# GuideSQL: Utilizing Tables to Guide the Prediction of Columns for Text-to-SQL Generation

Huajie Wang*, Lei Chen*†, Mei Li*, Mengnan Chen*
*School of Computer Science and Technology, East China Normal University, Shanghai, China
†Shanghai Key Laboratory of Multidimensional Information Processing, Shanghai, China
Email: 51184506041@stu.ecnu.edu.cn, lchen@cs.ecnu.edu.cn,
51184506023@stu.ecnu.edu.cn, 51174506001@stu.ecnu.edu.cn

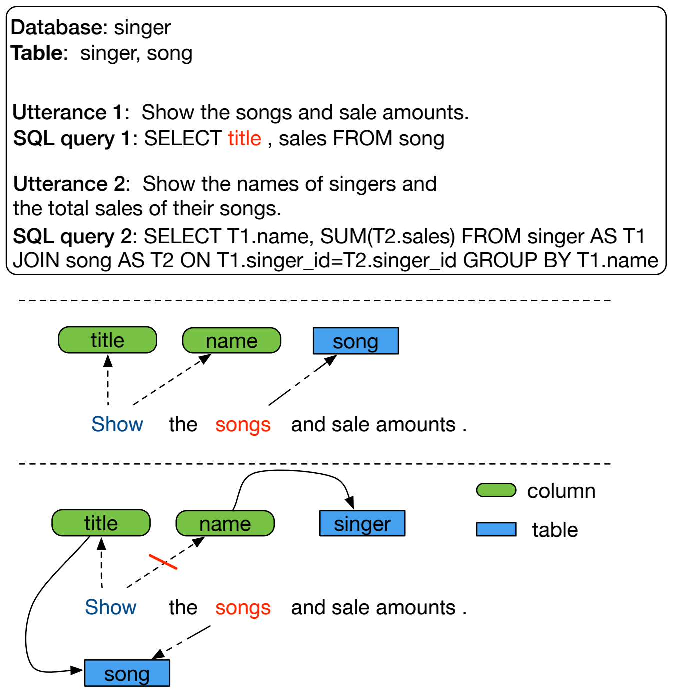
***Abstract*—Text-to-SQL is a task of synthesizing SQL queries from utterances. Most existing approaches of text-to-SQL rarely utilize tables to guide the prediction of SQL query. We present a novel approach called GuideSQL which predicts tables first and uses a pruning algorithm for removing the columns which don't belong to the predicted tables to avoid errors caused by misprediction of table-column dependencies. For reducing the prediction errors of tables, we use the top-K predicted tables to generate SQL queries and employ a string-matching algorithm to get the most reasonable one. Futhermore, a type linking mechanism is utilized to augment the relevance between utterances and schemas. On the challenging text-to-SQL benchmark SParC, we use previous query attention to get context-dependent information of SQL queries. GuideSQL obtains 36.3% question matching accuracy and 19.5% interaction matching accuracy on the dev set. With BERT augmentation, GuideSQL achieves 49.2% question matching accuracy and 31.6% interaction matching accuracy on the dev set, outperforms the previous state-of-the-art model by 2% question matching accuracy and 2.1% interaction matching accuracy.**

## I. Introduction

In recent years, there has been an increasing interest in text-to-SQL, i.e., translating a question to the corresponding SQL query. Text-to-SQL is a sub-task of semantic parsing and it has a long history [1]–[5]. Various neural approaches have been proposed in context-independent datasets such as WikiSQL [6] and Spider [7]. For example, SQLova [8] achieves more than 80% logical form accuracy on WikiSQL dev and test sets, and IRNet [9] achieves more than 60% and 50% exact matching accuracy on the Spider dev and test sets, respectively.

However, Yu et al. [10] present SParC, a new complex context-dependent and cross-domain text-to-SQL dataset based on Spider. Most studies in the field of text-to-SQL only focus on translating stand-alone utterances to SQL queries, and are hard to be extended on SParC which contains contextual dependencies. In addition, we found that most methods [9], [11], [12] predicted SQL queries in the normal order (e.g., SELECT title FROM song), without utilizing tables to guide the prediction of SQL queries (e.g., FROM song SELECT title). Suhr et al. [13] propose a seq2seq model with interaction-level encoder used on ATIS [14]. It is extended on SParC to get some new models such as CD-Seq2Seq [10] and EditSQL [15]. EditSQL achieves the new state-of-the-art performance,

Lei Chen is the corresponding author.



