ShadowGNN: Graph Projection Neural Network for Text-to-SQL Parser

Zhi Chen[1], **Lu Chen**[1]*, **Yanbin Zhao**[1], **Ruisheng Cao**[1], **Zihan Xu**[1],
Su Zhu[2] and **Kai Yu**[1]*
[1]X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
Shanghai Jiao Tong University, Shanghai, China
State Key Lab of Media Convergence Production Technology and Systems, Beijing, China
[2]AISpeech Co., Ltd., Suzhou, China
{zhenchi713, chenlusz, kai.yu}@sjtu.edu.cn

Abstract

Given a database schema, Text-to-SQL aims to translate a natural language question into the corresponding SQL query. Under the setup of cross-domain, traditional semantic parsing models struggle to adapt to unseen database schemas. To improve the model generalization capability for rare and unseen schemas, we propose a new architecture, ShadowGNN, which processes schemas at abstract and semantic levels. By ignoring names of semantic items in databases, abstract schemas are exploited in a well-designed graph projection neural network to obtain delexicalized representation of question and schema. Based on the domain-independent representations, a relation-aware transformer is utilized to further extract logical linking between question and schema. Finally, a SQL decoder with context-free grammar is applied. On the challenging Text-to-SQL benchmark Spider, empirical results show that ShadowGNN outperforms state-of-the-art models. When the annotated data is extremely limited (only 10% training set), ShadowGNN gets over absolute 5% performance gain, which shows its powerful generalization ability. Our implementation will be open-sourced at https://github.com/WowCZ/shadowgnn.

1 Introduction

Recently, Text-to-SQL has drawn a great deal of attention from the semantic parsing community (Berant et al., 2013; Cao et al., 2019, 2020). The ability to query a database with natural language (NL) engages the majority of users, who are not familiar with SQL language, in visiting large databases. A number of neural approaches have been proposed to translate questions into executable SQL queries. On public Text-to-SQL benchmarks (Zhong et al., 2017; Krishnamurthy et al., 2017), exact match accuracy even excesses more than 80%. However, the cross-domain problem for Text-to-SQL is a practical challenge and ignored by the prior datasets. To be clarified, a database schema is regarded as a domain. The domain information consists of two parts: the semantic information (e.g., the table name) of the schema components and the structure information (e.g., the primary-key relation between a table and a column) of the schema.

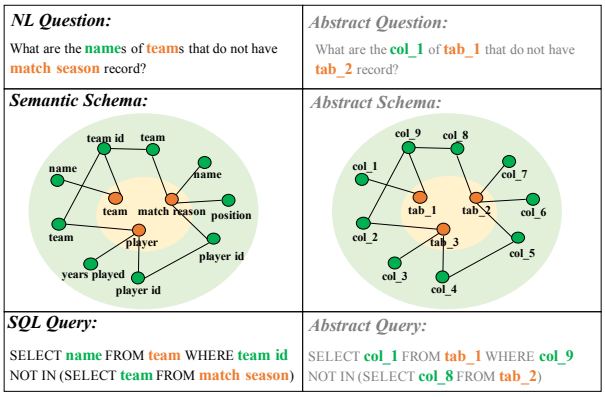

Figure 1: An example to demonstrate the impact of domain information. (b) is the human-labeled abstract representation of Text-to-SQL content from domain-aware example (a). The green nodes and orange nodes represent columns and tables respectively.

The recently released dataset, Spider (Yu et al., 2018), hides the database schemas of the test set, which are totally unseen on the training set. In this cross-domain setup, domain adaptation is challenging for two main reasons. First, the semantic information of the domains in the test and development set are unseen in the training set. On the given development set, 35% of words in database schemas do not occur in the schemas on the training set. It is hard to match the domain representations in the question and the schema. Second, there is

*The corresponding authors are Lu Chen and Kai Yu.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5567–5577
June 6–11, 2021.

a considerable discrepancy among the structure of the database schemas. Especially, the database schemas always contain semantic information. It is difficult to get the unified representation of the database schema. Under the cross-domain setup, the essential challenge is to alleviate the impact of the domain information.

First, it is necessary to figure out which role the semantic information of the schema components play during translating an NL question into a SQL query. Consider the example in Fig. 1(a), for the Text-to-SQL model, the basic task is to find out all the mentioned columns ($name$) and tables ($team$, $match\ season$) by looking up the schema with semantic information (named as semantic schema). Once the mentioned columns and tables in the NL question are exactly matched with schema components, we can abstract the NL question and the semantic schema by replacing the general component type with the specific schema components. As shown in Fig. 1(b), we can still infer the structure of the SQL query using the abstract NL question and the schema structure. With the corresponding relation between semantic schema and abstract schema, we can restore the abstract query to executable SQL query with domain information. Inspired by this phenomenon, we decompose the encoder of the Text-to-SQL model into two modules. First, we propose a ***G**raph **P**rojection **N**eural **N**etwork* (GPNN) to abstract the NL question and the semantic schema, where the domain information is removed as much as possible. Then, we use the relation-aware transformer to get unified representations of abstract NL question and abstract schema.

