Received July 3, 2020, accepted July 21, 2020, date of publication July 24, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011747

F-SQL: Fuse Table Schema and Table Content for Single-Table Text2SQL Generation

XIAOYU ZHANG[1], FENGJING YIN[1], GUOJIE MA[2], BIN GE[1], AND WEIDONG XIAO[1]

[1]Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China
[2]School of Software Engineering, East China Normal University, Shanghai 200000, China

Corresponding authors: Xiaoyu Zhang (xyzhang09@nudt.edu.cn) and Fengjing Yin (yinfengjing@nudt.edu.cn)

The work was supported by the NSFC under Grant 61872446, Grant 61902417, and Grant 71971212.

ABSTRACT Automatically parsing SQL queries from natural languages can help non-professionals access databases and improve the efficiency of information utilization. It is a long-term research issue and has recently received attention from the relevant communities. Although previous researches have provided some workable solutions, most of them only consider table schemas and natural language questions when parsing SQL queries, and do not use table contents. We observe that table contents can provide more helpful information for some user questions. In this paper, we propose a novel neural network approach, F-SQL, to focus on solving the problem of table content utilization. In particular, we employ the gate mechanism to fuse table schemas and table contents and get the more different representation about table schemas. We test this idea on the WikiSQL and TableQA datasets. Experimental results show that F-SQL achieves new state-of-the-art results on WikiSQL and TableQA.

INDEX TERMS Text2SQL, BERT, table schema, table content.

I. INTRODUCTION

Semantic parsing is to map natural languages to formal meaning representations. In particular, parsing SQL queries from natural languages is an important part of semantic parsing. Some researchers call it Text2SQL. Text2SQL can help non-professionals access databases and improve the efficiency of information utilization. It is a long-term research issue and has recently received attention from the relevant communities. Text2SQL requires the system to understand natural language questions and generate corresponding SQL queries.

Although there are already some solutions for Text2SQL task, most of them only consider table schemas and natural language questions when parsing SQL queries, and do not use table contents. We observe that table contents can provide more helpful information for some user questions. A typical example is shown in Figure 1. The blue font indicates the table schema. The black font and red font indicate the table content. The red font indicates the target value in the SQL query. The table schema consists of multiple column names. The table content contains multiple cells. For the given question "Where is Qingxiu Nancheng Department Store in Nanning?", since "Area" and "Region" have similar semantic information, if the model only considers table schema, it cannot distinguish them effectively. The table content of "Region" column contains the target value "Nanning". If the model can consider both the table schema and the table content, then columns with similar semantics would be easily distinguished.

In this work, we propose a novel approach, F-SQL, to focus on solving the problem of table content utilization. Based on SQL syntax, we employ the sketch-based approach [1] to generate SQL queries from natural language questions. The sketch-based approach is essentially a template filling method. The sketch can also be considered as the template. Our predefined SQL sketch is shown in Figure 2. It corresponds naturally to the syntactical structure of the SQL query. "SELECT" and "WHERE" represent keywords, and the blue parts indicate the slots. The sketch-based approach needs model to predict these slots to assemble SQL. Our proposed F-SQL contains multiple sub-models, and these sub-models respectively predict different slots in the sketch. Taking \$OP in Figure 2 as an example, it represents the conditional column operation, and its target set is ["<", ">", "==", "!="]. We use a classification sub-model to

The associate editor coordinating the review of this manuscript and approving it for publication was Weipeng Jing.

Database Table

Type	Area	Region	Name	Address
Merchandise	Guangxi	Fangchenggang	Jiahui Supermarket in Fangchenggang Port Area	No. 95-1 Xinggang Avenue
Merchandise	Guangxi	Nanning	Qingxiu Nancheng Department Store	No. 64 Minzu Avenue
Merchandise	Guangxi	Nanning	Baisha Nancheng Department Store	No. 20 Baisha Avenue, Nanning
……	……	……	……	……

Question: Where is Qingxiu Nancheng Department Store in Nanning?

SQL: SELECT Address WHERE Name= Qingxiu Nancheng Department Store AND Region=Nanning

Answer: No. 64 Minzu Avenue

FIGURE 1. A typical example.

```
SELECT   ($AGG   $COLUMN)^(*>=1)
WHERE    $WOP    ($COLUMN   $OP   $VALUE)^(*>=0)
```

FIGURE 2. Predefined sketch.

predict it. The sub-models depend on the pattern of dataset, and they share the same pre-trained encoder. Compared with the traditional sketch-based program synthesis approaches [3]–[5], our approach can be viewed as a neural approach.

The slots which need be filled in Figure 2 can be divided into five types: column($COLUMN), quantity(*), relationship($WOP), operation($AGG, $OP) and value($VALUE). These slots are not isolated from each other. By training them jointly, we can achieve better SQL generation performance. We observe that the most challenging part in parsing SQL queries is column prediction($COLUMN). Because the information contained in table schemas(column names) is limited, relying on table schemas alone is not enough to distinguish columns with similar semantics. For this challenge, we find table contents can provide differentiate information to help model effectively distinguish these columns. We use the gate mechanism to fuse table schemas and table contents, and get more different column representations.

We test our idea on the WikiSQL[1] and TableQA[2] datasets. WikiSQL is an English dataset and TableQA is a Chinese dataset. The syntax of SQL in WikiSQL is relatively simple. TableQA is built by Zhuiyi Technology based on business considerations in an AI competition. The competition is held in June 2019, and we participate in this AI competition and finally win the first place.Compared with WikiSQL, TableQA is more complicated.

In this work, we focus on the table content utilization on the single-table Text2SQL generation task. Our contributions can be summarized into three folds. First, we are the first to consider the table contents based on neural technology in the single-table SQL generation task. Second, we propose a novel neural network approach F-SQL. F-SQL uses the gate mechanism to fuse table schemas and table contents, and get more different column representations. This method can help the model to distinguish columns with similar semantics and achieve better SQL generation performance. Although the gate mechanism is relatively simple, it shows a surprising improvement. Finally, our proposed F-SQL bypasses the previous state-of-the-art approaches and achieves the new state-of-the-art results on WikiSQL and TableQA.

II. RELATED WORK

Semantic parsing is to transform natural languages into formal meaning representations. The transformed representations are highly relevant to applications, ranking from question answering [9] to robot control [10]. A research area which is very similar to our task is Text2Code [11], [12]. The goal of this task is to parse the executable programming language(such as Python) from natural languages. As a subtask of semantic parsing, Text2SQL has a long research history. Its goal is to parse SQL queries from natural languages. Text2SQL can help non-professionals access the databases, and can also be used as a solution based on database Q&A. Early researches [13]–[16] focus on domain databases and rely on domain experts to build corresponding parsing rules. These approaches generally involve more manual feature engineering. Although they work well on special databases, the generalization is weak and re-designed parsing rules are needed on new databases.

