S^{2}SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers

Anonymous ACL submission

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^{2}SQL**, injecting **S**yntax to question-**S**chema 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

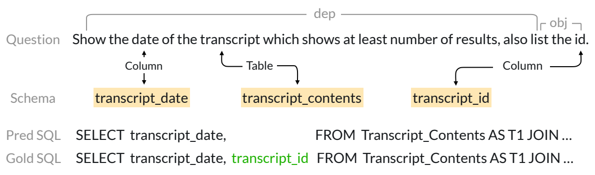
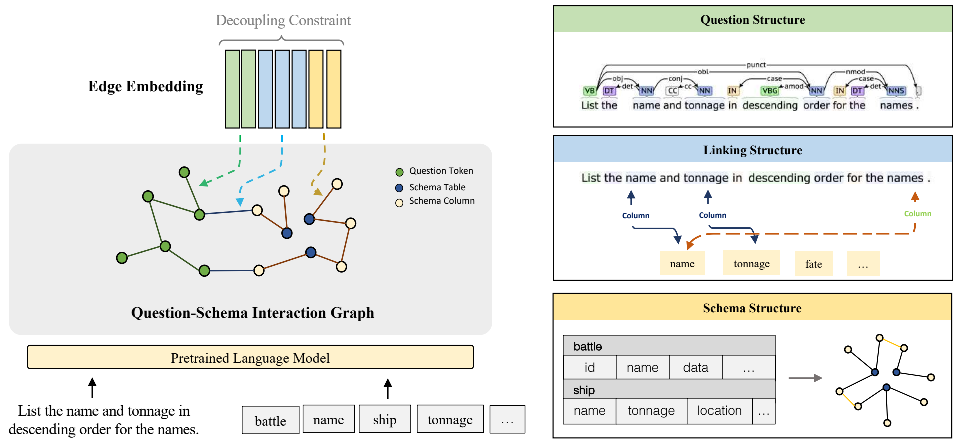

dep
obj
Question
Show the date of the transcript which shows at least number of results, also list the id.
Column
Table
Column
Schema
transcript_date
transcript_contents
transcript_id
Pred SQL
SELECT transcript_date, FROM Transcript_Contents AS T1 JOIN ...
Gold SQL
SELECT transcript_date, transcript_id FROM Transcript_Contents AS T1 JOIN ...

Figure 1: A typical bad case of the current graph-based methods. If the structure of question (dependencies) are not considered, the wrong SQL will be generated even if the linking is correct.

jointly models natural language question and structured 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 `date`, `id` and `transcript` between the question and schema, but fail to identify that `id` should also be included in the `SELECT` clause. On the other hand, with the help of the dependency tree, `date` and `id` are close to each other and thus should appear in the `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



Decoupling Constraint
Edge Embedding
Question Token
Schema Table
Schema Column
Question-Schema Interaction Graph
Pretrained Language Model
List the name and tonnage in descending order for the names.
battle
name
ship
tonnage
…
Question Structure
List the name and tonnage in descending order for the names .
Linking Structure
List the name and tonnage in descending order for the names .
Column
Column
Column
name
tonnage
fate
…
Schema Structure
battle
id
name
data
…
ship
name
tonnage
location
…

Figure 2: An overview of S^{2}SQL framework. The S^{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.

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^{2}SQL**, injecting **S**yntax to question-**S**chema graph encoder for Text-to-**SQL** parsers. S^{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^{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:

