Multimedia Systems (2024) 30:16
https://doi.org/10.1007/s00530-023-01201-y

REGULAR PAPER

SV^2-SQL: a text-to-SQL transformation mechanism based on BERT models for slot filling, value extraction, and verification

Chih-Yung Chang[1] · Yuan-Lin Liang[1] · Shih-Jung Wu[1] · Diptendu Sinha Roy[2]

Received: 24 February 2023 / Accepted: 8 December 2023

Abstract
Information retrieval from databases is challenging for a non-SQL-domain expert. Some previous studies have provided solutions for translating the natural language to SQL instruction, aiming to access the information in the database directly. However, most solutions are in English Natural Language. In addition, the accuracies of the existing works still need to be improved. This work presents a mechanism called SV^2-SQL, based on the pre-trained BERT. The proposed SV^2-SQL mainly consists of multiple deep-learning models, including select-where slot filling model (SWSF-model), value extraction model (VE-model), and verification (V-model). The SWSF-model handles the classification tasks for those fields that appear in the "Select" and "Where" clauses, and the VE-model extracts the values for the "Where" clause from the input. The V-model sorts out the unwanted candidates from two previous models and leaves only the ones with the highest possibility. The proposed SV^2-SQL also includes an algorithm for the inference process and allows the three models to be cooperative. Experimental results show that the proposed SV^2-SQL outperforms the existing studies in terms of precision, accuracy, and recall.

Keywords NL2SQL · Slot filling · BERT · SV^2-SQL · Information retrieval · Semantic parsing

1 Introduction

Semantic parsing is as considered one of the essential research topics in natural language processing. Semantic parsing is a task that transforms natural languages into lines of codes or instructions that machines can execute. In the past, there were some significant works related to this topic. For example, Text2Code [1], Text2Sparql [2], and Text2SQL [3]. Text2SQL has supported non-domain workers or employees to gain access to the databases by transforming natural languages into Structured Query Language (SQL). Specifically, the SQL here refers to Data Manipulation Language (DML), a type of instruction that manipulates the data in a database. The DML instructions consist of SELECT, INSERT, UPDATE, DELETE, and many more. To summarize, the NL2SQL task referred to converting natural languages to SELECT instructions to look up the data in a database. This task could support many business and customer service applications [4, 5].

There were many different methods to investigate this topic. The first method was rule-based, the earliest method, and did not require any neural network. This method created rules to generate the SQL query based on sentence patterns. For example, if the sentence contains words like "How many," there is a big chance that the SQL instruction will include "COUNT". The second method was sequence-to-sequence (Seq2Seq), which used deep neural networks such as encoder–decoder. The encoder part will convert the natural languages into a vector to translate the natural languages into SQL instruction. Then, the decoder part will generate the SQL instruction based on the encoded vector. The Seq2seq marks the first attempt to use a vector to learn the semantic representation from natural languages. Finally, the third method was Sequence-to-set (Seq2set), which used the improved version of the encoder–decoder neural networks.

Communicated by B. Bao.

✉ Chih-Yung Chang
cychang@mail.tku.edu.tw

Yuan-Lin Liang
809416018@gms.tku.edu.tw

Shih-Jung Wu
wushihjung@mail.tku.edu.tw

Diptendu Sinha Roy
diptendu.sr@nitm.ac.in

[1] Tamkang University, New Taipei City, Taiwan

[2] National Institute of Technology, Shillong, India

Published online: 16 January 2024

Seq2set required a sketch of SQL instruction, highlighting the components used in the instruction. Then, the Seq2set would predict which component category is the most suitable and fill in the slot of the components.

All three methods have both advantages and disadvantages. The rule-based method offered high accuracy if the natural language sentence contained specific patterns that easily met the conditions. If it were not the case, the accuracy would drop heavily. The Seq2seq method offered a new path that used deep learning to cope with the translation problem. However, an early encoder–decoder neural network offered low performance and required additional modules to improve the performance. The Seq2set changed the nature of the encoder–decoder neural networks and evolved into a different path. The path changes from direct translation to classification to offer better performance. However, it has difficulty when handling more complex SQL instructions such as having extra components ("GROUP BY", "HAVING", "ORDER BY") or nested instructions. The current Seq2set performs best with a simple sketch.

Although the NL2SQL research has much potential, data annotation requires SQL-domain annotators, leading to a small number of existing NL2SQL public datasets. The WikiSQL [6] is a large-scale crowd-sourced NL2SQL dataset consisting of 80,654 pairs of natural language questions and SQL instructions. The WikiSQL dataset's presentation attracted the attention of researchers in the NLP fields and produced many essential works. In WikiSQL, the questions and SQL instructions are in English, and the SQL instructions do not have any complex components as shown in Fig. 1.

This paper proposed a new mechanism called SV^2-SQL. The main focus of the proposed SV^2-SQL is to translate a sentence in the format of Chinese natural language to a SQL instruction which can access the data from a database table. Most previous studies took an English sentence as input and translated it to a SQL, aiming to access the corresponding data from a database. Different from previous studies, this work takes a Chinese sentence as input. The proposed SV^2-SQL mainly consists of three models. Each model handles different components in the SQL sketch. After predicting the key components, each model will return a list of results for using other models. This work also adapts the pre-trained language model BERT intending to achieve decent results when handling Chinese natural language queries. The contributions of this work are itemized below.

(1) *Multi-task mechanism.* The proposed mechanism can handle multiple tasks simultaneously. The aim is to improve the model's overall performance by leveraging the learning information and features from the tasks and sharing them between tasks.
(2) *Schema information enhancement.* The proposed mechanism employs a special token to enhance the semantic representations further. This tactic also helps improve the accuracy of the model's prediction.
(3) *Proposed an inference algorithm that allows the models to cooperate.* The algorithm constructs the SQL instruction based on the best possible outputs *V-model* chooses from the *SWSF-model* and *VE-model*. The proposed algorithm integrates two models and finally generates the required SQL instruction.
(4) *Replace the execution-guided decoding (EG) with V-model to reduce memory space.* The **EG** used the beam-search method to exhaust the possible un-executable SQL instructions, which increased the memory cost. Unlike **EG**, the *V-model* computes the similarity score of the components with the input query, and the algorithm will choose the components based on the similarity scores that are higher than the threshold value. As a result, a smaller and more stable memory space is required.

This mechanism also ensures that language localization is possible and reduces monotonous tasks when applied to business applications.

This paper organized the remaining sections as follows. Section 2 reviews the related work and their methods to tackle the challenges. Section 3 presents the problem statements and assumptions. Section 4 presents the architecture of the mechanism, including the networks of *SWSF-model*, *VE-model*, and *V-model*, along with the algorithm. Section 5 presents the performance improvements of the proposed mechanism against the previous studies. Finally, Section 6 presents the conclusions and future work.

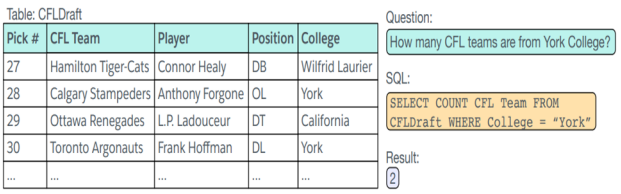
Table: CFLDraft

Pick #	CFL Team	Player	Position	College
27	Hamilton Tiger-Cats	Connor Healy	DB	Wilfrid Laurier
28	Calgary Stampeders	Anthony Forgone	OL	York
29	Ottawa Renegades	L.P. Ladouceur	DT	California
30	Toronto Argonauts	Frank Hoffman	DL	York
...	...	...	...	...

Question: How many CFL teams are from York College?

SQL: SELECT COUNT CFL Team FROM CFLDraft WHERE College = "York"

Result: 2

Fig. 1 Data sample in WikiSQL dataset [6]

2 Related work

Most early studies [13, 14] only handled the domain databases and relied on rules-based algorithms to perform the task. Later, the computer capabilities improved, and NL2SQL researchers began to adapt deep neural network technology and gained much attention from related communities.

The development of semantic representation learning techniques has recently reached a new height. The new state-of-art "Attention" technique [21] has semantic representation learning capabilities better than any previous technique and helps tremendously in this task. There were many pre-trained language models, such as BERT [12], ELMo [19], GPT [20], and more deep neural network models that used mentioned pre-trained language models and achieved better results. SQLova introduced the pre-trained model BERT [12], achieving significant results, more robust effects, and better performance. Some researchers used BERT as the Encoder to obtain the semantic representations of the queries [10]. In the RuleSQLova [27] study, the SQLova is improved further. Researchers used techniques such as knowledge distillation and fuzzy matching as rules to reduce the numeric column selection errors and "Where" clause values errors, respectively. Based on the semantic representations of the queries, three variant models, Shallow-Layer, Decoder-Layer, and NL2SQL-Layer, were proposed and achieved state-of-the-art results. The structure of NL2SQL-Layer was similar to SQLNet. The SQLova validated the effectiveness of dynamic representation learning techniques in the Text2SQL task. Researchers used BERT and proposed a simpler Text2SQL model. It introduced the [XLS] to replace the [CLS] tag and a new token [EMPTY] to represent the samples without the "Where" clause [11]. The study also included a ranking method for the column selection task using the Kullback–Leibler (KL) as an optimization function, improving the column selection accuracy.

The WikiSQL [6] dataset contained 80,654 samples on 24,241 tables from Wikipedia. Each sample contained an English natural language question and SQL instruction. The early studies [15, 16] were based on the neural semantic parsing approach, focusing on the direct translation (Seq2seq) approach to streamline the process. They divided the NL2SQL task into subtasks, which handle "select" clauses and "where" clauses separately. Each part is handled separately with different models.