WikiSQL [6] is one of the first large-scale Text2SQL databases, and it contains 80,654 pairs of natural language question and corresponding human-annotated SQL query. It involves 24,241 tables which are from Wikipedia. The large data size of WikiSQL can take neural technology, and it has newly attracted much attention from relevant communities. Note although the research on Text2SQL is valuable, manual data annotation is still difficult due to the professional syntax knowledge of SQL. This situation leads to a few large-scale Text2SQL datasets. Compared with real application scenarios, WikiSQL makes many simplifications. For example, WikiSQL assumes that the user question must contain the table cells. TableQA is the first large-scale Chinese Text2SQL dataset containing 45,918 pairs about question and annotated SQL query. Compared to WikiSQL, the question forms are highly differentiated on TableQA. TableQA does not impose any constraint on user questions.

Earlier researches based on neural semantic parsing consider the Text2SQL task as a sequence generation problem,

[1]https://github.com/salesforce/WikiSQL

[2]https://tianchi.aliyun.com/competition/entrance/231716/introduction?spm=5176.12281949.1003.1.503b2448m6OhlF&lang=en-us

and employ the sequence-to-sequence neural network structure [17] with attention mechanism [18] and copy mechanism [19]. They learn the mapping relationship between the user question and the SQL query through the "encoder-decoder" framework. Seq2SQL [6] breaks the SQL query into two components and the keywords of these two components are "Select" and "Where". It introduces the reinforcement learning mechanism to train the model, and then generates the two components of SQL query independently. The performance of Seq2SQL exceeds the general sequence-to-sequence Text2SQL generation model.

Some studies have found that using SQL syntax rules can constrain the output space and improve SQL generation performance. Xu *et al.* [1] thinks Seq2SQL is effected by the conditional order, but the conditional order does not affect the SQL execution result. They propose a sketch-based approach SQLNet. SQLNet takes the sequence-to-set method to further simplify the SQL query generation. It divides the Text2SQL task into 6 subtasks which predict the filled slots in the predefined sketch. Only the value in where clause is generated by the sequence-to-sequence structure, and others in the predefined sketch use the classification structure. Therefore, SQLNet is no longer effected by conditional order. Yu *et al.* [20] believe that additional entity type information can improve SQL generation performance, and propose TypeSQL model. TypeSQL introduces the entity information for every token of the question by the external knowledge. It also employs the sequence-to-set method based on the sketch and breaks the Text2SQL into 6 subtasks. The difference is that TypeSQL shares the parameters of some encoders, reducing from the original 12 encoders to 6 encoders. Coarse2Fine [21] takes a two-stage approach to generate SQL queries from natural languages. It first generates a rough intermediate representation, and then modifies the intermediate representation based on the predefined set of sketches. McCann *et al.* [22] regard the Text2SQL task as a multi-task question answering problem, and propose MQAN. MQAN is a multi-task question answering neural framework and it can jointly train multiple subtasks. Wang *et al.* [23] assume that all generated SQL queries are able to be executed by computers. They propose an execution-guided decoding strategy in the SQL query generation phase. This strategy need remove non-executable SQL queries from all candidate SQL queries based on table information. Shi *et al.* [24] propose IncSQL based on the sequence-to-action approach. IncSQL is able to encode syntax through the available actions in the predefined inventory.

Although neural network technology has been successfully applied to many natural language processing tasks, its performance is affected by the scale of high-quality training data. In general, the performance of neural networks becomes strong with the increase of high-quality training data. For the Text2SQL task, although available datasets such as WikiSQL already contain tens of thousands of training samples, its size still cannot cover all natural language expressions. Fortunately, pre-training techniques [25]–[27] using large-scale external data recently develops rapidly. and have showed promoting performance on many natural language tasks. Hwang *et al.* [7] use the pre-trained BERT to replace the static language model GloVe [28] as the encoder, and propose SQLova on the WikiSQL dataset. After the encoder, they introduce three structures, of which SQLova is the core and has achieved the outstanding results. SQLova is similar to SQLNet, except that the encoder is significantly different. SQLova uses pre-trained BERT, and SQLNet uses static language model GloVe. The performance comparison between SQLova and SQLNet proves the effectiveness of pre-training techniques on the Text2SQL task. In addition, Wang *et al.* claim that their proposed SQLova has exceeded human level. He *et al.* use MT-DNN [29] as the pre-trained weights for BERT and propose X-SQL. They think that MT-DNN can make the pre-trained model learn more label information by multi-task learning framework, and make the Text2SQL task benefit from the related label information captured by MT-DNN. The output structure of X-SQL is relatively simple, and BERT encoder plays the main fitting role. In addition, X-SQL replaces [CLS] with [XLS], and relearns the semantic information of the input sequence to get a better downstream representation for multiple subtasks.

Text2SQL studies have shown increasing results, but these studies do not make full use of database information. When generating SQL queries from natural languages, they only consider table schemas and natural language questions, and do not use table contents. This method of data utilization brings challenges to COLUMN prediction in the sketch. If the table schema contains columns with similar semantics, the previous approaches cannot distinguish them effectively based on semantic information of these columns. In this work, we explore the use of table contents.

III. METHODOLOGY

In this section, we introduce our proposed F-SQL model. First, we define the sketch-based Text2SQL task. Then, we describe the overall framework of F-SQL and related calculation details. In particular, 3.4 "Enhanced column representation" is the core in our work, focusing on how to enhance column representation by fusing table schemas and table contents. Finally, we introduce related implementation details.

A. PROBLEM DEFINITION

Text2SQL task is parsing SQL queries automatically from natural language questions. The syntax of SQL queries is relatively regular. The sketch-based approaches are the current main solutions. They essentially belong to the template-filling framework and generate SQL queries by predicting the slots in the predefined sketch. Compared with seq2seq approach, the sketch-based approach is simpler and under control. Our proposed F-SQL also adopts the sketch-based approach. For the single-table Text2SQL task, the SQL sketch we define is shown in Figure 2.

"SELECT" and "WHERE" represent keywords in the SQL queries. We assume that every SQL query must contain

