Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **7**, Number **2**, May **2022**

Cross-Lingual Database Q&A through Lexicon Enhancement and Relationship Awareness

Rui-Heng Liu*

Department of Computer Science
Xi'an Research Institute of Hi-Tech
Xi'an, 710038, China
*Corresponding Author:liuruih@foxmail.com

Xia Ye

Department of Computer Science
Xi'an Research Institute of Hi-Tech
Xi'an, 710038, China
yex_qing@126.com

Zeng-Ying Yue

Department of Computer Science
Xi'an Research Institute of Hi-Tech
Xi'an, 710038, China
zengying2020@gmail.com

Jin-Jin Zhang

Department of Computer Science
Xi'an Research Institute of Hi-Tech
Xi'an, 710038, China
wangzhao@st.xatu.edu.cn

Chen-Hao Zhu

Department of Computer Science
Xi'an Research Institute of Hi-Tech
Xi'an, 710038, China
972778371@qq.com

Received October 2021; revised December 2021

Abstract. *Reflecting user query intent through Structured Query Language(SQL) to answer questions on a knowledge base has been the focus of research in natural language processing and human-computer interaction. For complex relational database Q&A tasks in cross-lingual scenarios, we propose a semantic parsing model DBSQL. First, the shared features of natural language queries and database schemas are extracted. Next, to solve the problem of imbalance of minimal semantic units in cross-lingual text sequences, lexicon features are fused into character-level semantic representations by alignment attention mechanism. Then, the relationship features in database schema are captured through a relationship-aware attention mechanism, thus improving the ability to model complex queries. Ultimately, a hybrid pointer network is used to guide the generation of more flexible SQL statements. Experimental results on a public dataset shows that the proposed method can effectively improve the exact match accuracy of generated SQL statements and better parse the query intent.*
Keywords: cross-lingual question and answering, structured query language, database schema, semantic parsing, attention mechanism

1. **Introduction.** With the deep integration of artificial neural networks and natural language processing tasks, how to establish a more natural human-machine interaction between humans and machines has gradually become an emerging research hotspot. Especially for user questions on structured knowledge such as tables and relational databases across domains, an effective solution is to transform natural human statements into generic Structured Query Language (SQL) that can be executed on databases through semantic parsing techniques, which can build an interactive bridge between non-technical people and structured knowledge. This method can be applied to a variety of practical scenarios such as cross-domain data retrieval [1], intelligent Q&A(Question and Answer) [2].

To achieve the conversion from natural human language to machine-understandable logical queries, most approaches use an encoder-decoder structure similar to the translation tasks. The encoder is used to understand the query intent in natural language and establishes a corresponding entity mapping with structured knowledge that is the object of the query. The decoder generates SQL statements based on encoder's understanding to obtain answers. SQLNet [3] can generate corresponding SQL statements for a single table query in English scenario. It takes into account the limitations imposed by the query scope of a single table on the structure of SQL statements, in other words, SQL statements do not involve complex syntactic operations such as multi-table joins. To this end, SQLNet designs a generic SQL template that predicts the detailed information to form a complete SQL statement. However, this approach cannot be effectively extended to common relational database scenarios, as more advanced and flexible SQL syntaxes are often required to support the implementation of complex query intent when dealing with many database tables with join relationships. Considering that simple SQL templates cannot support the generation of complex query statements, SyntaxSQLNet [4] addresses the problem of difficult semantic parsing of complex relational databases in English scenarios by defining different syntax modules based on SQL syntax knowledge and recursively generating SQL statements through a tree structure. IRNet [5] uses a similar idea to SyntaxSQLNet, defining an intermediate transition layer between natural language and SQL statements, which establishes mapping relationships and guides the generation of SQL statements. RYANSQL [6] follows the idea of SQL templates in SQLNet and enhances the ability to generate complex SQL queries by introducing recursive methods. However, although existing approaches have been achieved some important advances in the task of semantic parsing on structured data, they suffer from three limitations as follows.

First, most works have focused on scenarios where natural language queries and database structures are in same language [6-9], while in-depth studies for cross-lingual scenarios are lacking. In practice, most widely used database management systems, such as Oracle, SQL Server, Access, etc., often use English to represent the names of tables and fields in relational databases, while the natural language used for querying can be a language other than English, such as Chinese. Figure 1 gives an example of converting a natural language query to a SQL statement in a multilingual scenario. For the Chinese query "返回抵达哈尔滨市的航班数量(Return the number of flights arriving in Harbin)", we need to generate the corresponding SQL statement based on the English database "Flight". The key to achieving this goal is, on the one hand, to establish a cross-lingual entity mapping between natural language queries and database tables and fields, such as "航班(Flight)" in the example, which corresponds to the database table "FLIGHES", "抵达(Arrival)" is an omitted expression for "抵达机场(Arrival Airport)" and corresponds to the field "DestAirport" in table "FLIGHES", "哈尔滨市(Harbin City)" corresponds to the field "City" in table "AIRPORTS", which specifies the value information "哈尔

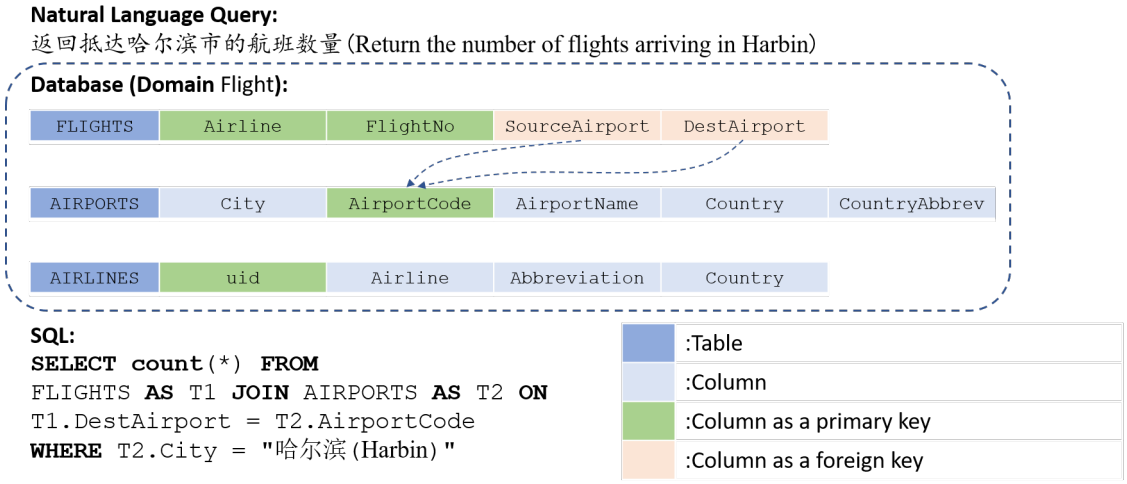

FIGURE 1. Example of converting a natural language query to a SQL statement in multilingual scenario.