In the Seq2SQL study [6], researchers considered the NL2SQL task as a translation task that converted the query into the SQL statement. The encoder–decoder framework learned how to map the query and the SQL statement. However, the Seq2SQL performance dropped when tackled with conditional values in the "where" clause. Xu et al. [2] proposed a sketch-based solution SQLNet, which transformed the NL2SQL task into six subtasks. These subtasks predicted and filled in the blue parts of the sketch. The SQLNet used the Seq2set approach to divide the NL2SQL task into six subtasks, each responsible for predicting different components of the SQL instruction. Among them, the conditional value prediction still used the seq2seq method, while others adopted the classification method.

Some researchers used the Seq2set method similar to the SQLNet study, but it employed the "type" information of each token in the query by a knowledge graph [7]. They also divided the NL2SQL task into numerous subtasks, which simplified the model encoders and reduced the original 12 encoders to 6 encoders, reducing the number of training parameters. In the Coarse2Fine study [8], the Text2SQL task is divided into two phases, first producing the intermediate presentation and then using the intermediate presentation to decode the "where" clause. In the MQAN study [9], researchers recommended a multi-task question and answer network structure that enabled jointly learning multi-task representation using various attention mechanisms. Wang et al. [17] proposed execution-guided decoding (EG), which executed all the generated SQL instructions using the beam-search function. The EG exhausted all the un-executable SQL instruction candidates and kept the one that was executable during the decoding phase. Other researchers proposed a sequence-to-action method that used predefined actions from a unique sketch to fill slot values [18]. Lately, a few studies have tried to tackle cross-domain challenges, including handling multiple tables. Some researchers proposed an encoder that consists of multiple modules. Each module learns different information representations such as utterance, schema discrepancy, and latent schema linking [24]. The study was evaluated on a new dataset called SparC [25] that has introduced a cross-domain semantic parsing challenge in which input queries require a long sequence of questions similar to conversations. However, this study only focuses on a single-domain database and does not handle cross-domain challenges. Template infilling was a valuable and pioneer approach in single-domain database. However, it has limitations in handling complex SQL queries. There has been a shift toward generation methods with the help of larger pre-trained language models. In the SeaD [26] study, researchers adopt a transformer-based seq2seq model with a large-size pre-trained language model combined with denoising and schema linking techniques to improve the SQL translation.

Some of the studies mentioned above combined with the EG algorithm could boost the performance of their models. The boost varied depending on which approach the studies applied, and the bigger the beam size the EG algorithm used, the more memory it cost. When handling the column selection and its related function, previous studies considered each classification task an individual task, resulting in the inability to find the relation between the column and the function. The proposed mechanism SV^2-SQL combines two tasks into one. As a result, it can simply find the relation between the column and the function. This work also proposes an algorithm that applies the deep-learning model to filter out the components with low possibilities before the execution of SQL instructions. The proposed algorithm

replaces the EG algorithm and requires a stable memory cost, as compared with EG algorithm (Table 1).

3 Assumption and problem statement

Assumes that there exists a database table $T_{\alpha\times\beta} = (F, V)$ consisting of column set F and value set V, where F consists of β columns and V consists of α rows of values. Let $F = \{F_j | 1 \le j \le \beta\}$ denote the set of β columns, where F_j denotes the j-th column name in F. Let $V = \{v_{i,j} | v_{i,j}, 1 \le i \le \alpha, 1 \le j \le \beta\}$ denote the two-dimensional set of values, where $v_{i,j}$ denote the value located in the i-th row of the j-th column. Herein, it is noticed that the values in the j-th column are said to be instances of column f_j. It is assumed that the table contents would not be changed quickly.

This paper aims to develop a text-to-SQL mechanism SV^2-SQL : $Q \to S$, where Q is a query in Chinese Nature Language format and S is a SQL instruction. Mechanism SV^2-SQL is a framework consisting of many deep-learning models which are trained based on the given query Q and T. Each model performs different tasks. Consider a set of x queries $Q = \{Q_1, \dots Q_x | 1 \le Q_i \le x\}$. For each query Q_i, there exists a correct answer which is the results obtained by putting SQL instruction S_i to table $T_{\alpha\times\beta}$. Let $Y = \{Y_1, \dots Y_x\}$ denote the x correct answers what user expects to obtain by the x correct SQL instruction $S = \{S_1, \dots S_x\}$. Let $\widehat{S} = \left\{\widehat{S}_1, \dots \widehat{S}_x\right\}$ be the x SQL instructions generated by mechanism SV^2-SQL. Let $\widehat{Y} = \widehat{S}(T_{\alpha\times\beta}) = \{\widehat{Y}_1, \dots \widehat{Y}_x\}$ denote the results obtained by applying SQL instruction $\widehat{S}$ to database table $T_{\alpha\times\beta}$.

Table 1 Comparison of the related works that focus on single-domain challenge

Related works	Pretrained model usage (BERT)	Extra techniques	EG algorithm
[6]	X	X	X
[7]	X	O	X
[8]	X	O	O
[9]	X	O	X
[10]	O	O	O
[11]	O	O	O
[18]	X	O	X
[27]	O	O	O
Our	O	O	X

The pre-trained model usage refers to the pre-trained language model for the encoder. The extra techniques column refers to the information enhancement of the input, and the EG algorithm refers to the usage of the EG algorithm

Consider the query Q_i. Let $Y_i = \{y_{i,1}, \dots y_{i,n}\}$ denote the labels of Q_i. Let $\widehat{Y}_i = \{\widehat{y}_{i,1}, \dots, \widehat{y}_{i,n}\}$ be the results obtained from the outputs of mechanism SV^2-SQL. In the following, the confusion matrix will be used to measure the performance of the mechanism SV^2-SQL in terms of recall, precision and $F1$-score. Let $\mathrm{TP}_{i,k}^{\mathcal{M}}$ be a Boolean valuable representing whether or not the predicted k-th element $\widehat{y}_{i,k}$ is in Y_i. That is,

$$\mathrm{TP}_{i,k}^{\mathcal{M}} = \begin{cases} 1 & \text{if } \widehat{y}_{i,k} \in Y_i \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Let $\mathrm{TP}_i^{\mathcal{M}}$ denote the true positive result of a prediction $\widehat{Y}_i$ and the label Y_i corresponding to the query Q_i. The value of $\mathrm{TP}_i^{\mathcal{M}}$ can be obtained according to Exp. (2):

$$\mathrm{TP}_i^{\mathcal{M}} = \sum_{k=1}^{n} \mathrm{TP}_{i,k}^{\mathcal{M}}. \tag{2}$$

Similarly, let $\mathrm{FP}_{i,k}^{\mathcal{M}}$ be a Boolean valuable representing whether or not the k-th predicted element $\widehat{y}_{i,k}$ is not in Y_i. That is,

$$\mathrm{FP}_{i,k}^{\mathcal{M}} = \begin{cases} 1 & \text{if } \widehat{y}_{i,k} \notin Y \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Let $\mathrm{FP}_i^{\mathcal{M}}$ denote the false-negative result of a prediction $\widehat{Y}_i$ and the label Y_i corresponding to the query Q_i. The value of $\mathrm{FP}_i^{\mathcal{M}}$ can be obtained according to Exp. (4):

$$\mathrm{FP}_i^{\mathcal{M}} = \sum_{k=1}^{n} \mathrm{FP}_{i,k}^{\mathcal{M}}. \tag{4}$$

Let $\mathrm{FN}_{i,k}^{\mathcal{M}}$ be a Boolean valuable representing whether or not the k-th label element $y_{i,k}$ is not in $\widehat{Y}$. That is,

$$\mathrm{FN}_{i,k}^{\mathcal{M}} = \begin{cases} 1 & \text{if } y_{i,k} \notin \widehat{Y} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Let $\mathrm{FN}_i^{\mathcal{M}}$ denote the false-negative result of a prediction $\widehat{Y}_i$ and the label Y_i corresponding to the query Q_i. The value of $\mathrm{FN}_i^{\mathcal{M}}$ can be obtained according to Exp. (6):

$$\mathrm{FN}_i^{\mathcal{M}} = \sum_{k=1}^{n} \mathrm{FN}_{i,k}^{\mathcal{M}}. \tag{6}$$

Let $\mathrm{TP}^{\mathcal{M}}$, $\mathrm{FP}^{\mathcal{M}}$, and $\mathrm{FN}^{\mathcal{M}}$ denote the true positive, false positive, and false negative of the label Y_i and prediction $\widehat{Y}_i$ corresponding to the query Q_i. The values of $\mathrm{TP}^{\mathcal{M}}$, $\mathrm{FP}^{\mathcal{M}}$ and $\mathrm{FN}^{\mathcal{M}}$ can be derived by Exp. (7), (8), and (9):

$$\mathrm{TP}^{\mathcal{M}} = \sum_{i=1}^{x} \mathrm{TP}_i^{\mathcal{M}}, \tag{7}$$

$$\mathrm{FP}^{\mathcal{M}} = \sum_{i=1}^{x} \mathrm{FP}_i^{\mathcal{M}}, \tag{8}$$

$$\text{and } \mathrm{FN}^{\mathcal{M}} = \sum_{i=1}^{x} \mathrm{FN}_i^{\mathcal{M}}. \tag{9}$$

Given a set of user queries x queries $Q = \{Q_1, \ldots Q_x | 1 \le Q_i \le x\}$, this paper aims to achieve high-accuracy results from the generated x SQL instructions $\widehat{S} = \left\{\widehat{S}_1, \ldots \widehat{S}_x\right\}$. As mentioned before, $\widehat{Y} = \{\widehat{Y}_1, \ldots \widehat{Y}_x\}$ and $Y = \{Y_1, \ldots Y_x\}$ denote the x query results and the x labels, respectively. Let $\mathrm{Pre}^{\mathcal{M}}$, $\mathrm{Rec}^{\mathcal{M}}$ and $F1^{\mathcal{M}}$ denote the total precisions, recalls and $F1$-measure of mechanism SV^2-SQL, we have,

$$\mathrm{Pre}^{\mathcal{M}} = \frac{\mathrm{TP}^{\mathcal{M}}}{\mathrm{TP}^{\mathcal{M}} +, \mathrm{FP}^{\mathcal{M}}}, \tag{10}$$

$$\mathrm{Rec}^{\mathcal{M}} = \frac{\mathrm{TP}^{\mathcal{M}}}{\mathrm{TP}^{\mathcal{M}} +, \mathrm{FN}^{\mathcal{M}}}, \tag{11}$$

$$F1^{\mathcal{M}} = \frac{2 * (\mathrm{Pre}^{\mathcal{M}} * \mathrm{Rec}^{\mathcal{M}})}{\mathrm{Pre}^{\mathcal{M}} + \mathrm{Rec}^{\mathcal{M}}}. \tag{12}$$

This paper aims to generate x SQL instructions from x input queries to achieve the maximal value of the $F1$-score. That is,

Objective:

$$\max_{\mathcal{M} \in M} F1^{\mathcal{M}}, \tag{13}$$

where M denotes the set of all possible mechanisms.

