# HIE-SQL: History Information Enhanced Network for Context-Dependent Text-to-SQL Semantic Parsing

**Anonymous NAACL submission**


## Abstract

Previous works of context-dependent text-to-SQL semantic parsing leverage context-dependence information either from interaction history utterances or the previous predicted SQL queries but fail in taking advantage of both since of the mismatch between natural language and logic-form SQL. In this work, we propose a **H**istory **I**nformation **E**nhanced text-to-**SQL** model (**HIE-SQL**) to exploit context-dependence information from both history utterances and the last predicted SQL query. In view of the mismatch, we treat natural language and SQL as two modalities and propose a bimodal pre-trained model to bridge the gap between them. Besides, we design a schema-linking graph to enhance connections from utterances and the SQL query to the database schema. We achieve new state-of-the-art results on the two context-dependent text-to-SQL benchmarks, SparC and CoSQL, at the writing time.


## 1 Introduction

Conversation user interfaces to databases have launched a new research hotspot in text-to-SQL semantic parsing (Zhang et al., 2019; Guo et al., 2019; Wang et al., 2020; Lin et al., 2020; Xu et al., 2021; Cao et al., 2021; Hui et al., 2021; Yu et al., 2021b). Most previous works focus on the context-independent text-to-SQL task. Some models (Wang et al., 2020; Scholak et al., 2021) even surprisingly work well on the context-dependent text-to-SQL task by just appending the interaction history utterances to the input. Especially, PICARD (Scholak et al., 2021) achieves state-of-the-art performances both in Spider (Yu et al., 2018), a cross-domain context-independent text-to-SQL benchmark, and CoSQL (Yu et al., 2019a), a cross-domain context-dependent text-to-SQL benchmark, before our work. However, every coin has two sides. That implies underachievement of the exploration of context information in context-dependent

$U_1$: List the name of the teachers and the courses assigned for them to teach.

```
S1 : SELECT T3.Name, T2.Course FROM course_arrange AS T1
     JOIN course AS T2 ON T1.Course_ID = T2.Course_ID
     JOIN teacher AS T3 ON T1.Teacher_ID = T3.Teacher_ID
```

$U_2$: Arrange this list with the teachers name in ascending order.

```
S2 : SELECT T3.Name, T2.Course FROM course_arrange AS T1
     JOIN course AS T2 ON T1.Course_ID = T2.Course_ID
     JOIN teacher AS T3 ON T1.Teacher_ID = T3.Teacher_ID
     ORDER BY T3.Name
```

$U_3$: Include teachers id in the same list.

```
S3 : SELECT T3.Name, T2.Course, T1.teacher_ID FROM course_arrange AS T1
     JOIN course AS T2 ON T1.Course_ID = T2.Course_ID
     JOIN teacher AS T3 ON T1.Teacher_ID = T3.Teacher_ID
     ORDER BY T3.Name
```

Figure 1: An example of context-dependent text-to-SQL interaction in CoSQL where $U_i$ is the utterance of turn $i$ and $S_i$ is the corresponding SQL query for $U_i$. The tokens with red color are the history information that should be considered in later predictions.

text-to-SQL semantic parsing.

Compared with context-independent text-to-SQL semantic parsing, context-dependent text-to-SQL semantic parsing are more challenging since of the various types of context dependence which make models vulnerable to parsing errors. As $R^2$SQL (Hui et al., 2021) considers, different context dependencies between two adjacent utterances require the model to establish dynamic connections between utterances and database schema carefully. Besides long-range dependence is also the case as the prediction of $S_3$ depends on "the name of the teachers and the courses" in $U_1$ in Figure 1. A workable proposition for that is to inherit context information from previous predicted SQL queries(Zhang et al., 2019; Wang et al., 2021). But it is not a piece of cake since of the mismatch between natural language and logic-form SQL. As Liu et al. (2020) conclude, roughly encoding the last predicted SQL query and utterances takes the wooden spoon in their evaluation of 13 existing context modeling methods.

In this paper, we propose HIE-SQL to make full use of both history interactive utterances and the last predicted SQL query. We first treat the logic-form SQL query as another modality with natural language. We present SQLBERT, a bimodal pre-trained model which is able to capture the semantic connection and bridge the gap between SQL and natural language.