- We investigate the importance of syntax in text-to-SQL and propose a novel and strong encoder for cross-domain text-to-SQL, namely S^{2}SQL.
- To induce the diverse edge embedding learning, we introduce the decoupling constraint, which further improves the performance.
- 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 $\mathcal{Q} = \{q_i\}_{i=1}^{|\mathcal{Q}|}$ and a schema $\mathcal{S} = \langle \mathcal{C}, \mathcal{T} \rangle$ consisting of columns $\mathcal{C} = \{c_1^{t_1}, c_2^{t_1}, \cdots, c_1^{t_2}, c_2^{t_2}, \cdots\}$ and tables $\mathcal{T} = \{t_i\}_{i=1}^{|\mathcal{T}|}$, 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 $\mathcal{G} = (\mathcal{V}, \mathcal{R})$, where $\mathcal{V} = \mathcal{Q} \cup \mathcal{T} \cup \mathcal{C}$ are nodes of three types $\{\mathcal{Q}, \mathcal{T}, \mathcal{C}\}$. The initial node embeddings matrix $\mathbf{X} \in \mathbb{R}^{|V^{|\mathcal{Q}|+|\mathcal{T}|+|\mathcal{C}|}| \times d}$ is obtained by flattening all question tokens and schema items into a sequence: $\texttt{[CLS]} q_1 q_2 \cdots q_{|Q|} \texttt{[SEP]} t_{10} t_1 c_{10}^{t_1} c_1^{t_1} c_{20}^{t_1} c_2^{t_1} \cdots t_{20} t_2 c_{10}^{t_2} c_1^{t_2} c_{20}^{t_2} c_2^{t_2} \cdots \texttt{[SEP]}$. The type information t_{i0} or c_{j0}^{ti} is inserted before each schema item. The edge $\mathcal{R} = \{R\}_{i=1,j=1}^{|X|,|X|}$ represents the known relation between two elements in the input nodes. The RGAT (relational graph attention transformers)

(Shaw et al., 2018; Wang et al., 2020; Cao et al., 2021) models the graph $\mathcal{G}$ and computes the output representation $\mathbf{z}$ by:

$$
\begin{aligned}
e_{ij}^{(h)} &= \frac{\mathbf{x}_i \mathbf{W}_q^{(h)} \left(\mathbf{x}_j \mathbf{W}_k^{(h)} + \mathbf{r}_{ij}^{K}\right)^{\top}}{\sqrt{d_z/H}}, \\
\alpha_{ij}^{(h)} &= \operatorname{softmax}\left\{e_{ij}^{(h)}\right\}, \\
\mathbf{z}_i^{(h)} &= \sum_{v_j^n \in \mathcal{N}_i^n} \alpha_{ij}^{(h)} \left(\mathbf{x}_j \mathbf{W}_v^{(h)} + \mathbf{r}_{ij}^{V}\right),
\end{aligned} \tag{1}
$$

where matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ are trainable parameters in self-attention, and $\mathcal{N}_i^n$ is the receptive field of node v_i^n.

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[1] 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 S^{2}SQL, 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 $\boldsymbol{D}(v_i, v_j)$ 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 $\boldsymbol{D}$, we model the syntactic relation R_{ij}^{question} between tokens v_i and v_j by one of the three previously defined abstract types:

$$
\mathcal{R}_{ij}^{\text{question}} = \begin{cases} \texttt{Forward}, & \text{if } \boldsymbol{D}(v_i, v_j) = 1 \\ \texttt{Backward}, & \text{if } \boldsymbol{D}(v_j, v_i) = 1 \\ \texttt{NONE}, & \text{otherwise.} \end{cases} \tag{2}
$$

Overall, as shown in Figure 2, S^{2}SQL models three structures in the graph $\mathcal{G}$:

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

[1]We use SpaCy toolkit to construct syntactic information: https://spacy.io/.