4 Proposed mechanism

It is well known that a standard sketch for *Select* SQL instruction has the following format:

The tokens "**Select**" and "**Where**" represent the starting clauses of a SQL instruction. The tokens marked in blue color in Fig. 2 indicate the slots that will be filled through proposed SV^2-SQL mechanism. The token *@Rel* represents the relation function between conditions in the "Where" clause, and the relation function *@Rel* consists of ["", "And", "Or"], where the "" indicates that there is only one condition and does not require any connection function. The *@S_Col* and *@W_Col* represent the columns in the "Select" and "Where" clauses, respectively. The token *@Agg* represents the aggregate function of selected columns. The aggregation functions generally used in SQL instruction are "Count", "Sum", "Average", "Min", and so on. The token *@op* represents the operator in a condition of the "Where" clause. The possible values of token *@op* are the elements including "=", "<", ">", "!=", and so on. The token *@Value* represents the value in the "Where" clause. The ground-true values of token *@Value* are the contents in table T. The notation $(x)^*$ allows multiple occurrences of x, where x could be the (*@Agg@S_Col*) in "Select" clause or the (*@W_Col@op@Value*) in "Where" clause.

Fig. 2 SQL instruction sketch

This section presents the proposed mechanism. The proposed mechanism called SV^2-SQL, consists of three AI models: The *Select-Where Slot filling* model (SWSF-model), *Value Extraction model* (VE-model), and *Verification model* (*V*-model). Figure 3 gives the network architecture of the designed SV^2-SQL. The goal of the proposed SV^2-SQL is to translate the user query into a semantically equivalent SQL instruction such that the query results of the SQL instruction can be the answer to the user query. To achieve this, the first model plays the role of slot filling, aiming to fill the column *@S_Col* and aggregate function *@Agg* slots in the "Select" clause. As shown in Fig. 3, the first model also fills the column *@W_Col* and operator *@op* slots in the "Where" clause. The second model aims to extract the mentioned value *@Value* from the user query. The third model plays a role of evaluation, aiming to verify the correctness of the extracted tokens from the first and the second models. The following figure presents the details of each model.

BERT is a fine-tuning based pre-trained bidirectional transformers model that uses a multi-head attention mechanism. The pre-trained BERT is trained using two unsupervised tasks: *Masked LM* (MLM) and *Next Sentence Prediction* (NSP). The goal of the Masked LM task is to capture a deep bidirectional representation. As for Next Sentence Prediction task, the goal is to capture the relationship between two sentences.

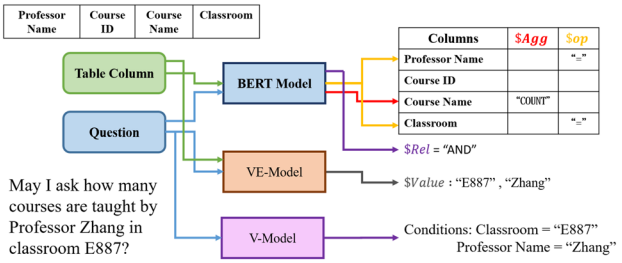

Fig. 3 The framework in details

4.1 The select-where slot filling model (SWSF-model)

The goal of the proposed mechanism is to translate it to SQL instruction. The structure of SQL itself was well known and pre-determined. This work assumes there is only one table in the database. Therefore, the first model only needs to capture the token's semantic representations and fill in the slots of the SQL sketch, including @*Agg*, @*S_Col*, @*Rel*, @*W_Col*, as well as @*Op* from the user query. Figure 4 presents the design of the SWSF-model, inputs, and outputs.

4.1.1 Input preprocessing

A SQL instruction is designed as the format of "select … where …". The SWSF-model aims to help understand more semantics of the query instruction. First, whether or not the selected column name in the query has the aggregation function. Second, what are the operations @*op* that should be translated and put in the "Where" clause. To achieve this, the formats of input and output are presented below. The input sequence consists of a *user query* and the *column names* in the table. In addition, the *column names* part also includes a unique tag to represent the aggregate pieces of information from the *column names*. That is, the format of the input:

"[CLS] Query [SEP] [COL] Column Names [SEP]".

The query consists of n-th Chinese characters $Q_i = \{q_1, \ldots q_n\}$ while the column names include a set of column names F_i in the tables after tokenized by *Wordpiece Tokenizer*. Each column name also consists of m-th Chinese characters $c_i = \{c_1, \ldots c_{|F_i|}\}$. The concatenation, starting with the [CLS], uses [SEP] token to separate the query and the column names. The Column Names might consist of more than one column name. The Column Names can be decomposed to distinguish them in the following format.

[CLS] $q_1, q_2, q_3, \ldots q_n$ [SEP] $f_{1,1}$, $f_{1,2}$, …, $f_{1,|F_1|}$ [SEP]
…
[SEP] $f_{\beta,1}$, $f_{\beta,2}$, …, $f_{\beta,|F_i|}$ [SEP].

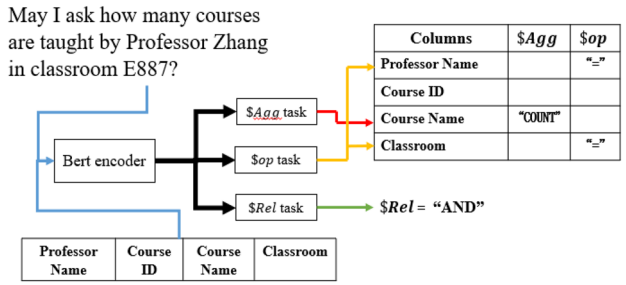

Fig. 4 The SWSF-model and subtasks

The BERT model applies the weights of Pretrained BERT *Whole Word Masking* model [23] as the initial weights of BERT.

4.1.2 The outputs of the SWSF-model tasks

As for the output, given a set of labels $L_i = \{l_1, \ldots l_\beta\}$, each column name f_β is mapped to label l_β correspondingly. The SWSF-model consists of three tasks: @*Agg* task, @*op* task, @*Rel* task. Assume aggregation @*Agg* has a number of classes including "None", "Min", "Max", "Avg", "Sum", and "Count". The BERT model embeds the input token, then creates the output, which is the hidden representation of the input tokens. The hidden representation is used as an input for the @*Agg* task. The @*Agg* task will verify if any class of aggregation should be applied to the column name f_i. For example, as shown in Fig. 4, the query of input is "*How many courses are taught by Professor Zhang in E787 classroom ?*" that is translated into Chinese language. Assume that there are a total of four column names. That is,

f_i = (f_1 = "Professor Name",

f_2 = "Course ID", f_3 = "Course Name",
f_4 = "Classroom").

In this case. The aggregation function "Count" should be applied to the third column f_3= "*Course Name*". This indicates that the corresponding values of labels $(l_1, \ldots, l_4)$ should be (0, 0, e("Count"), 0), where e() is an encoding function. Similar to @*Agg* task, assume the operations @*op* consists of the following classes: "=", "<", ">", "!=". The @*op* task will determine which class of operation @*op* should be labeled. Continue the example shown in Fig. 4. The output label of the @*op* task should be (l_1 = "=", $l_2 = 0, l_3 = 0, l_4$ = "="). The @*Rel* task will be a single-sentence classification.

4.1.3 BERT encoder and subtasks

The following represents a sequence of vectors, each of which is the output encoded by BERT.

$[H_{[CLS]}, H_{q_1}, \ldots, H_{q_n}, H_{[SEP]}, H_{f_{1,1}}, \ldots, H_{f_{1,|f_1|}}, H_{[SEP]},$
…
$H_{[SEP]}, H_{f_{\beta,1}}, \ldots, H_{f_{\beta,|f_i|}}, H_{[SEP]}]$.

The output sequence mainly consists of two crucial parts, including the query and headers of database tables. The user query has been transformed into an embedded vector format in the first part. That is, each H_{q_i} denote the d-dimensional semantic vector representing the embedded vector of each word appearing in the query. The column names are also

transformed into vector formats in the second part. That is, each $H_{f_{\beta,|f_i|}}$ denote the embedded vector of a word token, which is originally a token of the column name f_i.

Figure 5 represents the procedure of the SWSF-model. As shown in Fig. 5, the input of the SWSF-model contains the query and the columns in Chinese. Then the tokenizer automatically constructs three embedding layers, including Token embedding, Segment embedding as well as Mask embedding. The token embedding assigns a unique number for each input word. This number plays the role of ID to represent the original Chinese word as a unique number such that the word can be recognized and processed in advance. The segment embedding aims to assign each sentence with a unique notation to distinguish the two sentences. The Mask embedding aims to mark each input word with '$E_1^{\mathcal{M}}$' to represent the "actual word" or mark the word with '$E_0^{\mathcal{M}}$' to represent the "null" word. This implies that the later procedure can pass the word marked with '$E_0^{\mathcal{M}}$'.