**Database**: singer
**Table**: singer, song

**Utterance 1**: Show the songs and sale amounts.
**SQL query 1**: SELECT title , sales FROM song

**Utterance 2**: Show the names of singers and the total sales of their songs.
**SQL query 2**: SELECT T1.name, SUM(T2.sales) FROM singer AS T1 JOIN song AS T2 ON T1.singer_id=T2.singer_id GROUP BY T1.name

title
name
song
Show the songs and sale amounts .
title
name
singer
column
table
Show the songs and sale amounts .
song

Fig. 1. Top: An example of an interaction from SParC dataset. Middle: The ambiguity of predicting column (`title` or `name`) when without using guide mechanism. Bottom: The guide mechanism leads to predict the correct column `title` when table `song` is predicted.

but we found EditSQL generated table and column together (e.g., song.title) in a token during decoding phase, which would increase the prediction risk of tables. Meanwhile, we found that lacking the dependency between table and column in CD-Seq2Seq caused the errors in predicting columns when tables were mentioned in the question but columns showed ambiguity. Considering the interaction and its corresponding SQL queries in Fig. 1 top part, the column `title` is not easy to be predicted for the first utterance because columns `title` and `name` are similar, as shown in Fig. 1 middle part.

In this work, we propose a new model called GuideSQL based on CD-Seq2Seq. GuideSQL contains the guide mechanism which is used to solve the misprediction of table-column dependencies in predicted SQL queries. As shown in Fig. 1 bottom part, according to the *songs* in utterance, the range of

*Original*: `SELECT name FROM physician EXCEPT SELECT T2.name FROM appointment AS T1 JOIN physician AS T2 on T1.physician = T2.employeeid`

*IntermediateState*: `FROM physician SELECT name EXCEPT FROM appointment AS T1 JOIN physician AS T2 on T1.physician = T2.employeeid SELECT T2.name`

Fig. 2. An example of getting an intermediate state of the SQL query, and EXCEPT is a split token. The SQL contains two subqueries and the FROM clause is moved to the first place in each subquery.

query is selected as the table song. While the table song is predicted, we prune columns which don't belong to table song, then we can directly exclude the column name which is not in song to get the correct column title. In order to reduce the prediction errors of tables and to obtain the most reasonable SQL query, we use an effective string-matching algorithm to re-rank SQL queries which were generated by top-K predicted tables. In order to predict tables instead of columns first, we design an intermediate state of the SQL query shown in Fig. 2. In order to augment the relevance between utterances and schemas, we use a type linking mechanism to discern the type of entities (TABLE, COLUMN) in utterances and add type information to schema embeddings. In addition, we use previous query attention to get context-dependent information of SQL queries on SParC dataset. Experimental results show that GuideSQL achieves 36.3% question matching accuracy and 19.5% interaction matching accuracy on SParC dev set. In addition, GuideSQL with BERT [16] augmentation obtains 49.2% question matching accuracy and 31.6% interaction matching accuracy on SParC dev set, gains 2% question matching accuracy and 2.1% interaction matching accuracy improvements compared to the previous state-of-the-art model.

## II. Related Work

With the advent of large-scale cross-domain text-to-SQL data sets [6], [7], [10], instead of manually intervening in specific databases [17]–[19], methods based on neural network [8], [9], [11], [12], [20], [21] have received more and more attention. The most of approaches only apply to the context-independent tasks such as WikiSQL and Spider. SQLNet [20] propose seq2set to generate SQL queries on WikiSQL dataset. With BERT augmentation, a new model based on SQLNet named SQLova [8] achieves the state-of-the-art performance. But these methods are limited to support very simple queries which only contain SELECT and WHERE clauses. Based on SQL syntax, Yu et al. [21], Bogin et al. [12] and Guo et al. [9], etc., use seq2tree to generate SQL queries on Spider which does not have contextual dependencies.

However, the actual requirements are often context-dependent. In other words, the text-to-SQL task requires SQL queries to be obtained from context-dependent questions. Recently, Suhr et al. [13] propose a seq2seq model with interaction-level encoder to solve the context-dependent dataset ATIS [14], and this model is extended to SParC called CD-Seq2Seq [10]. They use a discourse state to maintain the entire interaction during encoding phase. In addition, during the decoding phase, they use position embeddings to record the position information for each utterance and propose a mechanism to copy segments from the previous query. Based on SyntaxSQL [21], SyntaxSQL-con [10] uses two bi-LSTM with different parameters to encode the previous utterance and current utterance. The decoder of SyntaxSQL-con utilizes the SQL syntax and generation history to generate SQL queries. Based on CD-Seq2Seq, EditSQL [15] utilizes utterance-table encoder and table-aware decoder to incorporate the utterances and schemas. In the decoder, EditSQL proposes an editing mechanism for deciding whether to copy a token from the previous query or insert a new token.