Our approach, named ShadowGNN, is evaluated on the challenging cross-domain Text-to-SQL dataset, Spider. Contributions are summarized as:

- We propose the ShadowGNN to alleviate the impact of the domain information by abstracting the representation of NL question and SQL query. It is a meaningful method to apply to similar cross-domain tasks.
- To validate the generalization capability of our proposed ShadowGNN, we conduct the experiments with limited annotated data. The results show that our proposed ShadowGNN can obtain absolute over 5% accuracy gain compared with state-of-the-art model, when the annotated data only has the scale of 10% of the training set.
- The empirical results show that our approach outperforms state-of-the-art models (66.1% accuracy on test set) on the challenging Spider benchmark. The ablation studies further confirm that GPNN is important to abstract the representation of the NL question and the schema.

2 Background

In this section, we first introduce relational graph convolution network (R-GCN) (Schlichtkrull et al., 2018), which is the basis of our proposed GPNN. Then, we introduce the relation-aware transformer, which is a transformer variant considering relation information during calculating attention weights.

2.1 Relational Graph Convolution Network

Before describing the details of R-GCN, we first give notations of relational directed graph. We denote this kind of graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ with nodes (schema components) $v_i \in \mathcal{V}$ and directed labeled edge $(v_i, r, v_j) \in \mathcal{E}$, where v_i is the source node, v_j is the destination node and $r \in \mathcal{R}$ is the edge type from v_i to v_j. $\mathcal{N}_i^r$ represents the set of the neighbor indices of node v_i under relation r, where v_i plays the role of the destination node.

Each node of the graph has an input feature $\mathbf{x}_i$, which can be regarded as the initial hidden state $\mathbf{h}_i^{(0)}$ of the R-GCN. The hidden state of each node in the graph is updated layer by layer with following step:

Sending Message At the l-th layer R-GCN, each edge (v_i, r, v_j) of the graph will send a message from the source node v_i to the destination node v_j. The message is calculated as below:

$$\mathbf{m}_{ij}^{(l)} = \mathbf{W}_r^{(l)} \mathbf{h}_i^{(l-1)}, \tag{1}$$

where r is the relation from v_i to v_j and $\mathbf{W}_r^{(l)}$ is a linear transformation, which is a trainable matrix. Following Equation 1, the scale of the parameter of calculating message is proportional to the number of the node types. To increase the scalability, R-GCN regularizes the message-calculating parameter with the basis decomposition method, which is defined as below:

$$\mathbf{W}_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} \mathbf{V}_b^{(l)}, \tag{2}$$

where B is the basis number, $a_{rb}^{(l)}$ is the coefficient of the basis transformation $\mathbf{V}_b^{(l)}$. For different edge

types, the basis transformations are shared and only the coefficient $a_{rb}^{(l)}$ dependents on r.

Aggregating Message After the message sending process, all the incoming messages of each node will be aggregated. Combined with Equations 1 and 2, R-GCN simply averages these incoming messages as:

$$\mathbf{g}_i^{(l)} = \sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}(\sum_{b=1}^{B}a_{rb}^{(l)}\mathbf{V}_b^{(l)})\mathbf{h}_j^{(l-1)}, \quad (3)$$

where $c_{i,r}$ equals to $|\mathcal{N}_i^r|$.

Updating State After aggregating messages, each node will update its hidden state from $\mathbf{h}_i^{(l-1)}$ to $\mathbf{h}_i^{(l)}$,

$$\mathbf{h}_i^{(l)} = \sigma(\mathbf{g}_i^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l-1)}), \quad (4)$$

where σ is an activation function (i.e., ReLU) and $\mathbf{W}_0^{(l)}$ is a weight matrix. For each layer of R-GCN, the update process can be simply denoted as:

$$\mathbf{Y} = \text{R-GCN}(\mathbf{X}, \mathcal{G}), \quad (5)$$

where $\mathbf{X} = \{\mathbf{h}_i\}_{i=1}^{|\mathcal{G}|}$, $|\mathcal{G}|$ is the number of the nodes and $\mathcal{G}$ is the graph structure.

2.2 Relation-aware Transformer

With the success of the large-scale language models, the transformer architecture has been widely used in natural language process (NLP) tasks to encode the sequence $X = [\mathbf{x}_i]_{i=1}^n$ with the *self-attention* mechanism. As introduced in Vaswani et al. (2017), a transformer is stacked by self-attention layers, where each layer transforms $\mathbf{x}_i$ to $\mathbf{y}_i$ with H heads as follows:

$$e_{ij}^{(h)} = \frac{\mathbf{x}_i\mathbf{W}_Q^{(h)}(\mathbf{x}_j\mathbf{W}_K^{(h)})^\top}{\sqrt{d_z/H}}, \quad (6)$$

$$\alpha_{ij}^{(h)} = \underset{j}{\text{softmax}}\{e_{ij}^{(h)}\}, \quad (7)$$

$$\mathbf{z}_i^{(h)} = \sum_{j=1}^{n}\alpha_{ij}^{(h)}\mathbf{x}_j\mathbf{W}_V^{(h)}, \quad (8)$$

$$\mathbf{z}_i = \text{Concat}(\mathbf{z}_i^{(1)}, \ldots, \mathbf{z}_i^{(H)}), \quad (9)$$