According to study [21], each word token of the three embeddings, including token, segment and mask embeddings, will be sent as an input of the BERT encoder to be transferred to a word vector. Let H_{x_i} denote the vector of i-th word token x_i. The BERT encoder will perform the attention calculation to capture the semantics representation between all possible word token x_i in the input query and column names. To capture the semantics representation, the query q_i, key k_i, and value v_i will be generated for each word token x_i based on its vector H_{x_i}. Let weights w_q, w_k and w_v be three different weights of v_{x_i}. The query q_i, key k_i, and value v_i will be generated by

$$q_i = w_q v_{x_i}, k_i = w_k v_{x_i}, \text{and } v_i = w_v v_{x_i}. \tag{14}$$

These weights can be learned based on matrix multiplications. Let Q, K, V be the matrices with a d_k dimension which contains all q_i, k_i and v_i of word token x_i. Then the "Q, K, V" computations will firstly perform the single head attention operation to get the score $\hat{s}_i$ for each query q_i to each transposed key k_j. A simple attention operation is presented as the following equation:

$$\hat{s}_{i,j} = q_i k_j^{\mathrm{T}}, \tag{15}$$

where $\hat{s}_{ij}$ is the result of the single head attention of query q_i to each key k_j for all possible word token x_j in the inputs. After the multiplication, each score $\hat{s}_{ij}$ will perform the normalization operation by applying the softmax($\hat{s}_{ij}$) function to guarantee that each score ranges in [0, 1] and the sum of all scores equals one. The output scores s_{ij} are called attention

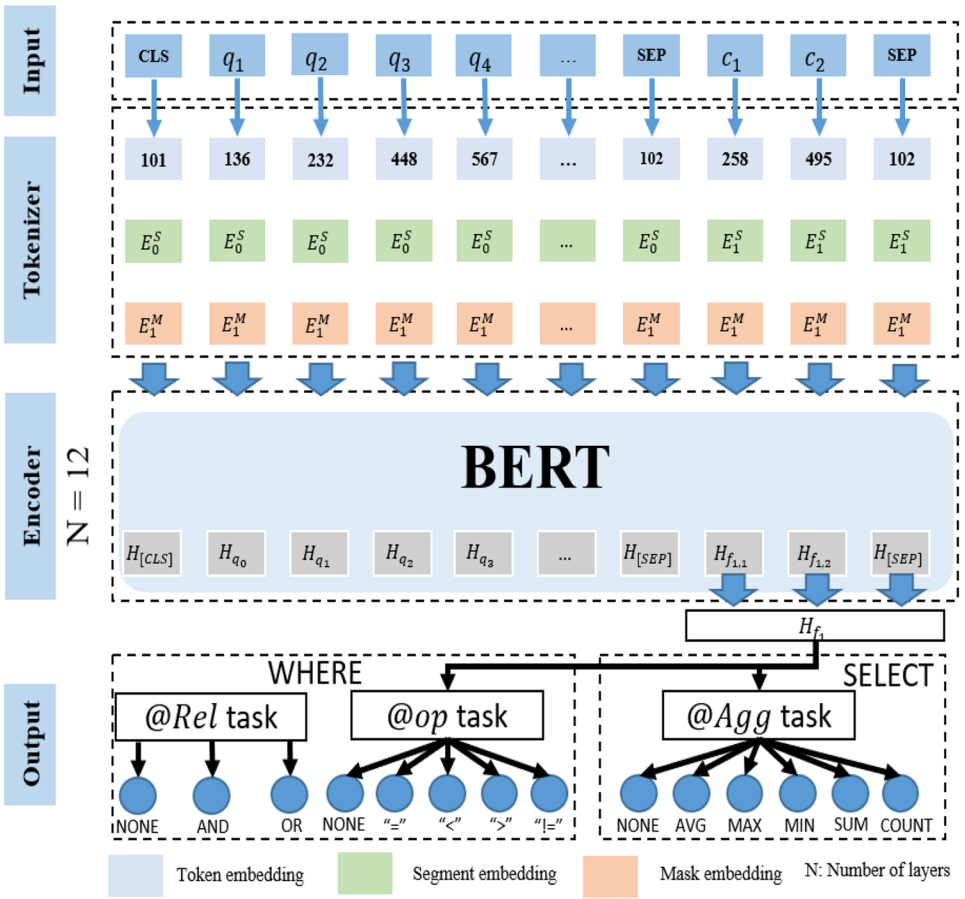

Fig. 5 The structure and the procedure of SWSF-model

scores which is the result multiplied by the normalized score and value v_j. Then the new word vector of the word token x_i will be generated by referring to the value of each word token using attention scores as their weights. Equation (15) reflects the operations for generating the attention score s_{ij}:

$$s_{i,j} = \text{softmax}\left(\frac{\hat{s}_{i,j}}{\sqrt{d_k}}\right). \tag{16}$$

The new vector $H_{x_i}^{\text{new}}$ of the word token x_i is, therefore, can be obtained by

$$H_{x_i}^{\text{new}} = [s_{i,1}v_1, \ldots, s_{i,j}v_j, \ldots]. \tag{17}$$

This result can be performed in parallel, which is called MultiHead(H_i) as depicted in follows.

$$\text{MultiHead}(H_i) = \text{Concat}(s_{i,1}v_1, s_{i,2}v_2, \ldots). \tag{18}$$

The result will be normalized to obtain the final vector representation.

Assume $H_{f_{\beta,F_i}}$ is the final vector representation of a token x_i from the column name part of the sequence. In order to train the output tasks, the representations $H_{f_{\beta,F_i}}$ will pass through a pooler layer to produce a single representation of the column name H_{f_β}. The representation H_{f_β} is embedded in the [COL] token. That is,

$$H_{f_\beta} = \text{Pooler}\left(H_{f_{\beta,1}}, H_{f_{\beta,2}}, \ldots, H_{f_{\beta,|f_i|}}\right). \tag{19}$$

The output tasks @*Agg* and @*op* are multi-label classifications based on the assumption that the table column names are known. Therefore, the output tasks will perform a combination of two subtasks: obtain the multi-label distribution of column name, then the category of @*Agg* or @*op* corresponding to that column name. Each output category corresponds to each column name of the data table. The @*Agg* corresponds to @S_Col one by one, and the category is fixed. Therefore, the set of labels is also the set of column names representation H_{f_β}. Each representation H_{f_β} represents a different classification task. The sigmoid function and the softmax function are finally applied to the last layer using a linear transformation to obtain the multi-label classes and the subset classes distribution over the intent representation H_{f_β}. Equation (18) represents the probability calculation of @S_Col and @*Agg* tasks:

$$P(\text{S\_Col}) = \text{Sigmoid}(W_{\text{S\_Col}}H_{f_\beta}), \tag{20}$$

$$P(Agg) = \text{Softmax}\left(W_{\text{agg}}H_{f_\beta}\right), \tag{21}$$

where both W_{agg} and $W_{\text{S\_Col}} \in R$ are the learning weight parameters for "Select" column and aggregation classification, respectively. The aggregation category includes an empty string indicating that the column name is selected, but not aggregated, along with other categories indicating that the column is selected and aggregated.

Similar to @*Agg* task, the @*op* task also has an output category corresponding @*op* to each "Where" column name @*W_Col*. The set of labels is the same set of column names representation H_{C_β}. Therefore, Eq. (19) represents the probability calculations of @*W_Col* and @*op* tasks:

$$P(W_{\text{Col}}) = \text{Sigmoid}\left(W_{W_{\text{Col}}}H_{C_\beta}\right), \tag{22}$$

$$P(\text{op}) = \text{Softmax}\left(W_{\text{op}}H_{C_\beta}\right), \tag{23}$$

where W_{op} and $W_{\text{W\_Col}} \in R$ are learning weight parameters for the "Where" column and operation classification, respectively. The operation category also includes an empty string indicating that the column is not selected. The remaining categories represent the selected "Where" column and corresponding operators.

The final @*Rel* task is a multi-class classification task. The task classifies the type of relation function, which appears in the "Where" clause when there are more than two "Where" columns are selected, or a single column is selected multiple times. Unlike @*Agg* and @*op* tasks mentioned above, the @*Rel* task requires the [CLS] token representation to be fed into the output layer for classification. The following equation represents the probability calculation of @*Rel* task,

$$P(Rel) = \text{Softmax}\left(W_{\text{rel}}H_{[\text{CLS}]}\right), \tag{24}$$

where W_{rel} is the learning weight parameter for relation classification.

The Cross-Entropy function was used to calculate the distance between the output probabilities of each task and ground-truth values. The loss function is the sum of cross-entropy loss for each task @*Agg*, @*op*, @*Rel*.

$$\text{Loss} = \text{Loss}_{\text{agg}} + \text{Loss}_{\text{op}} + \text{Loss}_{\text{rel}},$$

$$\text{Loss}_{\text{agg}} = -\frac{1}{N}\sum_{1}^{N} y_{\text{agg}}\log\left(\hat{y}_{\text{agg}}\right) + \left(1 - \hat{y}_{\text{agg}}\right)\log\left(1 - \hat{y}_{\text{agg}}\right),$$

$$\text{Loss}_{\text{op}} = -\frac{1}{N}\sum_{1}^{N} y_{\text{op}}\log\left(\hat{y}_{\text{op}}\right) + \left(1 - y_{\text{op}}\right)\log\left(1 - \hat{y}_{\text{op}}\right),$$

$$\text{Loss}_{\text{rel}} = -\frac{1}{N}\sum_{1}^{N} y_{\text{rel}}\log\left(\hat{y}_{\text{rel}}\right) + \left(1 - y_{\text{rel}}\right)\log\left(1 - \hat{y}_{\text{rel}}\right). \tag{25}$$

4.2 The value extraction model (VE-model)

The VE-model is shown in Fig. 6, which aims to extract the information and fill the @value from the query.

According to the study [22], the input sequence requires a pair of passage and a question. In this case, the column name and input question play the role of question and passage, respectively. This indicates that the answer (or label) should be found in the input question.

