IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 16, 2020, accepted February 27, 2020, date of publication March 2, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977613

M-SQL: Multi-Task Representation Learning 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 (404338476@qq.com) and Fengjing Yin (yinfengjing@nudt.edu.cn)

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

ABSTRACT Text2SQL can help non-professionals connect with databases by turning natural languages into SQL. Although previous researches about Text2SQL have provided some workable solutions, most of them extract values based on column representation. If there are multiple values in the query and these values belong to different columns, the previous approaches based on column representation cannot accurately extract values. In this work, we propose a new neural network architecture based on the pre-trained BERT, called M-SQL. The column-based value extraction is divided into two modules, value extraction and value-column matching. We evaluate M-SQL on a more complicated TableQA dataset, which comes from an AI competition. We rank first in this competition. Experimental results and competition ranking show that our proposed M-SQL achieves state-of-the-art results on TableQA.

INDEX TERMS Text2SQL, M-SQL, pre-trained, multi-task learning.

I. INTRODUCTION

Semantic parsing refers to transforming natural languages into other meaningful logical representations which could usually be executed by computers. Representative work includes Text2Code [1], Text2SQL [2], Text2Sparql [3], etc. Text2SQL is an important task in semantic parsing. It can help non-professionals connect with databases by turning natural languages into SQL. This task has many potential applications in real life, such as question answering [4], robot navigation [5] and so on. In this work, we focus on the Text2SQL task.

Although the Text2SQL research is very meaningful, data annotation relies on high expertise and requires annotators to master SQL syntax, resulting in a small number of existing Text2SQL public datasets. The WikiSQL [6] is a large-scale Text2SQL dataset, which includes 80,654 text and SQL human annotation pairs. The presentation of WikiSQL dataset has attracted the attention of a large number of researchers in related fields and has produced a lot of related work. There are two main approaches to deal with the Text2SQL task, Seq2seq and Sketch-based approach. The typical representative of Seq2seq approach is Seq2SQL [6] which treats Text2SQL task as a translation task of text to SQL statement. Seq2SQL encodes the text through an encoder(CNN or RNN) to obtain the semantic representation of the text, and uses a decoder(CNN or RNN) to decode the semantic representation of the text to generate the SQL statement. Seq2SQL is the first effective work on WikiSQL. However, this approach does not consider the syntax rules of SQL statements, and the SQL generation accuracy is low.

```
SELECT $AGG $COLUMN
WHERE $COLUMN $OP $VALUE
(AND $COLUMN $OP $VALUE)*
```

FIGURE 1. Sketch-based approach.

In WikiSQL, the SQL statements belong to a fixed pattern which is shown in Figure 1 [2]. The model does not need to predict everything in the SQL statements, just fill in the key contents(blue parts in Figure 1). This method is called Sketch-based approach. The first work of Sketch-based Text2SQL is SQLNet [2], which transforms the Text2SQL task into six subtasks. These subtasks predict the blue parts of the sketch which need be filled. For WikiSQL datasets, Sketch-based subsequent researches also use similar task divisions, such as TypeSQL [7], Coarse2Fine [8], MQAN [9], SQLova [10], X-SQL [11], etc. SQLova and X-SQL introduce the pre-trained model BERT [12], and they achieve more robust

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

Table

Player	No.	Nationality	Position	Years in Toronto	School/Club Team
Antonio Lang	21	United States	Guard-Forward	1999-2000	Duke
Voshon Lenard	2	United States	Guard	2002-03	Minnesota
Martin Lewis	32, 44	United States	Guard-Forward	1996-97	Butler CC(KS)
Brad Lohaus	33	United States	Guard-Center	1996	Iowa
Art Long	42	United States	Guard-Center	2002-03	Cincinnati

Sample data

Situation 1: WikiSQL: Who is the player that wears number 42? ComplexSQL: Who is the player that wears number 42, and which country?
Situation 2: WikiSQL: Who was born in 1996? ComplexSQL: Who was born in 1996 and played for the Duke?
Situation 3: WikiSQL: Who was born in 1996 and played for the Duke? ComplexSQL: Who was born in 1996 or played for the Duke?
Situation 4: WikiSQL: Which player's nationality is United States? ComplexSQL: Who is the American player?

FIGURE 2. WikiSQL data description.

effects and basically reach the limit that WikiSQL dataset can be reached.

For WikiSQL dataset, although some advanced researches (SQLova, X-SQL) have almost reached the limit, this does not prove that the single-table Text2SQL task has been completely solved. Compared to the actual situation, WikiSQL has a lot of simplifications. The samples in WikiSQL are shown in Figure 2. Simplified situations are as follows:

1) WikiSQL assumes that the number of selected columns can only be 1.
2) WikiSQL contains fewer multi-conditional samples.
3) WikiSQL assumes that the conditional relationship in where clause can only be AND, and does not consider OR.
4) WikiSQL assumes that the contents of the databases must appear in the query, but in the actual application, the user's query is casual, and the contents of the databases don't necessarily appear in the user's query.

According to the actual situation of the single-table Text2SQL task, Zhuiyi Technology holds an AI competition[1] in which it proposes a more complicated Chinese Text2SQL dataset, called TableQA. TableQA dataset contains 45,918 samples. Compared to WikiSQL, the query form is more complicated, including all modes of ComplexSQL in Figure 2. The number of selected columns can be more than one(situation 1). TableQA contains more multi-conditional samples(situation 2) and adds ''OR'' logic relationship(situation 3). The form of query is diverse, and the contents of the databases may not appear in the corresponding query(situation 4). ComplexSQL in Figure 2 is just an example of our hypothesis, not the actual data from TableQA. TableQA samples[2] are shown in Figure 3.