滨(Harbin)". On the other hand, it is also necessary to parse the query logic and convert it to the corresponding SQL operations. The "航班数量(Number of Flights)" in example indicates to count all flight information of the query, i.e. "COUNT(*)", and connecting two tables ("FLIGHES" and "AIRPORTS") that related to the query by "JOIN" operation. However, most of the existing semantic parsing methods for same language use string matching to obtain entity mapping relationships between heterogeneous data [5,6,8,10], which also means that these methods will not be applicable in cross-lingual scenarios. Luo et al. [11] solve the cross-lingual entity linking problem by generating candidate entities through translation and fusing entity features on different dimensions in web forms, but the translation process not only requires additional task cost, but also inaccurate translation results tend to cause error accumulation and propagation. Min et al. [12] attempt a cross-lingual word embedding approach to compensate for the incompatibility of word vectors between Chinese and English, but this approach only represents words independently, extracts shallow text representations, and ignores their context, which is not conducive to capturing long-term dependencies between entities.

Second, the syntactic differences between Chinese and English lead to the imbalance of minimum semantic units. While words in English are separated by spaces, there is no explicit lexicon boundary information in Chinese scenarios, and many approaches ignore this important difference when dealing with multilingual texts. For example, when processing text sequences containing Chinese and English, the WordPiece tokenization [13] method in multilingual pre-trained model M-BERT [14] takes a word as the smallest unit for English sequences and a character as the smallest unit for Chinese sequences. Specifically, assuming that the mixed English and Chinese input contains the words "航班(Flight)" and "Flight", the WordPiece tokenization will split the Chinese word "航班(Flight)" into "航(Sail)" and "班(Class)" characters, which loses the lexicon information in Chinese text, and the English "Flight" retains the original lexicon form, resulting in inconsistency between the minimal semantic units of two languages. For the semantic parsing task in Figure 1, the Chinese word "航班(Flight)" in query corresponds to the English word "FLIGHES" in database. However, after splitting the Chinese vocabulary, it is difficult for a model to detect the mapping relationship between heterogeneous data during contextual representation learning because the single Chinese characters "航(Sail)" and "班(Class)" do not have complete semantic information. Lei et al. [15] also find that

the model often confuses words with same semantic meaning due to different word splitting methods of WordPiece tokenization in the process of building schema links.

Finally, most existing methods have difficulty in explicitly characterizing the primary and foreign key relationships between different tables in a database. Unlike single table Q&A scenario, when dealing with relational databases, the queries raised by users often involve multiple tables, which are related to each other by fields with primary and foreign key attributes. In the sample database(Flight) shown in Figure 1, the field attributes in each table are marked using different colors and the relationships between the tables are indicated by dashed lines with arrows. There are two fields with foreign key attributes "SourceAirport" and "DestAirport" in table "FLIGHES", both of them refer to the field "AirportCode" which is the primary key in table "AIRPORTS". From natural language query and corresponding SQL statement in the example, we can see that although the final answer of query comes from the table "FLIGHES", it also needs to be combined with the table "AIRPORTS" to form a conditional constraint, i.e. "AIRPORTS.City = 哈尔滨(Harbin)", and the above primary and foreign key relationship ensure that SQL query actions can operate consistently across tables. It is worth noting that only a simple example is given in Figure 1. In a real scenario, a table can have multiple primary and foreign keys, which means it can be associated with multiple tables at the same time, and this poses a challenge for relational database modeling. Therefore, the model is required to be able to accurately perceive the associated tables based on natural language queries. BRIDGE [8] designs a meta-data method for representing table and field characteristics in relational databases. However, this method simply distinguishes whether each field in the table is a primary or foreign key, and ignores the specific relationships between them.

To overcome the above difficulties, we focus on the CSpiser dataset proposed in literature [12], a cross-lingual Q&A task based on a relational database, and propose a semantic parsing model DBSQL based on Chinese lexicon enhancement and database schema relationship awareness to generate SQL statements that reflect the user's query intent. In the process of encoding, in order to handle the cross-lingual semantic parsing scenario where natural language query is in Chinese and database schema is in English. First, a multilingual pretrained model M-BERT is used as an initialization encoder to unite two heterogeneous data and represent table and field information in database schema by a special tag-based serialization encoding method to obtain a global representation of different structural data. Then, a LSTM(Long Short-Term Memory) structure is used to capture the long-term dependencies between contexts in the jointly encoded sequences and obtain the common features of multilingual heterogeneous data, thus effectively avoiding the local errors caused by word embedding and the semantic bias caused by translation. Subsequently, with the common features, we design corresponding methods to further extract the private features of two kinds of heterogeneous data according to their syntactic and structural characteristics, respectively, which compensate the limitations of existing methods in multilingual semantic learning and database complex relationship modeling. Experimental results show that DBSQL achieves 54.6% exact matching accuracy on the publicly available dataset CSpider and achieves a 4.1% advantage over the existing state-of-the-art methods. In this paper, we make four contributions.

(1) We design a relational database semantic parsing model DBSQL for cross-lingual scenarios, which can effectively alleviate three limitations in most existing approaches, including limited language scenarios, inconsistency of the minimum semantic units, and the difficulty of characterizing complex tabular relationships in databases.

(2) We propose a heterogeneous data common feature extraction method. It obtains common features of cross-lingual heterogeneous data in same semantic space by combining

serialized encoding of custom tokens, which effectively bridges the language and data format gaps in database-based Q&A tasks.
(3) We propose a Chinese query feature enhancement method. Through an alignment attention mechanism, lexical information is incorporated into Chinese character representations to alleviate the problem of imbalance in the minimum semantic units when encoding multilingual sequences.
(4) We propose a relational database schema relationship-aware method. It explicitly defines the connection relationships of database tables and obtains their relational representations through a relational attention mechanism, which effectively describes the complex feature in relational databases.

2. **Related Work.** With the development of deep learning techniques, the semantic parsing task of transforming natural human utterances into machine-understandable logically structured representations has been extended to generalizable Natural Language Interface to Database(NLIDB) [16], which maps natural language queries posed by users to SQL statements that can be executed on relational databases. The database used for inference is not visible at the time of training, which requires the model to be generalizable and expandable across different domain data.

This task is earlier focused on scenarios where the query's knowledge base consisted of a single table [7,17]. However, in practice, due to the widespread use of database systems, people often expect to find answers directly from a database containing a large number of tables. This work is considered challenging [10] because it must not only be able to understand the intent of people's spoken questions, but also establish accurate mapping relationships between entities in queries and tables, fields, values in the database to generate logical SQL statements.