Initially, the input sequence will be tokenized and transformed into three embeddings: token embeddings, segment embeddings, and mask embeddings. After passing through 12 layers of the BERT encoder, the vector representation of the input sequence will be used to calculate the probabilities of the start and end positions of the answer extracted from the passage. The following represents the vector representation of the input sequence. That is,

$$H_{[\text{CLS}]}, H_{q_1}, \ldots, H_{q_n}, H_{[\text{SEP}]}, H_{f_{\hat{p},1}}, \ldots, H_{f_{\hat{p},|F_i|}}, H_{[\text{SEP}]}.$$

Let H_i denote the vector representation of the i-th token. The output layer of the model will determine the probability distributions of start and end positions for the answer span. Equation (21) calculates the distribution of the start position:

$$P(\text{Start}) = \text{Softmax}\left(W_{\text{start}} H_i\right). \tag{26}$$

The following equation calculates the distribution of the end position, that is:

$$P(\text{End}) = \text{Softmax}\left(W_{\text{end}} H_i\right), \tag{27}$$

where $W_{\text{start}}, W_{\text{end}} \in R$ both are learning weight parameters for start and end positions.

The loss function is the sum of cross-entropy losses for start and end positions. Assume L is the maximum length of the input sequence, which include both the passage and the question parts after tokenized if the start and end positions are $\boldsymbol{s}$ and $\boldsymbol{e}$, respectively, where values of s and e are in $[0, L]$. The loss function is calculated as follows:

$$\text{Loss} = \text{Loss}_{\text{start}} + \text{Loss}_{\text{end}},$$

$$\text{Loss}_{\text{start}} = -\frac{1}{N}\sum_{1}^{N} y_{\text{start}}\log\left(\hat{y}_{\text{start}}\right) + \left(1 - y_{\text{start}}\right)\log\left(1 - \hat{y}_{\text{start}}\right),$$

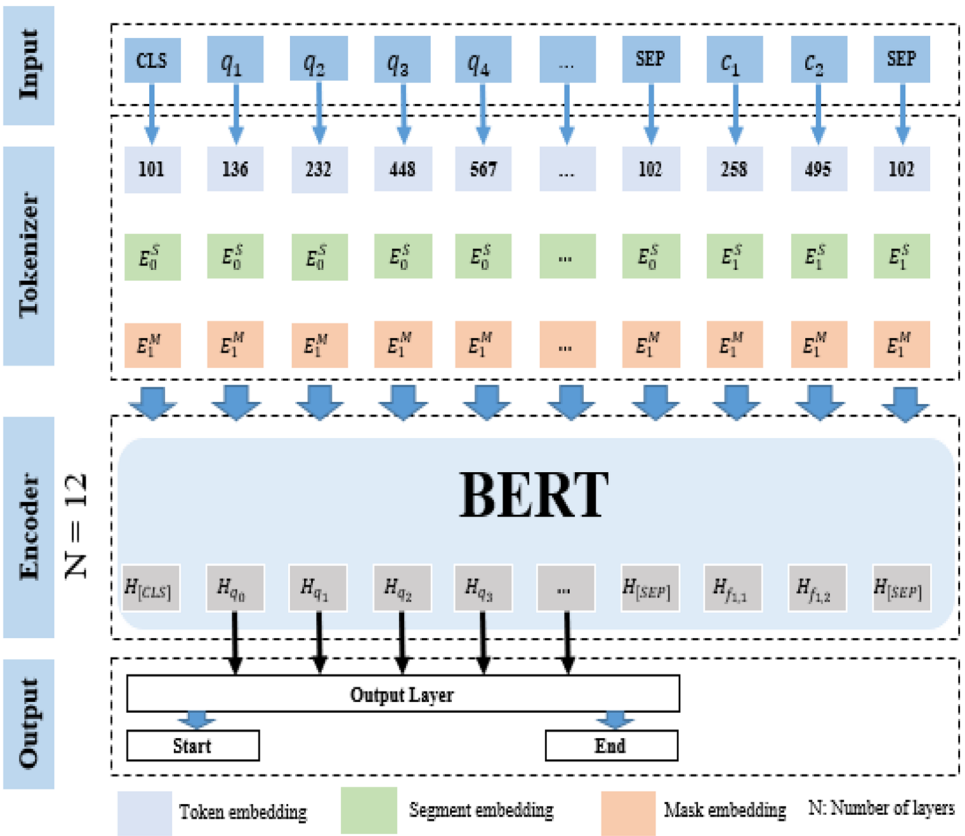

Fig. 6 The learning procedure of VE-model

$$\text{Loss}_{\text{end}} = -\frac{1}{N}\sum_{1}^{N} y_{\text{end}}\log\left(\hat{y}_{\text{end}}\right) + \left(1 - y_{\text{end}}\right)\log\left(1 - \hat{y}_{\text{end}}\right). \tag{28}$$

4.3 The verification model (V-model)

The final model aims to verify whether or not the combination of condition components in the "Where" condition clause is related to the user input query. This model exhausts the low possibilities of conditions and only keeps the conditions with the highest possibilities. The input sequence contains a pair of user input queries and a single condition clause. That is, the format of the input sequence is:

"[CLS] Query [SEP] Condition [SEP]"

Let $C = \{C_1, C_2, C_3, \ldots, C_n\}$ denote the condition clause that consists of a column name, operation function, and value. The format of the condition clause has a similar structure to SQL instruction. Figure 7 shows the learning procedure of the final model.

The input sequence will also be tokenized and transformed into three embeddings, similar to the first and second models. The vector representation which was generated after the input sequence passed through many layers of the BERT encoder as follows:

$$H_{[\text{CLS}]}, H_{q_1}, H_{q_2}, \ldots, H_{q_n}, H_{[\text{SEP}]}, H_{c_1}, H_{c_2}, \ldots, H_{c_n}, H_{[\text{SEP}]}.$$

The output layer of the model will calculate the semantic textual similarity between the pair of input sentences and the condition clause. Since this is a binary classification, the sigmoid function was applied in the output layer. Equation (23) represents the probability calculation of the output layer:

$$P(\text{Similarity}) = \text{Sigmoid}\left(W_{\text{sim}} H_i\right), \tag{29}$$

where $W_{\text{sim}} \in R$ is a learning weight parameter for the input query and condition clause.

The loss function is just a single cross-entropy loss as follows:

$$\text{Loss}_{\text{sim}} = -\frac{1}{N}\sum_{1}^{N} y_{\text{sim}}\log\left(\widehat{y_{\text{sim}}}\right) + \left(1 - y_{\text{sim}}\right)\log\left(1 - \widehat{y_{\text{sim}}}\right). \tag{30}$$

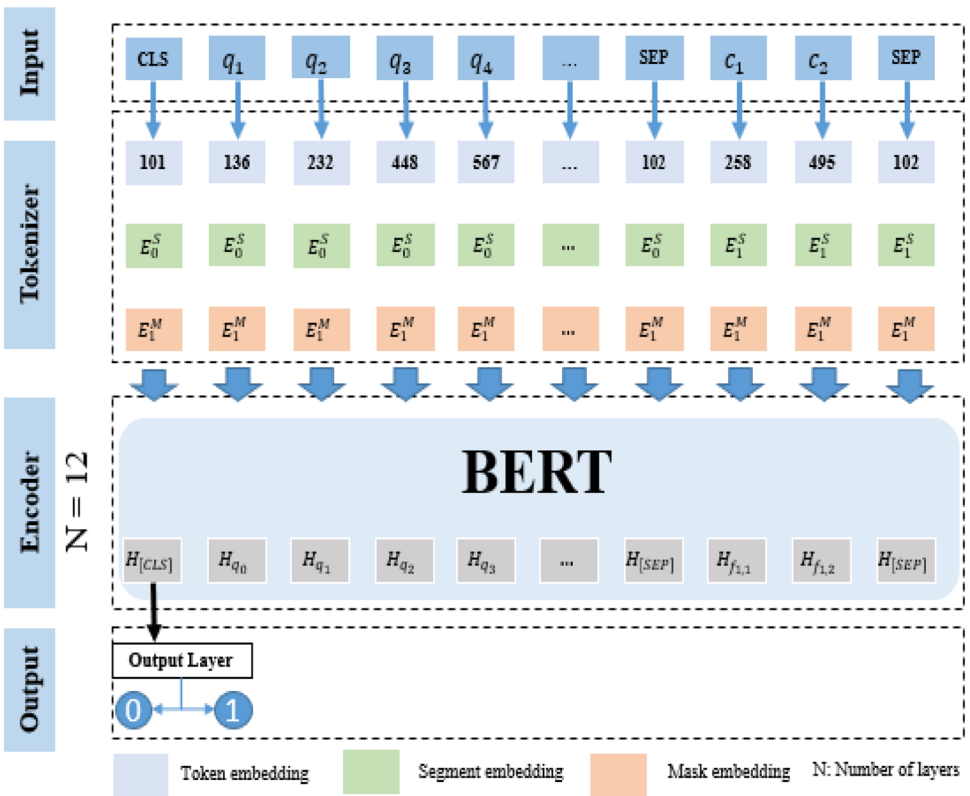

Fig. 7 The inputs, outputs, and learning procedure of V-model

4.4 The inference procedure of the mechanism

To understand the complete picture of how the mechanism works during the inference phase. Algorithm 1 shows the detailed procedure.

