Abstract

The task of converting a natural language question into an executable SQL query, known as text-to-SQL, is an important branch of semantic parsing. The state-of-the-art graph-based encoder has been successfully used in this task but does not model the question syntax well. In this paper, we propose S^2SQL, injecting Syntax to question-Schema interaction graph encoder for Text-to-SQL parsers, which effectively leverages the syntactic dependency information of questions in text-to-SQL to improve the performance. We also employ the decoupling constraint to induce diverse relational edge embedding, which further improves the network’s performance. Experiments on the Spider and robustness setting Spider-Syn demonstrate that the proposed approach outperforms all existing methods when pre-training models are used, resulting in a performance ranks first on the Spider leaderboard.

1 Introduction

Relational databases are ubiquitous and store a great amount of structured information. The interaction with databases often requires expertise on writing structured code like SQL, which is not friendly for users who are not proficient in query languages. Text-to-SQL aims to automatically translate natural language questions into executable SQL statements (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007; Berant et al., 2013; Li and Jagadish, 2014; Yaghmazadeh et al., 2017; Iyer et al., 2017).

Recently, a large-scale, multi-table, realistic text-to-SQL benchmark, Spider (Yu et al., 2018), has been released. The most effective and popular encoder architecture on Spider is the question-schema interaction graph (Wang et al., 2020). Built on that, many state-of-the-art models have been further developed (Chen et al., 2021; Cao et al., 2021). It

entangled database schema information, and uses some pre-defined relationships to carve out the interaction between them. However, we observed that the current graph-based model yet has two major limitations.

Syntactic Modelling. Jointly modeling syntax and semantics is a core problem in NLP. In the paradigm of deep learning, the role of syntax should be better understood for tasks in which syntax is a central feature (Ge and Mooney, 2005; Michalon et al., 2016; Zhang et al., 2019b; Zanzotto et al., 2020), including the text-to-SQL task. For example, Figure 1 shows that the baseline model can learn the correct linking among \texttt{date}, \texttt{id} and \texttt{transcript} between the question and schema, but fail to identify that \texttt{id} should also be included in the \texttt{SELECT} clause. On the other hand, with the help of the dependency tree, \texttt{date} and \texttt{id} are close to each other and thus should appear in the \texttt{SELECT} clause simultaneously. However, almost all available approaches treat the language question as a sequence, and syntactic information is ignored in neural network-based text-to-SQL models.

Entangled Edge Embedding. The question-schema interaction graph pre-defines a series of edges, and models them as learnable embeddings. These embeddings should be diverse by nature because each of them represents a different type of relations and has a different meaning. Previous
work (Brock et al., 2019; Zhang et al., 2020) has proved that the learnable embeddings are easy to be entangled and do not satisfy the diversity objective. In this paper, we propose S\textsuperscript{2}SQL, injecting Syntax to question-Schema graph encoder for Text-to-SQL parsers. S\textsuperscript{2}SQL models the syntactic labels from a syntactic dependency tree as additional edge embeddings. Motivated by the belief that if the structure of input can be reliably obtained and is a central feature of a task, models that explicitly exploit the structure can benefit. In this paper, we investigate and prove that properly introducing syntactic information into text-to-SQL can further improve the performance, and we provide a detailed analysis on why and how the proposed model works. Built on that, we propose a decoupling constraint to encourage the model to learn a diverse set of relation embeddings, which further enhances the network’s performance. We evaluate our proposed model on the challenging text-to-SQL benchmark Spider (Yu et al., 2018) and robustness setting Spider-Syn (Gan et al., 2021), and demonstrate that S\textsuperscript{2}SQL outperforms other graph-based models consistently when augmented with different pre-training models. In brief, the contributions of our work are three-fold:

- The empirical results show that our approach outperforms all the existing models on the challenging Spider benchmark. (Our processed data and code will be made publicly available to ensure the replicability).

2 The Proposed Method

2.1 Problem Definition

Given a natural language question $Q = \{q_i\}_{i=1}^{|Q|}$ and a schema $S = (C,T)$ consisting of columns $C = \{c_1, c_2, \ldots, c_1, c_2, \ldots\}$ and tables $T = \{t_i\}_{i=1}^{|T|}$, text-to-SQL aims to generate the SQL query $y$ for the question sentence. The de facto method for text-to-SQL employs an encoder-decoder architecture. In this paper we focus on improving the encoder part. For a detailed description of the decoder, please refer to the work of (Wang et al., 2020; Cao et al., 2021).