Most of the current works on this task focus on monolingual scenarios where both the user query and the database are in English or Chinese. RYANSQL [6] designs a fillable template for complex SQL statements in the Spider dataset [18], using a recursive approach to convert nested statements in SQL statements to a non-nested form. Instead, we take a more general and flexible generative approach to building SQL statements, thus not being bound by templates. To better establish mapping relationships between natural language queries and database schemas, RAT-SQL [10] uses a relation-aware approach to achieve explicit alignment of entities in natural language queries with database schemas and decodes them with a SQL syntax tree structure. We also consider the important aspect of relationship awareness, but unlike RAT-SQL, for cross-lingual scenarios we focus more on the primary and foreign key relational features in database schema without matching the entity relationships between queries and databases, and use a generative decoding approach based on a hybrid pointer network. BRIDGE [8] matches strings related to natural language queries from database table contents in the encoding process to enhance the mapping relationships between heterogeneous data. To avoid the drawback that string matching methods are not applicable to cross-lingual texts, we extract private features of two heterogeneous data separately and construct hybrid feature sequences, which can improve the semantic imbalance in cross-lingual scenarios and obtain more detailed database schema representations.

Influenced by existing database systems, the names of database tables and fields are usually expressed in English to improve compatibility during the construction of relational databases, but the languages used to interact with them are diverse. For such cases, Min et al. [12] constructed CSpider dataset, which is similar to Spider dataset. It requires the model to generate SQL statements that can be executed on database based on user queries, but the queries in CSpider are in Chinese and the database schemas retain the original

English format to make it more compatible for actual Chinese application scenario. However, migration from monolingual to cross-lingual tasks is difficult. This is because in addition to dealing with formatting differences between heterogeneous data, there is also a semantic divide between texts in different languages to be overcome. More importantly, in monolingual scenarios, most approaches tend to establish mapping relationships between queries and databases through explicit methods such as text matching [8,10,15,19], which will no longer be applicable in cross-lingual scenarios. Released along with CSpider is a cross-lingual model improved from SyntaxSQLNet, which obtains cross-lingual representations of heterogeneous data in same semantic space by multilingual word embedding, and decouples SQL statements into multiple components simultaneously combined with a syntax tree structure to form a pipeline task. We instead perform representation learning on cross-lingual heterogeneous data through a separate sequence-to-sequence task, thus avoiding the problem of error accumulation in pipeline tasks. There are also some works that use translation methods to unify different languages [11]. However, in the Q&A scenario of this paper, user questions are often characterized by colloquial phrases and field names in database lack contextual descriptions, which often produces poor translation results and leads to ambiguities and errors. Instead, we study the cross-lingual Q&A approach without introducing additional translation operations.

3. **Methodology.** A cross-lingual Q&A task based on a relational database can be defined as follows. Given a Chinese natural language query Q and an English relational database schema $D = \left\{T_1, ..., T_{|D|}\right\}$. The goal is to generate a SQL statement that satisfy the query intent, where the query input $Q = [q_1, ..., q_{|Q|}]$ is a Chinese sequence containing $|Q|$ characters, T_i refers to the name of the *i-th* table in database D, and $i \in \{1, ..., |D|\}$, $|D|$ is the number of tables in database D. For table T_i , we have $T_i = \left\{C_1^i, ..., C_{|T_i|}^i\right\}$, where C_j^i refers to the name of the *j-th* field in table T_i . At the same time, according to the database storage format, all fields are given a data type and denoted by TP_j^i , which includes numeric and text, and $j \in \{1, ..., |T_i|\}$,$|T_i|$ is the number of fields in table T_i. In addition to this, there are a number of fields that are marked as primary and foreign keys for indexing and joining tables.

The DBSQL model framework proposed in this paper is shown in Figure 2, which consists of three stages in general: encoding layer, middle layer and decoding layer. Specifically, for natural language queries, we design a Chinese lexicon enhancement method to solve the minimum semantic unit inconsistency problem in the encoding process of Chinese and English heterogeneous data. Through an attention mechanism based on lexicon alignment, each Chinese character is characterized by incorporating the semantic information of the vocabulary in which the character is located. For relational databases, a relationship-aware layer is designed to explicitly model the complex tabular relationships. We define primary and foreign key relationships between different fields and add this feature to attention mechanism used to characterize the database schema. Finally, we use the private features of two heterogeneous data to form a hybrid feature sequence which guides the subsequent decoding operation. In the decoding process, in order to address the limitation that the SQL template-based filler method cannot effectively generate SQL statements containing complex syntactic structures such as multi-table joins and nested queries, we use sequence generation to obtain more flexible forms of SQL statements by improving the pointer network structure and dynamically selecting the appropriate content from SQL proprietary vocabulary and mixed feature sequences.

The encoding layer jointly encodes two cross-lingual heterogeneous data to obtain common features(Section 3.1). The middle layer acquires private features(Section 3.2) of

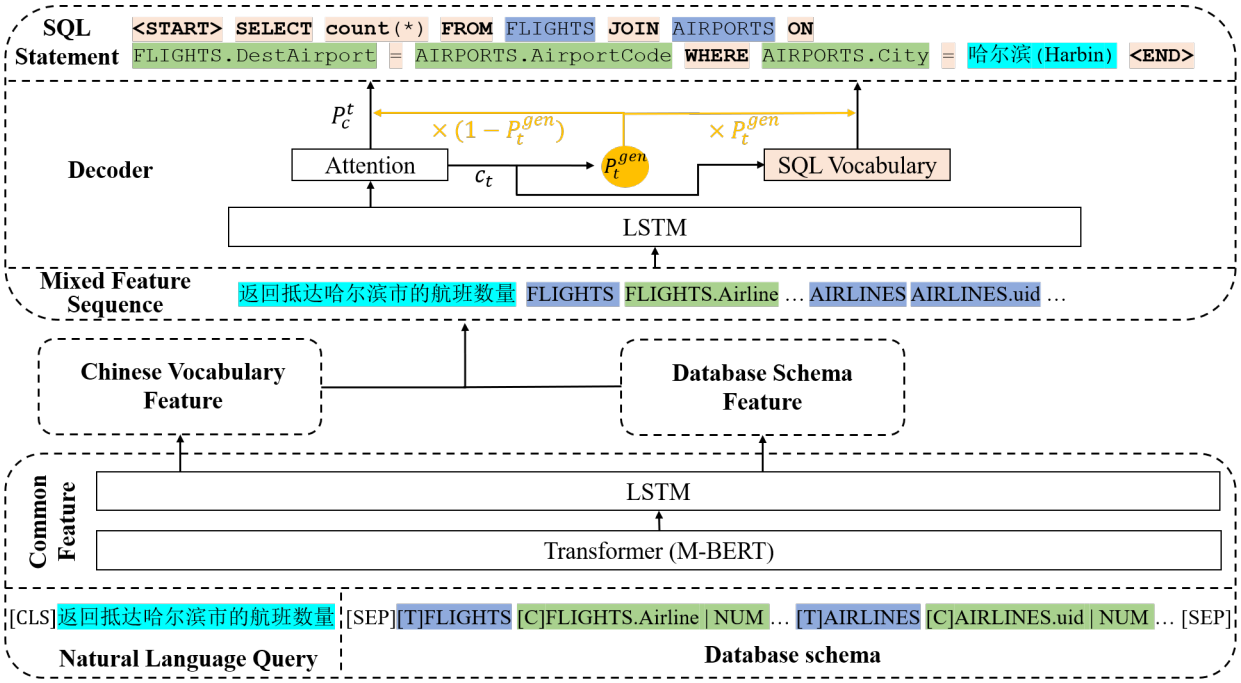

FIGURE 2. Framework of DBSQL.