Algorithm 1: inference procedure

Input: the *i*-th input query $Q_i = \{q_1, q_2, q_3, \ldots q_n\}$; column names $F_i = \{f_1, \ldots f_{\lvert F_i \rvert}\}$	
Output: SQL syntax	
1:	Preprocess Q_i, F_i
2:	Feed Q_i and F_i into SWSF-model $@S\_Col$, $@W\_Col$, $@Agg$, $@op$, $@Rel$ =SWSF(Q_i, F_i) Return a list L consisting of the results: $@S\_Col$ $@W\_Col$ $@Agg$ $@op$ $@Rel$.
3:	For each $@W\_Col$ in the list of $@W\_Col$ do {
4:	Feed Q_i and $@W\_Col$ into VE-model Return a list V consisting $@Value$ For each $@Value$ in list V do { New $@Value$ = processing($@Value$) Add new $@Value$ to new list V Return post-processed list V

5:	Combine results $@W\_Col$, $@op$, $@Value$ into condition clause C_i
	For each $@W\_Col$ in list L do {
	For each $@op$ in list L do {
	For each $@Value$ in new list V do {
	C_i = $@W\_Col$ + $@op$ + $@Value$
	Add C_i to list C
	Return list C

6:	Feed Q_i and C_i into V-model compute similarity. # Results in range of Sim = [0,1] For each $C_i \in$ C do { sim_i = V-model(Q_i, C_i)
7:	# depend on the threshold value T For each $C_i \in$ C do { $C_i = \arg(\max_T sim_i)$
8:	*sql* = Combine $@S\_Col$, $@Agg$, $@Rel$, C_i into SQL instruction.
9:	*Results*=Execute(*sql*)
10:	Return *Results* and SQL instruction *sql*

5 Experiments and analysis

The proposed mechanism is compared in the experiments with the existing mechanisms SQLNet and SQLova. The following describes the dataset used in the experiments and the performance results.

5.1 Dataset and data augmentation

This research uses the TableQA [19] dataset, a Chinese NL2SQL dataset created by Zhuiyi Technology. The TableQA contains around 40,000 training samples, 5000 validation samples, and 10,000 test samples. Like WikiSQL, each sample includes a table, a pair of Chinese Natural Language question and the equivalent SQL query. This work also includes two different augmented datasets. The first augmented dataset contains around 90,000 samples, with 30,000 samples for each training, validation, and test dataset. The second augmented dataset consists of 90,000 samples.

5.2 Training and inference results analysis

During the training phase, each model in the mechanism SV^2-SQL is trained separately. The *SWSF-model* uses the original data from TableQA. The *VE-model* uses the first augmented data, each sample's format containing related column and target value span, consisting of start and end positions in the input query. The V-model also uses the second augmented data, and the sample is in the sentence pair format.

The *VE-model* generates a list containing many candidate values based on the query. Each candidate value after post-processing will be evaluated with the ground-truth label and kept only the one with the highest score. Then, the experiment calculates the average score for all the validation and test datasets samples.

The SQLNet and SQLova methods achieved high performance with the WikiSQL dataset consisting of only the English natural language as input query. In comparison, the SV2-SQL, SQLNet, and SQLova, are evaluated on the TableQA dataset to compare the performance based on four metrics including accuracy, precision, recall, and *F*-measure. The evaluation is separated into two parts; the first compares the performance of the methods on six tasks using the metrics, and the second compares the logical form and execution performance with the same metrics. Figure 8 represents the evaluation of the first part.

As shown in Fig. 8, the performance of the methods started to drop on the @Value task. The TableQA dataset has many challenges that may affect the overall performance of the models, mainly related to the condition clauses, such as condition columns or values. The performances of existing methods SQLNet, SQLova, SQLova-Rules on this dataset drop compared with the WikiSQL dataset. A few cases in the TableQA dataset prove to be a challenge. For example, samples containing a synonym confuse the mechanism into selecting a different column, and a wrong one will lead to selecting the wrong function to match that column. Other cases containing multiple keywords which are highly similar also result in the

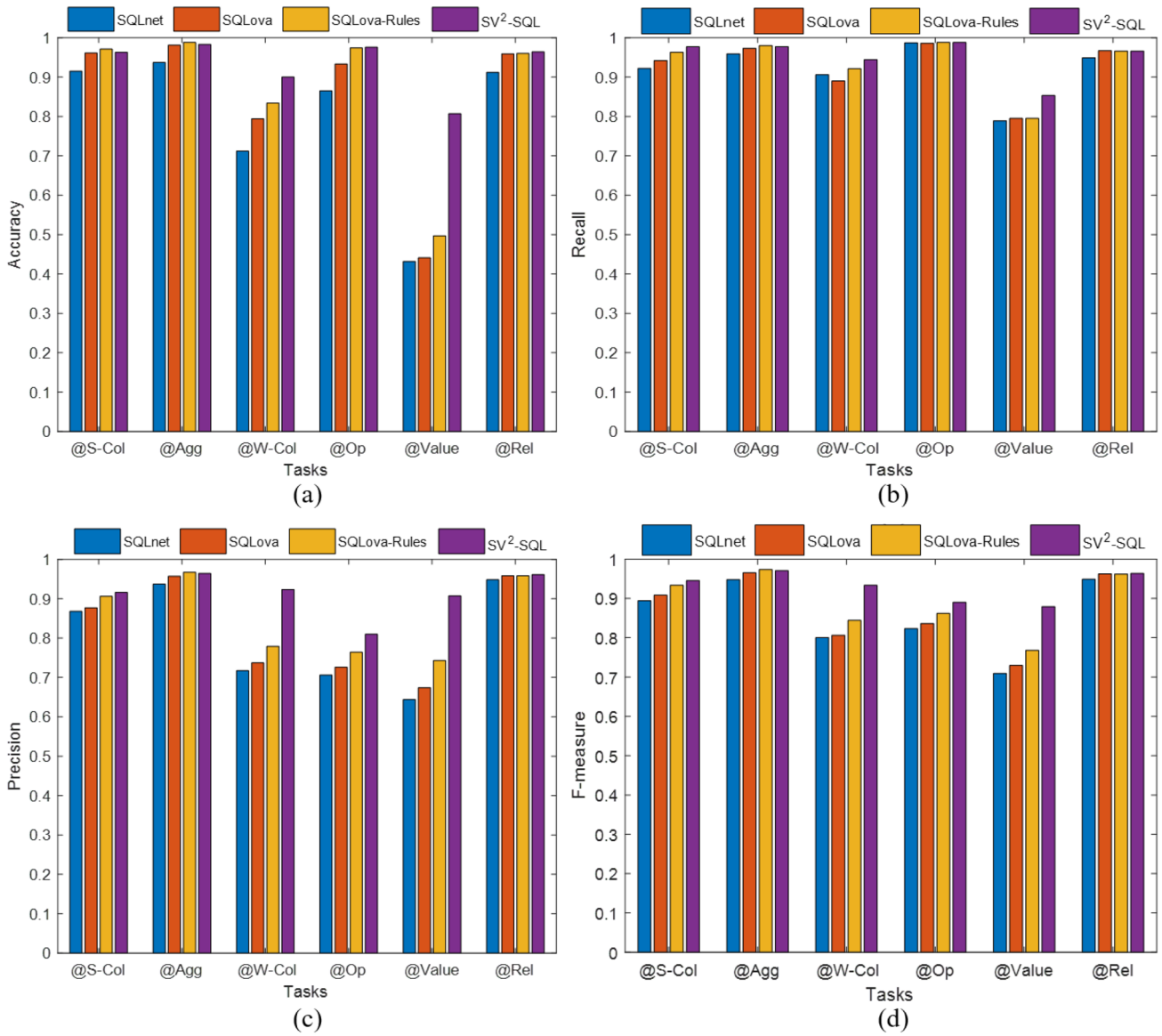

Fig. 8 Performance of four compared methods and **a** accuracy. **b** Precision. **c** Recall. **d** *F*-measure evaluations of the tasks. The notations @S_Col, @Agg, @W_Col, @Op, @Value, @Rel represent the 'Select' column, aggregation function, 'Where' column, Operation function, value, and Relation function, respectively

same mistakes. The input question contains no column-related references but only the target values and may or may not appear in many more complicated cases. The values, such as the numbers themselves, do not provide any meaning, making it hard for previous models to predict the relation between columns and values if they are not referred to in the query. Values with long-length characters, such as the name of companies or organizations, will also cause the models to be unable to predict the columns or the values. The proposed SV^2-SQL provides a way to tackle these challenges with the pre-trained BERT model, multi-tasking and column enhancement. BERT uses attention to focus on specific parts of the query, which helps tremendously but is not effective enough to tackle the synonym challenge. The column enhancement can tackle the challenge by using database content, which enhances the representation of information in columns, leading to an increased variation in column data. Finally, multi-tasking allows models to process and utilize information from different inputs to the best potential.

Figure 9 compares the performance of the value extraction task of the SV^2-SQL with other methods. The SV^2-SQL extracts the possible span of the target value. The target value might contain sub-values which will require Algorithm 1 to remove possible stopwords or punctuations and leave only the top values k. As shown in Fig. 9, when the

