3]. We set the max input length as 1024, generation max length as 128, and batch size as 32. We also adopt Adafactor (Shazeer and Stern 2018) as our primary optimizer with a linear decayed learning rate of 5e-5. During the experiment, GRAPHIX layers are mainly injected into the encoder to learn better representations for structural generalization. We evaluate our effectiveness of GRAPHIX-T5 across two main versions: T5-Large with approximately 800M parameters and T5-3B, with more than 3 Billion parameters literally. All experiments are conducted on one NVIDIA Tesla A100, which is available for most research centers.

Compared Methods Our model are compared mainly to mainstream strong baseline models such as GNNSQL (Bogin, Berant, and Gardner 2019), RATSQL (Wang et al. 2020a), GAZP (Zhong et al. 2020), BRIDEGE (Chen et al. 2020b), SMBOP (Rubin and Berant 2021), NatSQL (Gan et al. 2021b), LGESQL (Cao et al. 2021), S^2SQL (Hui et al. 2022) and T5+PICARD (Scholak, Schucher, and Bahdanau 2021) across the disparate datasets and settings.

[2] https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/graphix

[3] https://huggingface.co/

MODEL	SYN	DK	REALISTIC
GNN	23.6	26.0	-
IRNet	28.4	33.1	-
RAT-SQL	33.6	35.8	-
RAT-SQL + BERT	48.2	40.9	58.1
RAT-SQL + Grappa	49.1	38.5	59.3
LGESQL + ELECTRA	64.6	48.4	69.2
T5-large	53.6	40.0	58.5
GRAPHIX-T5-large	61.1 (↑ 7.5)	48.6 (↑ 8.6)	67.3 (↑ 8.8)
T5-3B	58.0	46.9	62.0
GRAPHIX-T5-3B	**66.9** (↑ 8.9)	**51.2** (↑ 4.3)	**72.4** (↑ 10.4)

Table 2: Exact match (EM) accuracy (%) on SYN, DK and REALISTIC benchmark.

MODEL	TEMPLATE	LENGTH	TMCD
T5-base	59.3	49.0	60.9
T5-3B	64.8	56.7	69.6
NQG-T5-3B	64.7	56.7	69.5
GRAPHIX-T5-3B	**70.1** (↑ 5.4)	**60.6** (↑ 3.9)	**73.8** (↑ 4.3)

Table 3: Exact match (EM) accuracy (%) on compositional dataset SPIDER-SSP.

4.2 Overall Performance

Results on SPIDER Table 1 displays the performance of GRAPHIX-T5 and other competitive baseline models on official SPIDER benchmark. First, we demonstrate that GRAPHIX-T5-3B with a constrained decoding module PICARD (Scholak, Schucher, and Bahdanau 2021) achieves the state-of-the-art on this challenging text-to-SQL benchmark. Also, it is evident that GRAPHIX-T5 is vastly superior to the vanilla T5 on large and 3B scales with a significant margin. This indicates that the structural generalization capability of the GRAPHIX layer is crucial for T5, such a text-to-text PLM to perform the text-to-SQL task.

Zero-shot Results on More Challenging Settings As shown in the Table 2, we further demonstrate the robustness of GRAPHIX-T5 when it confronts with more challenging and closer to realistic evaluations in SYN, DK, REALISTIC without any additional training. First of all, the results show that GRAPHIX-T5-3B outperforms other baseline models across all three datasets. Furthermore, we observe that GRAPHIX-T5-large and GRAPHIX-T5-3B surpass the performance of vanilla T5-large and T5-3B with a clear margin, respectively. This demonstrates that vanilla T5 is hungry for structural reasoning when dealing with more flexible and complicated questions for text-to-SQLs from real-world scenarios. And GRAPHIX can mitigate this problem.