Although both WikiSQL and TableQA belong to the single-table Text2SQL dataset, compared to WikiSQL, TableQA is more complicated and more realistic. State-of-the-art methods about WikiSQL, such as SQLova, X-SQL, do not handle TableQA very well. There are three main reasons. First, the task framework is different. TableQA need add two extra subtasks: one for predicting the number of selected columns and the other for predicting the conditional relationship in where clause. Second, SQLova and X-SQL are flawed. They extract values based on column representation. If there are multiple values in the query and these values belong to different columns, they cannot accurately extract values. Finally, TableQA's query form is more casual, and the contents of databases may not appear in the query.

We focus on the Text2SQL generation task on TableQA. We reconstruct the task framework of Text2SQL and propose M-SQL. M-SQL contains 8 sub-models, which are S-num, S-col, S-col-agg, W-num-op, W-col, W-col-op, W-col-val and W-cal-match. We would describe the details of M-SQL in section 3. Specially, our contributions in this work can be summarized into three folds.

1) First, for the more complicated single-table Text2SQL task TableQA, we extended the WikiSQL framework and proposed M-SQL.
2) Second, the existing models extract values based on column representation. This approach cannot accurately extract values for samples containing multiple values and multiple columns. We improved this extraction approach, and divided the column-based value extraction into two modules: value extraction and value-column matching.

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

[2]Download address: https://tianchi.aliyun.com/competition/entrance/231716/information

Data 1:

Query_en: The average daily volume of ChangSha in 2011 was 3.17, so what is the volume in the past week?

Query_zh: 长沙 2011 年平均每天成交量是 3.17，那么近一周的成交量是多少

SQL_en: SELECT 'Seven days trading' WHERE 'City' == ChangSha AND 'Daily trading' == 3.17

SQL_zh: SELECT '七日成交' WHERE '城市' == 长沙 AND '每日成交' == 3.17

Data 2:

Query_en: Please check the situation of the weekly fluctuations of SouFang and RenRen

Query_zh: 请查一查搜房网和人人网的周涨跌幅的情况

SQL_en: SELECT 'Weekly Fluctuation' WHERE 'Name == SouFang Or 'Name' == RenRen

SQL_zh: SELECT '周涨跌幅' WHERE '名称' == 搜房网 AND '名称' == 人人网

FIGURE 3. TableQA data samples.

3) Finally, we achieved the state-of-the-art results for TableQA task.

II. RELATED WORK

Research on Text2SQL has a long history [13], [14], and early researches focused on domain databases, relying on rules defined by domain experts. With the improvement of computer capabilities and the emergence of large-scale corpus, Text2SQL task can adopt deep neural network technology, and recently draws a lot of attention from related communities.

WikiSQL [6] is a large-scale Text2SQL corpus containing 80,654 samples on 24,241 tables from Wikipedia. WikiSQL samples are shown in Figure 2. The early researches [15], [16] based on the neural semantic parsing approach, focus on the Seq2seq approach and don't use SQL syntax to limit the output space. They divide the Text2SQL task into two parts, select clause and where clause. The two parts are independent of each other and are modelled separately. Seq2SQL [6] threats the Text2SQL task as a translation task which converts the query into the SQL statement. Through the encoding/decoding framework, the mapping between the query and the SQL statement is learned.

Some researchers have found that using SQL syntax can simplify the model and greatly improve the accuracy of SQL conversion by limiting the output space. Considering the problem that Seq2SQL is sensitive to conditional values in where clause, Xu *et al.* [2] proposes a sketch-based solution SQLNet. SQLNet uses the sequence-to-set approach to divide the Text2SQL task into six subtasks, each of which is responsible for predicting different parts of the SQL statement. Among them, the conditional value prediction still uses the seq2seq method, and the other parts adopt the classification method. Similar to SQLNet, TypeSQL [7] also uses the sequence-to-set structure, but it employs "type" information of each token in the query by knowledge graph. Although TypeSQL also divides the Text2SQL task into 6 subtasks, it simplifies the model encoders and reduces the original 12 encoders to 6 encoders, reducing the number of training parameters. Coarse2Fine [8] divides the Text2SQL task into two phases, first producing the intermediate presentation and then using the intermediate presentation to decode the where clause. MQAN [9] recommends a multi-task question and answer network structure which enables jointly learning multi-task representation by using various attention mechanisms. Wang *et al.* [17] proposes an execution guided decoding, which assumes that all generated SQL statements can be executed. It suggests to remove unexecutable SQL statements from the candidate SQL statements during the decoding phrase. IncSQL [18] proposes a sequence-to-action structure which fills slot values using feasible actions from the predefined inventory.

With the development of dynamic representation learning techniques(BERT [12], ELMo [19], GPT [20], etc.), more and more deep neural network solutions use dynamic representation learning models as encoders and achieve state-of-the-art results. SQLova [10] uses BERT as the encoder to encode queries, then obtain the semantic representations of the queries. Based on the dynamic representations of the queries, three variant models, Shallow-Layer, Decoder-Layer and NL2SQL-Layer, are proposed and achieve state-of-the-art results. The structure of NL2SQL-Layer is similar to SQLNet. SQLova validates the effectiveness of dynamic representation learning techniques in the Text2SQL task.

SELECT ($AGG $COLUMN)*
WHERE $WOP ($COLUMN $OP $VALUE)*

FIGURE 4. TableQA sketch.

X-SQL [11] uses MTDNN [21] to initialize BERT and proposes a simpler Text2SQL model. It uses the [XLS] to replace the [CLS] tag, and gets a better downstream representation. Based on the characteristics of the WikiSQL dataset, X-SQL employs [EMPTY] in samples without where clause. In addition, X-SQL regards the column selection as a ranking task, using the Kullback-Leibler(KL) as optimization target, improving the accuracy of the column selection. X-SQL has achieved state-of-the-art results on the WikiSQL dataset.

Although previous researches about Text2SQL have provided some workable solutions on WikiSQL dataset, most of them extract values based on column representation. If there are multiple values in query and these values belong to different columns, these approaches based on column representation cannot accurately extract values from the queries. Because WikiSQL contains few such samples, this flaw is minor. However, TableQA contains so many such samples that the defect is very obvious. Our proposed M-SQL can effectively improve the SQL generation accuracy for such samples.