natural language queries and database schemas respectively through Chinese vocabulary enhancement and database schema awareness according to the characteristics of different data formats. The decoding layer generates SQL statements by fusing private features of heterogeneous data and using them as input to the hybrid pointer network. The decoding layer fuses private features of heterogeneous data into a sequence of hybrid features and generates SQL statements through an improved pointer network(Section 3.3). In this section, we will describe the details of DBSQL from the bottom up, following the three stages mentioned above.

3.1. **Heterogeneous Data Common Feature Extraction.** Pre-trained models [14,20-21] have achieved better results on several natural language processing tasks by using a large-scale corpus to learn generic semantic representations. M-BERT is a multilingual pre-trained model [22] that uses the same architecture and training method as BERT [14], with the difference that BERT is trained on a single corpus, while M-BERT is trained on a corpus containing 104 languages, enabling multilingual shared contextual word embedding representation. In the process of capturing mapping relationships between entities in cross-lingual scenarios, in order to compensate for the inefficiency and high error of translation means, we design a heterogeneous data modeling approach based on M-BERT to jointly encode the Chinese query Q and the English database schema D. This method could obtain their contextual representations, and capture the mapping relationships between them implicitly through multi-heads self-attention mechanism in Transformer [23].

Due to the differences in structure as well as language conventions between free text and structured databases. It is also necessary to pre-process these two types of data separately before encoding. In Chinese query Q, some English terms are still interspersed, such as "返回VLDB会议的主页(Return to the home page of VLDB conference)", where "VLDB" is a rare term that does not exist in the vocabulary of M-BERT. For such unregistered words, the WordPiece method slices them into two subwords "VL" and "## DB" that exist in vocabulary. However, to facilitate subsequent lexicon alignment operations(Section 3.2), we modify the original word splitting mechanism to still take a

character-level segmentation for such words. In database schema D, we find that the fields of different tables are similar or even identical. For example, fields such as "id", "name" appear in most tables, while there are also some fields such as "gender", "sex" that are difficult to distinguish. Therefore, we use the format "Table.Column|Type" to expand the representation of each field in table, where "Column" indicates the field name, "Table" is the table name where the field is located, "Type" indicates the data type of the field, ". " and "|" are special symbols used to separate different information of the field. The entire input sequence S can be represented as follows.

$$\begin{aligned} S = &[\text{CLS}], Q, [\text{SEP}], [\text{T}], T_1, [\text{C}], T_1.C_1^1|TP_1^1, ..., \\ &T_1.C_{|T_1|}^1|TP_{|T_1|}^1, ..., [\text{T}], T_{|D|}, [\text{C}], T_{|D|}.C_1^{|D|}|TP_1^{|D|} \\ &, ..., T_{|D|}.C_{|T_{|D|}|}^{|D|}|TP_{|T_{|D|}|}^{|D|} \end{aligned} \tag{1}$$

Where the special tags [T] and [C] are used to indicate table and field information in database, respectively. The [CLS] and [SEP] tags defined by M-BERT are used to represent the global information of the entire input and to distinguish between the two formats of data, respectively.

Subsequently, the sequence S is fed into the M-BERT encoder with 12 multi-heads self-attention layers, and the vector of the last layer is taken as the final output $\boldsymbol{S}$. Next, a Bi-LSTM(Bi-directional LSTM) is used to obtain the long-term dependencies in the vector, which is calculated at each step as follows.

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{S}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{2}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{S}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{3}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{S}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{4}$$

$$\tilde{\boldsymbol{z}}_t = \tanh(\boldsymbol{W}_z \boldsymbol{S}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{5}$$

$$\boldsymbol{z}_t = \boldsymbol{f}_t \odot \boldsymbol{z}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{z}}_t \tag{6}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{z}_t) \tag{7}$$

Where $\boldsymbol{S}_t$ denotes the input to the sequence vector $\boldsymbol{S}$ at moment t. $\boldsymbol{i}_t \in [0, 1]$, $\boldsymbol{f}_t \in [0, 1]$, $\boldsymbol{o}_t \in [0, 1]$ denote the input gate, forget gate and output gate, respectively. $\sigma(\cdot)$ refers to a sigmoid function, and $\odot$ denotes element-wise product. $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{b}$ with different subscripts denote trainable parameters.

The common feature vector consists of the hidden layer vectors $\boldsymbol{h}_Q$ and $\boldsymbol{h}_D$ corresponding to Q and D, respectively, which are represented as follows.

$$\boldsymbol{h}_S = [\boldsymbol{h}_Q, \boldsymbol{h}_D] = \text{Bi - LSTM}(\boldsymbol{S}) \tag{8}$$

Where Bi-LSTM denotes bi-directional LSTM calculation. $\boldsymbol{h}_S \in \mathbb{R}^{|S| \times d}$, and $|S|$ denotes the length of the input sequence S and d denotes the vector dimension.

To simplify the database schema representation, we extract the semantic vectors of all tables and fields from $\boldsymbol{h}_D$, which are the hidden vectors corresponding to the special tokens [T] and [C] as the final data-base schema representation, and N_{TC} is the total number of [T] and [C].

3.2. **Heterogeneous Data Private Feature Extraction.**

Chinese Lexicon Enhancement Based on Aligned Attention Mechanism. Li et al. [24] demonstrate the effectiveness of incorporating lexicon information into character-level vectors. To address the minimum semantic unit imbalance problem in cross-lingual text encoding, inspired by their work, we introduce a Chinese lexicon enhancement module based on an alignment attention mechanism. For the vector $\boldsymbol{h}_Q \in \mathbb{R}^{|Q|\times d}$, which corresponds to the natural language query Q in the common feature $\boldsymbol{h}_S$. The attention distribution between characters is first calculated by scaling the dot product as follows.

$$\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{h}_Q \tag{9}$$

$$\boldsymbol{\alpha} = \text{softmax}\left(\frac{(\boldsymbol{W}_Q\boldsymbol{Q})^{\mathrm{T}}(\boldsymbol{W}_K\boldsymbol{K})}{\sqrt{d/H_1}}\right) \tag{10}$$

Where $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ denote the trainable parameter matrix, and H_1 denotes the number of heads of the attention mechanism.