Results on Compositional Generalization As shown in Table 3, on SPIDER-SSP, the grammar-based inductive T5 model provided by (Shaw et al. 2021), named NQG-T5, has no obvious advantages over vanilla T5, which indicates that the grammar of natural language is not helpful to enhance T5 for compositional generation. However, GRAPHIX-T5 helps the T5 gain the SQL knowledge and makes it less vulnerable to these modifications through the effective fusion of

MODEL	SPIDER easy	SPIDER medium	SPIDER hard	SPIDER extra	SPIDER all	SYN easy	SYN medium	SYN hard	SYN extra	SYN all	REALISTIC easy	REALISTIC medium	REALISTIC hard	REALISTIC extra	REALISTIC all
T5-large	85.5	70.9	55.2	41.6	67.0	69.0	56.8	46.3	30.2	53.6	79.8	68.0	44.4	28.9	58.5
GRAPHIX-T5-large	**89.9**	**78.7**	**59.8**	**44.0**	**72.6**	**75.8**	**67.5**	**50.6**	**33.1**	**61.1**	**88.1**	**77.3**	**50.5**	**40.2**	**67.3**
T5-3B	89.5	78.3	58.6	40.4	71.6	74.2	64.5	48.0	27.8	58.0	85.3	73.4	46.5	27.8	62.0
GRAPHIX-T5-3B	**91.9**	**81.6**	**61.5**	**50.0**	**75.6**	**80.6**	**73.1**	**52.9**	**44.6**	**66.9**	**93.6**	**85.7**	**52.5**	**41.2**	**72.4**

Table 4: Exact matching (EM) accuracy by varying the levels of difficulty of the inference data on three main benchmarks.

MODEL	EM	EX
(a) RAT-SQL + BERT	69.7	-
(b) T5-large	67.0	69.3
(c) GNN-T5-large	51.6	54.5
(d) GRAPHIX-T5-large		
w/ BRIDGE Mode	**72.7**	**75.9**
w/ NO-MATCH Mode	71.1	74.2
w/ DOUBLE-GRAPH	72.0	74.7

Table 5: Ablation study for the variant GNN + PLM tactics on cross-domain text-to-SQLs, echoing Figure 2, (a) is RAT-SQL, (b) is vanilla T5, (c) is GNN-T5 and (d) is GRAPHIX.

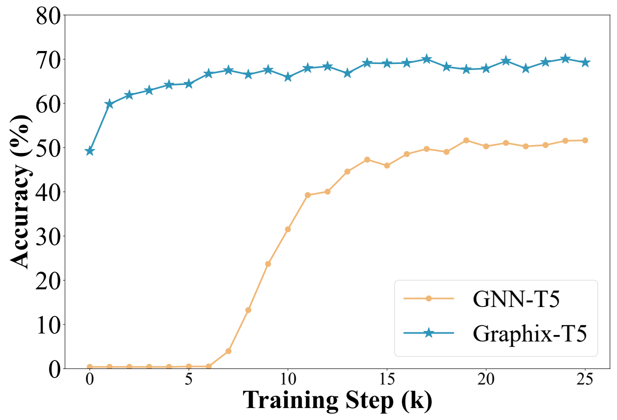


Figure 4: The performance of the validation sets during the convergence of GRAPHIX-T5 and GNN-T5 on SPIDER. It can be clearly demonstrated that GNN-T5 has extremely unsatisfactory performance, due to catastrophic forgetting.

structural information.

Results on Complex Queries As presented in Table 4, we also compare the more precise performance results of GRAPHIX-T5 to the vanilla T5 in four separate SQL difficulty levels splitted by SPIDER officially, in order to better comprehend the performance improvements. We observe that GRAPHIX-T5 is more capable of handling harder text-to-SQL cases, as illustrated in the **Hard** and **Extra-hard** examples, indicating that structural bias training is beneficial to hard text-to-SQL cases.

4.3 Ablation Study

As shown in Table 5, to better validate the function of each component of GRAPHIX-T5, ablation studies are performed in large version and expected to answer the following questions.