2.2 Question-Schema Interaction Graph

The joint input questions and schema items can be viewed as a graph $G = (V,R)$, where $V = Q \cup T \cup C$ are nodes of three types $\{Q,T,C\}$. The initial node embeddings matrix $X \in \mathbb{R}^{(|V|+|T|+|C|) \times d}$ is obtained by flattening all question tokens and schema items into a sequence: $[\text{CLS}]q_1q_2\ldots q_{|Q|}\text{[SEP]}t_{10}t_{12}c_{10}c_{12}t_{20}c_{20}\ldots\text{[SEP]}$. The type information $t_{ij}$ or $c_{ij}$ is inserted before each schema item. The edge $R = \{R_{ij}\}_{i=1,j=1}^{|X|,|X|}$ represents the known relation between two elements in the input nodes. The RGAT (relational graph attention transformers)

Figure 2: An overview of S\textsuperscript{2}SQL framework. The S\textsuperscript{2}SQL has three relation types to represent the known syntactic information, linking structure and schema information. There structure are integrated into the question-schema interaction graph by learnable edge embedding with decoupling constraints.
(Shaw et al., 2018; Wang et al., 2020; Cao et al., 2021) models the graph $G$ and computes the output representation $z$ by:

$$
e_{ij} = \frac{x_i W_q^h (x_j W_k^h + v_{ij})^\top}{\sqrt{d_i h}}$$
$$a_{ij} = \text{softmax} \{ e_{ij} \},$$
$$x_i = \sum_{v_j \in \mathcal{N}_i} a_{ij} (x_j W_v^h + v_{ij}),$$

where matrices $W_q, W_k, W_v$ are trainable parameters in self-attention, and $\mathcal{N}_i$ is the receptive field of node $v_i$.

Injecting Syntax The previous work mainly focuses on using linking structure and schema structure in the encoder, in which the structure of the question is ignored. We proposed an effective approach to integrate syntactic dependency information into the graph. A straightforward idea is to treat all dependent types directly as a new edge type. However, the dependency parser will return 55 different dependency types. Such a large number of edge types will significantly increase the number of relational embedding parameters in S2SQL, leading to over-fitting. In order to address this, similar to (Vashishth et al., 2018), we induct dependency types into three abstract relations, Forward, Backward and NONE. In addition, in order to ensure the simplicity of edge embedding, we only consider the first-order relationship. By stacking multi-layer transformers, the model implicitly captures the multi-order relationship without deliberate construction. Specifically, we compute the distance $D(v_i, v_j)$ between any two tokens $v_i$ and $v_j$ from the question. This distance is set to the first-order distance between $v_i$ and $v_j$ if they have the aforementioned dependency types, and 0 otherwise. Based on this first-order distance $D$, we model the syntactic relation $R^{\text{question}}_{ij}$ between tokens $v_i$ and $v_j$ by one of the three previously defined abstract types:

$$R^{\text{question}}_{ij} = \begin{cases} 
\text{Forward}, & \text{if } D(v_i, v_j) = 1 \\
\text{Backward}, & \text{if } D(v_j, v_i) = 1 \\
\text{NONE}, & \text{otherwise}
\end{cases}$$

Overall, as shown in Figure 2, S2SQL models three structures in the graph $G$:

- **Question Structure** $R^{\text{question}}$: relations that represent syntactic dependency between two question tokens.
- **Linking Structure** $R^{\text{linking}}$: relations that align entity references in question to corresponding schema columns or tables.
- **Schema Structure** $R^{\text{schema}}$: relations within a database schema, e.g., foreign key.

Decoupling Constraint. There are $k$ known edges in $R$ and each is represented as a relation embedding. Intuitively, these edge embedding $r = [r_1, r_2, ..., r_k]$ should be diverse because they have different semantic meanings. To avoid the potential risk of coupling edge embedding during optimization, we introduce the orthogonality condition (Brock et al., 2019) to $r$:

$$\mathcal{L}(r) = \left\| r^\top r \odot (1 - I) \right\|_F^2,$$

where 1 denotes a matrix with all elements being set to 1 and $I$ is the identity matrix.