$$\bar{\mathbf{y}}_i = \text{LayerNorm}(\mathbf{x}_i + \mathbf{z}_i), \quad (10)$$

$$\mathbf{y}_i = \text{LayerNorm}(\bar{\mathbf{y}}_i + \text{FC}(\text{ReLU}(\text{FC}(\bar{\mathbf{y}}_i)))), \quad (11)$$

where h is the head index, d_z is the hidden dimension of $\mathbf{z}_i^{(h)}$, $\alpha_{ij}^{(h)}$ is attention probability, Concat denotes the concatenation operation, LayerNorm is layer normalization (Ba et al., 2016) and FC is a full connected layer. The transformer function can be simply denoted as:

$$\mathbf{Y} = \text{Transformer}(\mathbf{X}), \quad (12)$$

where $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{|X|}$ and $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{|X|}$ and $|X|$ is the sequence length.

Relation-aware transformer (RAT) (Shaw et al., 2018) is an important extension of the traditional transformer, which regards the input sequence as a labeled, directed, fully-connected graph. The pairwise relations between input elements are considered in RAT. RAT incorporates the relation information in Equation 6 and Equation 8. The edge from element $\mathbf{x}_i$ to element $\mathbf{x}_j$ is represented by vectors $\mathbf{r}_{ij,K}$ and $\mathbf{r}_{ij,V}$, which are represented as biases incorporated in self-attention layer, as follows:

$$e_{ij}^{(h)} = \frac{\mathbf{x}_i\mathbf{W}_Q^{(h)}(\mathbf{x}_j\mathbf{W}_K^{(h)} + \mathbf{r}_{ij,K})^\top}{\sqrt{d_z/H}}, \quad (13)$$

$$\alpha_{ij}^{(h)} = \underset{j}{\text{softmax}}\{e_{ij}^{(h)}\}, \quad (14)$$

$$\mathbf{z}_i^{(h)} = \sum_{j=1}^{n}\alpha_{ij}^{(h)}(\mathbf{x}_j\mathbf{W}_V^{(h)} + \mathbf{r}_{ij,V}), \quad (15)$$

where $\mathbf{r}_{ij,K}$ and $\mathbf{r}_{ij,V}$ are shared in different attention heads. For each layer of RAT, the update process can be simply represented as:

$$\mathbf{Y} = \text{RAT}(\mathbf{X}, \mathcal{R}), \quad (16)$$

where $\mathcal{R} = \{R\}_{i=1,j=1}^{|X|,|X|}$ is the relation matrix among the sequence tokens and R_{ij} means the relation type between i-th token and j-th token.

Both R-GCN and RAT have been successfully applied into Text-to-SQL tasks. Bogin et al. (2019a) utilizes R-GCN to encode the structure of the semantic schema to get the global representations of the nodes. Wang et al. (2020) considers not only the schema structure but also the schema link between the schema and the NL question. They proposed a unified framework to model the representation of the schema and the question with RAT. However, they do not explicitly explore the impact of the domain information. In the next section, we will introduce our proposed GPNN and explain how to use GPNN to get the abstract representation of the schema and the question.

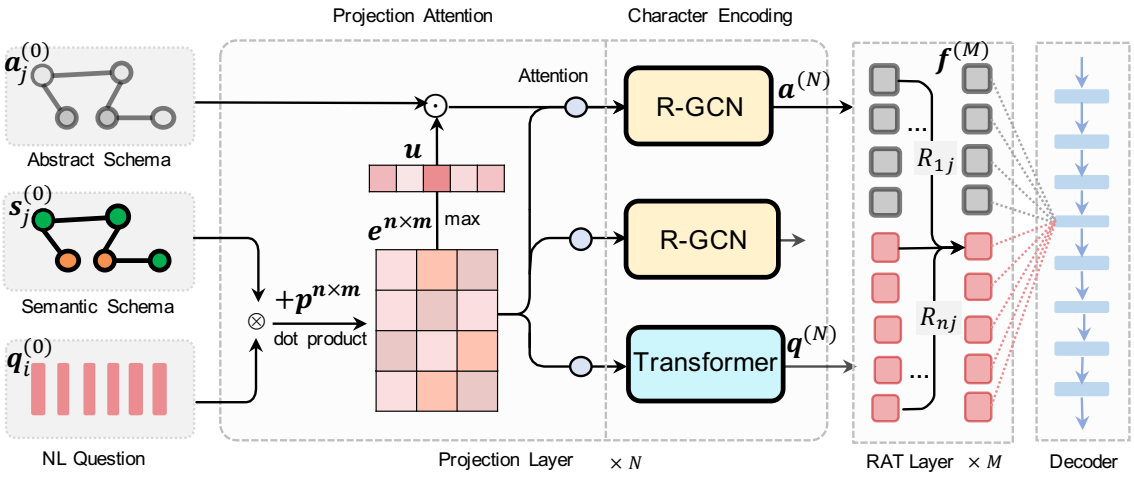

Figure 2: The structure of our proposed ShadowGNN. ShadowGNN has three kinds of input: abstract schema, semantic schema, and natural language question. The encoder of ShadowGNN consists of two module: a stack of graph projection layers and a stack of relation-aware self-attention layers. To clarify the introduction of GPNN layer, we ignore the pretrained model RoBERTa in the figure.