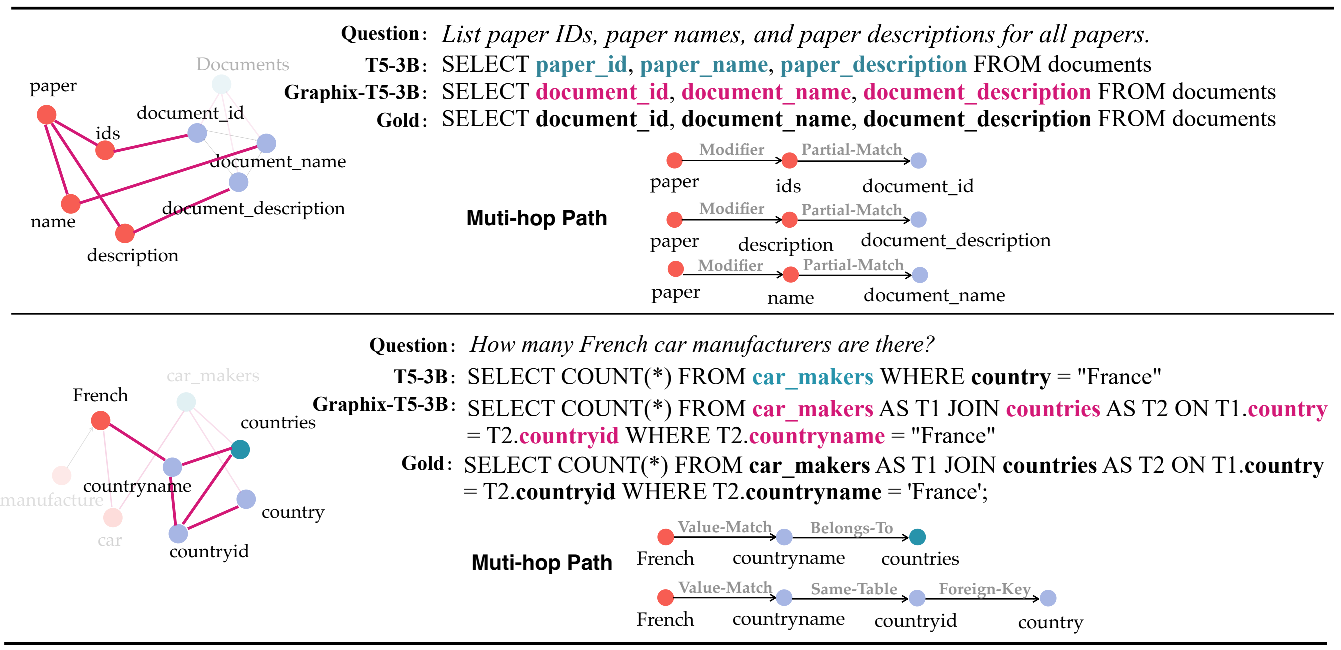
[1] How effective is BRIDGE MODE ? GRAPHIX-T5-large with BRIDGE MODE can achieve the better performance than with NO-MATCH Mode via reducing the number of noisy neighbors. It indicates that NO-MATCH mode will greatly increase the number of noisy neighbors, resulting in higher risk of over-smoothing issues (Chen et al. 2020a).

[2] Could GRAPHIX be incorporated into decoder ? With DOUBLE-GRAPH means that GRAPHIX-T5 incorporate GRAPHIX layer into the both encoder and decoder. The result reveals that adding GRAPHIX layers to the decoder does not lead to any improvements. Since decoder is an auto-regressive model, which only considers the history tokens when generating the current token. However, GRAPHIX-T5, which can forecast the information of future tokens by global linking, may disrupt this characteristic leading to the negative impact on the decoder. Therefore, we propose that the best tactic is to only incorporate GRAPHIX layers into the encoder.

[3] Is GRAPHIX superior than other architecture variants ? Echoing Figure 2, we access the performance of 4 categories of models using PLMs on SPIDER. According to Table 5 (c), the performance of GNN-T5 has decreased by roughly 20% when compared to GRAPHIX-T5, proving that GNN-T5 meets the catastrophic forgetting problem (French 1999). Since the accuracy of the GNN-T5 continues to be 0 in the first thousands of steps and the performance decreases significantly from vanilla T5, as shown in Figure 4, it is evident that all pre-trained knowledge from T5 would be forgotten. In contrast, the result verifies the advantages of GRAPHIX-T5 that can avoid catastrophic forgetting and augment generalization capability.