The type linking we proposed is inspired by the type recognization mechanisms in TypeSQL [22] and IRNet [9]. TypeSQL utilizes type information based on knowledge to recognize entities mentioned in an utterance, then concatenates embeddings of words and their corresponding types to the encoder. IRNet only recognizes the partial matching and exact matching of tables and columns mentioned in a question, then takes the average of the type and words embeddings as the input. Our method is similar to IRNet, but we remove the partial matching case.

## III. Methodology

Our model is based on CD-Seq2Seq [10], along with three novel components. (1) We use guide mechanism to limit the prediction space of columns and re-ranking mechanism to improve exact matching accuracy. (2) We use type linking to augment the relevance between utterances and schemas. (3) We add previous query attention and schema attention to the decoder. Fig. 3 illustrates the architecture of GuideSQL.

### A. Type Linking

We use an effective method based on string-matching to implement type linking and define two types (TABLE, COLUMN) of entities that may be mentioned in a question. First, all the n-grams of length 1-6 are enumerated in a question. Then, we traverse the n-grams from length 6 to 1. If an n-gram exactly matches a table name or a column name, we mark this n-gram as TABLE or COLUMN. If an n-gram matches both TABLE and COLUMN, we prioritize TABLE. Once an n-gram is assigned, we will recognize the next part of question which does not overlap the detected n-gram. In addition, we add TABLE and COLUMN embeddings to schemas for predicting the correct type of schema entries.

### B. Schema Embedding

Let $E_c$ denotes the average embedding of table words or column words, and $E_\tau$ denotes the type embedding. We take the average of $E_c$ and $E_\tau$ embeddings to generate the schema

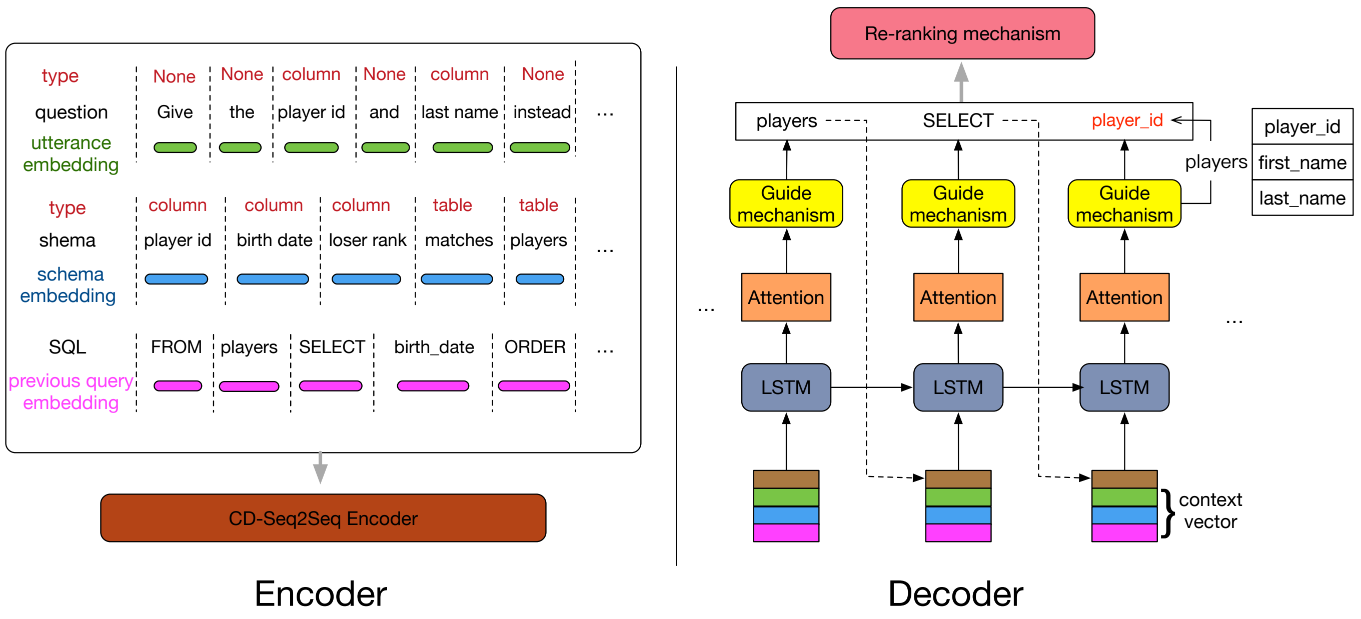


Fig. 3. Our model architecture is based on CD-Seq2Seq [10]. The inputs to the encoder are schema embedding, utterance embedding and previous query embedding. The decoder has been extended to include schema attention, previous query attention, guide mechanism and re-ranking mechanism. As shown in the figure, the table ‘players’ is used to guide prediction of the column ‘player_id’.