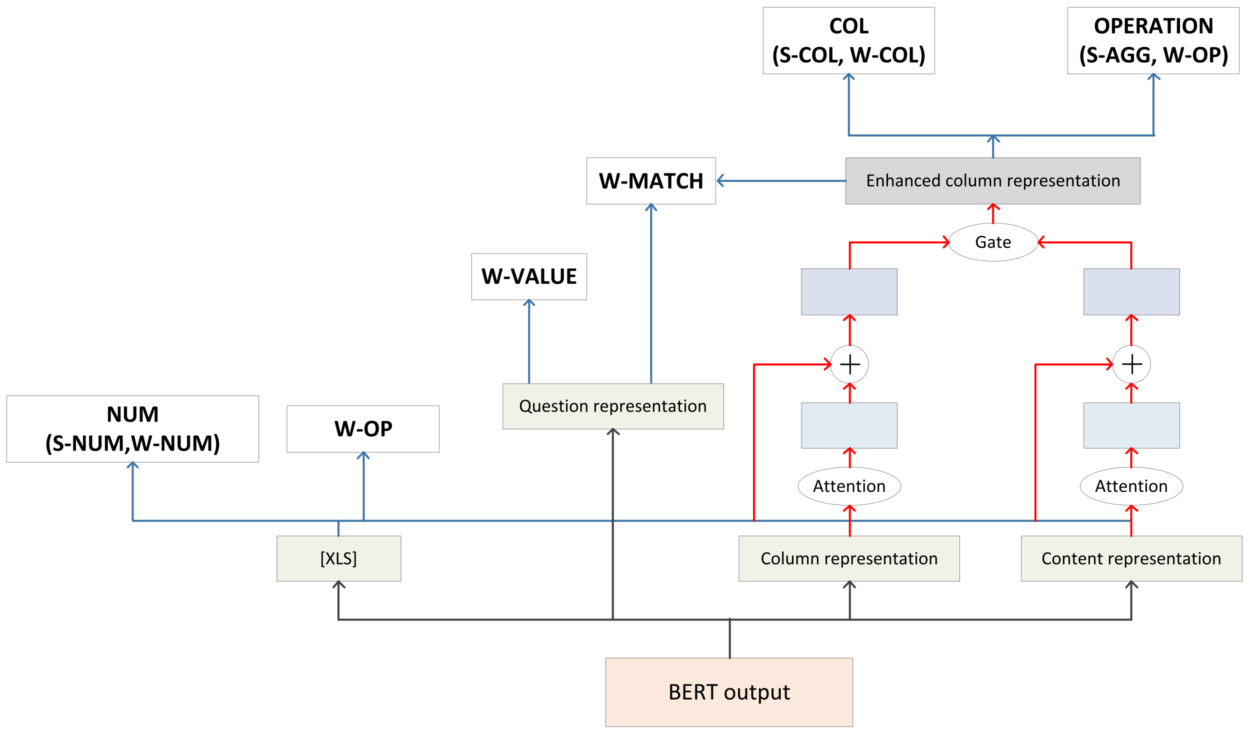

FIGURE 3. F-SQL neural network architecture.

the "SELECT" keyword. The blue tokens starting with "\$" are slots to fill. These slots can be divided into five types: column(\$COLUMN), quantity(*), relationship(\$WOP), operation(\$AGG, \$OP) and value(\$VALUE).

\$COLUMN represents the column name in the table. We refer to the column appearing in select clause and the column appearing in where clause as selected column and conditional column respectively. The model needs to determine whether the columns in the table appear in select clause or where clause. This task can be viewed as two binary classification problems. Let $c_1, c_2, \ldots, c_m$ be columns in the table, $s_1, s_2, \ldots, s_m$ be whether the corresponding column appears in select clause, $w_1, w_2, \ldots, w_m$ be where the corresponding column appears in where clause. In particular, $s_i \in [0, 1]$ and $w_i \in [0, 1]$. 1 means yes.

"*" is the number of columns to be predicted. The numbers of target columns in the SQL query are not fixed, and the model needs to predict the corresponding numbers. We can determine the final selected columns and conditional columns based on the predicted numbers and the probabilities of the selected columns and conditional columns. We assume that each SQL query contains at least one select column, and the number of conditional columns is not limited. The prediction of the number of related columns can be regarded as a classification problem.

\$AGG and \$OP are column-related operations, respectively aggregation operations and conditional operations. \$AGG aggregation set is ["", "AVG", "MAX", "MIN", "COUNT", "SUM"], and "" means there is no related aggregation. \$OP conditional operation set is ["<", ">", "==", "!="]. \$WOP is the relationship between conditions in where clause. When the number of conditional columns is 1, \$WOP is NULL. When the SQL query contains multiple conditional columns, \$WOP can be "AND" or "OR".

\$VALUE is the value associated with the conditional column in where clause. We consider value prediction as a problem of extracting value from the question. The value extracted from the question is not necessarily the cell in the table. So for the string-type column, we select a cell from the table based on user question and extracted value as the final value to generate SQL.

The slots in the sketch are not isolated from each other. We consider these slot prediction tasks as multiple subtasks, and jointly train them to establish the connection between them. The basic task of Text2SQL is to generate SQL queries based on user questions and table schemas. We take the table contents as an additional input. In general, the input of our model contains user question, table schemas(columns) and table contents(cells). The SQL queries are assembled from results of multiple subtasks based on the predefined sketch.

B. F-SQL

The sketch-based approach uses multiple sub-models to fill slots in the predefined sketch. The neural network structure of F-SQL is shown in Figure 3. It contains three components, encoder, enhanced column representation, multiple subtask outputs. We use BERT as our encoder.

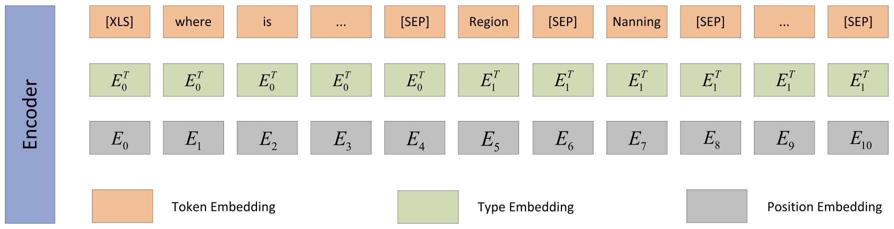

FIGURE 4. **BERT encoder structure.**

Column prediction is one of the biggest challenges in the single-table Text2SQL task. Some tables contain columns with similar semantics, and previous approaches are not able to distinguish them by column names alone. In the enhanced column representation, F-SQL uses the attention mechanism to obtain the column representation and content representation. Then it takes the gate mechanism to fuse them to obtain the enhanced column representation. For columns with similar semantics, the enhanced column representation is more discriminating due to the addition content information.

In order to fill the slots in the sketch, our F-SQL contains multiple sub-models, which can be divided into 5 types. Note these sub-models share the same BERT encoder.

- **NUM**. It is responsible for predicting the number of columns and includes two sub-models S-NUM and W-NUM, respectively corresponding to the selected columns and conditional columns.
- **COL**. It predicts the columns and includes sub-models S-COL and W-COL, respectively corresponding to the selected columns and conditional columns.
- **OPERATION**. It predicts column-related operations, including aggregation operations S-COL and conditional operations W-COL.
- **W-OP**. It predicts the relationship between conditions in where clause.
- **W-VALUE**. It extracts target values from the user question.
- **W-MATCH**. It determines which column the extracted value belongs to.