3 Method

Text-to-SQL models take the NL questions $Q = \{q_i\}_{i=1}^{n}$ and the semantic schema $G = \{s_j\}_{j=1}^{m}$ as the input. In our proposed ShadowGNN, the encoder has been decomposed into two modules. The first module filters the specific domain information with a well-designed graph projection neural network (GPNN). The second module leverages relation-aware transformer to further get unified representations of question and schema. This two-phase encoder of ShadowGNN simulates the inference process of a human when translating a question to a SQL query under cross-domain setup: abstracting and inferring.

3.1 Graph Projection Neural Network

In this subsection, we introduce the structure of GPNN. As we discussed, the schema consists of database structure information and domain semantic information. GPNN looks at the schema from these two perspectives. Thus, GPNN has three kinds of inputs, abstract schema, semantic schema, and NL question. The input of the abstract schema is the type (table or column) of the schema nodes without any domain information, which can be regarded as a projection of semantic schema. Each node in the abstract schema is represented by a one-hot vector $\mathbf{a}_j^{(0)}$, which has two dimensions.

For semantic schema and NL question, we first use pretrained language model RoBERTa (Liu et al., 2019) to initialize their representations. We directly concatenate NL question and semantic schema together, which formats as " [CLS] question [SEP] tables columns [SEP]". Each node name in the semantic schema may be tokenized into several sub-tokens or sub-words. We add an average pooling layer behind the final layer of the RoBERTa to align the sub-tokens to the corresponding node. We indicate the initial representation of NL question and semantic schema as $\mathbf{q}_i^{(0)}$ and $\mathbf{s}_j^{(0)}$.

The main motivation of GPNN is to abstract the representations of question and schema. The abstract schema has been distilled from the semantic schema. The essential challenge lies on abstracting question representation. There are two separate operations in each GPNN layer: **Projection Attention** and **Character Encoding**. The projection attention of GPNN is to take the semantic schema as the bridge, where question updates its representation using abstract schema but attention information is calculated with the vectors of semantic schema. The character encoding is to augment the structure representation of the question sentence and the schema graph.

Projection Attention In each GPNN layer, there is first an attention operation between NL question and semantic schema, as follows:

$$e_{ij} = \mathbf{q}_i^{(l)} \mathbf{W}_Q^{(l)} (\mathbf{s}_j^{(l)} \mathbf{W}_K^{(l)})^\top, \quad (17)$$

$$\alpha_{ij} = \underset{j}{\text{softmax}}\{e_{ij}\}, \quad (18)$$

where $\mathbf{W}_Q^{(l)}$ and $\mathbf{W}_K^{(l)}$ are trainable parameters at l-th projection layer and $\mathbf{e}^{n\times m} = \{e_{ij}\}_{i=1,j=1}^{n,m}$ is the matrix of the weight score. n is the length of the question, and m is the number of schema nodes.

Before operating attention mechanism, inspired by (Bogin et al., 2019a), we first calculate the maximum values $\mathbf{u}$ of attention probability,

$$\mathbf{u}_j = \max_i \{\alpha_{ij}\}, \tag{19}$$

where the physical meaning of $\mathbf{u}_j$ is the most probability that the j-th component of the schema is mentioned by the question. We distinct the initial representation of the abstract schema by multiplying $\mathbf{u}$ on l-th layer abstract schema representation $\mathbf{a}^{(l)}$ in element-wise way, $\hat{\mathbf{a}}^{(l)} = \mathbf{a}^{(l)} \cdot \mathbf{u}$.

When updating the question representation, we take the representation of augmented abstract schema $\hat{\mathbf{a}}^{(l)}$ as key value of attention at l-th layer of GPNN,

$$\mathbf{b}_i = \sum_{j=1}^{m} \alpha_{ij} \hat{\mathbf{a}}_j^{(l)} \mathbf{W}_V^{(l)}, \tag{20}$$

$$\bar{\mathbf{q}}_i^{(l+1)} = \text{gate}(\mathbf{b}_i) * \mathbf{b}_i + (1 - \text{gate}(\mathbf{b}_i)) * \mathbf{q}_i^{(l)}, \tag{21}$$

where $\text{gate}(\cdot) = \text{sigmoid}(\text{Linear}(\cdot))$ and $\mathbf{W}_V^{(l)}$ is trainable weight. When updating semantic schema, we take the transpose of the above attention matrix as the attention from schema to question,

$$\hat{\mathbf{e}}^{m \times n} = (\mathbf{e}^{n \times m})^\top = \{\hat{e}_{ij}\}_{i=1,j=1}^{m,n}. \tag{22}$$

Similar to the update process of question from Equation 17- 21, the update process of semantic schema $\bar{\mathbf{s}}^{(l+1)}$ takes $\hat{\mathbf{e}}^{m \times n}$ as attention score and $\mathbf{q}^{(l)}$ as attention value. We can see that we only use the augmented abstract schema to update the question representation. In this way, the domain information contained in question representation will be removed. The update process of the abstract schema $\bar{\mathbf{a}}^{(l+1)}$ is the same as the semantic schema updating, where their attention weight $\hat{\mathbf{e}}^{m \times n}$ on the question $\mathbf{q}^{(l)}$ is shared. Noting that the input of attention operation for the abstract schema is the augmented abstract representation $\hat{\mathbf{a}}$.