embedding. We define $\mathbf{h}^S$ as the schema embedding and calculate it as:

$$\mathbf{h}^S = Avg(E_c + E_\tau) \quad . \tag{1}$$

## C. Input Encoder

Let $x_i = [(x_{i,1}, \tau_{i,1}), \cdots, (x_{i,L}, \tau_{i,L})]$ denotes the non-overlap span sequence of a question in turn $i$ and $L$ is the length of $x_i$, $x_{i,j}$ denotes the $j^{th}$ span tokens in turn $i$ and $\tau_{i,j}$ denotes the type of $x_{i,j}$ which we obtained in section III-A. First, we take the average of $x_{i,j}$ and $\tau_{i,j}$ embeddings as the span embeddings $e_x^{i,j}$. Then, we implement the interaction-level encoder similar to CD-Seq2Seq. We use a bi-directional LSTM [23] and the forward LSTM is defined by:

$$\mathbf{h}_{i,j}^{\overrightarrow{E}} = \mathrm{LSTM}^{\overrightarrow{E}} \left( \left[ e_x^{i,j}; \mathbf{h}_{i-1}^I \right], \mathbf{h}_{i,j-1}^{\overrightarrow{E}} \right) \quad , \tag{2}$$

where $\mathbf{h}_{i,j}^{\overrightarrow{E}}$ is the $j^{th}$ forward hidden state in turn $i$. $\mathbf{h}_{i-1}^I$ is the discourse state in turn $i-1$ which can maintain and update over the entire interaction. The backward LSTM $\mathbf{h}_{i,j}^{\overleftarrow{E}}$ is modified analogously. We compute a hidden state $\mathbf{h}_{i,j}^E = [\mathbf{h}_{i,j}^{\overrightarrow{E}}; \mathbf{h}_{i,j}^{\overleftarrow{E}}]$ for each span embedding $e_x^{i,j}$, and the discourse state $\mathbf{h}_i^I$ is computed as:

$$\mathbf{h}_i^I = \mathrm{LSTM}^I \left( \mathbf{h}_{i,L}^E, \mathbf{h}_{i-1}^I \right) \quad . \tag{3}$$

## D. Decoder

An LSTM decoder with attention [24], [25] is used to generate SQL queries in GuideSQL. In addition, we use embeddings $\phi^I$ to record the relative position information and they are learnt for each possible distance $0 \ldots h$ from the current utterance.

We compute the attention between the decoder hidden state and the interaction encoder hidden state as:

$$\begin{aligned} s_k(t,j) &= \mathbf{h}_k^D \mathbf{W}_{\text{token-att}} [\mathbf{h}_{t,j}^E; \phi^I(i-t)] \\ \alpha^{token} &= softmax(s) \\ \mathbf{c}_k^{token} &= \sum_{t=i-h}^{i} \sum_{j=1}^{|x_t|} \alpha_k^{token}(t,j) [\mathbf{h}_{t,j}^E; \phi^I(i-t)] \quad , \end{aligned} \tag{4}$$

where $\alpha^{token}$ is the attention weight and $\mathbf{h}_k^D$ means the hidden state of decoder in step $k$, $|x_t|$ means the length of $x_t$ and $\mathbf{c}_k^{token}$ is the attention vector. $\mathbf{W}_{\text{token-att}}$ is the trainable parameter.

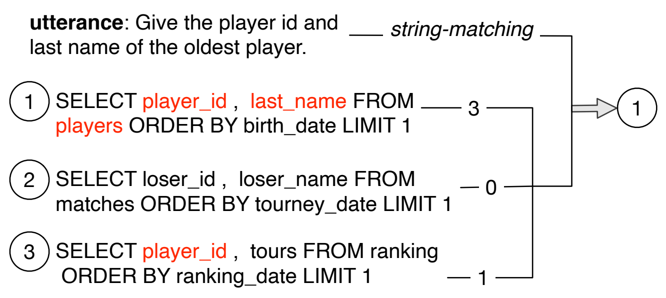
The attention between the decoder hidden state and the schema is calculated as follows:

$$\begin{aligned} s_k(l) &= \mathbf{h}_k^D \mathbf{W}_{\text{schema-att}} \mathbf{h}_l^S \\ \alpha^{schema} &= softmax(s) \\ \mathbf{c}_k^{schema} &= \sum_{l=1}^{|schema|} \alpha_k^{schema}(l) \mathbf{h}_l^S \quad , \end{aligned} \tag{5}$$

where $|schema|$ denotes the length of $\mathbf{h}^S$, $\mathbf{W}_{\text{schema-att}}$ is the trainable parameter. The context vector $\mathbf{c}_k$ is a concatenation of $\mathbf{c}_k^{token}$ and $\mathbf{c}_k^{schema}$:

$$\mathbf{c}_k = [\mathbf{c}_k^{token}; \mathbf{c}_k^{schema}] \quad . \tag{6}$$

At each decoding step, the decoder choose to predict a SQL keyword or a schema entry (table or column). We first use a heuristic algorithm to get the type of prediction for the current



utterance: Give the player id and last name of the oldest player.
string-matching
1 SELECT player_id , last_name FROM players ORDER BY birth_date LIMIT 1
3
2 SELECT loser_id , loser_name FROM matches ORDER BY tourney_date LIMIT 1
0
3 SELECT player_id , tours FROM ranking ORDER BY ranking_date LIMIT 1
1
1