In order to obtain the lexicon information of Chinese input sequences, for Chinese natural language queries $Q = [q_1, ..., q_{|Q|}]$, We divided Q into non-overlapping lexicon chunks using the word division tool Hanlp, which is calculated as follows.

$$\phi\,(Q) = [V_1, V_2, ..., V_{|V|}] \tag{11}$$

Where $\phi\,(\bullet)$ refers to the Chinese word separation operation.$V_i = [q_s, q_{s+1}, ..., q_{s+|V_i|-1}]$ denotes the i-th word in the query and the length of V_i is $|V_i|$, s is the index number of the character in Q, $|V|$ is the number of words after splitting, and $|V| \leq |Q|$.

Subsequently, the attention distribution $\boldsymbol{\alpha}$ is aligned with the word separation results and the attention distribution in the same word is aggregated and calculated as follows.

$$\rho(\boldsymbol{\alpha}) = [\boldsymbol{\alpha}_1^1, ..., \boldsymbol{\alpha}_1^{|V_1|}, ..., \boldsymbol{\alpha}_{|V|}^1, ..., \boldsymbol{\alpha}_{|V|}^{|V_{|V|}|}] \tag{12}$$

Where $\rho(\bullet)$ refers to word alignment and aggregation operations. $\boldsymbol{\alpha}_i^j$ refers to the weight of the attention distribution corresponding to the j-th character in the word V_i ,and $i \in [1, ..., |V|]$, $j \in [1, ..., |V_i|]$.

To integrate the attention weights of individual characters in the same word, we use the same hybrid pooling approach [25]. The attention weight α_i^V of word V_i after integration is calculated as follows.

$$\boldsymbol{\alpha}_i^V = [\boldsymbol{\alpha}_i^1, ..., \boldsymbol{\alpha}_i^{|V_i|}] \tag{13}$$

$$\boldsymbol{A}_i = \boldsymbol{\lambda} p_1(\boldsymbol{\alpha}_i^V) + (1 - \boldsymbol{\lambda}) p_2(\boldsymbol{\alpha}_i^V) \tag{14}$$

Where $p_1(\bullet)$ and $p_2(\bullet)$ denote the average pooling and maximum pooling operations, respectively, and $\boldsymbol{\lambda}$ is a trainable parameter.

In the process of incorporating word information on character-level vectors, for the case where a single character remains after splitting, we no longer consider applying influence to it. Therefore, depending on the number of characters after separation, we design an attention mask matrix $\boldsymbol{\eta} \in \mathbb{R}^{|Q|\times|Q|}$, which is initialized as an all-zero matrix. For any word V_i, given the following mask strategy, to compute its corresponding $\boldsymbol{\eta}_i \in \mathbb{R}^{|Q|\times|V_i|}$.

$$\boldsymbol{\eta}_i = \begin{cases} 1 \quad , |V_i| = 1 \\ \dfrac{\boldsymbol{A}_i}{\boldsymbol{\alpha}_i^V}, |V_i| > 1 \end{cases} \tag{15}$$

Then, according to the mask strategy, the attention distribution $\boldsymbol{A}$ containing lexicon information is incorporated into the character-level vector, and the fused character vector

$\boldsymbol{h}_Q^A \in \mathbb{R}^{|Q|\times d}$ is calculated. $\boldsymbol{h}_Q^A$ will be superimposed on the initial encoding vector $\boldsymbol{h}_Q$ to obtain the private feature $\boldsymbol{h}_Q^E \in \mathbb{R}^{|Q|\times d}$ of the query, which is calculated as follows.

$$\boldsymbol{h}_Q^A = (\boldsymbol{A}\boldsymbol{\eta})(\boldsymbol{W}_V\boldsymbol{V}) \tag{16}$$

$$\boldsymbol{h}_Q^E = \text{Concat}(\boldsymbol{h}_Q, \boldsymbol{h}_Q^A) = [\boldsymbol{h}_Q : \boldsymbol{h}_Q^A] \tag{17}$$

Where $\boldsymbol{\eta} = [\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_{|V|}]$, $\boldsymbol{W}_V$ is a trainable parameter matrix, and Concat means vector splicing operation.

Database Schema Representation Based on Relational Awareness. In a relational database, tables are not independent of each other, but are connected by fields with primary and foreign key attributes. Specifically, the primary key is used as a unique identifier for each record, ensuring the uniqueness of the data, while the foreign key is used to associate different tables, ensuring the consistency of the data. From the example in Figure 1, we can find that a natural language query often involves multiple tables, and in the corresponding SQL statement, these tables are connected by the "JOIN...ON..." SQL operation based on the primary and foreign key relationships between them.

However, the database schema relationship is not explicitly defined in the process of extracting common features of heterogeneous data, thus easily causing the model to overlook this detail, which is not conducive to cope with query scenarios involving multiple tables. Therefore, after obtaining the common features $\boldsymbol{h}_S$, to enable the model to perceive such pre-existing relational features in encoding phase. Inspired by previous work [10,26], we introduce a relationship-aware layer that explicitly incorporates the primary and foreign key relationships to the feature vector $\boldsymbol{h}_{TC} \in \mathbb{R}^{N_{TC}\times d}$ by a special attention mechanism. Eventually, the private characteristics of the database schema D are further obtained.

Table 1. Description of the database schema relationship.

Object x	Object y	Mark	Description
Field	Table	CT-x-PK-y	x is a primary key of y.
		CT-x-FK-y	x is a foreign key of y.
Table	Field	TC-y-PK-x	y is a primary key of x.
		TC-y-FK-x	y is a foreign key of x.
Field	Field	CC-x-FK-y	x is a foreign key of y.(x and y are both fields)
		CC-y-FK-x	y is a foreign key of x.(x and y are both fields)
Table	Table	TT-x-FK-y	A foreign key for y exists in x.
		TT-y-FK-x	A foreign key for x exists in y.

In order to distinguish primary and foreign key relationships in relational databases in detail, we define a binary relationship set R for different objects, which is described as shown in Table 1. For any two objects x and y in the database schema, a trainable relation matrix $\boldsymbol{R}_{x,y} = [\boldsymbol{r}_{x,y}^1, ..., \boldsymbol{r}_{x,y}^{N_R}]$ is used to represent the unidirectional relation, where $\boldsymbol{r}_{x,y}^i$ denotes a trainable word embedding of the i-th binary relation in the set R of unidirectional relations from x to y, and N_R is the total number of relations in the set R. It is worth noting that in the relation matrix $\boldsymbol{R}_{x,y}$, if a relation does not exist in R, its corresponding binary relation representation is assigned to zero.