Besides, we propose a history information enhanced schema-linking graph to represent the relations among current utterance, interaction history utterances, the last predicted query, and corresponding database schema. Considering it is weird to shift a topic back and forth in an interaction, we assume that the long-range dependence is successive. In that case, we can leverage the long-range dependence from the last predicted query. Therefore, unlike the previous schema-linking graph just with utterances and database schema (Hui et al., 2021), the last predicted query takes part in our graph.

At the time of writing, our model ranks first on both two large-scale cross-domain context-dependent text-to-SQL leaderboards, SparC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a).

## 2 HIE-SQL

### 2.1 Preliminaries

**Task Definition.** Given the current user utterance $u_\tau$, interaction history $h_\tau = [u_1, u_2, ..., u_{\tau-1}]$, the schema $D = \langle T, C\rangle$ of the target database such that the set of tables $T = \{t_1, ..., t_{|T|}\}$ and the set of columns $C = \{c_1, ..., c_{|C|}\}$, our goal is to generate the corresponding SQL query $s_\tau$.

**Model Architecture.** Figure 2 shows the framework of HIE-SQL. We will introduce it in four modules: **Multimodal Encoder**, **SQL Encoder** (SQLBERT), **HIE-Layers**, and **Decoder**.

### 2.2 Multimodal Encoder

Inspired by the efficiency of the works (Kiela et al., 2019; Tsimpoukelli et al., 2021) to solve the multimodal problems, we build an additional pre-trained Encoder named SQLBERT (we will detail it in the following section) to pre-encode SQL query. Then we learn weights $W \in R^{N\times M}$ to project the N-dimensional SQL query embeddings to M-dimensional token input embedding space of the language model:

$$\mathcal{S} = Wf(s_{\tau-1}), \tag{1}$$

where $f(\cdot)$ is the last hidden state output of SQLBERT.

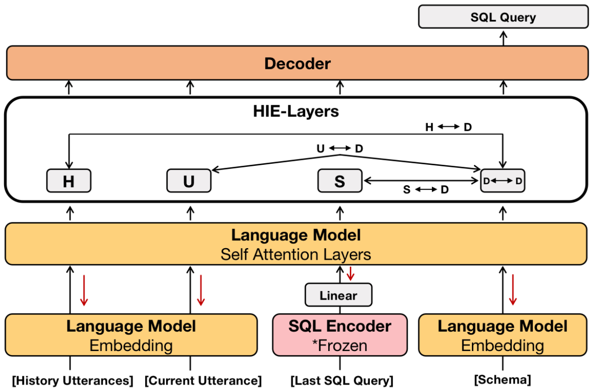


Figure 2: Structure and components of HIE-SQL. The red arrows represent the direction of back propagation during the training stage, witch means parameters of SQL Encoder will not be updated during training. Linear represents one fully connected layer. And we use SQLBERT as the SQL Encoder in the structure.

We arrange the input format of HIE-SQL as $x = ($`[CLS]`$,\mathcal{U},$ `[CLS]`$,\mathcal{S},$ `[SEP]`$,\mathcal{T},$ `[SEP]`$,\mathcal{C})$ in which

$$\begin{aligned} \mathcal{U} &= (u_1, \texttt{[CLS]}, u_2, ..., \texttt{[CLS]}, u_\tau), \\ \mathcal{T} &= (t_1, \texttt{[SEP]}, t_2, ..., \texttt{[SEP]}, t_{|T|}), \\ \mathcal{C} &= (c_1, \texttt{[SEP]}, c_2, ..., \texttt{[SEP]}, c_{|C|}). \end{aligned} \tag{2}$$

All the special separator tokens and language word tokens in $x$ are converted to the word embedding by embedding layer of the language model. Gathering the embeddings of natural language and SQL, we feed them to self-attention blocks in a language model. In the training stage, we directly take the golden SQL query of the last turn as an input SQL query and set $\mathcal{S}$ to empty for the first turn. As for the inference stage, we apply the SQL query generated by HIE-SQL in the last turn.