Fig. 4. An example of re-ranking mechanism. There are three generated SQL queries from decoder and we use string-matching algorithm to choose the first SQL query whose matching number is the largest, which is 3.

step. Once we get the current predicted type, we define $\mathbf{h}^C$ as follows:

$$\mathbf{h}^C = \begin{cases} \mathbf{h}^S_{table}, & \text{if } type \text{ is } table \\ \mathbf{h}^S_{column}, & \text{otherwise} \end{cases} , \tag{7}$$

where $\mathbf{h}^S_{table}$ is the schema embedding only contains table embeddings, $\mathbf{h}^S_{column}$ only contains column embeddings which belong to the previous predicted tables. Then we calculate the output probability distribution as follows:

$$\begin{aligned} \mathbf{o}_k &= \tanh([\mathbf{h}^D_k; \mathbf{c}_k]\mathbf{W}_o) \\ \mathbf{m}^{\text{SQL}} &= \mathbf{o}_k \mathbf{W}_{\text{SQL}} + \mathbf{b}_{\text{SQL}} \\ \mathbf{m}^{\text{schema}} &= \mathbf{o}_k \mathbf{W}_{\text{schema}} \mathbf{h}^C \\ P(y_k) &= softmax([\mathbf{m}^{\text{SQL}}; \mathbf{m}^{\text{schema}}]) \end{aligned} , \tag{8}$$

where $\mathbf{m}^{\text{SQL}}$ and $\mathbf{m}^{\text{schema}}$ are SQL keywords scores and schema entries scores respectively. $\mathbf{W}_o$, $\mathbf{b}_{\text{SQL}}$, $\mathbf{W}_{\text{SQL}}$ and $\mathbf{W}_{\text{schema}}$ are trainable parameters.

### *E. Re-ranking*

We found that using tables to guide prediction of columns could cause many errors when the tables were predicted incorrectly. In addition, the correct table always appears in the top-K table prediction spaces. Hence, we first use top-K tables to predict columns for generating SQL queries. Then we calculate the similarity between the SQL query and the utterance. We found that even if the predicted tables were similar, the columns from different SQL queries had obvious differences. Hence, we use the string-matching algorithm which only calculates the matching number between SQL tokens and utterance tokens. Finally, we choose the SQL query of maximun matching number as the target SQL query which can improve the accuracy. Fig. 4 illustrates an example of re-ranking mechanism.

### *F. Previous Query Attention*

Inspired by EditSQL, we add previous query attention to the context vector which can slightly improve interaction matching accuracy. First, we encode previous predicted SQL query in a bi-LSTM. Then we calculate the attention between the decoder hidden state and the previous predicted query as:

TABLE I
EXACT MATCHING ACCURACY ON SPIDER DEV SET.

| Model | Dev Set |
|---|---|
| SyntaxSQLNet (augment) [21] | 24.8 |
| RCSQL [26] | 28.5 |
| EditSQL [15] | 36.4 |
| GNN [12] | 40.7 |
| Global-GNN [27] | 52.7 |
| IRNet [9] | 53.2 |
| EditSQL (BERT) [15] | 57.6 |
| IRNet (BERT) [9] | 61.9 |
| GuideSQL | 36.8 |

$$\begin{aligned} s_k(p) &= \mathbf{h}^D_k \mathbf{W}_{\text{query-att}} \mathbf{h}^Q_p \\ \alpha^{query} &= softmax(s) \\ \mathbf{c}^{query}_k &= \sum_{j=1}^{|q|} \alpha^{query}_k(p) \mathbf{h}^Q_p \end{aligned} , \tag{9}$$

where $\mathbf{h}^Q_p$ is the $p^{th}$ token encoder hidden state of the previous predicted SQL query. $\mathbf{W}_{\text{query-att}}$ is the trainable parameter and $|q|$ is the length of previous SQL query. Then we change the context vector as:

$$\mathbf{c}_k = [\mathbf{c}^{token}_k; \mathbf{c}^{schema}_k; \mathbf{c}^{query}_k] \; . \tag{10}$$