C. ENCODER

The model uses BERT as the encoder. Input sequence consists of three components, question, table schema and table content. Table schema means the multiple column names in the table. Column name and corresponding content are concatenated in order. Note GPU capacity is limited, and all content cells cannot be processed at the same time. So for each column, we use the cell most similar to the question as a substitute for table content. For the example in Figure 1, the question is "Where is QingXiu Nancheng Department Store in Nanning?", and the content of the "Region" column is ["Fangchenggang", "Nanning", "Nanning"...]. For the "Region" column, the cell most similar to the user question is "Nanning". The model uses "Nanning" as a substitute for the table content of the "Region" column. We use Rouge-L as the similarity for question and table content. The input can be expressed as follows:

$$[XLS], Q_1, Q_2, \ldots, Q_L, [SEP], S_{11}, S_{12}, \ldots, [SEP], C_{11}, C_{12}, \ldots, [SEP], S_{21}, S_{22}, \ldots, [SEP], C_{21}, C_{22}, \ldots, [SEP], \ldots, [SEP], S_{n1}, S_{n2}, \ldots, [SEP], C_{n1}, C_{n2}, \ldots, [SEP] \quad (1)$$

where [XLS] is the first token of the input sequence, indicting the input sequence information. [SEP] is the separator which separates the question, table schema and table content. Q_i and L represent the i-th token of the question and question length respectively. S_i is the i-th column name, and C_i is the i-th column content. In particular, S_{ij} is the j-th token of the i-th column name and C_{ij} is the j-th token of the i-th column content. n indicates the number of columns in the table.

BERT contains three embedding weight matrices, corresponding to token, type and position. BERT encoder structure is described in Figure 4. Token embedding, Type embedding and Position embedding encode word information, type information and position information of the input sequence respectively. In order to distinguish question and column, the model uses E_0^T to identify the question, and E_1^T to identify the column. In addition, the original BERT uses [CLS] tag as the first token in the input sequence, and its semantic vector has been pre-trained on external large-scale corpus. We think that since subsequent subtasks depend on the [CLS] representation, taking pre-trained [CLS] would reduce the convergence effect and make the model get the local optimal solution earlier. We take their method from He *et al.* [8]. We replace [CLS] with [XLS] and retrain the input sequence representation [XLS] on the target Text2SQL task.

D. ENHANCED COLUMN REPRESENTATION

The model uses the BERT encoder to encode the input sequence to obtain the BERT output vector. As shown in Figure 3, the BERT output vector contains four parts, [XLS] sequence information representation, question representation, column representation, content representation. We use h to label them as follows.

$$h_{[XLS]}, h_{q1}, h_{q2}, \ldots, h_{qL}, h_{[SEP]}, h_{s11}, h_{s12}, \ldots, h_{[SEP]}, h_{c11}, h_{c12}, \ldots, h_{[SEP]}, h_{s21}, h_{s22}, \ldots, h_{[SEP]}, h_{c21}, h_{c22}, \ldots, h_{[SEP]}, \ldots, h_{[SEP]}, h_{sn1}, h_{sn2}, \ldots, h_{[SEP]}, h_{cn1}, h_{cn2}, \ldots, h_{[SEP]} \quad (2)$$

where $h_{[XLS]}$ is the first token representation output by BERT and it represents the sequence information. h_q and h_{qi} represent the question representation and the i-th token representation of the question respectively. h_s and h_{sij} represent the column representation and the j-th token representation of the i-th column name respectively. h_c and h_{cij} represent the content representation and the j-th token representation of i-th column content respectively. L and n represent the length of the question sequence and the number of columns in the table respectively.

Each column name or column content may contain multiple tokens. Some tokens are highly relevant to the question, while others are just the opposite. These tokens should not be treated equally when calculating column representation and content representation. In order to highlight the tokens related to the question, the model uses the attention mechanism to learn new representations for each column name and column content. The column representation with attention is calculated as follows.

$$s_{sij} = dot(U_1 h_{[XLS]}, V_1 h_{sij}) \tag{3}$$
$$a_{sij} = \frac{exp(s_{sij})}{\sum_{j=1}^{n_i} exp(s_{sij})} \tag{4}$$
$$r_{si} = \sum_{j=1}^{n_i} a_{sij} h_{sij} \tag{5}$$

where U_1, V_1 are learnable parameters. *dot* is the dot product used to calculate the similarity of two vectors. s is the similarity and a is the attention weight. Subscript ij represents the j-th token of the i-th column name. r_{si} represents the i-th column representation with attention.

Similar to the column, the content representation with attention is calculated as follows.

$$s_{cij} = dot(U_2 h_{[XLS]}, V_2 h_{cij}) \tag{6}$$
$$a_{cij} = \frac{exp(s_{cij})}{\sum_{j=1}^{n_i} exp(s_{cij})} \tag{7}$$
$$r_{ci} = \sum_{j=1}^{n_i} a_{cij} h_{cij} \tag{8}$$

where U_2, V_2 are learnable parameters. Subscript ij represents the j-th token of the i-th column content. r_{ci} is the i-th content representation with attention.

He *et al.* [8] propose that adding sequence information $h_{[XLS]}$ to r_{si} can help model better align question and column. We think their opinion is effective. Our model contains multiple subtasks and $h_{[XLS]}$ is the input of many subtasks. Adding $h_{[XLS]}$ can also strengthen the connection between multiple subtasks and systematically improve the SQL generation performance. We add $h_{[XLS]}$ to r_{si} and r_{ci}.

$$r_{si} = r_{si} + h_{[XLS]} \tag{9}$$
$$r_{ci} = r_{ci} + h_{[XLS]} \tag{10}$$

We use the gate mechanism to fuse the column representation r_{si} and the content representation r_{ci}, and get the new enhanced column representation r_{ei}. We observe that although table content can provide helpful information in SQL query generation, not every cell is beneficial. The gate mechanism can be used to control the flow of information, and selectively incorporate the content information into the column representation. The successful application of the gate mechanism on LSTM [31] and GRU [32] has proven that the gate mechanism is an effective method to control the flow of information. The fusion calculation based on the gate mechanism is as follows.

$$\theta = sigmoid(U_3 r_{si} + V_3 r_{ci}) \tag{11}$$
$$r_{ei} = \theta \odot r_{si} + (1 - \theta) \odot r_{ci} \tag{12}$$

where U_3, V_3 are learnable parameters. r_{ei} is the i-th enhanced column representation obtained by fusing the column representation r_{si} and the content representation r_{ci}.

E. PREDICTION

Our proposed sketch is highly aligned with the SQL syntax. The model predicts the slots in the predefined sketch and assembles these slots to generate SQL query. The model uses the [XLS] sequence representation $h_{[XLS]}$, the enhanced column representation r_e and the question representation h_q from the BERT output to predict slots in the sketch. Our proposed model contains multiple sub-models. These sub-models are jointly trained and share the same underlying encoder, as shown in Figure 3.

We use NUM sub-models to predict the number of columns. In particular, we use S-NUM to predict the number of selected columns and the target output space is determined by the SQL pattern in the dataset. Similar, we use W-NUM to predict the number of conditional columns. The calculation details of S-NUM and W-NUM are the same, only the parameters are different. We model them as classification problems. If the size of the target space is 2, the model uses sigmoid as the activation function, otherwise softmax is used as the activation function. Taking TableQA as an example, the calculation method of W-NUM is as follows.

$$P^{W-NUM} = softmax(W_1 h_{[XLS]}) \tag{13}$$

where W_1 is the learnable parameter. P^{W-NUM} indicates output probability of the number of conditional columns.