III. METHODOLOGY

In this section, we introduce our proposed M-SQL. First, we define the task of sketch-based Text2SQL. Then, we describe the overall framework of M-SQL and related calculation details. Finally, we introduce the execution-guided decoding in the infer stage.

A. PROBLEM DEFINITION

The purpose of Text2SQL task is to automatically convert queries into machine-executable SQL statements. The sketch-based approach is the current mainstream solution for Text2SQL, which can take full advantage of the syntax rules of SQL statements. SQL statements have a fixed syntax format. In order, it consists of SELECT, WHERE, AND/OR keywords which are followed by the contents to be filled. The SQL sketch of WikiSQL is shown in Figure 1. TableQA is relatively complicated, and its SQL sketch is shown in Figure 4. The sketch-based Text2SQL task need fill the blue parts of Figure 4. Through the analysis of TableQA, we assume that the SQL statement generation rules are as follows:

- SQL statement template is shown in Figure 4.
- SELECT and WHERE represent SQL keywords, and we assume that every SQL statement must contain SELECT and WHERE.
- $WOP represents the conditional column relationship in where clause, and the relationship set is [“”, “AND”, “OR”]. “” means that there is no relationship between multiple conditional columns.
- $COLUMN indicates the column name of the databases. We name the column of select clause as selected column, the column of where clause as conditional column.
- $AGG represents the operation of selected columns, and the operation set is [“”, “AVG”, “MAX”, “MIN”, “COUNT”, “SUM”]. “” means no operation.
- $OP represents the operation of conditional columns, and the operation set is [“>”, “<”, “==”, “! =”]
- $VALUE represents the value of conditional columns, and the string-type values must be the contents of the databases. In addition, some queries are spoken, not necessarily containing target values.
- * indicates the quantity. We assume that the number set of selected columns is [1, 2], and the number set of conditional columns is [1, 2, 3].

B. M-SQL

The basic framework of our proposed M-SQL is a multi-task learning architecture. Its overall architecture is shown in Figure 5. M-SQL consists of three parts, encoder, column representation and several sub-models. In the encoder, we use the BERT-wwm-ext [22]. Chinese words have more coherence and semantic information than Chinese characters. When training BERT, Google’s BERT treats the Chinese character as a masking unit, while BERT-wwm-ext treats the Chinese word as a masking unit. BERT-wwm-ext uses the whole word masking strategy, which can better learn Chinese word vector representation. Related researches [22] show that, compared with Google’s BERT, BERT-wwm-ext has better effects on Chinese NLP tasks. In addition, the training corpus of BERT-wwm-ext is larger and has 5.4B more words than the original Wiki data.

Selected columns and conditional columns are the bottleneck the of sketch-based Text2SQL task. We use the “CONTENT REINFORCING LAYER” in X-SQL as the column semantic representation. The entire M-SQL model contains 8 sub-models. S-num, S-col and S-col-agg predict the select clause of the SQL statement. W-num-op, W-col, W-col-op, W-col-val and W-val-match predict the where clause of the SQL statement.

- S-num, predicting the number of the selected columns. The prediction set is [1, 2], and it is the two-class classification problem.
- S-col, predicting the selected columns. It is the single classification problem, and needs to determine whether every column in the databases is selected. In the inference stage, the model determines the selected columns according to the number of the selected columns and the output probability.
- S-col-agg, predicting the operations associated with the selected columns.
- W-num-op, predicting the relationship of the conditional columns in where clause and the number of the conditional columns. It is the seven-class classification problem. The prediction set is obtained from the TableQA dataset enumeration.