4.4 Case Study

To illustrate the effectiveness of GRAPHIX qualitatively, two examples are displayed in Figure 5, which are sampled randomly from SYN. Figure 5 shows the comparison of predicted SQLs by vanilla T5-3B and GRAPHIX-T5-3B. We can observe that GRAPHIX can generate correct SQLs even in the hard scenarios. That is because that, even with a small number of keywords overlapped, GRAPHIX-T5 can accurately identify counterpart column or table objects and generate a high-quality SQL through multi-hop reasoning and structural grounding. For example, in the first case, vanilla T5-3B picks the incorrect columns `paper_id`, `paper_name`, and `paper_description`, which even


Question: *List paper IDs, paper names, and paper descriptions for all papers.*
T5-3B: SELECT **paper_id**, **paper_name**, **paper_description** FROM documents
Graphix-T5-3B: SELECT **document_id**, **document_name**, **document_description** FROM documents
Gold: SELECT **document_id**, **document_name**, **document_description** FROM documents

Question: *How many French car manufacturers are there?*
T5-3B: SELECT COUNT(*) FROM **car_makers** WHERE **country** = "France"
Graphix-T5-3B: SELECT COUNT(*) FROM **car_makers** AS T1 JOIN **countries** AS T2 ON T1.**country** = T2.**countryid** WHERE T2.**countryname** = "France"
Gold: SELECT COUNT(*) FROM **car_makers** AS T1 JOIN **countries** AS T2 ON T1.**country** = T2.**countryid** WHERE T2.**countryname** = 'France';

Documents
paper
document_id
ids
document_name
name
document_description
description
Muti-hop Path
paper → Modifier → ids → Partial-Match → document_id
paper → Modifier → description → Partial-Match → document_description
paper → Modifier → name → Partial-Match → document_name
car_makers
French
countries
countryname
manufacture
country
car
countryid
Muti-hop Path
French → Value-Match → countryname → Belongs-To → countries
French → Value-Match → countryname → Same-Table → countryid → Foreign-Key → country

Figure 5: Case study: two illustrative cases sampled randomly from SYN. It shows that multi-hop reasoning can help GRAPHIX-T5 generate more correct SQLs in terms of semantic meanings and database schema structures.

don't appear in the table documents. This implies that vanilla T5-3B is unable to reach the target schema elements without the capability of structural grounding when confronting challenging text-to-SQLs. Instead, GRAPHIX-T5-3B can correspond the question entities to the correct column names through multi-hop paths presented in the Figure 5. In the second case, vanilla T5-3B misidentifies the `country` as their target column, however, `"France"` only appears in the column `countryname` of the table `countries`. This suggests T5-3B is only able to generate semantically valid SQLs, which fails to take into account the real database structure. On contrary, GRAPHIX-T5 can produce truly valid SQLs in terms of both questions and databases via a successful mixture of semantic and structural information during training.

5 Related Works

The basic principle of a cross-domain text-to-SQL parser is to build an encoder to learn the representations of the questions and schemas, while employing a decoder to generate SQLs with the information learnt in the encoder (Qin et al. 2022a). In particular, IRNET (Guo et al. 2019) proposes to design an encoder to learn the representations of questions and schemas respectively via an attention-based Bi-LSTM and a decoder to predict SQLs via encoded intermediate representations. Later, the graph-based encoders have been successfully proved its effectiveness in text-to-SQL tasks, for example, some works (Bogin, Berant, and Gardner 2019; Chen et al. 2021) construct the schema graph and enhance the representations of inputs. RATSQL (Wang et al. 2020a), SDSQL (Hui et al. 2021b), LGESQL (Cao et al. 2021), S^{2}SQL (Hui et al. 2022) further improve structural reasoning through modelling relations between schema and questions. R^{2}SQL (Hui et al. 2021a), SCORE (Yu et al. 2021) and STAR (Cai et al. 2022) enhance structural reasoning for context-dependent text-to-SQL parsing. These works are performed by the PLM independently building the semantic features, followed by the graph-based module injecting the structural information. However, such training strategy is just effective to encoder-based PLMs (*i.e.* , BERT (Devlin et al. 2019).