We use W-OP to predict the relationship between conditions in where clause. When the number of conditional columns is greater than 1, the relationship between conditions is "OR" or "AND". When the SQL query has only one conditional column, the relationship is "NULL". We regard it as a classification problem and related calculation is as follows.

$$P^{W-OP} = softmax(W_2 h_{[XLS]}) \tag{14}$$

where W_2 is the learnable parameter. P^{W-OP} is the output probability of the relationship between conditions in where clause.

We use COL sub-models to predict columns. S-COL predicts the selected columns and it determines whether each column in table schema would be selected. W-COL predicts the conditional columns and it determine whether each column in table schema would appear in where clause. The calculation methods of S-COL and W-COL are the same and related parameters are different. They take the enhanced column representation r_e as the input. The calculation detail of S-COL is as follows.

$$P_i^{S-COL} = sigmoid(W_3 r_{ei}) \tag{15}$$

where W_3 is the learnable parameter. r_{ei} is the i-th enhanced column representation in table schema. P_i^{S-COL} is the probability that the i-th column appears in select clause. The model selects the ns columns with highest probability as the selected columns based on Equation 15. ns represents the predicted number about selected columns.

We use OPERATION to predict column-related operations. In particular, S-AGG is used to predict the aggregation operation related to the selected column and the target output space is ["", "AVG", "MAX", "MIN", "COUNT", "SUM"]. We use OP to predict the operation related to the conditional column and the target output space is [">", "<", "==", "!="]. Both S-AGG and OP take the enhanced column representation r_e as the input. They are calculated in the same way, and related parameters are different. The calculation about S-AGG is as follows.

$$P_i^{S-AGG} = softmax(W_4 r_{ei}) \tag{16}$$

where W_4 is the learnable parameter. P_i^{S-AGG} is the aggregation probability associated with i-th selected column.

We treat value prediction as an information extraction problem. Most of the previous researches extract corresponding values based on the column representation. We observe that for SQL queries containing multiple conditional columns, these approaches are not able to accurately differentiate conditional columns with similar semantic. We use the processing method in M-SQL [2] to divide the value extraction based on the column representation into two modules, W-VALUE and W-MATCH. W-VALUE does not consider the column representation and directly extracts all values from the question. Our model takes the joint training framework and the loss of each subtask should not differ too much. So W-VALUE uses 0-1 labelling instead of CRF [33] which has a large loss. 0-1 labelling means that every token in the question is marked as 0 or 1, and 1 means that this token needs to be extracted.

$$P_i^{W-VALUE} = sigmoid(W_5 h_{qi}) \tag{17}$$

where W_7 is the learnable parameter. $P_i^{W-VALUE}$ is the probability i-th token being extracted in the question. h_q represents the question representation and h_{qi} is the i-th token representation of question.

The model uses W-VALUE to extract all values from the question. Then W-MATCH determines whether the extracted value and the conditional column match. We consider W-MATCH as a matching problem. If the value does not match the conditional column, this pair is labelled 0, otherwise 1. The value is a segment which the model extracts from the question and we use the mean of this segment representation h_q to represent the value.

$$h_v = \frac{\sum_{i=s}^{e} h_{qi}}{e-s+1} \tag{18}$$

$$score_i = sigmoid(u \cdot tanh(W_6 h_v + W_7 r_{ei})) \tag{19}$$

where s and e are the start and end index of the value in the question. $e-s+1$ indicates the length of the extracted value and h_v indicates the value representation. W_6, W_7 are learnable parameters. $score_i$ represents the matching score of i-th conditional column and the value. We assign the value to the conditional column with the highest matching score.

F. IMPLEMENTATION DETAILS

In order to make our experiments reproducible, we will introduce some implementation details.

1) LOSS

Our model takes multi-task learning framework with multiple subtasks. These subtasks are essentially classification problems, 0-1 binary classification or multi-class classification. We use the cross-entropy of the output probability and the truth label as the optimization goal. Taking the S-COL as an example, its loss is as follows.

$$loss^{S-COL} = -\frac{1}{n}\sum_{i=1}^{n}(y_i log P_i^{S-COL} + (1-y_i) log(1-P_i^{S-COL})) \tag{20}$$

In this function, P_i^{S-COL} represents the probability that the i-th column in table schema is selected. n is the number of columns in the table schema. Loss of our model F-SQL is the sum of the loss of multiple subtasks. We do not use high-loss structures such as CRF in subtasks. This can make the loss of multiple subtasks as close as possible to improve the convergence performance. Note the initial loss of subtask W-MATCH is high. As the training progress, its loss gradually decreases to the level of other subtasks.

2) WEIGHT SHARING

Subtasks in F-SQL share underlying framework weights, including BERT encoder and the enhanced column representation. Some subtasks use the same input, for example both S-NUM and W-OP take the [XLS] sequence representation as the input. We observe that subtasks are not isolated from each other. The model can establish connections between subtasks by sharing the underlying weights, and help other models converge to further improve SQL generation performance. In addition, weight sharing can reduce the number of parameters and improve the speed of model training.

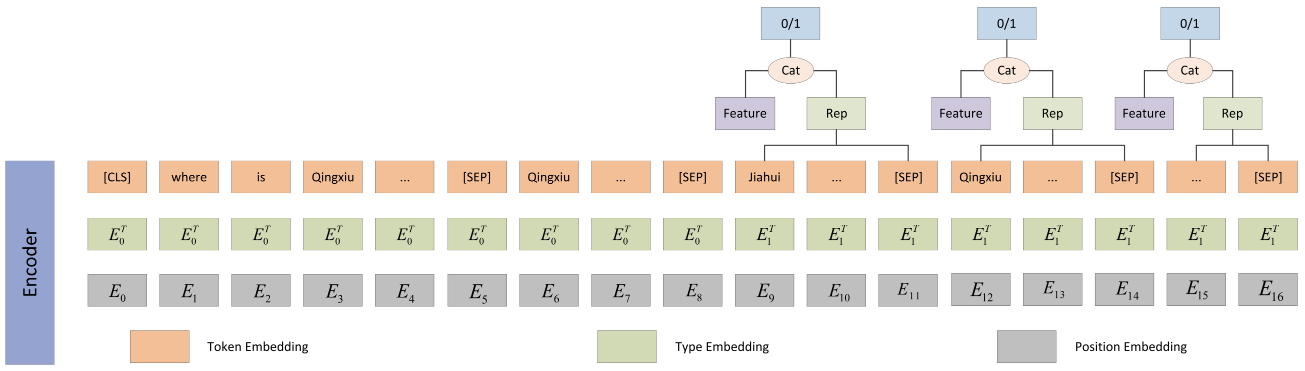

FIGURE 5. Semantic retrieval model structure.