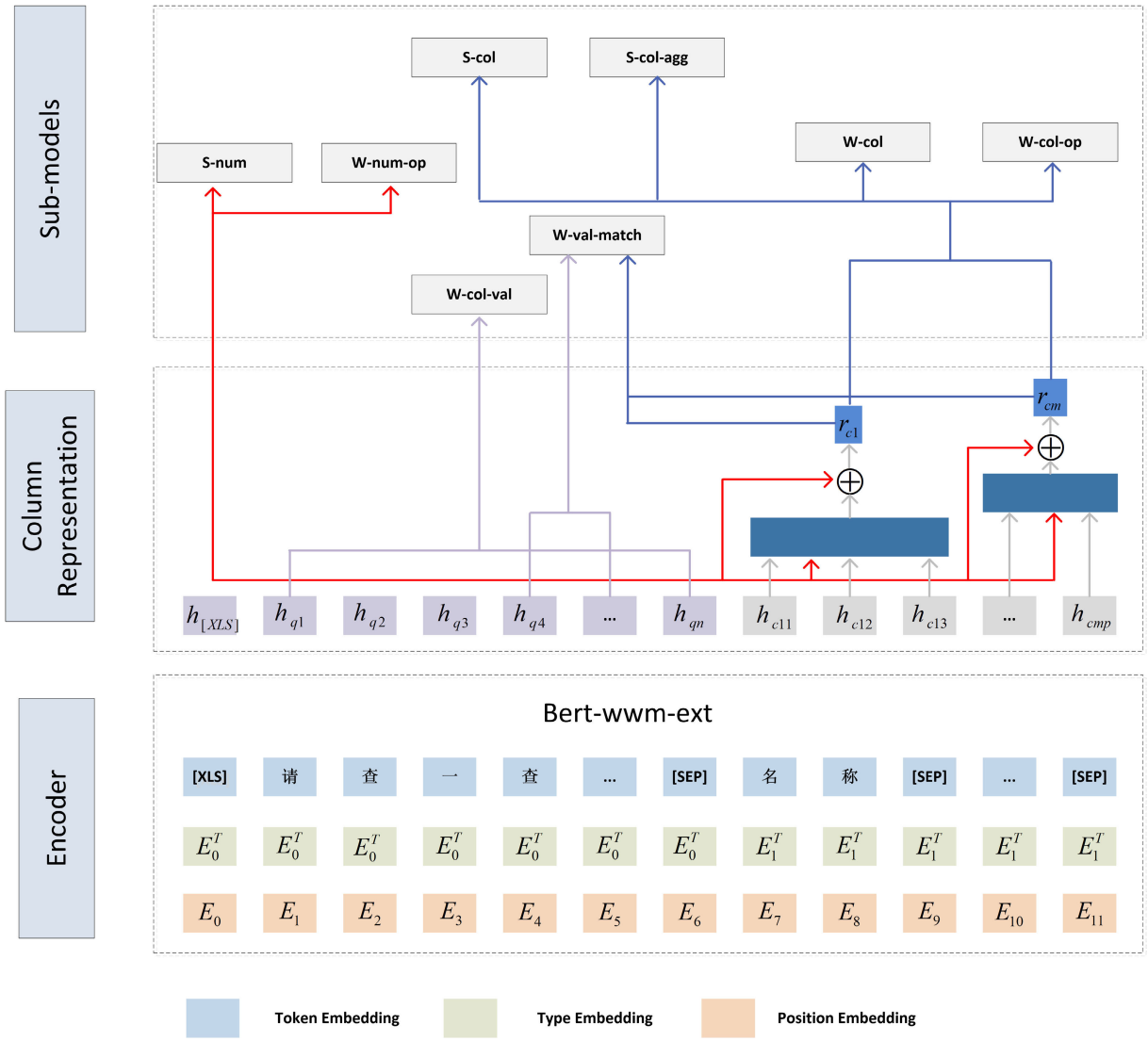

FIGURE 5. M-SQL neural network architecture.

- W-col, predicting the conditional columns. Similar to S-col, it is the single classification problem.
- W-col-op, predicting the operations associated with the conditional columns.
- W-col-val, extracting the values associated with the conditional columns from the queries.
- W-val-match, matching the conditional columns and the extracted values.

C. ENCODER

In the encoder, we concatenate the query with all columns in the databases and use [SEP] token to separate each other. Compared with English, Chinese words contain more semantic information. We use BERT-wwm-ext as the initial weights of BERT. The input sequence is as follows:

$$[XLS], T_1, T_2, \ldots, T_L, [SEP], H_{11}, H_{12}, \ldots, [SEP], \ldots, [SEP]H_{n1}, H_{n2}, \ldots, [SEP] \quad (1)$$

where [XLS] and [SEP] are special tags. [XLS] represents the sequence information, and [SEP] represents the separator. T_i is the i-th token in the query. L is the sequence length of the query. n is the number of columns in the databases, and H_{ni} represents the i-th token of the n-th column.

There are three types about token embedded information, as shown in the encoder of Figure 5. Token embedding encodes word information. Type embedding encodes the type information of the input sequence. E_0^T stands for the queries and E_1^T stands for the columns. Position embedding encodes the position information of the input sequence. In addition, Initializing sequence information by pre-trained model will loss some information and fall into local minimum. So we replace [CLS] with [XLS] tag. The sequence information learning([XLS]) is handed over downstream tasks, and this processing can help model learn more diverse representation.

Both X-SQL and SQLova use BERT. Compared with SQLova, X-SQL's structure is simpler, but it can generate SQL better. We think the complex structure behind BERT would weaken the convergence of the entire model. Similar to X-SQL, after BERT encoder, we don't use complex structures. We think BERT is enough to learn multi-task representation.

D. COLUMN REPRESENTATION

The input sequence is encoded by BERT to obtain the semantic vector, labeled $[h_{[XLS]}, h_{q1}, h_{q2}, \ldots, h_{qn}, h_{[SEP]}, h_{c11}, h_{c12}, \ldots, h_{[SEP]}, \ldots, h_{cm1}, \ldots, h_{[SEP]}]$.The dimension of the semantic vector is d. The input sequence contains a query sequence and multiple column sequences. Each sequence is connected by [SEP] tag. $h_{[XLS]}$ is the representation of the special token [XLS]. h_{qi} is the representation of the i-th token in the query. h_{cmi} is the representation of the i-th token in the m-th column. n is the length of the query. m is the number of the columns in the databases. Similar to X-SQL, we use global information $h_{[XLS]}$ to enhance the semantic representation of each column by attention mechanism.

The attention weights about the global information $h_{[XLS]}$ and the t-th column are as follows:

$$s_{ti} = dot(Uh_{[XLS]}, Vh_{cti}) \tag{2}$$

$$a_{ti} = \frac{s_{ti}}{\sum_{j=1}^{n_t} s_{tj}} \tag{3}$$

Both $U, V \in R^{d\times d}$, dot is the dot product. n_t is the number of tokens in the t-th column. s_{ti} is the similarity between $h_{[XLS]}$ and the i-th token in the t-th column. a_{ti} is the attention weight of the i-th token in the t-th column.

The representation of the t-th column is:

$$\overline{r}_{ct} = \sum_{i=1}^{n_t} a_{ti} h_{cti} \tag{4}$$

$$r_{ct} = \overline{r}_{ct} + h_{[XLS]} \tag{5}$$

where n_t is the length of the t-th column. The final column representation r_{ct} is obtained by adding $\overline{r}_{ct}$ and $h_{[XLS]}$. X-SQL thinks that adding $h_{[XLS]}$ can provide better alignment with the particular neural language query being asked. However, we consider that the whole framework is a multi-task learning, and [XLS] is the input of the other two subtasks S-num, W-num-op. The main role about adding $h_{[XLS]}$ is not to directly improve the column representation ability, but to build a relationship between multiple subtasks and improve the multi-task learning ability.