Recently, the text-to-text PLM T5 has been proven effectiveness in text-to-SQL (Shaw et al. 2021; Qin et al. 2022c). Besides, (Scholak, Schucher, and Bahdanau 2021) designs a constrained decoding process, namely PICARD, to refuse erroneous tokens during the beam-search phase. Xie et al. (2022) further injects the knowledge from other structural knowledge grounding tasks into T5 with multi-task to boost performance on text-to-SQL. Despite effectiveness, these methods still struggle to generate SQLs in the more challenging and complex scenarios without explicit and implicit structural information.

6 Conclusion

In this paper, we proposed an effective architecture to boost the capability of structural encoding of T5 cohesively while keeping the pretrained T5's potent contextual encoding ability. In order to achieve this goal, we designed a Graph-Aware semi-pretrained text-to-text PLM, namely GRAPHIX-T5, to augment the multi-hop reasoning for the challenging text-to-SQL task. The results under the extensive experiments demonstrate the effectiveness of GRAPHIX-T5, proving that structural information is crucial for the current text-to-text PLMs for complicated text-to-SQL cases.

Acknowledgements

We thank Dr. Tao Yu and Tianbao Xie for evaluation of our work on SPIDER leaderboard. We thank Dr. Bailin Wang and Dr. Bowen Li for constructive suggestions. Reynold Cheng, Jinyang Li, Nan Huo, and Wenyu Du were supported by the University of Hong Kong (Project 104006830), the Guangdong–Hong Kong-Macau Joint Laboratory Program 2020 (Project No: 2020B1212030009), and the Innovation Wing Two Research fund. Jinyang Li was also supported by HKU Presidential PhD Scholar Programme and Alibaba Group through Alibaba Research Intern Program. Chenhao Ma was supported in part by Shenzhen Science and Technology Program under grant No.ZDSYS20211021111415025.

References

Bogin, B.; Berant, J.; and Gardner, M. 2019. Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. In *Proc. of ACL*.

Cai, R.; Xu, B.; Zhang, Z.; Yang, X.; Li, Z.; and Liang, Z. 2018. An Encoder-Decoder Framework Translating Natural Language to Database Queries. In *Proc. of IJCAI*.

Cai, R.; Yuan, J.; Xu, B.; and Hao, Z. 2021. SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL. In *Proc. of NeurIPS*.

Cai, Z.; Li, X.; Hui, B.; Yang, M.; Li, B.; Li, B.; Cao, Z.; Li, W.; Huang, F.; Si, L.; and Li, Y. 2022. STAR: SQL Guided Pre-Training for Context-dependent Text-to-SQL Parsing. In *Proc. of EMNLP Findings*.

Cao, R.; Chen, L.; Chen, Z.; Zhao, Y.; Zhu, S.; and Yu, K. 2021. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In *Proc. of ACL*.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2020a. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *Proc. of AAAI*.

Chen, X.; Meng, F.; Li, P.; Chen, F.; Xu, S.; Xu, B.; and Zhou, J. 2020b. Bridging the Gap between Prior and Posterior Knowledge Selection for Knowledge-Grounded Dialogue Generation. In *Proc. of EMNLP*.

Chen, Z.; Chen, L.; Zhao, Y.; Cao, R.; Xu, Z.; Zhu, S.; and Yu, K. 2021. ShadowGNN: Graph Projection Neural Network for Text-to-SQL Parser. In *Proc. of NAACL*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL*.

Fang, Y.; Sun, S.; Gan, Z.; Pillai, R.; Wang, S.; and Liu, J. 2020. Hierarchical Graph Network for Multi-hop Question Answering. In *Proc. of EMNLP*.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*.

Gan, Y.; Chen, X.; Huang, Q.; Purver, M.; Woodward, J. R.; Xie, J.; and Huang, P. 2021a. Towards Robustness of Text-to-SQL Models against Synonym Substitution. In *Proc. of ACL*.

Gan, Y.; Chen, X.; and Purver, M. 2021. Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization. In *Proc. of EMNLP*.

Gan, Y.; Chen, X.; Xie, J.; Purver, M.; Woodward, J. R.; Drake, J.; and Zhang, Q. 2021b. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Proc. of EMNLP Findings*.

Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; and Zhang, D. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proc. of ACL*.