Unlike the definition of RAT-SQL, the relationships defined here only consider the primary and foreign key relationships, and not the containment and inclusion relationships between tables and fields. This is because such relationships have been distinguished by the special markers [T] and [C] during the common feature extraction. In addition to this,

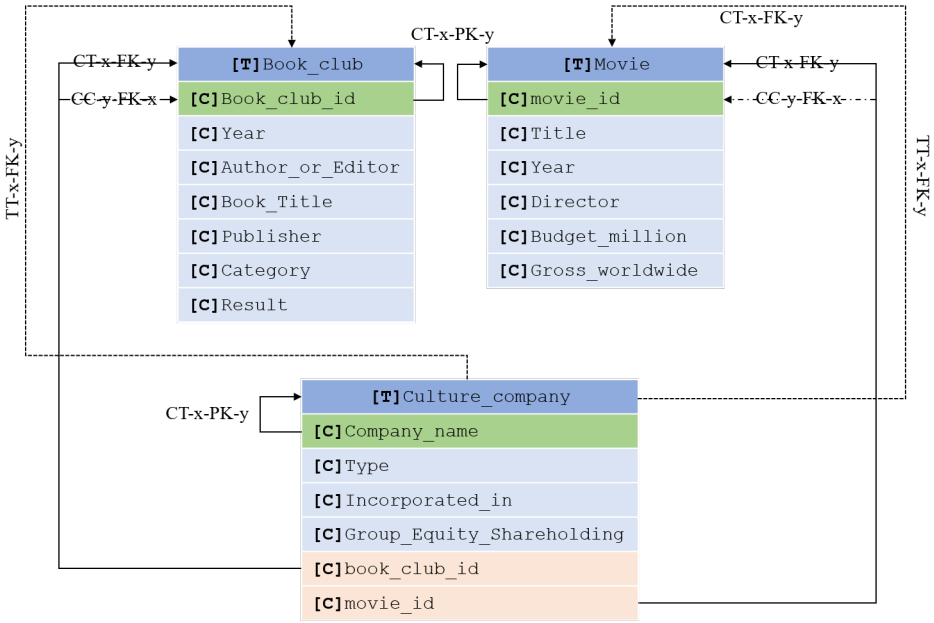

FIGURE 3. Example of database schema relationship-awareness.

the feature vectors corresponding to the special tokens [T] and [C] are used to represent the table and field names in database schema, respectively, during the computation of the relational awareness layer. Figure 3 gives an example of representing some of the primary and foreign key relationships in a database schema. We use the same way as the example in Figure 1 to distinguish database components with different colors, while the relationships between them are pointed out using different types of lines according to the definitions in Table 1(Only some of relationships are shown in Figure 3 to promote readability of the graph).

The design of the relationship-aware layer is carried out based on a self-attention mechanism, which aims to incorporate relational characteristics as a priori knowledge when calculating the attention weights between two objects in the database schema. Specifically, the attention weight $\boldsymbol{\beta}_{m,n}$ between the m-th object and the n-th object is first calculated as follows.

$$\boldsymbol{\beta}_{m,n} = \text{softmax}\left(\frac{(\boldsymbol{W}_Q^R \boldsymbol{h}_{TC}^m)(\boldsymbol{W}_K^R \boldsymbol{h}_{TC}^n + \boldsymbol{R}_{m,n}^K)^T}{\sqrt{d/H_2}}\right) \tag{18}$$

Where $m, n \in [1, ..., N_{TC}]$, and $\boldsymbol{h}_{TC}^m$ denotes the vector of hidden layers corresponding to the m-th special label. $\boldsymbol{W}_Q^R, \boldsymbol{W}_K^R \in \mathbb{R}^d$ are trainable parameter matrices, $\boldsymbol{R}_{m,n}^K$ is a trainable relationship matrix between m and n. H_2 indicates the number of attention heads.

And then the attention distribution $\boldsymbol{z}_m$ for each object is calculated by using the scaled dot product, which is calculated as follows.

$$\boldsymbol{z}_m = \sum_{n=1}^{N_{TC}} \boldsymbol{\beta}_{m,n}(\boldsymbol{W}_V^R \boldsymbol{h}_{TC}^n + \boldsymbol{R}_{m,n}^V) \tag{19}$$

Where $\boldsymbol{h}_{TC}^n$ denotes the vector of hidden layers corresponding to the n-th special label. $\boldsymbol{W}_V^R \in \mathbb{R}^d$ is a trainable parameter matrix. $\boldsymbol{R}_{m,n}^V$ is a trainable relationship matrix between m and n, and $\boldsymbol{R}_{m,n}^V = \boldsymbol{R}_{m,n}^K$.

The attention distribution of all objects is aggregated to the matrix $\boldsymbol{z}$, which is then concatenated with the input feature vector $\boldsymbol{h}_{TC}$ to obtain the final database schema private feature $\boldsymbol{h}_{TC}^{E}$

$$\boldsymbol{z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_{N_{TC}}] \tag{20}$$

$$\boldsymbol{h}_{TC}^{E} = \text{Concat}(\boldsymbol{h}_{TC}, \boldsymbol{z}) = [\boldsymbol{h}_{TC} : \boldsymbol{z}] \tag{21}$$

Where Concat is the vector splicing operation

3.3. **Hybrid Pointer Network Decoding.** Unlike other languages, SQL is a standard computer language with a rigorous logical structure and proprietary syntax rules. Inspired by the existing work [8,27], we employ a hybrid pointer generation network to generate SQL statements by mixing information from different sources during the decoding process. In the example in Figure 2, we mark the source of the components with different colors in SQL statement and show their usage in various parts of the model. It can be found that the sources composing the SQL statements consist of two main parts, which include mixed feature sequence and SQL vocabulary. The mixed feature sequence consists of the natural language query and database information corresponding to the model input, and it should be noted that the input to decoder is actually a vector form of the mixed feature sequence, containing private features of the two heterogeneous data. The SQL vocabulary contains 90 commonly used SQL keywords, functions and symbols, such as "SELECT", "WHERE", and "GROUP BY", as well as the sequence start/stop tokens "<START>"and "<END>". In each decoding step, an attention mechanism is used to focus on the key information in input mixed sequence vector and to determine the source of the generated sequence through control unit P_t^{gen} . As shown in the SQL statement in Figure 2, the sequences"<START>", "SELECT", etc. are from the SQL vocabulary, while "FLIGHTS", "AIRPORTS.City", etc. are from the database schema-related sequences in mixed feature sequence, and "哈尔滨(Harbin)" is from the natural language query-related sequences in mixed feature sequence. The calculations in the decoding process are explained in detail below.