### 2.3 SQLBERT

We propose SQLBERT, a bimodal pre-trained model for natural language and SQL, and develop it by using the same model architecture as RoBERTa$_{BASE}$ (Liu et al., 2019).

**Input Format.** To alleviate the difficulty of training and resolve inconsistencies between natural language and schema, we append the question-relevant database schema to the concatenation of SQL query and question. We represent the whole input sequence into the format as $x = ($ `[CLS]`$, s_1, s_2, ..., s_n,$ `[SEP]`$, q_1, ..., q_m,$ `[SEP]`$,$ $t_1 : c_{11}, c_{12}, ...,$ `[SEP]`$, t_2 : c_{21}, ...,$ `[SEP]`$, ...)$, in which $s$, $q$, $t$, and $c$ are the tokens of SQL query, question, tables, and columns respectively.

| | U | H | S |
|---|---|---|---|
| C | U−C−EM<br>U−C−PM<br>U−C−VM | H−C−EM<br>H−C−PM<br>H−C−VM | S−C−EC<br>S−C−UC |
| T | U−T−EM<br>U−T−PM | H−T−EM<br>H−T−PM | S−T−ET<br>S−T−UT |

Table 1: Edge types between current utterance $U$, interaction history $H$, SQL $S$, and database schema $D$ (Columns $C$ and Tables $T$). We set two match types between the language tokens of $U$, $H$, and $D$: **EM** for Exact Match, **PM** for Partial Match. When using database contents, we set **VM** (Value Match) for exactly matching the value of columns. As for SQL $S$, we simply match the words of tables and columns that appear in it to the target database schema: **EC** (Equal Columns) and **UC** (Unequal Columns) for columns, **ET** (Equal Tables) and **UT** (Unequal Tables) for tables. And we omit the pre-existing relations in schema such as the foreign-key relation (C-C-FK) in the table.

**Training Objective.** The main training objective of SQLBERT is the masked language modeling (MLM). Specifically, we utilize a special objective referenced span masking (Sun et al., 2019) by sampling 15% independent span in SQL clause except the reserved word (e.g., SELECT, FROM, WHERE). We describe the masked span prediction loss as

$$\mathcal{L}(\theta) = \sum_{k=1}^{n} -log\mathcal{P}_{\theta}(s_k^{mask}|s^{\setminus mask}, q, t, c), \quad (3)$$

where $\theta$ stands for the model parameters, $s_k^{mask}$ is the masked span of SQL input, $s^{\setminus mask}$ is the unmasked part. The detail of data we use to train SQLBERT is shown in Appendix A.

### 2.4 HIE-Layers

**Schema-Linking Graph.** To explicitly encode the complex relational database schema, we convert it to a directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = C \cup T$ and $\mathcal{E}$ represents the set of pre-existing relations within columns and tables such as the foreign-key relation. In addition, we also consider the unseen linking to the schema in the contexts. Specifically, we define the context-dependent schema-linking graph $\mathcal{G}_c = \langle \mathcal{V}_c, \mathcal{E}_c \rangle$ where $\mathcal{V}_c = C \cup T \cup U \cup H \cup S$ and $\mathcal{E}_c = \mathcal{E} \cup \mathcal{E}_{U \leftrightarrow D} \cup \mathcal{E}_{H \leftrightarrow D} \cup \mathcal{E}_{S \leftrightarrow D}$. The additional relation edges are listed in Table 1. We show an example of the proposed schema-linking graph in Appendix B.

**Graph Encoding.** We follow the work (Wang et al., 2020) to encode schema-linking graph via Relative Self-Attention Mechanism (Shaw et al., 2018). We show its details in Appendix C.

### 2.5 Decoder

To build the decoder of HIE-SQL, we apply the same work as Wang et al. (2020) propose, which generates SQL as an abstract syntax tree via LSTM (Hochreiter and Schmidhuber, 1997). We recommend the reader to refer to the work (Yin and Neubig, 2017) for details.

## 3 Experiment

### 3.1 Setup

**Setting.** Since the weights of SCoRe (Yu et al., 2021b) have not been open sourced, we initialize the weights of Language Model with GraPPa (Yu et al., 2021a). We stack 8 HIE-layers on top of the Language Model. And we use R-Drop (Liang et al., 2021) as our regularization strategy in training. Specific hyper-parameters and training setting are shown in Appendix D.