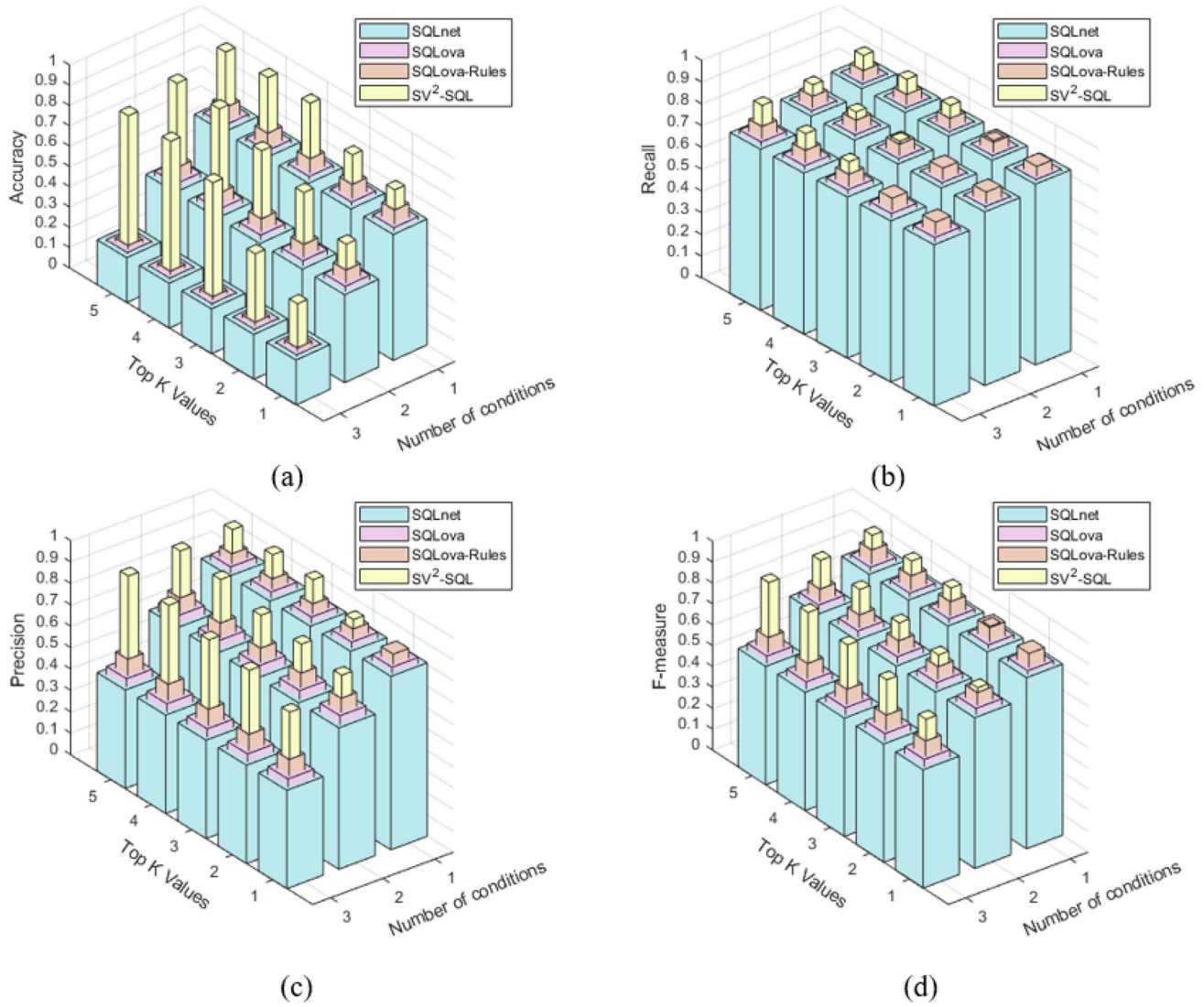

Fig. 9 Performance of four compared methods and **a** accuracy. **b** Precision. **c** Recall. **d** *F*-measure evaluations of the value extraction with top-*k*, where *k* is the number of the possible values extracted

number of conditions increases, it is more difficult for models to predict the target values as there are more possible values in the input query. Figure 9c compares the performance of methods in terms of value extraction rate. In a few cases, SV^2-SQL extracts the spans that may not contain a specific value leading to the target value not being extracted, resulting in lower recall scores than others. However, most cases that SV^2-SQL predicted are correct and previous methods have no technique to determine whether the extracted value relates to the specific column, which leads to lower precision, as shown in Fig. 9b.

Figure 10 compares the SQL query's logic form and execution results. The performance of the logical form is measured by the percentage of instances in which every element of the predicted SQL is precisely accurate. In contrast, the execution performance is based on the percentage of cases that produce the same result as the ground-truth SQL after execution. The threshold values were set for the *V-model* to filter out the non-executable SQL instruction candidates. The V-model will compute the similarity score between the input query and the 'Where' conditions combined from the first two models. The combinations with scores higher than the threshold value will be used to construct the SQL instruction candidates. Then, the SQL instruction candidates executed results will be evaluated along with the ground-truth results. At a threshold value of 0.5, the V-model seemed to include more possible combinations since some actual correct conditions have a lower score than incorrect ones. However, V-model will include unnecessary combinations when the threshold

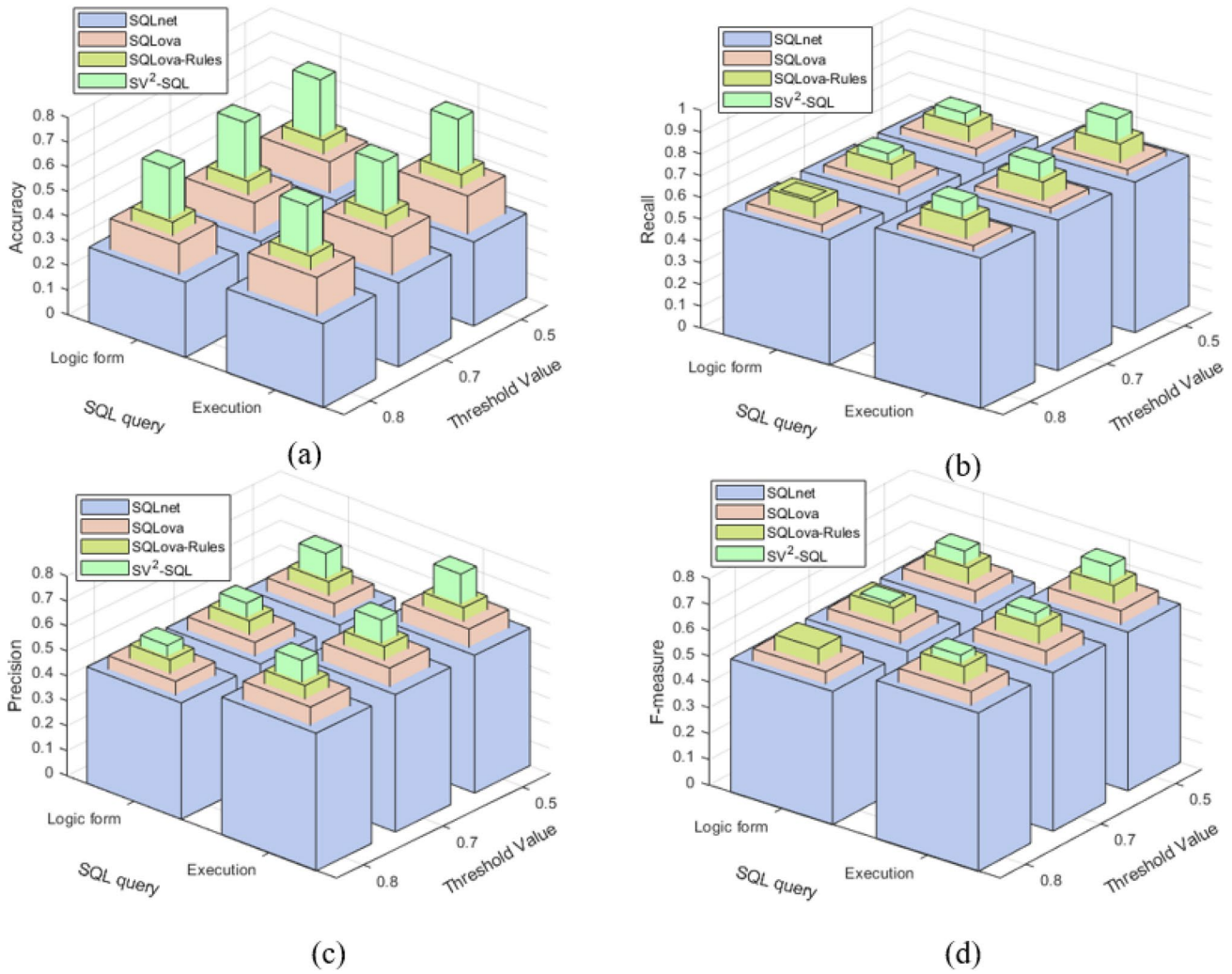

Fig. 10 Performance of four compared methods and **a** accuracy. **b** Recall. **c** Precision. **d** *F*-measure evaluations of the SQL query's logic form and execution results

Table 2 Experiment results of the methods SQLNet, SQLova, SQLova-Rules, and SV^2-SQL on TableQA dataset

Methods	@S-Col	@Agg	@W-Col	@Op	@Value	@Rel	Logic form	Execution	Mean
SQLnet	0.909	0.927	0.702	0.857	0.428	0.903	0.305	0.341	0.671
SQLova	0.951	0.971	0.784	0.921	0.434	0.949	0.425	0.491	0.740
SQLova-Rules	**0.971**	**0.988**	0.834	0.972	0.496	0.958	0.475	0.541	0.779
SV^2-SQL	0.963	0.983	**0.900**	**0.976**	**0.807**	**0.964**	**0.703**	**0.749**	**0.883**

value is lower than 0.5, leading to performance drops. The differences in results between 0.7 and 0.8 threshold values are minimal. Previous methods have no technique to determine whether the conditions are actually relevant.

Table 2 shows the overall accuracy of all tasks. The numbers in bold highlight the highest result compared to other results in each column. There are tasks that all methods can predict without any problems, especially the aggregation @*Agg*, operation @Op and relation @Rel functions, which can achieve around 90% accuracy and the same as the column selection @S_Col in the 'Select' clause. The proposed SV^2-SQL has slightly different accuracy, with around 0.1% up to 0.5% compared to SQLnet and SQLova, but slightly lower than SQLova-Rules. Besides the mentioned tasks,

Table 3 The results of the ablation study

SV^2-SQL			Test dataset	
SWSF-model	VE-model	V-model	Logic form (%)	Execution (%)
BERT-base	BERT-base	BERT-base	0.683	0.708
BERT-base	BERT-BiLSTM-CRF [28]	BERT-base	0.695	0.734
BERT-base	BERT-pointer [29]	BERT-base	0.727	0.752
BERT-wwm	BERT-wwm-	BERT-wwm	0.703	0.749
BERT-wwm	BERT-wwm BiLSTM-CRF	BERT-wwm	0.715	0.765
BERT-wwm	BERT-wwm-pointer	BERT-wwm	0.739	0.787

SQLnet, SQLova, and SQLova-Rules have performance drops to 42%, 43%, and 31% accuracies when predicting the columns and values in the 'Where' clause. The SV^2-SQL can achieve 90% and 80% accuracies in @W_Col and @Op tasks, respectively, surpassing SQLnet, SQLova and SQLova-Rules methods. The average performance of the proposed SV^2-SQL is 88.3%.