Character Encoding We have used the projection attention mechanism to update the three kinds of vectors. Then, we combine the characters of schema and NL question and continue encoding schema and question with $\text{R-GCN}(\cdot)$ function and $\text{Transformer}(\cdot)$ function respectively, as shown in Fig. 2.,

$$\mathbf{a}^{(l+1)} = \text{R-GCN}(\bar{\mathbf{a}}^{(l+1)}, G), \tag{23}$$

$$\mathbf{s}^{(l+1)} = \text{R-GCN}(\bar{\mathbf{s}}^{(l+1)}, G), \tag{24}$$

$$\mathbf{q}^{(l+1)} = \text{Transformer}(\bar{\mathbf{q}}^{(l+1)}). \tag{25}$$

Until now, the projection layer has been introduced. Graph projection neural network (GPNN) is a stack of the projection layers. After GPNN module, we get the abstract representation of the schema and the question, indicated as $\mathbf{a}^{(N)}$ and $\mathbf{q}^{(N)}$.

3.2 Schema Linking

The schema linking (Guo et al., 2019; Lei et al., 2020) can be regarded as a kind of prior knowledge, where the related representation between question and schema will be tagged according to the matching degree. There are 7 tags in total: Table Exact Match, Table Partial Match, Column Exact Match, Column Partial Match, Column Value Exact Match, Column Value Partial Match, and No Match. The column values store in the databases. As the above description, the schema linking can be represented as $\mathcal{D} = \{d_{ij}\}_{i=1,j=1}^{n,m}$, which d_{ij} means the match degree between i-th word of question and j-th node name of schema. To integrate the schema linking information into GPNN module, we calculate a prior attention score $\mathbf{p}^{n \times m} = \text{Linear}(\text{Embedding}(\mathbf{d}_{ij}))$, where $\mathbf{d}_{ij}$ is the one-hot representation of match type d_{ij}. The attention score in Equation 17 is updated as following:

$$e_{ij} = \mathbf{q}_i^{(l)} \mathbf{W}_Q^{(l)} (\mathbf{s}_j^{(l)} \mathbf{W}_K^{(l)})^\top + p_{ij}, \tag{26}$$

where p_{ij} is the prior score from $\mathbf{p}^{n \times m}$. The prior attention score is shared among all the GPNN layers.

3.3 RAT

If we split the schema into the tables and the columns, there are three kinds of inputs: question, table, column. RATSQL (Wang et al., 2020) leverages the relation-aware transformer to unify the representation of the three inputs. RATSQL defines all the relations $\mathcal{R} = \{R_{ij}\}_{i=1,j=1}^{(n+m),(n+m)}$ among the three inputs and uses the $\text{RAT}(\cdot)$ function to get unified representation of question and schema. The details of the defined relations among three components are introduced in RATSQL (Wang et al., 2020). The schema linking relations are the subset of $\mathcal{R}$. In this paper, we leverage the RAT to further unify the abstract representation of question $\mathbf{q}^{(N)}$ and schema $\mathbf{a}^{(N)}$, which is generated by previous GPNN module. We concatenate sentence sequence $\mathbf{q}^{(N)}$ and schema sequence $\mathbf{a}^{(N)}$ together into a longer sequence representation, which is the initial input of RAT module. After RAT module, the final

unified representation of question and schema is indicated as:

$$\mathbf{f}^{(M)} = \mathrm{RAT}(\mathrm{concat}(\mathbf{q}^{(N)}, \mathbf{a}^{(N)}), \mathcal{R}). \quad (27)$$

3.4 Decoder with SemQL Grammar

To effectively constrain the search space during synthesis, IRNet (Guo et al., 2019) designed a context-free SemQL grammar as the intermediate representation between NL question and SQL, which is essentially an abstract syntax tree (AST). SemQL recovers the tree nature of SQL. To simplify the grammar tree, SemQL in IRNet did not cover all the keywords of SQL. For example, the columns contained in GROUPBY clause can be inferred from SELECT clause or the primary key of a table where an aggregate function is applied to one of its columns. In our system, we improve the SemQL grammar, where each keyword in SQL sentence is corresponded to a SemQL node. During the training process, the labeled SQL needs to be transferred into an AST. During the evaluation process, the AST needs to recovered as the corresponding SQL. The recover success rate means the rate that the recovered SQL totally equals to labeled SQL. Our improved grammar raises the recover success rate from 89.6% to 99.9% tested on dev set.

We leverage the coarse-to-fine approach (Dong and Lapata, 2018) to decompose the decoding process of a SemQL query into two stages, which is similar with IRNet. The first stage is to predict a skeleton of the SemQL query with skeleton decoder. Then, a detail decoder fills in the missing details in the skeleton by selecting columns and tables.

4 Experiments

In this section, we evaluate the effectiveness of our proposed ShadowGNN than other strong baselines. We further conduct the experiments with limited annotated training data to validate the generalization capability of the proposed ShadowGNN. Finally, we ablate other designed choices to understand their contributions.