3 Experiments

3.1 Experiment Setup

Datasets and Evaluation Metrics. We conduct experiments on Spider (Yu et al., 2018) and Spider-Syn (Gan et al., 2021). Spider is a large-scale, complex, and cross-domain text-to-SQL benchmark. Spider-Syn is derived from Spider, by replacing their schema-related words with manually selected synonyms that reflect real-world question paraphrases. For evaluation, we followed the official evaluation to report exact match accuracy. Please refer to Appendix A.1 and A.2 for implementation details and baselines.

3.2 Results and Analyses

Overall Performance. We first compare S2SQL with other state-of-the-art models on Spider. As shown in Table 1, we can see that S2SQL outperforms all existing models. Remarkably, the accuracy of S2SQL + RoBERTa on the hidden test set is 67.1%, which is 2.8% higher than the strong baseline RAT + RoBERTa. Similarity, the accuracy of the SoTA model LGESQL + ELECTRA is 72.0% on the hidden test set, and 75.1% on the development set, while S2SQL + ELECTRA can reach 72.1% test and 76.4 development accuracy. Table 2 shows results on the development set for RAT and S2SQL with Table-based pre-training models. We can see that S2SQL outperforms RAT consistently when augmented with different pre-training models, including RoBERTa (Liu et al., 2019), GraPPa.
Table 1: The exact match accuracy on the Spider dev and test set. — indicates that the test set performance cannot be obtained due to the number of submission limit.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev.</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global-GNN (Bogin et al., 2019)</td>
<td>52.7</td>
<td>47.4</td>
</tr>
<tr>
<td>Edit-SQL (Zhang et al., 2019a)</td>
<td>57.6</td>
<td>53.4</td>
</tr>
<tr>
<td>Bertand-DR (Kelkar et al., 2020)</td>
<td>57.9</td>
<td>54.6</td>
</tr>
<tr>
<td>IRNet v2 (Guo et al., 2019)</td>
<td>63.9</td>
<td>55.0</td>
</tr>
<tr>
<td>BRIDGE (Lin et al., 2020)</td>
<td>70.0</td>
<td>65.0</td>
</tr>
<tr>
<td>RATESQL + BERT (Wang et al., 2020)</td>
<td>69.7</td>
<td>65.6</td>
</tr>
<tr>
<td>ShadowGNN + RoBERTa (Chen et al., 2021)</td>
<td>72.3</td>
<td>66.1</td>
</tr>
<tr>
<td>RAT + RoBERTa (Wang et al., 2020)</td>
<td>69.7</td>
<td>64.3</td>
</tr>
<tr>
<td>S2SQL + RoBERTa</td>
<td>71.4</td>
<td>67.1</td>
</tr>
<tr>
<td>w/o DC</td>
<td>70.9</td>
<td>-</td>
</tr>
<tr>
<td>LGESQL + ELECTRA (Cao et al., 2021)</td>
<td>75.1</td>
<td>72.0</td>
</tr>
<tr>
<td>S2SQL + ELECTRA</td>
<td>76.4</td>
<td>72.1</td>
</tr>
<tr>
<td>w/o DC</td>
<td>75.8</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Comparison on S2SQL under the different table-based pre-training models on Spider Dev set.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAT + GraPPa (Yu et al., 2021)</td>
<td>71.5</td>
</tr>
<tr>
<td>S2SQL + GraPPa</td>
<td>73.4</td>
</tr>
<tr>
<td>RAT + GAP (Shi et al., 2021)</td>
<td>71.8</td>
</tr>
<tr>
<td>S2SQL + GAP</td>
<td>72.7</td>
</tr>
</tbody>
</table>

Table 3: The accuracy on the Spider-Syn dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global-GNN (Bogin et al., 2019)</td>
<td>23.6</td>
</tr>
<tr>
<td>IRNet (Guo et al., 2019)</td>
<td>28.4</td>
</tr>
<tr>
<td>RATESQL (Wang et al., 2020)</td>
<td>33.6</td>
</tr>
<tr>
<td>RATESQL + BERT (Wang et al., 2020)</td>
<td>48.2</td>
</tr>
<tr>
<td>RATESQL + Grappa (Wang et al., 2020)</td>
<td>49.1</td>
</tr>
<tr>
<td>S2SQL + Grappa</td>
<td>51.4</td>
</tr>
</tbody>
</table>