Decoupling Constraint. There are k known edges in $\mathcal{R}$ and each is represented as a relation embedding. Intuitively, these edge embedding $\mathbf{r} = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_k]$ should be diverse because they have different semantic meanings. To avoid the potential risk of coupling edge embedding $\mathbf{r}$ during optimization, we introduce the orthogonality condition (Brock et al., 2019) to $\mathbf{r}$:

$$
\mathcal{L}(\mathbf{r}) = \left\|\mathbf{r}^{\top}\mathbf{r} \odot (\mathbf{1} - I)\right\|_{\mathrm{F}}^2, \tag{3}
$$

where $\mathbf{1}$ 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 S^{2}SQL with other state-of-the-art models on Spider. As shown in Table1, we can see that S^{2}SQL outperforms all existing models. Remarkably, the accuracy of S^{2}SQL + 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 S^{2}SQL + ELECTRA can reach 72.1% test and 76.4 development accuracy. Table 2 shows results on the development set for RAT and S^{2}SQL with Table-based pre-training models. We can see that S^{2}SQL outperforms RAT consistently when augmented with different pre-training models, including RoBERTa (Liu et al., 2019), GraPPa

Model	Dev.	Test
Global-GNN (Bogin et al., 2019)	52.7	47.4
Eidt-SQL (Zhang et al., 2019a)	57.6	53.4
Bertand-DR (Kelkar et al., 2020)	57.9	54.6
IRNet v2 (Guo et al., 2019)	63.9	55.0
BRIDGE (Lin et al., 2020)	70.0	65.0
RYANSQL (Choi et al., 2020)	70.6	60.6
RATSQL + BERT (Wang et al., 2020)	69.7	65.6
ShadowGNN + RoBERTa (Chen et al., 2021)	72.3	66.1
RAT + RoBERTa (Wang et al., 2020)	69.7	64.3
S^{2}SQL + RoBERTa	71.4	**67.1**
w/o DC	70.9	-
LGESQL + ELECTRA (Cao et al., 2021)	75.1	72.0
S^{2}SQL + ELECTRA	**76.4**	**72.1**
w/o DC	75.8	-

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.

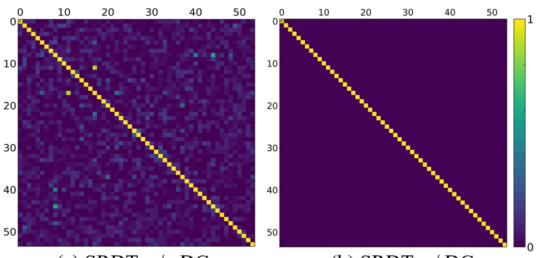


Figure 3: The similarity matrix of different relation embeddings with and without DC. The lighter the color, the higher the similarity (entangled embeddings).

(Yu et al., 2021) and GAP (Shi et al., 2021). In addition, as shown in Table 3, S^{2}SQL demonstrates improvement on the robustness dataset.

Ablation Study. The last row of Table 1 shows that removing the decoupling constraint causes a 0.5% performance drop on the development set. This implies that decoupling entangled embeddings helps to improve the performance. To examine the impact of the decoupling constraint, we visualize the cosine similarity between any two relation embeddings. As shown in Figure 3, we observe that the decoupling constraint eliminates the entangling phenomenon (darker colors) and produces a more diverse set of embeddings.

Qualitative Analysis. To further show the important role of syntactic structures of questions for modelling text-to-SQL as well as the necessity to model the structures explicitly, we conduct qualitative analysis in Appendix A.3.

Model	Dev.
RAT + GraPPa (Yu et al., 2021) †	71.5
S^{2}SQL + GraPPa	**73.4**
RAT + GAP (Shi et al., 2021) †	71.8
S^{2}SQL + GAP	**72.7**

Table 2: Comparison on S^{2}SQL under the different table-based pre-training models on Spider Dev set.

Model	Acc.
Global-GNN (Bogin et al., 2019)	23.6
IRNet (Guo et al., 2019)	28.4
RATSQL (Wang et al., 2020)	33.6
RATSQL + BERT (Wang et al., 2020)	48.2
RATSQL + Grappa (Wang et al., 2020)	49.1
S^{2}SQL + Grappa	**51.4**

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

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. Bogin 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 ShadowGNN 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 (S^{2}SQL) 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

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-sql parsing. In *EMNLP-IJCNLP*.