4.1 Experiment Setup

Dataset & Metrics We conduct the experiments on the Spider (Yu et al., 2018), which is a large-scale, complex and cross-domain Text-to-SQL benchmark. The databases on the Spider are split into 146 training, 20 development and 40 test. The human-labeled question-SQL query pairs are divided into 8625/1034/2147 for train/development/test. The test set is not available for the public, like all the competition challenges. We report the results with the same metrics as (Yu et al., 2018): exact match accuracy and component match accuracy.

Approaches	Dev.	Test
Global-GNN (Bogin et al., 2019b)	52.7%	47.4%
R-GCN + Bertrand-DR (Kelkar et al., 2020)	57.9%	54.6%
IRNet v2 (Guo et al., 2019)	63.9%	55.0%
RATSQL v3 + BERT-large (Wang et al., 2020)	69.7%	65.6%
RATSQL♣ + RoBERTa-large	70.2%	64.0%
GPNN + RoBERTa-large	69.9%	65.7%
ShadowGNN + RoBERTa-large	**72.3%**	**66.1%**

Table 1: The exact match accuracy on the development set and test set. ♣ means the model is implemented by us, where the only difference is the encoder part compared with the proposed ShadowGNN model.

Baselines The main contribution of this paper lies on the encoder of the Text-to-SQL model. As for the decoder of our evaluated models, we improve the SemQL grammar of the IRNet (Guo et al., 2019), where the recover success rate raises from 89.6% to 99.9%. The SQL query first is represented by an abstract syntax tree (AST) following the well-designed grammar (Lin et al., 2019). Then, the AST is flattened as a sequence (named SemQL query) by the deep-first search (DFS) method. During decoding, it is still predicted one by one with LSTM decoder. We also leverage the coarse-to-fine approach to the decoder as IRNet. A skeleton decoder first outputs a skeleton of the SemQL query. Then, a detail decoder fills in the missing details in the skeleton by selecting columns and tables. R-GCN (Bogin et al., 2019a; Kelkar et al., 2020) and RATSQL (Wang et al., 2020) are two other strong baselines, which improve the representation ability of the encoder.

Implementations We implement ShadowGNN and our baseline approaches with PyTorch (Paszke et al., 2019). We use the pretrained models RoBERTa from PyTorch transformer repository (Wolf et al., 2019). We use Adam with default hyperparameters for optimization. The learning rate is set to 2e-4, but there is 0.1 weight decay for the learning rate of pretrained model. The hidden sizes of GPNN layer and RAT layer are set to 512. The dropout rate is 0.3. Batch size is set to 16. The layers of GPNN and RAT in ShadowGNN encoder are set to 4.

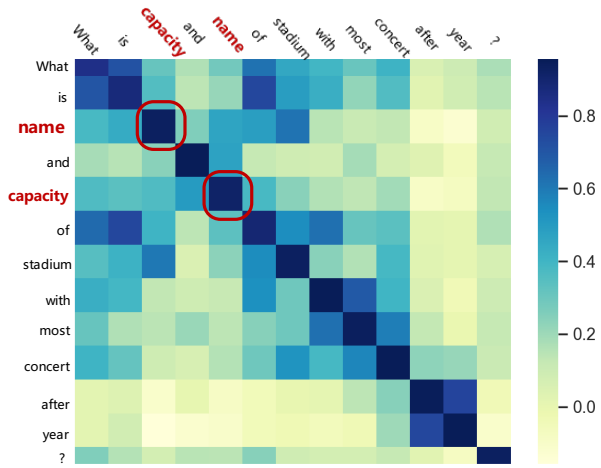

Figure 3: The cosine similarity of two questions. The positions of “name" and ’capacity’ in the two questions are exchanged.

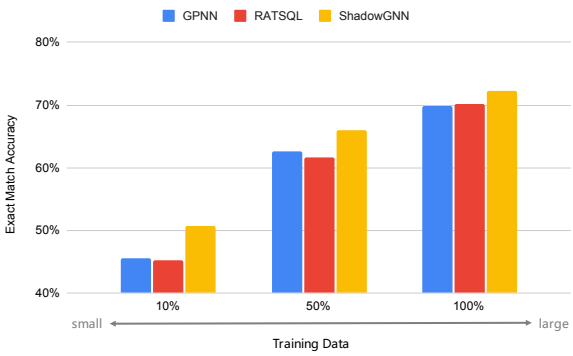

Figure 4: The exact match accuracy of GPNN, RATSQL and ShadowGNN on the limited training datasets. The limited training datasets are randomly sampled from fully training dataset with 10%, 50% and 100% sampling probability.

4.2 Experimental Results

To fairly compared with our proposed ShadowGNN, we implement RATSQL (Wang et al., 2020) with the same coarse-to-fine decoder and RoBERTa augmentation of ShadowGNN model. We also report the performance of GPNN encoder on test set. The detail implementations of these two baselines show as following:

- **RATSQL♣** RATSQL model replaces the four projection layers with another four relation-aware self-attention layers. There are totally eight relation-aware self-attention layers in the encoder, which is consistent with orignal RAT-SQL setup (Wang et al., 2020).