First, two private features $\boldsymbol{h}_{Q}^{E}$ and $\boldsymbol{h}_{TC}^{E}$ of heterogeneous data are concatenated together to obtain mixed feature $\boldsymbol{h}_{S}^{E} = [\boldsymbol{h}_{Q}^{E}, \boldsymbol{h}_{TC}^{E}] \in \mathbb{R}^{N \times d}$, and $N = |Q| + N_{TC}$. We also define the output SQL sequence as $Y = [y_1, ..., y_M]$, whose length is M. Subsequently, $\boldsymbol{h}_{S}^{E}$ is fed into a single-layer LSTM encoder, and each step t is computed as follows.

$$\boldsymbol{h}_{t}^{enc} = \text{Bi - LSTM}(\boldsymbol{h}_{t-1}^{enc}, \boldsymbol{h}_{t}^{E}) \tag{22}$$

$$\boldsymbol{H}^{enc} = [\boldsymbol{h}_{1}^{enc}, ..., \boldsymbol{h}_{N}^{enc}] \tag{23}$$

Where $\boldsymbol{h}_{t}^{E}$ is the vector of $\boldsymbol{h}_{S}^{E}$ input to encoder at step t, $\boldsymbol{h}_{t}^{enc}$ is the hidden state after the encoding corresponding to $\boldsymbol{h}_{t}^{E}$, and $\boldsymbol{H}^{enc}$ is the hidden state of all input sequences.

When decoding, the last moment hidden state of the LSTM encoder is taken as the initial input of the decoder and $\boldsymbol{h}_{0}^{dec} = \boldsymbol{h}_{N}^{enc}$. In order to generate sequences with biased attention to decoder input, we use an attention mechanism to dynamically compute context vector, and use another unidirectional LSTM as a decoder to generate output sequence, which is computed as follows.

$$\boldsymbol{\alpha}_{i}^{t} = \text{softmax}(\frac{(\boldsymbol{W}_{q}\boldsymbol{h}_{i}^{\text{enc}})^{\text{T}}(\boldsymbol{W}_{k}\boldsymbol{h}_{t-1}^{\text{dec}})}{\sqrt{d}}) \tag{24}$$

$$\boldsymbol{c}_t = \sum_{i=1}^{N} \boldsymbol{\alpha}_{i}^{t}\boldsymbol{h}_{i}^{\text{enc}} \tag{25}$$

$$\boldsymbol{h}_{t}^{\text{dec}} = \text{LSTM}(\boldsymbol{h}_{t-1}^{\text{dec}}, [\boldsymbol{e}_{y_{t-1}}, \boldsymbol{c}_t]) \tag{26}$$

Where $\boldsymbol{h}_t^{\text{dec}}$ denotes the hidden layer state of decoder at step t. $\boldsymbol{e}_{y_{t-1}}$ refers to the word embedding of predicted target sequence y_{t-1} at step $t-1$. At each step t, the decoding action has two choices: copying information from the mixed feature sequence or selecting components from the SQL vocabulary. The specific computation process is as follows.

$$P_t^{\text{gen}} = \sigma(\boldsymbol{W}_c \boldsymbol{c}_t + \boldsymbol{W}_h \boldsymbol{h}_t^{\text{dec}} + \boldsymbol{b}) \tag{27}$$

$$P_v^t = \text{softmax}(\boldsymbol{W}_2(\boldsymbol{W}_1[\boldsymbol{h}_t^{dec}, \boldsymbol{c}_t] + \boldsymbol{b}_1) + \boldsymbol{b}_2) \tag{28}$$

$$P_c^t = \sum_{i:X_i=w} \boldsymbol{\alpha}_i^t \tag{29}$$

$$P(y_t = w) = P_t^{\text{gen}} P_{\text{v}}(w) + (1 - P_t^{\text{gen}}) P_c^t \tag{30}$$

Where $\boldsymbol{W}$ and $\boldsymbol{b}$ with different subscripts are trainable parameter matrices. P_t^{gen} is a control unit that decides the next decoding action and selects the source of next generated sequence at moment t. $P_v^t(w)$ denotes the probability distribution of the sequence w generated from the SQL vocabulary at moment t. P_c^t determines the probability distribution of copying components from the mixed feature sequence at moment t by summing the probabilities of all corresponding table and field positions in the attention distribution $\boldsymbol{\alpha}^t$. X_i refers to the special labels such as [T] and [C], which denote the table and field name for database in the original input, respectively.

The loss function for whole training process is the negative log likelihood of target SQL sequence g_t , which is calculated as follows.

$$L = -\frac{1}{T} \sum_{t=0}^{T} \log(P(y_t = g_t)) \tag{31}$$

To avoid greedy search from falling into local optimal solutions during decoding, in inference stage, we use a beam search method to obtain a larger view at decoding time. When generating each sequence, the top k sequences with the highest probability are considered simultaneously.

4. **Experiments.**

4.1. **Dataset.**

The dataset used for the experiments in this paper is CSpider, which is the only known publicly available multilingual cross-domain Text-to-SQL dataset. The construction of CSpider originated from the English dataset Spider, with the difference that it uses Chinese natural language queries, while the database language is English. CSpider contains more than 9600 natural language query and SQL statement pairs involving 166 databases, with an average of 5.28 tables per database, and the databases are not cross-used in the process of dividing the train, development and test set. According to the complexity of SQL statements, the difficulty is classified according to four levels, such as Easy, Medium, Hard and Extra Hard. In this paper, we use same data classification method as released version, and since the final test set is not publicly available, the analysis is performed on development set to ensure the fairness of experimental results.

4.2. **Evaluation Metric.**

The evaluation metric in this paper is exact match accuracy of SQL statements. Assuming that the total number of test samples in the dataset is N, exact match accuracy refers to the number of generated SQL statements that formally match the ground truth exactly as a proportion of the total number of samples, which is $\text{ACC}_{\text{ex}} = N_{em}/N$. To further investigate the performance of generated results on different SQL components, we also evaluate the component matching accuracy, which is the *F1*

score of each SQL keyword component in generated SQL statements. The calculation is as follows.

$$PRE = \frac{TP}{TP + FP} \tag{32}$$

$$REC = \frac{TP}{TP + FN} \tag{33}$$

$$F1 = \frac{2 \times PRE \times REC}{PRE + REC} \tag{34}$$

Where *TP, TN, FP* and *FN* denote true positive, true negative, false positive and false negative, respectively. *PRE* and *REC* denote precision and recall, respectively.

4.3. **Implementation Details.** The model DBSQL proposed in this paper is developed on pytorch, and the version of the pre-trained model used for common feature extraction is bert-base-multilingual-uncased. Xavier [28] is used to initialize the model parameters, Adam is used as the training optimizer and the learning rate is 0.0002. The loss function is in the form of cross-entropy. The same practice as BRIDGE is used for training, which is to randomly disrupt the order of tables in same database during encoding, and some of the hyperparameters are set as shown in Table 2.

Table 2. Part of the model hyperparameter settings.