**Datasets.** We conduct experiments on two cross-domain context-dependent text-to-SQL datasets, SparC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). The statistic details of the datasets can be obtained in Appendix E.

**Evaluation Metrics.** The main metric we used to measure model performance in SparC and CoSQL is interaction match (IM), which requires all output SQL queries in the whole round of interaction to be correct. We also use question match (QM) to evaluate the accuracy of every single question.

### 3.2 Experiment Result

Results of our proposed HIE-SQL model are shown in Table 2. In terms of interaction match, our model achieves state-of-the-art performances on both SparC and CoSQL. For CoSQL, compared with the previous state-of-the-art (Scholak et al., 2021), a rule-based auto-regressive method based on the large pre-trained model-T5-3B (Raffel et al., 2020) which contains 2.8 billion parameters, HIE-SQL improves IM of development set by 4.5% and IM of the test set by 0.9% with only 580M parameters. Besides, HIE-SQL surpasses RAT-SQL + SCoRe in all metrics of SparC and CoSQL. This demonstrates that properly integrating interaction

| Model | SparC Dev | | SparC Test | | CoSQL Dev | | CoSQL Test | |
|---|---|---|---|---|---|---|---|---|
| | **QM** | **IM** | **QM** | **IM** | **QM** | **IM** | **QM** | **IM** |
| EditSQL + BERT (Zhang et al., 2019) | 47.2 | 29.5 | 47.9 | 25.3 | 39.9 | 12.3 | 40.8 | 13.7 |
| IGSQL + BERT (Cai and Wan, 2020) | 50.7 | 32.5 | 51.2 | 29.5 | 44.1 | 15.8 | 42.5 | 15.0 |
| IST-SQL + BERT (Wang et al., 2021) | - | - | - | - | 44.4 | 14.7 | 41.8 | 15.2 |
| $R^2$SQL + BERT (Hui et al., 2021) | 54.1 | 35.2 | 55.8 | 30.8 | 45.7 | 19.5 | 46.8 | 17.0 |
| RAT-SQL† + SCoRe (Yu et al., 2021b) | 62.2 | 42.5 | 62.4 | 38.1 | 52.1 | 22.0 | 51.6 | 21.2 |
| T5-3B + PICARD† (Scholak et al., 2021) | - | - | - | - | **56.9** | 24.2 | **54.6** | 23.7 |
| HIE-SQL + GraPPa (ours) | **64.7** | **45.0** | **64.6** | **42.9** | 56.4 | **28.7** | 53.9 | **24.6** |

Table 2: Performances of various models in SparC and CoSQL. QM and IM stand for question match and interaction match respectively. The models with † are proposed for the context-independent text-to-SQL task and applied to the context-dependent text-to-SQL task by just appending interaction history utterances to the input.

| Model | SparC | | CoSQL | |
|---|---|---|---|---|
| | **QM** | **IM** | **QM** | **IM** |
| HIE-SQL | 64.7 | **45.0** | 56.4 | **28.7** |
| w/o SQL query | **65.8** | 44.3 | **56.5** | 23.9 |
| w/o SQLBERT | 63.9 | 44.7 | 54.8 | 26.3 |
| w/o $\mathcal{E}_{H\leftrightarrow D}$ | 64.0 | 44.3 | 56.0 | 26.3 |

Table 3: Ablation study of HIE-SQL in development sets of SparC and CoSQL. As for ablation on SQL query, we drop the SQL query and only feed utterances and database schema to the model. As for ablation on SQLBERT, we directly concatenate the tokens of SQL query and other context tokens for the input of the language model. And w/o $\mathcal{E}_{H\leftrightarrow D}$ means we treat historical utterances like the current utterance in our schema-linking.

utterances and predicted SQL queries is an effective way to enhance the model's ability for Context-Dependent text-to-SQL Semantic Parsing. We test the robustness of HIE-SQL for the samples with different turn index and difficulty in Appendix F.