Andrew Brock, J. Donahue, and K. Simonyan. 2019. Large scale gan training for high fidelity natural image synthesis. In *ICLR*.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations. In *ACL*.

Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021. Shadowgnn: Graph projection neural network for text-to-sql parser. In *NAACL*.

Donghyun Choi, M. 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*, abs/2004.03125.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-sql. *ArXiv*, abs/2010.12773.

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*.

Ruifang Ge and R. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *CoNLL*.

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 *ACL*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *ACL*.

Amol Kelkar, R. Relan, V. Bhardwaj, Saurabh Vaichal, and P. Relan. 2020. Bertrand-dr: Improving text-to-sql using a discriminative re-ranker. *ArXiv*, abs/2002.00557.

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*.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

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

Adam Paszke, S. Gross, Francisco Massa, A. Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Ohad Rubin and Jonathan Berant. 2021. Smbop: Semi-autoregressive bottom-up semantic parsing. In *NAACL-HLT*.

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

Peng Shi, Patrick Ng, Zhi guo Wang, Henghui Zhu, Alexander Hanbo Li, J. Wang, C. D. Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *AAAI*.

Shikhar Vashishth, Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Pratim Talukdar. 2018. Dating documents using graph convolution networks. In *ACL*.

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 *ACL*.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. 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 3774–3785.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2760–2766.

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

Navid Yaghmazadeh, YUEPENG WANG, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *PACMPL*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL*.

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

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Y. Tan, Xinyi Yang, Dragomir Radev, R. Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. In *ICLR*.

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*.

F. M. Zanzotto, Andrea Santilli, Leonardo Ranaldi, Dario Onorati, P. Tommasino, and Francesca Fallucchi. 2020. Kermit: Complementing transformer architectures with encoders of explicit syntactic interpretations. In *EMNLP*.

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

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019a. Editing-based SQL query generation for cross-domain context-dependent questions. In *EMNLP-IJCNLP*.

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

Ziming Zhang, Wenchi Ma, Y. Wu, and Guanghui Wang. 2020. Self-orthogonality module: A network architecture plug-in for learning orthogonal filters. In *WACV*.

A 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 Standford 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 whuich 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 S^{2}SQL model with those created by the baseline model LGESQL. We notice that S^{2}SQL 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 S^{2}SQL predicts the result well.

Question	*List the name and tonnage in alphabetical descending order for the names.*
Baseline	SELECT name, tonnage FROM ship ORDER BY **tonnage** DESC
S^{2}SQL	SELECT name, tonnage FROM ship ORDER BY **name** DESC
Gold	SELECT name, tonnage FROM ship ORDER BY name DESC
Syntax	(name, tonnage, CONJ), (order, names, NMOD)
Question	*What is the total population and average area of countries in the continent of North America whose area is bigger than 3000 ?*
Baseline	SELECT sum(population) , avg(surface_area) FROM country where **surface_area** = "North America" and surface_area > 3000
S^{2}SQL	SELECT sum(population) , avg(surface_area) FROM country where **continent** = "North America" and surface_area > 3000
Gold	SELECT sum(population) , avg(surface_area) FROM country where continent = "North America" and surface_area > 3000
Syntax	(continent, America, NMOD)
Question	*Show the date of the transcript which shows the least number of results, also list the id.*
Baseline	SELECT transcript_date FROM Transcript_Contexts AS T1 JOIN . . .
S^{2}SQL	SELECT transcript_date, **transcript_id** FROM Transcript_Contexts AS T1 JOIN . . .
Gold	SELECT transcript_date, transcript_id FROM Transcript_Contexts AS T1 JOIN . . .
Syntax	(show, list, DEP) (show, date, OBJ) (list, id, OBJ)

Table 4: Case study: some comparisons with baseline (LGESQL) show that S^{2}SQL can generate more accurate SQL, where syntax column represents useful syntactic information in the generation of S^{2}SQL.