- **GPNN** Compared with ShadowGNN, GPNN model directly removes the relation-aware transformer. There are only four projection layers in the encoder, which can get better performance than eight layers.

Table 1 presents the exact match accuracy of the novel models on development set and test set. Compared with the state-of-the-art RATSQL, our proposed ShadowGNN gets absolute 2.6% and 0.5% improvement on development set and test set with RoBERTa augmentation. Compared with our implemented RATSQL♣, ShadowGNN can still stay ahead, which has absolute 2.1% and 2.1% improvement on development set and test set. ShadowGNN improved the encoder and SemQL grammar of IRNet obtains absolute 11.1% accuracy gain on test set. As shown in Table 1, our proposed pure GPNN model achieves comparable performance with state-of-the-art approach on test set. Compared with other GNN-based models (Global-GNN and R-GCN), GPNN gets over 10% improvement on development set and test set. To the best of our knowledge, our proposed GPNN gets the best performance on Spider dataset among all the GNN-based models.

4.3 Generalization Capability

We design an experiment to validate the effectiveness of the graph projection neural network (GPNN). Considering a question “What is name and capacity of stadium with most concert after year ?", which has been preprocessed, “name" and “capacity" are column names. We exchange their positions and calculate the cosine similarity with the representations of the final GPNN layer in ShadowGNN model. Interestingly, we find that “name" has the most similar with “capacity", as shown in Figure 3. The semantic meaning of the two column names seems to be removed that the representations of the two column names only dependent on the existed positions. It indicates the GPNN can get the abstract representation of the question.

To further validate the generalization ability of our proposed ShadowGNN, we conduct the experiments on the limited annotated training datasets. The limited training datasets are sampled from fully training dataset with 10%, 50% and 100% sampling rate. As shown in Figure 4, there is a large performance gap between RATSQL and ShadowGNN, when the annotated data is extremely limited only occupied 10% of the fully training dataset. Shad-

Approaches	Easy	Medium	Hard	Extra Hard	All
R-GCN (Kelkar et al., 2020)	70.4%	54.1%	35.6%	28.2%	50.7%
R-GCN♣	78.9%	63.2%	46.6%	29.8%	58.7%
R-GCN+RAT	85.0%	70.9%	56.3%	32.7%	65.6%
GPNN	**87.5%**	74.9%	59.2%	41.6%	69.9%
RATSQL♣	87.1%	74.9%	57.5%	**46.4%**	70.2%
ShadowGNN	**87.5%**	**78.0%**	**61.5%**	45.8%	**72.3%**

Table 2: The match accuracy of the ablation methods at four hardness levels on development set. ♣ means the model is implemented by us.

owGNN outperforms RATSQL and GPNN with over 5% accuracy rate on development set. Under this limited training data setup, we find an interesting phenomenon that the convergence speed of ShadowGNN is much faster than the other two models. As described in Section 3, the two-phase encoder of ShadowGNN simulates the inference process of a human when translating a question to a SQL query: abstracting and inferring. The experiments on limited annotated training datasets show these two phases are both necessary, which not only can improve the performance but also speed up the convergence.

4.4 Ablation Studies

We conduct ablation studies to analyze the contributions of well-designed graph projection neural network (GPNN). Except RATSQL and GPNN models, we implement other two ablation models: R-GCN and R-GCN+RAT. First, we introduce the implementations of the ablation models.

- **R-GCN♣** We directly remove the projection part in the GPNN. When updating the question representation, we use the representation of semantic schema as attention value instead of abstract representation.
- **R-GCN+RAT** In this model, there are four R-GCN layers and four relation-aware self-attention layers. To be comparable, the initial input of R-GCN is the sum of semantic schema and abstract schema.

The decoder parts of these four ablation models are the same as the decoder of ShadowGNN. We present the accuracy of the ablation models at the four hardness levels on the development set, which is defined in (Yu et al., 2018). As shown in Table 2, ShadowGNN can get the best performance at three hardness levels. Compared with R-GCN (Kelkar et al., 2020), our implemented R-GCN based on SemQL grammar gets higher performance. Compared with R-GCN+RAT model, ShadowGNN still gets the better performance, where the initial input information is absolutely the same. It denotes that it is necessary and effective to abstract the representation of question and schema explicitly.

5 Related Work

Text-to-SQL Recent models evaluated on Spider have pointed out several interesting directions for Text-to-SQL research. An AST-based decoder (Yin and Neubig, 2017) was first proposed for generating general-purpose programming languages. IR-Net (Guo et al., 2019) used a similar AST-based decoder to decode a more abstracted intermediate representation (IR), which is then transformed into an SQL query. RAT-SQL (Wang et al., 2020) introduced a relation-aware transformer encoder to improve the joint encoding of question and schema, and reached the best performance on the Spider (Yu et al., 2018) dataset. BRIDGE (Lin et al., 2020) leverages the database content to augment the schema representation. RYANSQL (Choi et al., 2020) formulates the Text-to-SQL task as a slot-filling task to predict each SELECT statement. EditSQL (Zhang et al., 2019), IGSQL (Cai and Wan, 2020) and R^{2}SQL (Hui et al.) consider the dialogue context during translating the utterance into SQL query. GAZP (Zhong et al., 2020) proposes a zero-shot method to adapt an existing semantic parser to new domains. PIIA (Li et al., 2020) proposes a human-in-loop method to enhance Text-to-SQL performance.