Table 3 represents the results of the ablation study. The ablation study is conducted to evaluate the performance of the proposed method. The evaluation includes changing the pre-trained language model BERT and BERT-wwm, which was trained with different tokenization methods. The VE-model includes additional layers of neural networks such as Bidirectional Long Short-Term Memory (BiLSTM) and Conditional Random Fields (CRFs), which increase the overall accuracy of logical form and execution. The accuracy is increased further when combined with the pointer network. This also increase overall logic form and execution accuracies.

Figure 11 represents a few highlighted cases the proposed SV^2-SQL failed to predict. For example, the columns mismatched errors often happen due to high similarity between column names in the database schema or due to ambiguities in language or differences in terminology between users and the database schema. Natural language queries sometimes involve long, complex named entities, such as product names, locations, or people's names. The basic VE-model may not accurately extract these entities, leading to failed or incomplete query translations. However, when combining the VE-model with additional layers or pointer networks, the errors are reduced. A typical case that can happen is the wrong operator selection error because users may have many ways to ask a similar question using synonyms or paraphrasing. In the English translation, the words are the same, but in the Chinese Language, the query includes two different ways to express the meaning of the "more than" word, such as "larger than". This causes the proposed SV^2-SQL to misinterpret operator intent in natural language queries.

6 Conclusion

This work presents the mechanism for a single-table Chinese Natural Language to SQL instruction tasks. With the combination of multi-class and multi-label tasks, the models and the implementation of a pre-trained language model such as BERT to help capture semantic information can improve the overall performance. This work includes a proposed inference algorithm that the models can work together. The proposed mechanism SV^2-SQL helps make an application for business class data management. The evaluations reveal that the proposed mechanism performs better than existing precision, accuracy, and recall methods.

Fig. 11 Examples of case analysis from TableQA dataset

Error type: Columns mismatched
Input: 麻烦帮我看看，11年本益比超过10，12年和13年的本益比预计也会超过10的有几家公司啊?
English translation: Could you please tell me how many companies have a P/E ratio of over 10 in 2011, and the P/E ratio in 2012 and 2013 is also expected to exceed 10?
Prediction: SELECT COUNT(公司) WHERE EPS2011 > '10' AND EPS2012E > '10' AND EPS2013E > '10'
Ground-truth: SELECT COUNT(公司) WHERE PE2011 > '10' AND PE2012E > '10' AND PE2013E > '10'

Error type: Failed to extract long name entity
Input: 郭文君的中国散裂中子源首次打靶成功获得了几等奖
English translation: Guo Wenjun's China Spallation Neutron Source successfully won several prizes for its first target-shooting
Prediction: SELECT 奖项 WHERE 作品 = '中国散裂中子源首' AND 作者 = '郭文君'
Ground-truth: SELECT 奖项 WHERE 作品 = '中国散裂中子源首次打靶成功' AND 作者 = '郭文君'

Error type: Wrong operator selection
Input: 小哥哥，你能告诉我那些均票价在三十五块以上的，或人均场次大于8的电影叫什么名吗
English translation: Brother, can you tell me the names of those movies whose average ticket price is more than 35 yuan, or whose average number of screenings per capita is more than 8?
Prediction: SELECT 影片名称 WHERE 平均票价 = '35' OR 人均场次 > '8'
Ground-truth: SELECT 影片名称 WHERE 平均票价 > '35' OR 人均场次 > '8'

Author contributions Conceptualization: C-YC, Y-LL; methodology: C-YC, Y-LL, S-JW; formal analysis: C-YC, Y-LL; writing—original draft preparation: C-YC, Y-LL; writing—review and editing: DSR.

Funding The authors did not receive support from any organization for the submitted work. No funding was received to assist with the preparation of this manuscript. No funding was received for conducting this study. No funds, grants, or other support was received.

Data availability The data that support the findings of this study are openly available in ZhuiyiTechnology/TableQA at https://github.com/ZhuiyiTechnology/TableQA, reference number [19].

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. The authors have no conflicts of interest to declare that are relevant to the content of this article. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

References

1. Hagedorn, B., Stoltzfus, L., Steuwer, M., Gorlatch, S., Dubach, C.: High performance stencil code generation with lift. In: Proceedings of the 2018 International Symposium on Code Generation and Optimization (CGO 2018). Association for Computing Machinery, New York, NY, USA, pp. 100–112. (2018). https://doi.org/10.1145/3168824
2. Xu, X., Liu, C., Song, D.: SQLNet: generating structured queries from natural language without reinforcement learning (2017). arXiv:1711.04436. [Online]
3. Thakor, P., Sasi, S.: Ontology-based sentiment analysis process for social media content. Procedia Comput. Sci. **53**, 199–207 (2015)
4. Guo, D., Tang, D., Duan, N.: Dialog-to-action: conversational question answering over a large-scale knowledge base. In: Proceedings of Advances in Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, pp. 2942–2951 (2018)
5. Gupta, S., Shah, R., Mohit, M., Kumar, A., Lewis, M.: Semantic parsing for task oriented dialog using hierarchical representations. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium, pp. 2787–2792 (2018)
6. Zhong, V., Xiong, C., Socher, R.: Seq2SQL: generating structured queries from natural language using reinforcement learning (2017). arXiv:1709.00103. [Online]
7. Yu, T., Li, Z., Zhang, Z., Zhang, R., Radev, D.: TypeSQL: knowledge-based type-aware neural Text-to-SQL generation. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies, New Orleans, Louisiana, USA, pp. 588–594 (2018)
8. Dong, L., Lapata, M.: Coarse-to-fine decoding for neural semantic parsing. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Melbourne, Australia, pp. 731–742 (2018)
9. McCann, B., Shirish Keskar, N., Xiong, C., Socher, R.: The natural language decathlon: multi-task learning as question answering (2018). arXiv:1806.08730. [Online]
10. Hwang, W., Yim, J., Park, S., Seo, M.: A comprehensive exploration on WikiSQL with table-aware word contextualization (2019). arXiv:1902.01069. [Online]
11. He, P., Mao, Y., Chakrabarti, K., Chen, W.: X-SQL: representation with context (2019). arXiv:1908.08113. [Online]
12. Devlin, J., Chang, M.W., Lee, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies, Minneapolis, Association for Computational Linguistics, Minnesota, USA, pp. 4171–4186 (2019)
13. Warren, D.H.D., Pereira, F.C.N.: An efficient easily adaptable system for interpreting natural language queries. Comput. Linguist. **8**(3–4), 110–122 (1982)
14. Ritchie, I., Thanisch, P.: MASQUE/sql-an efficient and portable natural language Query interface for relational database. In: Proceedings of 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, Scotland, p. 327 (1993)
15. Dong, L., Lapata, M.: Language to logical form with neural attention. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, pp. 33–43 (2016)
16. Jia, R., Liang, P.: Data recombination for neural semantic parsing. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, pp. 12–22 (2016)
17. Wang, C., Tatwawadi, K., Brockschmidt, M., Huang, P.-S., Mao, Y., Polozov, O., Singh, R.: Robust Text-to-SQL generation with execution-guided decoding (2018). arXiv:1807.03100. [Online]
18. Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., Chen, W.: IncSQL: training incremental Text-to-SQL parsers with non-deterministic oracles (2018). arXiv:1809.05054. [Online]
19. Sun, N., Yang, X., Liu, Y. (2020). TableQA: a Large-Scale Chinese Text-to-SQL Dataset for Table-Aware SQL Generation. arXiv:2006.06434. Accessed 20 Feb 2022. [Online]
20. Radford, A., Narasimhan, K., Salimans, T.: Improving language understanding by generative pre-training (2018). [Online]. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf. Accessed 29 July 2022
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), pp. 6000–6010. Curran Associates Inc., Red Hook (2017)
22. Hu, Z.: Question answering on SQuAD with BERT. Stanford University (2019). [Online]. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15792151.pdf. Accessed 15 Apr 2022
23. Cui, Y., Che, W., Liu, T., Qin, B., Yang, Z., Wang, S., Hu, G.: Pre-training with whole word masking for Chinese BERT. IEEE/ACM Trans. Audio Speech Lang. Process. **29**, 3504–3514 (2021). https://doi.org/10.1109/TASLP.2021.3124365
24. Wang, R.Z., Ling, Z.H., Zhou, J.B., Hu, Y.: A multiple-integration encoder for multi-turn text-to-SQL semantic parsing. IEEE/ACM Trans. Audio Speech Lang. Process. **29**, 1503–1513 (2021). https://doi.org/10.1109/TASLP.2021.3070726
25. Yu, T., Zhang, R., Yasunaga, M., Tan, Y.C., Lin, X.V., Li, S., Er, H., Li, I., Pang, B., Chen, T., Ji, E., Dixit, S., Proctor, D., Shim, S., Kraft, J., Zhang, V., Xiong, C., Socher, R., Radev, D.: (2019) SParC: cross-domain semantic parsing in context.

In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4511–4523. Association for Computational Linguistics, Florence
26. Xu, K., Wang, Y., Wang, Y., Wang, Z., Wen, Z., Dong, Y.: SeaD: end-to-end text-to-SQL generation with schema-aware denoising. In: Findings of the Association for Computational Linguistics: NAACL 2022, pp. 1845–1853. Association for Computational Linguistics, Seattle (2022)
27. Han, S., Gao, N., Guo, X., Shan, Y.: RuleSQLova: improving text-to-SQL with logic rules. In: 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, pp. 1–8 (2022). https://doi.org/10.1109/IJCNN55064.2022.9892938
28. Kuo, C.-Y., Lu, E.J.-L.: A BiLSTM-CRF entity type tagger for question answering system. In: 2021 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bandung, Indonesia, pp. 161–166 (2021). https://doi.org/10.1109/IAICT52856.2021.9532562
29. Mukherjee, R., Nayak, T., Butala, Y., Bhattacharya, S., Goyal, P.: PASTE: a tagging-free decoding framework using pointer networks for aspect sentiment triplet extraction. In: Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics. (2021) [Online]