E. SUB-TASK OUTPUT

Our proposed M-SQL uses BERT semantic representation and the column representation to predict the blue parts in Figure 4. Similar to the work of Xu *et al.* [2] and He *et al.* [11], we divide the Text2SQL task on TableQA dataset into several sub-tasks. Each subtask is responsible for predicting different parts of the SQL statements. The dependencies between subtasks are shown in Figure 5.

Subtask S-num predicts the number of the selected columns. The prediction set is [1, 2]. Subtask W-num-op predicts the relationship between the conditional columns and the number of the conditional columns. The prediction set is [''null-1'', ''OR-1'', ''AND-1'', ''OR-2'', ''AND-2'', ''OR-3'', ''AND-3'']. ''null'', ''OR'' and ''AND'' represent the relationship between the conditional columns. ''1'', ''2'' and ''3'' represent the number of the conditional columns. Two subtasks use the global information $h_{[XLS]}$ as the input. S-num is a two-class classification problem, and W-num-op is a seven-class classification problem. S-num and W-num-op are modelled as follows.

$$p_1 = sigmoid(W_1 h_{[XLS]}) \tag{6}$$

$$p_2 = softmax(W_2 h_{[XLS]}) \tag{7}$$

where p_1 and p_2 represent the output probabilities of S-num and W-num-op respectively. W_1 and W_2 are learnable parameters. $W_1 \in R^{1\times d}$, and $w_2 \in R^{7\times d}$. Note, S-num and W-num-op only depend on $h_{[XLS]}$.

Subtask S-col predicts the selected columns. Subtask W-col predicts the conditional columns in where clause. The two subtasks use the column representation r_c as the input. The probability that the i-th column belong to the target column is as follows.

$$p_3 = sigmoid(W_3 r_{ci}) \tag{8}$$

$$p_4 = sigmoid(W_4 r_{ci}) \tag{9}$$

where p_3 represent the probability that the i-th column is the selected column. p_4 represent the probability that the i-th column is the conditional column. W_3 and W_4 are learnable parameters. Both W_3 and $W_4 \in R^{1\times d}$. S-col and W-col only depend r_c.

Subtask S-col-agg predicts the operations associated with the selected columns. Subtask W-col-op predicts the operations associated with the conditional columns. Similar to S-col, The two subtasks use the column representation r_c as the input. The target set of S-col-agg is [''', ''AVG'', ''MAX'', ''MIN'', ''COUNT'', ''SUM''], and it is a six-class classification problem. The target set of W-col-op is [''>'', ''<'', ''=='', ''! ='']. and it is a four-class classification problem. The operation probabilities associated with the target columns are computed as follows.

$$p_5 = softmax(W_5 r_{ci}) \tag{10}$$

$$p_6 = softmax(W_6 r_{ci}) \tag{11}$$

where p_5 represents the probability of the operation associated with the i-th selected column. p_6 represents the probability of the operation associated with the i-th conditional column. W_5 and W_6 are learnable parameters. $W_5 \in R^{6\times d}$ and $W_6 \in R^{4\times d}$. Both only depend r_c.

Subtask W-val extracts all values from the query at once. The value is a sub-segment of the query. Unlike previous solutions, We don't take the idea of extracting values based on the column representation. Intuitively, it is a good idea to extract values based on the column representation. However,

this way does not effectively distinguish different columns. If there are multiple values in the query and these values belong to different columns, the previous methods based on the column representation cannot accurately extract values. Based on the above considerations, we don't consider the column information when extracting values. We use two sub-models to extract values associated with the conditional columns, W-col-val and W-val-match. W-col-val ignores conditional columns and extracts all values from the queries at once. We try three different ideas in value extraction. The first is the mainstream idea of information extraction, BERT (+BILSTM) + CRF. The second is similar to the idea of machine reading comprehension [23], using the pointer network to extract values. The third is our proposed 0-1 labeling. Each position of the query is labeled 0/1, where 0 indicates that it does not belong to the value and 1 indicates that it belongs to the value. Through experiments, we found that the third idea 0-1 labeling achieved the best performance. The probability that the i-th token in the query belongs to the value is computed as.

$$p_7 = sigmoid(W_7 h_{qi}) \tag{12}$$

where p_7 represents the probability that the i-th token in the query need be extracted. W_7 is learnable parameter. $W_7 \in R^{1\times d}$. h_{qi} is the BERT representation of the i-th token in the query.

Subtask W-val-match matches the extracted values and the conditional columns, then determines which conditional columns the extracted values belongs to. It is a matching problem. If the value is related to the conditional column, this pair ''value, column'' is marked as 1; otherwise, it is marked as 0. We use the mean of the value span representation as the value representation, and use column representation rc_i as the representation of the i-th conditional column. We assume that the start index and end index of the extracted value in the query are s and e. The match score about the extracted value and the i-th conditional column is as follows.

$$h_v = \frac{\sum_{i=s}^{e} h_{qi}}{l} \tag{13}$$

$$match_i = sigmoid(u \cdot tanh(W_8 h_v + W_9 rc_i)) \tag{14}$$

where h_v is the value representation. $match_i$ is the match score about the extracted value and the i-th conditional column. W_8, W_9 and u are learnable parameters. W_8 and $W_9 \in R^{d\times d}$. $u \in R^{1\times d}$. l represents the length of the extracted value span.

F. EXECUTION-GUIDED DECODING

We divide the Text2SQL task on TableQA into several subtasks. We predict the blue parts of the SQL statement template(Figure 4) by these subtasks. Taking subtask S-col-agg as an example, subtask S-col-agg predicts the operation($OP) associated with the selected column. The target set is [''', ''AVG'', ''MAX'', ''MIN'', ''COUNT'', ''SUM'']. It is a six-class classification task. The output probability of each class is calculated as Equation 10. We choose the class with the highest probability as the output. There are some restrictions on the construction of SQL statements, such as string-type column cannot have numeric operations(<, >). So we use the execution-guided decoding strategy [17] to remove unreasonable SQL statements from the candidate SQLs in the SQL generation stage. In **select** clause, we assume that when the selected column is string-type, the aggregation operator cannot be the numeric operator, such as SUM, MIN, MAX. Similarly, in **where** clause, we assume that, when the conditional column is string-type, the aggregation operator cannot be the numeric operator(>, <). Through data analysis, we find that the selected columns and conditional columns are not coincident. We view this discovery as a filtering rule. We filter the SQL candidates which do not meet the above rules, and select the SQL statement with the highest join probability as the final output.