3) FINAL VALUE

The question may not contain the table cell exactly, and it just contains synonyms for table cells. For the string-type column, the value extracted from the question cannot directly fill the $VALUE in the sketch. The model needs to determine the final value from the table content. We use the semantic retrieval model to determine the final value. The structure of the semantic retrieval model is shown in Figure 5. The input consists of three parts, user question, extracted value and table content. Feature in Figure 5 represents the statistical features between user question, extracted value and table content. We concatenate the statistical features and the BERT encoding about table content as the representation of the table content. Finally, the model determines which cell is the final value according to the representation of the table content. We choose the table cell with the highest output probability as the final value.

IV. EXPERIMENT

In this section, We evaluate our proposed F-SQL on WikiSQL and TableQA. In the following, we first introduce experiment setting, including dataset, evaluation methods and parameter setting. Secondly we compare F-SQL with other state-of-the-art approaches in SQL query generation performance. Then we show the performance about M-SQL and F-SQL on various subtasks. Finally we explore several ways to utilize table content.

A. EXPERIMENT SETTING

Manual data annotation is still difficult due to the professional syntax knowledge of SQL. To our knowledge, WikiSQL and TableQA are the two largest datasets in the single-table Text2SQL task. WikiSQL is an English dataset and contains 80,654 question and annotated SQL query pairs. WikiSQL is proposed in 2017. TableQA is a Chinese dataset and contains 45,918 question and annotated SQL query pairs. TableQA is proposed by Zhuiyi Technology in an AI competition. Compared with WikiSQL, TableQA is more complicated. It does not impose any restrictions on the question and the value in the sketch may not appear in the question. We use WikiSQL and TableQA as our experimental data. Note Zhuiyi Technology agree to open TableQA after the competition is over. Unfortunately, only training and validation data are currently open and test data is not yet accessible. In order to effectively evaluate the performance of our model, we divide the validation data equally into two portions, one for adjusting parameters and supervising training, and another for testing performance.

We use three metrics to compare our F-SQL and other state-of-the-art approaches on WikiSQL and TableQA dataset. **Logical-form accuracy**. It compares the predicted SQL query with ground truth and verifies if they match exactly. Since the order of the conditions in where clause does not affect the execution result of the SQL query, we do not consider the order of conditions. This process can eliminate false negatives due to conditional order. **Execution accuracy**. It compares the results of the predicted SQL query and ground truth and verifies if they are equal. Compared with Logical-form accuracy, its results are high and some false positives may be generated. **Mean accuracy**. It is the mean of the first two metrics. Compared with them, Mean accuracy is more suitable. For the expediency of description, in the rest of this paper, we use **LX**, **X** and **MX** to represent Logical-form accuracy, Execution accuracy and Mean accuracy.

We implement our proposed F-SQL model based on the pre-trained BERT with Whole Word Masking and use Pytorch as our backend. We count the length of input sequence in datasets and observe that the maximum length does not exceed 512. This means that BERT can directly process the input sequence. In order to avoid losing information, we do not truncate the input sequence and take the maximum length of the input sequence in dataset as input length. We use BertAdam with $lr = 2e-5$ as our optimizer and use the early stopping strategy to supervise the training process.

B. OVERALL PERFORMANCE

Table 1 shows the experimental results of our proposed F-SQL and some state-of-the-art models on WikiSQL and

TABLE 1. The overall performance on TableQA.

Model	WikiSQL Test LX(%)	WikiSQL Test X(%)	WikiSQL Test MX(%)	TableQA Test LX(%)	TableQA Test X(%)	TableQA Test MX(%)
SQLNet [1]	-	68.0	-	61.4	67.2	64.3
Coarse2Fine [21]	71.7	78.5	75.1	72.6	76.7	74.7
MQAN [22]	75.4	81.4	78.4	74.8	78.8	76.8
SQLova [7]	80.7	86.2	83.5	81.7	85.8	83.8
X-SQL [8]	83.3	88.7	86.0	83.3	87.6	85.5
M-SQL [2]	83.7	88.6	86.2	88.8	91.6	90.2
F-SQL	**85.6**	**91.4**	**88.5**	**90.4**	**93.2**	**91.8**

TABLE 2. The performance(%) of subtasks on WikiSQL.

Model	S-NUM	S-COL	S-AGG	W-NUM	W-OP	W-COL	OP	W-V-M	Test LX
M-SQL	-	**97.2**	90.7	98.7	-	96.6	99.1	**99.5**	83.7
F-SQL	-	96.8	**90.9**	**99.3**	-	**98.7**	**99.2**	99.4	**85.6**

TABLE 3. The performance(%) of subtasks on TableQA.

Model	S-NUM	S-COL	S-AGG	W-NUM	W-OP	W-COL	OP	W-V-M	Test LX
M-SQL	**99.5**	97.5	**98.9**	98.4	**98.8**	98.2	**99.1**	97.0	88.8
F-SQL	**99.5**	**97.7**	**98.9**	**98.6**	**98.8**	**99.1**	99.0	**97.6**	**90.4**

TableQA. We use bold to indicate the highest score in each column. Except for F-SQL, other models don't use the table content as additional input. In addition, we don't use the execution-guided decoding strategy [23] to generate SQL. Although the execution-guided decoding strategy can improve the performance of SQL generation, it reduces the efficiency of SQL generation and brings unfairness to the comparison between models.

The experimental results in Table 1 show that our proposed F-SQL is advanced and achieves the new state-of-the-art performance on WikiSQL and TableQA. F-SQL reaches 88.5% MX and 91.8% MX on WikiSQL and TableQA test data respectively. Compared to M-SQL, F-SQL shows +1.9% LX, +2.8% X and +2.3% MX on WikiSQL test data. F-SQL shows +1.6% LX, +1.6% X and 1.6% MX on TableQA test data. These improvements prove the effectiveness of our approach. We think that these improvement mainly come from two points, the use of table content and the gate mechanism module. Table contents can provide more support information for the model. The gate mechanism can help the model to selectively fuse information. For SQL query parsing task, we think that not all table contents are useful. Some table contents can provide gain to the model, while others are harmful.