Moreover, we use another layer to score the previous query tokens:

$$\mathbf{m}^{\text{prev}} = \mathbf{o}_k \mathbf{W}_{\text{prev}} \mathbf{h}^Q \; , \tag{11}$$

where $\mathbf{m}^{\text{prev}}$ is previous query tokens scores and $\mathbf{W}_{\text{prev}}$ is the trainable parameter. Then the output probability distribution is modified as follows:

$$P(y_k) = softmax([\mathbf{m}^{\text{prev}}; \mathbf{m}^{\text{SQL}}; \mathbf{m}^{\text{schema}}]) \; . \tag{12}$$

## IV. EXPERIMENT

In this section, we evaluate the effectiveness of GuideSQL on both Spider and SParC semantic parsing datasets. Spider contains 8659, 1034, 2147 interactions and SParC contains 3034, 421, 842 interactions for training, development and testing. As the test sets of SParC and Spider are unreleased, all evaluations are performed on the publicly available dev sets. We use exact matching accuracy to evaluate our model on Spider dataset. On SParC, question matching accuracy (the exact set matching score over all questions) and interaction matching accuracy (the exact set matching score over all interactions) are used to evaluate our model. We do not use any data augmentation for fair comparison. In addition, we conduct an ablation study to analysis the contribution of each mechanism on SParC dev set. Our code is publicly available.[1]

[1]https://github.com/cfhaiteeh/GuideSQL

TABLE II
OVERALL RESULTS OF QUESTION MATCHING ACCURACY AND INTERACTION MATCHING ACCURACY ON SPARC DEV SET.

| Model | Question Matching Dev Set | Interaction Matching Dev Set |
|---|---|---|
| SyntaxSQL-con [10] | 18.5 | 4.3 |
| CD-Seq2Seq [10] | 17.1 | 6.7 |
| EditSQL [15] | 33.0 | 16.4 |
| EditSQL(BERT) [15] | 47.2 | 29.5 |
| **GuideSQL(BERT)** | **49.2** | **31.6** |
| GuideSQL | 36.3 | 19.5 |
| - query attention | 36.0 | 18.5 |
| - query attention - re-ranking | 33.3 | 16.6 |
| - query attention - re-ranking - guide | 19.7 | 7.8 |

TABLE III
ACCURACY OF QUESTION MATCHING ON SPARC DEV SET BY DIFFERENT HARDNESS LEVELS.

| Model | Goal Difficulty | | | |
|---|---|---|---|---|
| | Easy | Medium | Hard | Extra Hard |
| | (481) | (441) | (145) | (133) |
| SyntaxSQL-con [10] | 38.9 | 7.3 | 1.4 | 0.7 |
| CD-Seq2Seq [10] | 35.1 | 7.0 | 2.8 | 0.8 |
| **GuideSQL(BERT)** | **70.5** | **43.1** | **26.2** | **17.3** |
| GuideSQL | 57.8 | 28.8 | 14.5 | 7.5 |

TABLE IV
ACCURACY OF QUESTION MATCHING ON SPARC DEV SET BY DIFFERENT TURNS.

| Model | Turn # | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | >=4 |
| | (421) | (421) | (269) | (89) |
| SyntaxSQL-con [10] | 38.6 | 11.6 | 3.7 | 1.1 |
| CD-Seq2Seq [10] | 31.4 | 12.1 | 7.8 | 2.2 |
| **GuideSQL(BERT)** | **65.1** | **46.8** | **36.1** | **25.0** |
| GuideSQL | 52.7 | 33.5 | 23.0 | 12.5 |

## *A. Experimental Setup*

The two datasets contain two different aspects of text-to-SQL task, hence we use different components of the model to address them.

For Spider, we don't use the interaction-level component which contains discourse state, position state and previous query attention.

Conversely, we evaluate our model on SParC with all mechanisms. We set hyperparameters $h$ = 5 for the position state and $K$ = 5 for the re-ranking mechanism. We use 50-dimensional position embeddings which are initialized from a random uniform distribution $U[-0.1, 0.1]$ and are fixed during training.

More configurations both on Spider and SParC are as follows. Our model is implemented in PyTorch [28] and we use Adam [29] for optimization. We use 300-dimensional SQL keyword embeddings which are initialized from a random uniform distribution and are also fixed. We use the pretrained GloVe word embeddings [30] for utterance embeddings, schema embeddings and previous query embeddings, all of them without being fixed. The other hyperparameters are the same as those of CD-Seq2Seq in our model and we use the official evaluation script[2] to calculate accuracy.