IV. EXPERIMENT

Compared to the real single-table SQL generation scenario, the WikiSQL dataset has made a lot of simplifications. Hwang *et al.* [10] claims that their proposed SQLova model has surpassed the human level on WikiSQL. Compared with SQLova, He *et al.* [11] proposed X-SQL achieves better performance. We can think that the SQL generation task on WikiSQL is basically solved. So our experiments are no longer based on WikiSQL, but only for the more complicated TableQA.

TableQA dataset contains 45,918 ''query-SQL'' pairs. Compared to WikiSQL, the query form is more complicated, including all modes of ComplexSQL in Figure 2. The number of the selected columns can be more than one(situation 1). TableQA contains more multi-conditional samples(situation 2) and adds ''OR'' logic relationship(situation 3). The queries are diverse, and the contents of the databases may not appear in the corresponding queries(situation 4). TableQA is built by Zhuiyi Technology in an AI competition. Zhuiyi Technology promises to open all data, but as of submission, only train and evaluation data are available, and test data has not yet been open. We split the evaluation data into two parts: one for parameter tuning and the other for testing. The numbers of train data, evaluation data and test data are 41,522, 2,198 and 2,198 respectively.

We use Chinese BERT with Whole Word Masking [22] as our pre-trained weights. The maximum length which BERT can handle is 512. However, the maximum length of our training samples does not exceed 300, so we directly use the maximum length of training samples as the input length of BERT. Because we have only two hyper-parameters, batch size and learning rate, we neither use hyper-parameters optimization nor grid search. We try several classic values, batch size set [16, 32], and learning rate set [2e-5, 5e-5]. We find that when the batch size is 32 and the learning rate is 2e-5, the model can get the best performance. We use Pytorch [24] as our deep learning framework. The computing resource is a 2080ti GPU, and the training time of single epoch is about

TABLE 1. The performance of various models on TableQA.

Model	Dev LX(%)	Dev X(%)	Dev MX(%)	Test LX(%)	Test X(%)	Test MX(%)
SQLNet [6]	61.28	66.20	63.74	61.42	67.24	64.33
Coarse2Fine [8]	72.98	76.89	74.94	72.61	76.71	74.66
MQAN [9]	75.66	79.21	77.44	74.84	78.75	76.80
SQLova [10]	81.39	85.26	83.33	81.71	85.76	83.74
X-SQL [11]	82.85	86.99	84.92	83.30	87.58	85.44
M-SQL(ours)	**89.13**	**91.86**	**90.50**	**89.31**	**92.13**	**90.72**
M-SQL-Ens(ours)	**90.54**	**93.40**	**91.97**	**90.49**	**93.31**	**91.90**

TABLE 2. The performance of sub-tasks on TableQA test data.

	S-num(%)	S-col(%)	S-col-agg(%)	W-num-op(%)	W-col(%)	W-col-op(%)	W-col-value(%)	Test LX(%)
M-SQL(ours)	99.50	97.82	98.91	97.45	98.50	99.10	96.95	89.31
M-SQL-Ens(ours)	99.55	98.36	98.91	97.68	99.09	99.27	97.00	90.49

20 min. By early stopping strategy, the model needs 4 or 5 epochs to converge.

We use three metrics to evaluate SQL generation accuracy. **Logical-form accuracy(LX)**. We directly compare the generated SQL statement with the ground truth, and check whether they match each other. **Execution accuracy(X)**. We execute the generated SQL statement and the ground truth to get the SQL query results, and check whether their results match each other. **Mean accuracy(MX)**. Mean of Logical-form accuracy and Execution accuracy.

A. ACCURACY

TableQA is built by Zhuiyi Technology in an AI competition. We participate in this competition and finally win the first place. Competition results can be found here.[3] Because test data has not yet been open, we split the evaluation data into two parts, one as the evaluation data and the other as test data. The Logical-form accuracy(LX), Execution accuracy(X) and Mean accuracy(MX) of our proposed M-SQL and several models on new evaluation dataset and new test dataset are shown in Table 1.The order of conditions in where clause is not considered in calculating the logical-form accuracy. We use Ens to represent ensemble results.

Our proposed M-SQL outperforms other models by a large margin. Compared to X-SQL model, M-SQL shows +6.01% LX, +4.55% X and +5.28% MX on test data. Interestingly, we use the early topping strategy based on evaluation performance, but achieve better performance on test data. M-SQL achieves 90.72% MX on test data. We guess the reason why test data performs better than evaluation data is that test data has better data quality.

In order to better understand M-SQL in detail, the logical-form accuracy of sub-tasks is shown in Table 2. All sub-tasks show >= 96%. The S-num achieves the highest accuracy 99.50%, and W-col-value achieves the lowest accuracy 96.95%. After error analysis, we find that the errors samples are mainly concentrated on value synonyms, and string-based matching would lead to higher error rate. For example, the value extracted from the query is ''wind chimes'', and the value candidates is [''wind chimes'', ''Feingxing brand wind chimes'', ...]. String-based matching tends to select ''wind chimes'', but the ground truth is ''Fengxing brand wind chimes''.