### 3.3 Ablation Study

We provide ablation studies to examine the contribution of each component of HIE-SQL. As shown in Table 3, Our full model achieves about 5 points and 1 point improvement of IM in CoSQL and SparC respectively compared with the model without the last SQL query input. The pre-encoding SQL query by SQLBERT can further improve the performance. It confirms SQLBERT's ability to efficiently represent SQL features. In addition, $\mathcal{E}_{H\leftrightarrow D}$ also plays a positive role.

It is worth noting that the last SQL query as input benefits the performance on IM which is converse

| Dataset | Model | T-F | F-T | T-T |
|---|---|---|---|---|
| SparC | HIE-SQL | 125 | 88 | **383** |
| | w/o SQL query | 132 | 104 | 379 |
| CoSQL | HIE-SQL | 140 | 106 | **278** |
| | w/o SQL query | 161 | 128 | 254 |

Table 4: The counts of different switches in the pairs of adjacent predicted SQL queries. T-F stands for the match of the former predicted query and unmatch of the later predicted query with golden queries. F-T stands for the reverse case. T-T is the case of both matching.

on QM. Table 4 shows that our model with SQL query has a higher rate of continuous match, but a lower rate of switching from mismatch to match. It illustrates that our model does use the SQL information and is sensitive to the accuracy of the last predicted SQL query. Since of the exposure bias during inference, the matched last SQL query will provide effective guidance for the model, but once prediction goes wrong, the errors tend to persist. We also offer some case study in Appendix G to further demonstrate the superiority of HIE-SQL.

## 4 Conclusion

We present HIE-SQL which targets at explicitly capturing the context-dependence from both interaction history utterances and the last predicted SQL query. With the help of SQLBERT and the proposed schema-linking graph, HIE-SQL bridges the gap between the utterances and predicted SQL. Taken together, HIE-SQL achieves consistent improvements on the context-dependent text-to-SQL task and achieves new state-of-the-art results on two famous context-dependent text-to-SQL datasets, SparC and CoSQL.

## References


Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *Proceedings of the 2015 Conference of International Semantic Web (ISWC), Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 425–441.

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, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: line graph enhanced text-to-SQL model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, volume 1, pages 2541–2555.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (EMNLP-Findings)*, pages 1536–1547.

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 Conference of the Association for Computational Linguistics (ACL)*, volume 1, pages 4524–4535.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 13116–13124.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. 2019. Supervised multimodal bitransformers for classifying images and text. In *Visually Grounded Interaction and Language (ViGIL), NeurIPS 2019 Workshop*.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-Drop: Regularized dropout for neural networks. *arXiv preprint arXiv:2106.14448*.

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 (EMNLP-Findings)*, pages 4870–4888.

Qian Liu, Bei Chen, Jiaqi Guo, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020. How far are we from effective context modeling? an exploratory study on semantic parsing in context. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3580–3586.

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*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 (NAACL-HLT)*, volume 2, pages 464–468.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *arXiv preprint arXiv:2106.13884*.

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 (ACL)*, pages 7567–7578.

Runze Wang, Zhen-Hua Ling, Jingbo Zhou, and Yu Hu. 2021. Tracking interaction states for multi-turn text-to-SQL semantic parsing. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 13979–13987.

Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon J. D. Prince, and Yanshuai Cao. 2021. Optimizing deeper transformers on small datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, volume 1, pages 2089–2102.

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 (ACL)*, pages 440–450.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021a. GraPPa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations (ICLR)*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard

Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP/IJCNLP)*, pages 1962–1979.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021b. SCoRe: Pre-training for context representation in conversational semantic parsing. In *9th International Conference on Learning Representations (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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3911–3921.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, volume 1, pages 4511–4523.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. 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 5337–5348.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# A Training Datas for SQLBERT

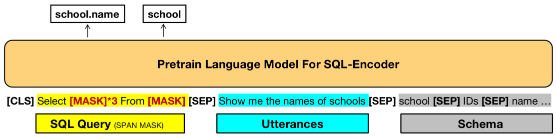


Figure 3: Input format and training objective of SQLBERT.