Hui, B.; Geng, R.; Ren, Q.; Li, B.; Li, Y.; Sun, J.; Huang, F.; Si, L.; Zhu, P.; and Zhu, X. 2021a. Dynamic Hybrid Relation Network for Cross-Domain Context-Dependent Semantic Parsing. In *Proc. of AAAI*.

Hui, B.; Geng, R.; Wang, L.; Qin, B.; Li, Y.; Li, B.; Sun, J.; and Li, Y. 2022. S^{2}SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers. In *Proc. of ACL Findings*.

Hui, B.; Shi, X.; Geng, R.; Li, B.; Li, Y.; Sun, J.; and Zhu, X. 2021b. Improving Text-to-SQL with Schema Dependency Learning. In *arXiv:2103.04399*.

Qin, B.; Hui, B.; Wang, L.; Yang, M.; Li, J.; Li, B.; Geng, R.; Cao, R.; Sun, J.; Si, L.; Huang, F.; and Li, Y. 2022a. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. In *arXiv:2208.13629*.

Qin, B.; Wang, L.; Hui, B.; Geng, R.; Cao, Z.; Yang, M.; Sun, J.; and Li, Y. 2022b. Linking-Enhanced Pre-Training for Table Semantic Parsing. In *arXiv:2111.09486*.

Qin, B.; Wang, L.; Hui, B.; Li, B.; Wei, X.; Li, B.; Huang, F.; Si, L.; Yang, M.; and Li, Y. 2022c. SUN: Exploring Intrinsic Uncertainties in Text-to-SQL Parsers. In *Proc. of COLING*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*.

Rubin, O.; and Berant, J. 2021. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. In *Proc. of NAACL*.

Scholak, T.; Schucher, N.; and Bahdanau, D. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proc. of EMNLP*.

Shaw, P.; Chang, M.-W.; Pasupat, P.; and Toutanova, K. 2021. Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both? In *Proc. of ACL*.

Shazeer, N.; and Stern, M. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proc. of ICML*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Proc. of NeurIPS*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proc. of ICLR*.

Wang, B.; Shin, R.; Liu, X.; Polozov, O.; and Richardson, M. 2020a. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proc. of ACL*.

Wang, K.; Shen, W.; Yang, Y.; Quan, X.; and Wang, R. 2020b. Relational Graph Attention Network for Aspect-based Sentiment Analysis. In *Proc. of ACL*.

Wang, L.; Qin, B.; Hui, B.; Li, B.; Yang, M.; Wang, B.; Li, B.; Huang, F.; Si, L.; and Li, Y. 2022. Proton: Probing Schema Linking Information from Pre-trained Language Models for Text-to-SQL Parsing. In *Proc. of KDD*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proc. of EMNLP*.

Xie, T.; Wu, C. H.; Shi, P.; Zhong, R.; Scholak, T.; Yasunaga, M.; Wu, C.-S.; Zhong, M.; Yin, P.; Wang, S. I.; Zhong, V.; Wang, B.; Li, C.; Boyle, C.; Ni, A.; Yao, Z.; Radev, D.; Xiong, C.; Kong, L.; Zhang, R.; Smith, N. A.; Zettlemoyer, L.; and Yu, T. 2022. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. *ArXiv preprint*.

Xu, X.; Liu, C.; and Song, D. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *ArXiv preprint*.

Yaghmazadeh, N.; Wang, Y.; Dillig, I.; and Dillig, T. 2017. SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*.

Yu, T.; Li, Z.; Zhang, Z.; Zhang, R.; and Radev, D. 2018a. TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation. In *Proc. of NAACL*.

Yu, T.; Zhang, R.; Polozov, A.; Meek, C.; and Awadallah, A., Hassan. 2021. SCoRe: Pre-Training for Context Representation in Conversational Semantic Parsing. In *Proc. of ICLR*.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; Zhang, Z.; and Radev, D. 2018b. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proc. of EMNLP*.

Zelle, J. M.; and Mooney, R. J. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proc. of AAAI*.

Zhong, V.; Lewis, M.; Wang, S. I.; and Zettlemoyer, L. 2020. Grounded Adaptation for Zero-shot Executable Semantic Parsing. In *Proc. of EMNLP*.