C. SUBTASK PERFORMANCE

Table 2 and Table 3 show the performance of subtasks about M-SQL and F-SQL on WikiSQL and TableQA. Compared with TableQA, WikiSQL is relatively simple. In WikiSQL, each SQL query contains only one selected column. For SQL queries containing multiple conditions, the relationship between the conditions is only "AND". So the model does not need to predict S-NUM and W-OP subtasks on WikiSQL. The 9-th column "W-V-M" indicates the accuracy of the value prediction in where clause. It is the joint accuracy of the subtasks W-VALUE and W-MATCH. Subtasks have an order dependency when generating SQL quires. So it is not scientific to directly calculate the accuracy of each subtask. We take the subtask S-AGG as an example. The upstream subtask of S-AGG is S-COL. The model first needs to predict the selected columns through S-COL, and then uses S-AGG to predict the corresponding aggregation. If the model predicts the selected columns incorrectly, its corresponding S-AGG prediction would be meaningless. For downstream subtasks, we use the accuracy based on "conditional probability" as their performances. We still use S-AGG as an example. We assume that the correct number of samples for S-COL is n_1, and the correct number of samples for both S-COL and S-AGG is n_2. The performance of S-AGG is calculated as $\frac{n_2}{n_1}$

The experimental results in Table 2 and Table 3 show that our proposed F-SQL achieves the best performance on multiple subtasks. In conditional column prediction(W-NUM, W-COL), the performance gap of M-SQL and F-SQL is large. Compared with M-SQL, F-SQL shows +0.6% for W-NUM on WikiSQL and +0.2% for W-NUM on TableQA. F-SQL shows +2.1% for W-COL on WikiSQL and +0.9% for W-COL on TableQA. These improvements prove that our proposed approach is effective. Additional table contents can improve the performance about conditional column prediction. However we find that the performance of F-SQL on the selected column prediction is unstable. Compared with M-SQL, F-SQL shows −0.4%, +0.2% for S-COL on WikiSQL and TableQA. We think this phenomenon is caused by invalid table contents. The use of additional table contents brings interference to the prediction of selected columns. There are samples where the selected column and conditional column are easily confused. For these samples, the model incorrectly predicts the conditional column as the selected column. In addition, F-SQL reaches 97.6% about subtask "W-V-M". "W-V-M" is the bottleneck of Text2SQL on TableQA. After analysing the errors, we find that "W-V-M" prediction errors mainly focus on synonyms. Since the user question form is not limited, for the string-type conditional column, the model needs to search in table according to the question and the extracted value. We use the semantic retrieval model to search the final value. Compared with the Logistic Regression based on statistical features, the semantic retrieval method help F-SQL improve by 0.6% for W-V-M on TableQA.

D. TABLE CONTENT UTILIZATION

In order to mine table contents in depth, we explore various table content utilization methods based on the BERT pre-trained model. The performance of various table content utilization methods on TableQA is shown in Table 4. **None** means no table contents are used. **Concatenation** represents the concatenation of the column name and the table content. **Filter Concatenation** is based on **Concatenation**. If the similarity between the table content and the question is above the threshold, **Filter Concatenation** concatenates the column name and the table content. **Weight** represents weighting the column representation and the content representation instead of the gate mechanism. **Gate mechanism** is our proposed method. Note **Concatenation** and **Filter Concatenation** use M-SQL structure, and they concatenate the column name and the table content in the input. **Weight** and **Gate mechanism** use the same structure except the gate mechanism.

TABLE 4. The performance of various table content utilizations on TableQA.

Method	Test LX(%)	Test X(%)	Text MX(%)
None	89.2	91.9	90.6
Concatenation	88.9	91.5	90.2
Filter concatenation	89.9	92.4	91.2
Weight	90.0	92.6	91.3
Gate mechanism	**90.4**	**93.2**	**91.8**

The experimental results in Table 4 show that the table content utilization method based on the gate mechanism achieves the best performance. Interestingly, **Concatenation** reduces performance compared with not using table contents, showing -0.4% MX on the test dataset. We think that some table contents are not beneficial and directly concatenating the column name and the table content would bring redundant information and reduce model performance. Comparing **Filter concatenation** with **None**, **Filter concatenation** shows +0.6% MX on the test dataset. Filter concatenation uses the threshold to filter table content and only concatenates table content above the threshold. This approach effectively reduces the introduction of redundant information. Comparing **Filter concatenation** with **Weight**, **Weight** shows +0.1% MX on the test dataset. **Filter concatenation** uses table content to reinforce column information in the input, and the model treats the original column name and table content fairly. This treatment is not a perfect method. **Weight** treats the original column name and table content with different weights and gets the better enhanced column representation. Compared to **Weight**, **Gate mechanism** shows +0.5% MX on the test dataset. We consider the gate mechanism to be an advanced method of information fusion. It can selectively fuse original column name and table content, and fuse more useful information instead of all information. The experimental results in Table 4 also further prove the effectiveness of the gate mechanism.

V. DISCUSSION

In this paper, we focus on the single-table Text2SQL generation task. We consider the gate mechanism to fuse the table schema and table content. Our proposed F-SQL has achieved the new state-of-the-art results on WikiSQL and TableQA. In this section, we would discuss two topics, additional computational cost and table content utilization.

A. ADDITIONAL COMPUTATIONAL COST

We use the table content as an additional input of the model to enhance column representation. This method can effectively improve the performance of SQL query generation. Compared with M-SQL, F-SQL shows +2.3% MX, +1.6% MX on WikiSQL and TableQA. Although the additional table content can bring better performance, it will also reduce the efficiency of SQL generation. For each column in the table, the table content is a list which contains multiple cells. Since BERT can only process input sequences up to 512 in length and GPU capacity is limited, the model cannot process all cells at the same time. In response to this problem, we take a trick. For the table content related to each column, we select the cell with the highest similarity to the user question as a substitute. This tricky approach effectively reduces GPU capacity consumption and computational cost. We count the time consumption before and after using the table content on WikiSQL. When the model does not use table content, the training time about each epoch is 1281 seconds, and the inference time is 158 seconds on test data. When the model uses table content, the training time about each epoch is 2247 seconds, and the inference time is 211 seconds on test data. Additional table content utilization increases 75.4% training time and 33.5% inference time. Considering the performance improvement, we think that additional computational consumption is acceptable and meaningful.

B. TABLE CONTENT UTILIZATION

We use the gate mechanism to fuse table schema and table content. Due to the limitation of BERT length and GPU capacity, We use the cell with the highest similarity to the question as a substitute for the table content. We use Rouge-L as the similarity calculation method. This tricky approach effectively reduces the length of the input sequence and GPU capacity consumption. However, this approach has two drawbacks. First, Rouge-L measures the similarity of two texts from the string perspective, not semantics. For some synonyms with low coincidence, Rouge-L cannot accurately represent the similarity between them. This means that similarity calculation based on Rouge-L is not a perfect way. Secondly, the similarly between the target cell and the question may not be the highest, and it may be in the second or third place.The model selects the most similar cell from the table content, which may miss valid information. Moreover, it may bring negative effects to the model and reduce model performance. Considering the two shortcomings mentioned above, we would explore the more suitable similarity calculation method and how to use more table cells in the limited input length in the future. For the similarity method, we consider the approach in the graph applications [34]–[36] as a suitable alternative, but this approach may bring higher computational costs. For the use of more table cells, we observe that not every cell is useful. The model does not fuse the table contents for all columns, only for the possible target columns. Then we select multiple cells as the representative of the table content for the possible target columns. This way may further improve the use of information in the table contents and reduce redundant information.