Unlike SCoRe (Yu et al., 2021b), which uses multiple open-source text-to-SQL datasets (WIKITABLES (Bhagavatula et al., 2015), WikiSQL (Zhong et al., 2017), Spider, SparC, and CoSQL) and data synthesis methods to obtain a large amount of pre-training data, we train SQLBERT only with the datasets including Spider, SparC and CoSQL. For each sample in Spider, we only use its question, SQL query, and the corresponding database schema. As for SparC and CoSQL, which is a context-dependent version, we simply concatenate the current utterance with the history utterances to build the question input. The size of the training data is about 34,000. A better understanding of the input format of SQLBERT can be obtained from Figure 3.

# B Schema-Linking Graph Example

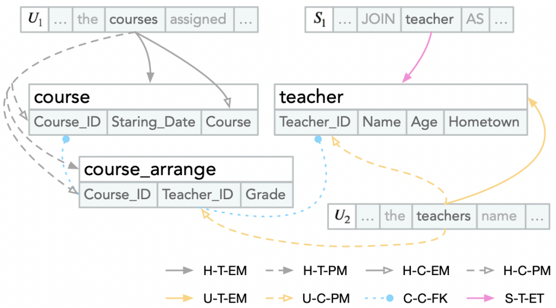


Figure 4: An example of the schema-linking graph for the prediction of $S_2$ in Figure 1.

As show in figure 4, the graph is a subgraph of the whole schema-linking graph. We only respectively choose one token in the history utterance ($U_1$), the current utterance ($U_2$), and the last predicted SQL query ($S_1$) in the example. Besides, we omit all unequal relation edges (S-C-UC and S-T-UT).

# C Relative Self-Attention Mechanism

Relative Self-Attention Mechanism rebuilds the calculation of the self-attention module in the transformer layers as follows:

$$
\begin{aligned}
e_{ij} &= \frac{x_i W^Q (x_j W^K + r_{ij}^K)^T}{\sqrt{d_z}}, \\
\alpha_{ij} &= \underset{j}{softmax}\{e_{ij}\}, \\
z_i &= \sum_{j=1}^{n} \alpha_{ij} (x_j W^V + r_{ij}^V).
\end{aligned} \tag{4}
$$

It consist of 8 transformer layers, whose self-attention modules are described above. Specifically, we initialize a learned embedding for each type of edge defined above. For every input sample, we build a relation matrix $\mathcal{R} \subseteq (L \times L)$ where L is the length of the input token. $\mathcal{R}^{(i,j)}$ represents the relation type between $i-$th and $j-$th input tokens. While computing the relative attention, we set the $r_{ij}^K = r_{ij}^V = \mathcal{R}_e^{(i,j)}$ where $\mathcal{R}_e^{(i,j)}$ is the corresponding embedding of $\mathcal{R}^{(i,j)}$.

# D Training Setting

We use Adam optimizer to conduct the parameter learning and set the learning rate of $1e^{-5}$ for fine-tuning GraPPa and $1e^{-4}$ for HIE-Layers and Decoder. The learning rate linearly increases to the setting point at first $max\_steps/8$ steps, then decreases to 0 at $max\_steps = 50000$ with 24 training batch-size. As for SQLBERT, we fine-tune CodeBERT$_{BASE}$ (Feng et al., 2020) on the dataset we described in Section A. We set the learning rate as $1e^{-5}$, a batch size of 64, and train SQLBERT for 10 epochs. The shape of learned weights of the linear layer applied to the output of SQLBERT is $768 \times 1024$. We only need one V100 (32G) GPU to train our model. While inferring, we set the beam size to 3.

# E Details of SparC and CoSQL datasets.

| Dataset | CoSQL | SparC |
|---|---|---|
| System Response | ✔ | ✗ |
| Interaction | 3007 | 4298 |
| Train | 2164 | 3034 |
| Dev | 293 | 422 |
| Test | 551 | 842 |
| User Questions | 15598 | 12726 |
| Vocab | 9585 | 3794 |
| Avg Turn | 5.2 | 3.0 |

Table 5: Details of SparC and CoSQL datasets.

# F Performances of HIE-SQL on Different Turns

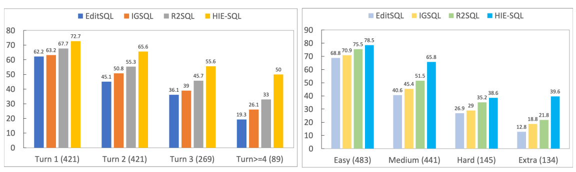


Figure 5: Performances of previous works and HIE-SQL in different turns (left) and different difficulty levels (right) on SparC.