[3]address:https://tianchi.aliyun.com/competition/entrance/231716/rankingList/1?lang=en-us

B. ABLATION STUDY

For the Text2SQL generation task, we try a lot of tricks. In order to understand the importance of each trick, we perform ablation study, and relevant results are shown in Table 3. We use Chinese BERT with Whole Word Masking [22] as our pre-trained weights. Compared with general BERT [12], Chinese BERT with Whole Word Masking show 0.54%+ MX on test data. The experimental results show that this pre-trained BERT can learn better Chinese semantic representation for Text2SQL. Similar to X-SQL, we try to replace the [CLS] with the [XLS] tag. On test data, we achieve 0.45%+ MX. We think that retraining sequence representation can get better performance. M-SQL uses sequence representation[XLS] to participate in multiple subtasks. Using pre-trained [CLS] will cause the sequence representation to fall into a local optimal point, while using [XLS] to relearn, the model can get better fitting performance.

The BERT input contains three types: word, position and type. Similar to machine reading comprehension, we use type information to distinguish the query and column in the input sequence. We mark this method as 2-type. Compared with no type information, 2-type can show 0.25%+ MX on test data. Following the work of X-SQL, we try to use 3-type to distinguish query, string-type column and real-type column in the input sequence. However, we find that M-SQL cannot converge by using 3-type. We think that pre-trained BERT

TABLE 3. The results of ablation study.

Model	Test LX(%)	Test X(%)	Test MX(%)
M-SQL	89.31	92.13	90.72
– BERT-wwm-ext + BERT-base	88.90	91.45	90.18
– [XLS] + [CLS]	88.90	91.63	90.27
– 2-type	89.13	91.81	90.47
– 2-type + 3-type	\	\	\
– enhance	88.81	91.63	90.22
– BERT-0/1 + BERT-CRF	87.85	90.63	89.24
– BERT-0/1 + BERT-BILSTM-CRF	87.99	90.63	89.31
– BERT-0/1 + BERT-pointer	88.90	91.81	90.36
– lr + rouge-L	85.90	88.49	87.20
– lr + svr	74.75	77.02	75.89
– lr + bayes	86.35	88.67	87.51

Chinese version

商户类型	地区	区域	商户名称	地址
百货	广西	防城港	防城港港口区家惠超市	兴港大道 95-1 号
百货	广西	南宁	青秀南城百货	民族大道 64 号
百货	广西	南宁	白沙南城百货公司	南宁市白沙大道 20 号
.....	.....	.....	.....	.....

QUERY：青秀南城百货有限公司在南宁的哪个位置？
SQL：SELECT 地址 WHERE 商户名称=青秀南城百货 AND 区域=Nanning
Answer：民族大道 64 号

English version

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
.....	.....	.....	.....	.....

QUERY：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 6. A typical data sample in TableQA.

have not trained 3-type information, and limited TableQA data cannot make M-SQL fully be trained.

We find that many databases have similar columns with high semantic similarity, such as ‘‘area’’ and ‘‘region’’. The model cannot effectively distinguish these similar columns by only using the column names. A typical data sample is shown in Figure 6. Both ‘‘area’’ and ‘‘region’’ columns contain location contents, and the model cannot make accurate judgement about them. We try to use the contents of databases to enhance the information representation of columns, so that the information information of columns has a higher difference. We mark this method as ‘‘enhance’’ in Table 3. Specifically, for each column of databases, we search the cell with the highest similarity to the query. If the similarity exceeds a certain threshold, we use ‘‘,’’ to concatenate column and cell content as a new representation. We use rouge-L as the similarity calculation method, and the threshold is set to 0.6. Through above processing, the ‘‘region’’ column can be enhanced to ‘‘region, Nanning’’. With the enhanced columns, M-SQL shows 0.50%+ MX on test data.

In the value extraction part, we try three different ideas. In the first idea, we consider the value extraction as a sequence labeling problem BIEO [25]. The model uses CRF [26] to decode the sequence. Pure CRF shows 89.24% MX on test data, and we mark this method as ''BERT-CRF'' in Table 3. Before CRF, the model which is smoothed by an additional BILSTM [27] shows 89.31% MX on test data, and this method is marked as ''BERT-BILSTM-CRF''. The first idea is not effective. M-SQL is a multi-task learning framework. The loss of each task should be as close as possible. If the loss of one task is too high, it will affect the convergence of other tasks. The CRF has higher loss and reduces the overall performance. In the second idea, similar to machine reading comprehension, we try to predict the start and end index of extracted value in the query, and this method shows 90.36% MX on test data. We mark the second idea as ''BERT-pointer'' in Table 3. We think, compared with the accuracy of value extraction, the convergence relationship between multiple subtasks is more important. The range of loss between multiple subtasks should be as close as possible, and the difference should not be too large. In the third idea, we use 0/1 labeling. Each position of the query is labeled 0/1, where 1 means the token belongs value, 0 means not. M-SQL achieves the best performance by this simple 0/1 labeling.

In TableQA, the queries have various forms, and the target values don't necessarily appear in the queries. After extracting values from the queries, for string-type columns, model needs to confirm final values from databases based on the queries and extracted values. We try two methods, rule matching and machine leaning matching. Rule matching refers to directly values from databases with the highest similarity to the extracted values. We use rouge-L as the similarity calculation method. Through rule matching, the model shows 87.20% MX on test data. Machine learning matching uses machine learning to select values from databases based on statistical features. We build five statistical features, rouge-L recall and precision between extracted values and contents of databases, rouge-L recall and precision between queries and contents of databases, and the number of character coincidence between extracted values and contents of databases. We try three statistical learning methods, logistic regression(lr), support vector(svr), and bayes(bayes). The method based on logistic regression has achieved best performance, and shows 90.72% MX on test data.

V. DISCUSSION

The main task of our work is single-table Text2SQL generation. Text2SQL is also known as NL2SQL. The mainstream method is to use multiple sub-models to predict the filled contents of SQL statement based on the SQL template. Our proposed M-SQL is a multi-task learning framework. In this section, we will discuss two topics, multi-task learning and database utilization.