4 Related Work

Extensive work has been conducted on improving the encoder and decoder (Yin and Neubig, 2017; Wang et al., 2019; Guo et al., 2019; Choi et al., 2020; Kelkar et al., 2020; Rubin and Berant, 2021) as well as table-based pre-training (Yin et al., 2020; Yu et al., 2021; Deng et al., 2020; Shi et al., 2021; Wang et al., 2021). The main contribution of our paper lies in improving the encoder of the text-to-SQL model. Among the encoder-related work, Guo et al. (2019) introduced the schema linking, which aimed to recognize the columns and the tables mentioned in a question. BRIDGE (Lin et al., 2020) leveraged the database content to augment the schema representation. Bogen et al. (2019) employed GNN to derive the representation of the schema and softly selected a set of schema nodes that are likely to appear in the output query. Then, Chen et al. (2021) propose Shadow-GNN to abstract the representation of question and schema with attention. The most recent approaches RAT (Wang et al., 2020) and LGESQL (Cao et al., 2021) achieved the best performance on Spider through relation-aware transformer encoder. Unlike these works, we investigated the impact of the syntactic structures during the encoding stage.

5 Conclusion

We present syntax-enhanced question-schema graph encoder (S2SQL) that can effectively model syntactic information for text-to-SQL and introduce the decoupling constraint to induce the diverse relation embedding. The proposed model achieves new state-of-the-art performance on the widely used benchmark, Spider and Spider-Syn.
References


Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-SQL models against synonym substitution. In ACL.


Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. VLDB.

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 EMNLP.


Olivier Michalon, Corentin Ribeyre, Marie Candito, and Alexis Nasr. 2016. Deeper syntax for better semantic parsing. In COLING.


Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In NAACL.


Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In ACL.


Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In EMNLP.


John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In AAAI.


Y. Zhang, Rui Wang, and Luo Si. 2019b. Syntax-enhanced self-attention-based semantic role labeling. In EMNLP-IJCNLP.

Appendix

A.1 Implementation Details.

We utilize PyTorch (Paszke et al., 2019) to implement our proposed model. During pre-processing, the input of questions, column names, and table names are tokenized and lemmatized with the Stanford Nature Language Processing toolkit. For a fair comparison with baselines, we configure it with the same set of hyper-parameters, e.g., stacking 8 self-attention layers, setting dropout to 0.1. The position-wise feed-forward network has an inner layer dimension of 1024. Inside the decoder, we use rule embeddings of size 128, node type embeddings of size 64, and a hidden size of 512 inside the LSTM with a dropout of 0.21.

A.2 Baseline Models.

We conduct experiments on Spider and Spider-Syn and compare our method with several baselines including:

- **RYANSQL** (Choi et al., 2020) is a sketch-based slot filling approach which is proposed to synthesize each SELECT statement for its corresponding position.
- **RATSQL + BERT** (Wang et al., 2020) is a relation aware schema encoding model in which the question-schema interaction graph is built by n-gram patterns.
- **ShadowGNN + RoBERTa** (Chen et al., 2021) processes schemas at abstract and semantic levels with domain-independent representations.
- **BRIDGE** (Lin et al., 2020) represents the question and schema in a tagged sequence where a subset of the fields are augmented with cell values mentioned in the question.
- **LGESQL + ELECTRA** (Cao et al., 2021) a line graph enhanced Text-to-SQL model to mine the underlying relational features without constructing metapaths. It was the SOTA model in the Spider leaderboard before ours.

A.3 Qualitative Analysis.

In Table 4, we compare the SQL queries generated by our S²SQL model with those created by the baseline model LGESQL. We notice that S²SQL performs better than the baseline system, especially in the case of question understanding that depends on syntax structure. For example, in the first case where the order and name have NMOD relation, baseline fails to For example, in the first example, both name and tonnage can be linked correctly, but the baseline fails to capture the structure present in name and order, resulting in a generation error, while S²SQL predicts the result well.
Table 4: Case study: some comparisons with baseline (LGESQL) show that S^2SQL can generate more accurate SQL, where syntax column represents useful syntactic information in the generation of S^2SQL.