We test the performance on different turns and at different difficulty levels of utterances. As shown in Figure 5, with the increase of turns, the lead of our model gets greater and greater. When the indexes of turns are greater than or equal to 4, the accuracy of HIE-SQL is 17% higher than that of $R^2$SQL. It demonstrates that the main contribution of introducing SQL query is to improve the robustness of the model to long interaction. HIE-SQL is also robust to the varying difficulty levels of utterances. Our model performs equally in hard and extra hard levels, and achieves 39.6% accuracy on the extra hard level, which is 17.8% higher than that of $R^2$SQL.

# G Case Study

| | |
|---|---|
| $U_{a1}$ | Which cartoon aired first? |
| HIE-SQL<br>RAT-SQL | SELECT title FROM cartoon ORDER BY original_air_date asc LIMIT 1<br>SELECT title FROM cartoon ORDER BY original_air_date asc LIMIT 1 |
| $U_{a2}$ | What was the last cartoon to air? |
| HIE-SQL<br>RAT-SQL | SELECT title FROM cartoon ORDER BY original_air_date desc LIMIT 1<br>SELECT title FROM cartoon ORDER BY original_air_date desc LIMIT 1 |
| $U_{a3}$ | What channel was it on? |
| HIE-SQL<br>RAT-SQL | SELECT channel FROM cartoon ORDER BY original_air_date desc LIMIT 1<br>SELECT channel FROM cartoon ORDER BY original_air_date desc LIMIT 1 |
| $U_{a4}$ | What is the production code? |
| HIE-SQL<br>RAT-SQL | SELECT production_code FROM cartoon ORDER BY original_air_date desc LIMIT 1<br>SELECT production_code FROM cartoon ORDER BY original_air_date asc LIMIT 1 |
| $U_{b1}$ | List the name of the teachers and the courses assigned for them to teach. |
| HIE-SQL<br>RAT-SQL | SELECT Name, Course FROM ...<br>SELECT Name, Course FROM ... |
| $U_{b2}$ | Arrange this list with the teachers name in ascending order |
| HIE-SQL<br>RAT-SQL | ELECT Name, Course FROM ... ORDER BY Name Asc<br>ELECT Name, Course FROM ... ORDER BY Name Asc |
| $U_{b3}$ | Include teachers ID in tha same list |
| HIE-SQL<br>RAT-SQL | SELECT Name, Course, Teacher_ID FROM ... ORDER BY Name Asc<br>SELECT Teacher_ID FROM ... ORDER BY Teacher_ID Asc |
| $U_{c1}$ | What is the name of the poker player with the highest earnings? |
| HIE-SQL<br>RAT-SQL | SELECT Name FROM ... ORDER BY Earnings Desc LIMIT 1<br>SELECT Name FROM ... ORDER BY Earnings Desc LIMIT 1 |
| $U_{c2}$ | What about the poker player with the lowest earnings? |
| HIE-SQL | ... FROM poker_player JOIN people ON People_ID = People_ID<br>ORDER BY Earnings Asc LIMIT 1 |
| RAT-SQL | ... FROM poker_player JOIN people ON People_ID = People_ID<br>ORDER BY Earnings Asc LIMIT 1 |
| $U_{b3}$ | What was his best finish? |
| HIE-SQL | SELECT Best_Finish FROM poker_player<br>JOIN people ON People_ID = People_ID ORDER BY ... |
| RAT-SQL | SELECT Best_Finish FROM poker_player ORDER BY ... |

Table 6: Examples in CoSQL. $U_{ij}$ is the input utterance of turn $j$ of example $i$ with corresponding predictions of HIE-SQL and RAT-SQL following. All predictions of HIE-SQL are the ground truth queries in the case. As the examples show, RAT-SQL fails to distinguish the right one from two long-range dependences in $U_{a1}$ and $U_{a2}$ in the first example and fails to inherit the query information from $U_{b2}$ in $U_{b3}$. By contrast, HIE-SQL inherits the right context-dependence from the last predicted query to avoid the confusion.