## *B. BERT Embeddings*

To further study the effectiveness of GuideSQL, we use BERT [16] on SParC dataset to encode interactions, schemas and previous queries. Our BERT embeddings are similar to EditSQL [15] and SQLova [8], we use pretrained small cased BERT model and the sequence is fed into the pretrained BERT model as follows:

$$\texttt{[CLS]},\ X_i, \texttt{[SEP]},\ T_1,\ \texttt{[SEP]},\ \ldots,\ T_n,\ \texttt{[SEP]},\ C_1\ \texttt{[SEP]},\ \ldots,\ C_m,\ \texttt{[SEP]}$$

where `[CLS]` and `[SEP]` are split tokens, $X_i$ is the utterance tokens. $T_j$ contains the table $t_j$ and columns which belong to $t_j$, $C_k$ only contains column tokens. All of $X_i$, $T_j$ and $C_k$ contain type information. We use the hidden states which are produced from BERT as embeddings. In addition, we use a bi-LSTM to encode the BERT embeddings of $T_j$ and $C_k$. Then we take the final hidden state of the bi-LSTM over BERT embedding $T_j$ as the table $t_j$ embedding and the final hidden state of the bi-LSTM over BERT embedding $C_k$ as the column $C_k$ embedding. We found using BERT could reduce the prediction errors of tables obviously, hence we don't use re-ranking mechanism when using BERT embeddings.

## *C. Results*

Table I shows the results of the exact matching accuracy on Spider set. We found that our model only uses guide and re-ranking mechanisms can outperform some recent neural models. Table II shows the results of question matching accuracy and interaction matching accuracy by comparing GuideSQL with previous models on SParC. In addition, GuideSQL outperforms EditSQL without BERT augmentation, and there are 3.3% question matching accuracy and 3.1% interaction matching accuracy improvements on the dev set. When using BERT augmentation, GuideSQL achieves 49.2% question matching accuracy and 31.6% interaction matching accuracy on the dev set, which outperforms the EditSQL by

[2]https://github.com/taoyds/sparc

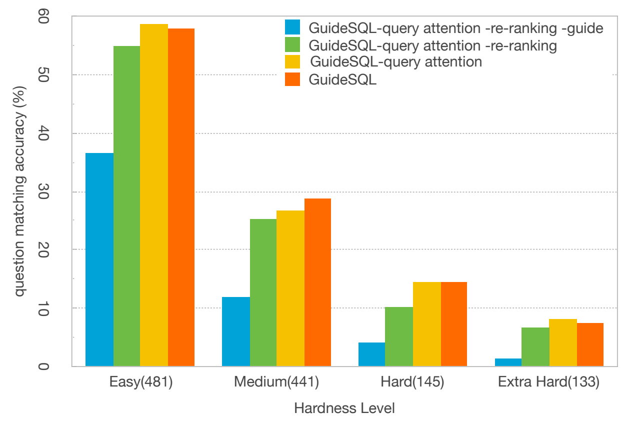


Fig. 5. Effects of different mechanisms in GuideSQL at different hardness levels on SParC dev set.

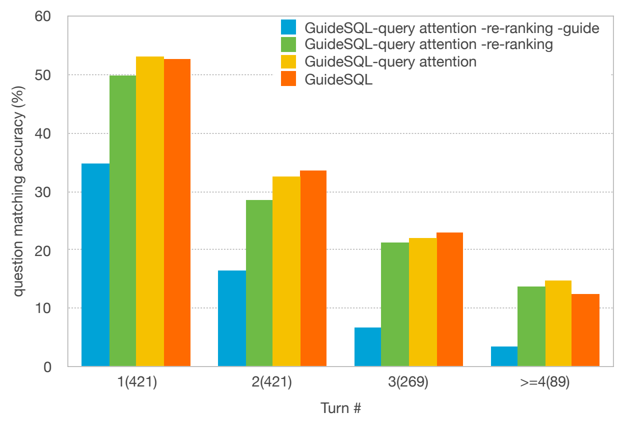


Fig. 6. Effects of different mechanisms in GuideSQL at different turns on SParC dev set.

2% question matching accuracy and 2.1% interaction matching accuracy on the dev set.

We further study the performance of GuideSQL in different hardness levels according to the official classification. There are 481, 441, 145, 133 questions for easy, medium, hard and extra hard levels respectively. As shown in Table III, our model outperforms baseline models which provide the data in all four hardness levels on dev set. To gain how utterance position effects the performance, we measure our model in different turns by official classification. There are 421, 421, 269, 89 questions for turn 1 to 4. As shown in Table IV, GuideSQL outperforms baseline models in all turns on dev set. Moreover, utterances in the later turns have a greater dependence on previous turns and have a greater risk for error propagation.

### D. Ablation Study

We ablate the major novel mechanisms of GuideSQL to assess their impacts on SParC dev set and we only consider performances without BERT augmentation. First, we remove the previous query attention and Table II shows that the performance drops by 0.3% in question matching accuracy and drops by 1% in interaction matching accuracy. Then, we remove the re-ranking mechanism, and it can be found that question matching accuracy drops by 2.7% and interaction matching accuracy drops by 1.9% in Table II. Finally, Table II shows removing the guide mechanism causes large drops in performance to 19.7% of question matching accuracy and 7.8% of interaction matching accuracy respectively.