A. MULTI-TASK LEARNING

Our proposed M-SQL is a multi-task learning framework which can jointly train multiple subtasks. We try to train partial sub-models in M-SQL and find some interesting phenomena. If S-num model is trained separately, it cannot converge. If S-num and W-num-op are jointly trained, both can converge, but the accuracy is lower. S-num shows 98.32% LX and W-num-op shows 96.22% LX on test data. This experimental results show that S-num and W-num-op are highly correlated and can help each other converge. We try to jointly train sub-models which are not related to the values(S-num, W-num-op, S-col, S-col-agg, W-col, W-col-op), and the joint accuracy is 90.67% LX on test data. For comparison, we train all models. The first 6 sub-models(S-num, W-num-op, S-col, S-col-agg, W-col, W-col-op) have a joint accuracy of 91.45% LX on test data. Experimental results show that value-related models can promote convergence of unrelated models. Finally, we can draw a conclusion that the multiple sub-tasks of our proposed M-SQL can promote each other and help other sub-models to converge.

In addition, we find that S-num, W-num and W-op all use [XLS] representation as the input. W-op is used to predict the conditional relationship in where clause and prediction set is [''', OR, AND]. We try to merge these three sub-models. When the three sub-models are independent, M-SQL shows 88.94% LX on test data. When merging W-num and W-op, M-SQL shows 89.31% LX on test data. When merging three sub-models, M-SQL shows 88.13% LX on test data. Experimental results show that W-num and W-op have higher correlation, and merging them can improve M-SQL's performance.

B. DATABASE UTILIZATION

We concatenate the columns with similar contents of the databases to get more differentiated column expressions. Through the new column expressions, M-SQL effectively distinguishes similar columns and obtains better SQL generation performance. Although database utilization method can improve the performance of M-SQL, the databases have not been fully utilized. We use rouge-L between the queries and contents of the databases as the similarity calculation method. This method belongs to the string-type matching. It does not involve the semantic matching and cannot match synonyms. In addition, this method can improve the difference between column expressions, but not every content which is concatenated is beneficial, and it may introduce noise information to the column expression. If the model uses gate mechanism to fuse the columns and contents of the databases, we think that M-SQL would get further improvement.

VI. CONCLUSION AND FUTURE WORK

In this work, we propose M-SQL for the more complicated single-table SQL generation task. We use TableQA as experimental data. Compared to the WikiSQL, TableQA is more complicated and more in line with the actual application. Our proposed M-SQL is a multi-task joint learning framework. The focus of model design is how to improve the accuracy of multiple sub-tasks by joint learning. We introduce each sub-model of M-SQL in Section III and briefly analyse related experiments in Section IV. TableQA data comes from an AI

competition, and we rank first in this competition. Unfortunately, the official test data is not yet public, so we split the evaluation dataset into two parts: one for evaluating and the other for testing. Both experimental results and competition ranking demonstrate the superiority of our proposed M-SQL.

In the future, we will continue our research from the following two aspects.

- In this work, we concatenate columns and contents of databases to improve the difference of column expressions. Although this method improves the performance of SQL generation, the use of databases is insufficient and inaccurate. In the future, we will try to use the fusion mechanism to fuse contents of databases into the column expressions.
- Our proposed M-SQL improves the performance of single-table SQL generation. The problem whether the query can be answered for the certain database has not been studied yet. It is an indispensable step in actual SQL generation. We will study this problem in the future.

ACKNOWLEDGMENT

The authors would like to thank Zhuiyi Technology for their TableQA dataset. They would also like to thank Cui *et al.* for sharing the Chinese pre-trained BERT.

REFERENCES

[1] B. Hagedorn, L. Stoltzfus, M. Steuwer, S. Gorlatch, and C. Dubach, "High performance stencil code generation with lift," in *Proc. Int. Symp. Code Gener. Optim. (CGO)*, 2018, pp. 100–112.
[2] X. Xu, C. Liu, and D. Song "SQLNet: Generating structured queries from natural language without reinforcement learning," 2017, *arXiv:1711.04436*. [Online]. Available: https://arxiv.org/abs/1711.04436
[3] P. Thakor and S. Sasi, "Ontology-based sentiment analysis process for social media content," *Procedia Comput. Sci.*, vol. 53, pp. 199–207, Jan. 2015.
[4] D. Guo, D. Tang, and N. Duan, "Dialog-to-action: Conversational question answering over a large-scale knowledge base," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2942–2951.
[5] S. Gupta, R. Shah, M. Mohit, A. Kumar, and M. Lewis, "Semantic parsing for task oriented dialog using hierarchical representations," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2787–2792.
[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] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "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.
[8] 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.
[9] 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
[10] 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
[11] 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
[12] J. Devlin, M. W. Chang, and K. Lee, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 4171–4186.
[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, 1982.
[14] I. Ritchie and P. Thanisch, "MASQUE/sql-an efficient and portable natural language Query interface for relational database," in *Proc. 6th Int. Conf. Ind. Eng. Appl. Artif. Intell. Expert Syst.*, 1993, p. 327.
[15] L. Dong and M. Lapata, "Language to logical form with neural attention," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 33–43.
[16] R. Jia and P. Liang, "Data recombination for neural semantic parsing," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 12–22.
[17] 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
[18] 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
[19] 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. Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 2227–2237.
[20] 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
[21] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," 2019, *arXiv:1901.11504*. [Online]. Available: http://arxiv.org/abs/1901.11504
[22] 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
[23] S. Wang and J. Jiang, "Machine comprehension using match-LSTM and answer pointer," 2016, *arXiv:1608.07905*. [Online]. Available: http://arxiv.org/abs/1608.07905
[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Proc. NIPS Workshop*, 2017.
[25] J. Yang, S. Liang, Y. Zhang, "Design challenges and misconceptions in neural sequence labeling," in *Proc. 27th Int. Conf. Comput. Linguistics*, 2018, pp. 3879–3889.
[26] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2001.
[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

XIAOYU ZHANG received the M.S. degree from the National University of Defense Technology, Changsha, 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, 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.