VI. CONCLUSION AND FUTURE WORK

In this paper,we focus on the single-table Text2SQL generation task and propose F-SQL. Our proposed F-SQL achieves the new state-of-the-art results on WikiSQL and TableQA. F-SQL uses the gate mechanism to fuse table schema and table content. This fusion effectively strengthens the column representation and improves SQL query generation performance. F-SQL is a multi-task learning model, including multiple sub-models. We describe F-SQL and its sub-models in detail in Section 3. In Section 4, we introduce the relevant experimental results and analyse them. The experimental results prove the effectiveness of our proposed F-SQL. We discuss computational cost and table content utilization in Section 5.

In the future, we will continue to study in the following two directions.

- Limited by the Maximum length of BERT and GPU capacity, the table content utilization method we propose is not perfect yet. We will continue to study the table content utilization.
- In Text2SQL applications, whether user question can be parsed is an important step. No one has studied this topic yet. We will study this topic.

ACKNOWLEDGMENT

The authors would like to thank Zhuiyi Technology for TableQA dataset and WkiSQL dataset provided by Zhong *et al.*

REFERENCES

[1] X. Xu, C. Liu, and D. Song, "SQLNet: Generating structured queries from natural language without reinforcement learning," 2017, *arXiv:1711.04436*. [Online]. Available: http://arxiv.org/abs/1711.04436

[2] X. Zhang, F. Yin, G. Ma, B. Ge, and W. Xiao, "M-SQL: Multi-task representation learning for single-table Text2sql generation," *IEEE Access*, vol. 8, pp. 43156–43167, 2020.

[3] A. Solar-Lezama, "Combinatorial sketching for finite programsg," in *Proc. 12th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2006, pp. 404–415.

[4] R. Alur, "Syntax-guided synthesis," in *Proc. Formal Methods Comput.-Aided Des.*, Oct. 2013, pp. 1–4.

[5] J. Bornholt, "Optimizing synthesis with metasketchesg," in *Proc. 43rd Annu. ACM SIGPLAN-SIGACT Symp. Princ. Program. Lang.*, 2016, pp. 775–788.

[6] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," 2017, *arXiv:1709.00103*. [Online]. Available: http://arxiv.org/abs/1709.00103

[7] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on WikiSQL with table-aware word contextualization," 2019, *arXiv:1902.01069*. [Online]. Available: http://arxiv.org/abs/1902.01069

[8] P. He, Y. Mao, K. Chakrabarti, and W. Chen, "X-SQL: Reinforce schema representation with context," 2019, *arXiv:1908.08113*. [Online]. Available: https://arxiv.org/abs/1908.08113

[9] L. Zettlemoyer and M. Collins, "Online learning of relaxed CCG grammars for parsing to logical formg," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn. (EMNLP-CoNLL)*, 2007, pp. 678–687.

[10] S. Tellex *et al.*, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proc. Nat. Conf. Artif. Intell.*, 2011, pp. 1507–1514.

[11] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 440–450.

[12] M. Rabinovich, M. Stern, and D. Klein, "Abstract syntax networks for code generation and semantic parsing," in *Proc. 55th Annu. Meeting Assoc. for Comput. Linguistics*, 2017, pp. 1139–1149.

[13] D. H. D. Warren and F. C. N. Pereira, "An efficient easily adaptable system for interpreting natural language queries," *Comput. Linguistics*, vol. 8, nos. 3–4, pp. 110–122, Jul./Dec. 1982.

[14] I. Thanisch, "Masque/sql-an efficient and portable natural language Query interface for relational databasesg," in *Proc. 6th Int. Conf. Held*, Edinburgh, Scotland, 1993, p. 327.

[15] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proc. 8th Int. Conf. Intell. User Interface*, 2003, pp. 149–157.

[16] A. Giordani and A. Moschitti, "Translating questions to SQL queries with generative parsers discriminatively reranked," in *Proc. COLING*, 2012, pp. 401–410.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–7.

[19] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2692–2700.

[20] T. Yu, "TypeSQL: Knowledge-based type-aware neural text-to-SQL generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 588–594.

[21] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 731–742.

[22] B. McCann, N. Shirish Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," 2018, *arXiv:1806.08730*. [Online]. Available: http://arxiv.org/abs/1806.08730

[23] C. Wang, K. Tatwawadi, M. Brockschmidt, P.-S. Huang, Y. Mao, O. Polozov, and R. Singh, "Robust Text-to-SQL generation with execution-guided decoding," 2018, *arXiv:1807.03100*. [Online]. Available: http://arxiv.org/abs/1807.03100

[24] T. Shi, K. Tatwawadi, K. Chakrabarti, Y. Mao, O. Polozov, and W. Chen, "IncSQL: Training incremental Text-to-SQL parsers with non-deterministic oracles," 2018, *arXiv:1809.05054*. [Online]. Available: http://arxiv.org/abs/1809.05054

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.

[26] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. for Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 2227–2237.

[27] A. Radford, K. Narasimhan, and T. Salimans. (2018). *Improving Language Understanding by Generative Pre-Training*. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper

[28] J. Pennington, R. Socher, and C. D. M. Glove, "Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[29] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4487–4496.

[30] Y. Cui, W. Che, T. Liu, B. Qin, Z. Yang, S. Wang, and G. Hu, "Pre-training with whole word masking for chinese BERT," 2019, *arXiv:1906.08101*. [Online]. Available: http://arxiv.org/abs/1906.08101

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN Encoder–Decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.

[33] J. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[34] Y. Chen, X. Zhao, X. Lin, Y. Wang, and D. Guo, "Efficient mining of frequent patterns on uncertain graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 287–300, Feb. 2019.

[35] X. Zhao, C. Xiao*, X. Lin, W. Zhang, and Y. Wang, "Efficient Structure Similarity Search: A Partition-based Approach," The VLDB Journal, 2018, vol. 27, no. 1, pp. 53–78.

[36] W. Zeng, X. Zhao, J. Tang, and H. Shang, "Collective list-only entity linking: A graph-based approach," *IEEE Access*, vol. 6, pp. 16035–16045, 2018.

XIAOYU ZHANG received the M.S. degree from the National University of Defense Technology, Changsha, Hunan, China, where he is currently pursuing the Ph.D. degree with the Science and Technology on Information System Engineering Laboratory. His research interests include natural language processing, machine learning, and sequence-to-sequence learning.

FENGJING YIN received the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2011. He is currently a Lecturer with the College of Systems Engineering, National University of Defense Technology. His research interests include social networks analysis and data mining.

GUOJIE MA received the Ph.D. degree from the University of Technology Sydney, Sydney, Australia. She is currently a Postdoctoral Research Fellow with East China Normal University. Her research interests include big data analysis for finance, fintech, and knowledge graph.

BIN GE was born in Shandong, China, in 1979. He received the M.S. and Ph.D. degrees from the National University of Defense Technology. He is currently an Associate Professor with the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology. His research interests include natural language processing and social computing.

WEIDONG XIAO was born in Harbin, China, in 1968. He received the M.S. and Ph.D. degrees in management science and engineering from the National University of Defense Technology. He is currently a Professor with the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology. His research interests include intelligence analytics, natural language processing, and knowledge graph.