In addition, we evaluate the effects of different mechanisms at different hardness levels and different turns. Fig. 5 and Fig. 6 show that the guide mechanism makes a significant contribution in GuideSQL. Furthermore, how to use schema structure takes an important part in text-to-SQL task.

## V. Conclusion

We present GuideSQL that uses predicted tables to guide the prediction of columns in text-to-SQL task and it outperforms the previous state-of-the-art model on SParC. We found that guide mechanism can effectively improve the performance on predicting columns when tables were predicted. The re-ranking mechanism can reduce the prediction errors of tables. Experimental results show that our model improves the performance not only in predicting simple queries, but also in predicting nested, complex queries in unseen databases.

## References

[1] D. H. D. Warren and F. Pereira, "An efficient easily adaptable system for interpreting natural language queries," *American Journal of Computational Linguistics*, vol. 8, pp. 110–122, 1981.

[2] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases - an introduction," *Natural Language Engineering*, vol. 1, pp. 29–81, 1995.

[3] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability," in *COLING*, 2004.

[4] A. Giordani and A. Moschitti, "Generating sql queries using natural language syntactic dependencies and metadata," in *NLDB*, 2012.

[5] C. Wang, A. Cheung, and R. Bodik, "Synthesizing highly expressive sql queries from input-output examples," in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, vol. 52, no. 6, 2017, pp. 452–466.

[6] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *CoRR*, vol. abs/1709.00103, 2017.

[7] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *EMNLP*, 2018.

[8] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on wikisql with table-aware word contextualization," *CoRR*, vol. abs/1902.01069, 2019.

[9] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang, "Towards complex text-to-sql in cross-domain database with intermediate representation," in *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4524–4535.

[10] T. Yu, R. Zhang, M. Yasunaga, Y. C. Tan, X. V. Lin, S. Li, I. L. Heyang Er, B. Pang, T. Chen, E. Ji, S. Dixit, D. Proctor, S. Shim, V. Z. Jonathan Kraft, C. Xiong, R. Socher, and D. Radev, "Sparc: Cross-domain semantic parsing in context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019.

[11] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *ACL 2018: 56th Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2018, pp. 731–742.

[12] B. Bogin, J. Berant, and M. Gardner, "Representing schema structure with graph neural networks for text-to-sql parsing," in *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4560–4565.

[13] A. Suhr, S. Iyer, and Y. Artzi, “Learning to map context-dependent sentences to executable formal queries,” in *NAACL HLT 2018: 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2018, pp. 2238–2249.

[14] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, “The atis spoken language systems pilot corpus,” *Proceedings of the workshop on Speech and Natural Language - HLT ’90*, 1990. [Online]. Available: http://dx.doi.org/10.3115/116580.116613

[15] R. Zhang, T. Yu, H. Er, S. Shim, E. Xue, X. V. Lin, T. Shi, C. Xiong, R. Socher, and D. Radev, “Editing-based sql query generation for cross-domain context-dependent questions,” in *2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *NAACL*, 2019.

[17] D. H. D. Warren and F. C. N. Pereira, “An efficient easily adaptable system for interpreting natural language queries,” *Computational Linguistics*, vol. 8, no. 3, pp. 110–122, 1982.

[18] F. Li and H. V. Jagadish, “Constructing an interactive natural language interface for relational databases,” *very large data bases*, vol. 8, no. 1, pp. 73–84, 2014.

[19] N. Yaghmazadeh, Y. Wang, I. Dillig, and T. Dillig, “Sqlizer: query synthesis from natural language,” *PACMPL*, vol. 1, pp. 63:1–63:26, 2017.

[20] X. Xu, C. Liu, and D. X. Song, “Sqlnet: Generating structured queries from natural language without reinforcement learning,” *ArXiv*, vol. abs/1711.04436, 2018.

[21] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. R. Radev, “Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task,” in *EMNLP*, 2018.

[22] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, “Typesql: Knowledge-based type-aware neural text-to-sql generation,” *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018. [Online]. Available: http://dx.doi.org/10.18653/v1/n18-2093

[23] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, p. 1735–1780, Nov 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[24] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

[25] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv preprint arXiv:1508.04025*, 2015.

[26] D. Lee, “Clause-wise and recursive decoding for complex and cross-domain text-to-sql generation,” in *2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[27] B. Bogin, M. Gardner, and J. Berant, “Global reasoning over database structures for text-to-sql parsing,” in *2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[29] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

[30] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.