Graph Neural Network Graph neural network (GNN) (Li et al., 2015) has been widely applied in various NLP tasks, such as text classification (Chen et al., 2020b; Lyu et al., 2021), text generation (Zhao et al., 2020), dialogue state tracking (Chen et al., 2020a; Zhu et al., 2020) and dialogue policy (Chen et al., 2018a,b, 2019, 2020c,d). It also has been used to encode the schema in a more structured way. Prior work (Bogin et al., 2019a) constructed a directed graph of foreign key relations in the schema and then got the corresponding schema representation with GNN. Global-GNN (Bogin et al., 2019a) also employed a GNN to derive the representation of the schema and softly select a set of schema nodes that are likely to appear in the output query. Then, it discriminatively re-ranks the top-K queries output from a generative decoder. We proposed Graph Projection Neural Network (GPNN), which is able to extract the abstract representation of the NL question and the

semantic schema.

Generalization Capability To improve the compositional generalization of a sequence-to-sequence model, SCAN (Lake and Baroni, 2018) (**S**implified version of the **C**omm**AI** **N**avigation tasks) dataset has been published. SCAN task requires models to generalize knowledge gained about the other primitive verbs ("walk", "run" and "look") to the unseen verb "jump". Russin et al. (2019) separates syntax from semantics in the question representation, where the attention weight is calculated based on syntax vectors but the hidden representation of the decoder is the weight sum of the semantic vectors. Different from this work, we look at the semi-structured schema from two perspectives (schema structure and schema semantics). Our proposed GPNN aims to use the schema semantics as the bridge to get abstract representation of the question and schema.

6 Conclusion

In this paper, we propose a graph project neural network (GPNN) to abstract the representation of question and schema with simple attention way. We further unify the abstract representation of question and schema outputted from GPNN with relative-aware transformer (RAT). The experiments demonstrate that our proposed ShadowGNN can get excellent performance on the challenging Text-to-SQL task. Especially when the annotated training data is limited, our proposed ShadowGNN gets more performance gain on exact match accuracy and convergence speed. The ablation studies further indicate the effectiveness of our proposed GPNN. Recently, we notice that some Text2SQL-specific pretrained models have been proposed, e.g., TaBERT (Yin et al., 2020) and GraPPa (Yu et al., 2020). In future work, we will evaluate our proposed ShadowGNN with these adaptive pretrained models.

Acknowledgements

We thank the anonymous reviewers for their thoughtful comments. This work has been supported by No. SKLMCPTS2020003 Project.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-sql parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Global reasoning over database structures for text-to-sql parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3650–3655.

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.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of ACL*, pages 51–64, Florence, Italy.

Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. 2020. Unsupervised dual paraphrasing for two-stage semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6806–6817.

Lu Chen, Cheng Chang, Zhi Chen, Bowen Tan, Milica Gašić, and Kai Yu. 2018a. Policy adaptation for deep reinforcement learning-based dialogue management. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 6074–6078. IEEE.

Lu Chen, Zhi Chen, Bowen Tan, Sishan Long, Milica Gašić, and Kai Yu. 2019. Agentgraph: Toward universal dialogue management with structured deep reinforcement learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9):1378–1391.

Lu Chen, Boer Lyu, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020a. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *AAAI*, pages 7521–7528.

Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. 2018b. Structured dialogue policy with graph neural networks. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1257–1268.

Lu Chen, Yanbin Zhao, Boer Lyu, Lesheng Jin, Zhi Chen, Su Zhu, and Kai Yu. 2020b. Neural graph matching networks for chinese short text matching.

In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6152–6158.

Zhi Chen, Lu Chen, Xiaoyuan Liu, and Kai Yu. 2020c. Distributed structured actor-critic reinforcement learning for universal dialogue management. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2400–2411.

Zhi Chen, Xiaoyuan Liu, Lu Chen, and Kai Yu. 2020d. Structured hierarchical dialogue policy with graph neural networks. *arXiv preprint arXiv:2009.10355*.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *arXiv preprint arXiv:2004.03125*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742.

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.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. Dynamic hybrid relation network for cross-domain context-dependent semantic parsing. *arXiv preprint arXiv:2101.01686*.

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

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

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Yuntao Li, Bei Chen, Qian Liu, Yan Gao, Jian-Guang Lou, Yan Zhang, and Dongmei Zhang. 2020. "what do you mean by that?"-a parser-independent interactive approach for enhancing text-to-sql. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6913–6922.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. 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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Boer Lyu, Lu Chen, Su Zhu, and Kai Yu. 2021. Let: Linguistic knowledge enhanced graph transformer for chinese short text matching. *arXiv preprint arXiv:2102.12671*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

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.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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.

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 5341–5352.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020. Line graph enhanced amr-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 732–741.

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. *CoRR*, abs/1709.00103.

Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020. Efficient context and schema fusion networks for multi-domain dialogue state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 766–781.