Hyperparameter	Value
Maximum input sequence length / character	512
Common feature extraction vector / dimension	768
LSTM encoder hidden layer vector / dimension	400
LSTM decoder hidden layer vector / dimension	400
Chinese query lexicon enhancement attention head count H_1	8
Database schema relationship-aware attention head count H_2	8
Beam search size	16
Training step	200000
Batch size	8

4.4. **Results and Discussion.** In this paper, we mainly compare with 7 advanced methods. SyntaxSQLNet uses a sequence-to-tree architecture that enables awareness of generation history and fields in the table during the decoding process by building a network of syntax trees. Min et al. [12] improve on it and use it as a baseline model for CSpider, while the effects of different word separation and word embedding methods on the model are also compared. To obtain a more detailed comparison, we retrain the model based on their publicly available code.

RYANSQL defines a generic sketch structure for the generation of complex SQL statements. To improve the sketch coverage of SQL statements, a recursive approach is used and the statement location code is introduced to convert nested SQL statements into non-nested form.

DG-MAML [29] improves model performance in zero-sample scenarios by designing a meta-learning framework that reconstructs the train and test sets, and also proposes a base parser that integrates schema linking [10] and multilingual word embedding.

RAT-SQL uses schema linking to explicitly align entities in user queries with table and field names in the database to enhance relational mapping between heterogeneous data. Since explicit text matching is not possible in multilingual scenario, schema linking is not considered in this experiment.

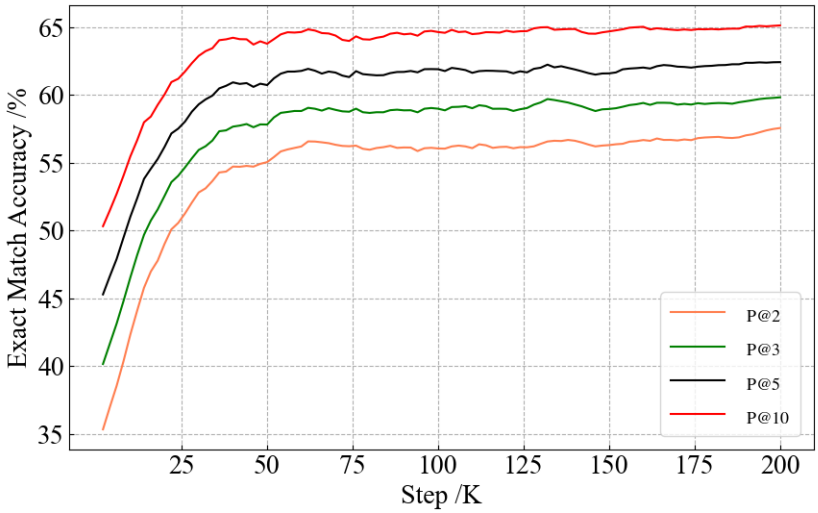

FIGURE 4. Comparison of results with different number of candidates.

BRIDGE enhances the mapping relationships between heterogeneous data by extracting values involved in queries from database content through explicit text matching methods. Since it is originally designed for English scenarios, we modify it by eliminating the value extraction session and introducing M-BERT to accommodate multilingual scenarios.

TABLE 3. Performance comparison of different models on dev set.

Model	ACC_{ex}
Word embedding	
SyntaxSQLNet	17.6
Base Parser	31.0
Base Parser + DG-MAML	35.5
Pre-trained model(M-BERT)	
RYANSQL	41.3
RAT-SQL	41.4
Base Parser + DG-MAML	50.1
BRIDGE	50.5
DBSQL(Ours)	54.6

Table 3 gives a comparison of the exact matching accuracy of the different models on the dev set. It can be seen that the exact matching accuracy of DBSQL reaches 54.6%, which achieves an absolute advantage of 4.1% compared with the state-of-the-art BRIDGE results. It is also found that the overall performance of the model is significantly improved after the introduction of the pre-trained model compared to the approach using word embedding

Since the sampling method of beam search is used in the decoding process, we further explore the correct prediction of SQL statements with different number of candidate results and denote the exact match-ing accuracy in the top k candidate results by P@k. Figure 4 gives the variation of exact matching accuracy of the model on dev set with the number of training steps in same training when the number of candidate results varies. It can be seen that as the candidate range increases, the correct answer can still be matched from other non-first candidates and when $k = 10$, the exact match accuracy of the

candidate results reaches more than 65%, which shows that DBSQL has the potential to be further improved.

Table 4. Performance comparison of models on different SQL components(dev set).

Component	SyntaxSQLNet			BRIDGE			DBSQL(Ours)		
	PRE	REC	F1	PRE	REC	F1	PRE	REC	F1
SELECT	44.1	44.0	44.0	73.4	72.6	73.0	75.7	75.1	75.4
SELECT(Without AGG)	45.0	45.0	45.0	74.4	73.6	74.0	76.6	76.0	76.3
WHERE	22.8	22.4	22.6	67.7	63.3	65.4	70.8	68.2	69.5
WHERE(Without OP)	29.6	29.2	29.4	71.3	66.7	68.9	74.3	71.6	72.9
GROUP(Without HAVING)	42.3	41.4	41.9	70.4	71.4	70.9	73.3	74.4	73.8
GROUP	34.1	33.3	33.7	65.7	66.7	66.2	69.0	70.0	69.5
ORDER	45.1	46.5	45.8	70.2	70.8	70.5	71.1	71.7	71.4
AND/OR	95.0	96.7	95.8	98.4	99.6	99.0	98.8	99.5	99.2
IUEN	6.8	4.2	5.2	29.0	25.4	27.1	42.6	32.4	36.8
KEYWORDS	71.0	69.4	70.2	82.1	79.9	81.0	85.8	82.3	84.0

To further investigate the specific performance of DBSQL on different SQL statement components. We choose SyntaxSQLNet, the baseline model released with CSpider, and the current state-of-the-art BRIDGE model as the comparison objects, and conduct detailed comparison experiments on dev set, the results are shown in Table 4. It should be noted that AGG refers to the aggregate operation functions in SQL statement, which includes MAX, MIN, SUM, COUNT, AVG, and NONE for indicating no operation. INEN evaluates a series of components used for nesting, which includes INTERCEPT, UNION, EXCEPT, and NONE for indicating no nested relationships. KEYWORDS is used to evaluates keywords in each component of the SQL statement, such as SELECT, WHERE, GROUP BY and ORDER BY.

From the experimental results, it can be seen that DBSQL achieves significant improvement compared with SyntaxSQLNet on some common SQL components such as SELECT, WHERE, GROUP, ORDER, etc., and retains certain advantages compared with the state-of-the-art BRIDGE. For some infrequent complex components, such as IUEN which contains nested relationships, these components are usually found in difficult samples to reflect the complex query intent of users, and are one of the difficulties of this task. The experimental results show that SyntaxSQLNet has significantly lower performance on IUEN component compared to other components (*PRE*: 6.8%, *REC*: 4.2%, *F1*: 5.2%) and is barely able to organize nested statements efficiently. However, DBSQL improves such phenomena, and achieves 35.8%, 28.2% and 31.6% improvement in three evaluation metrics compared to SyntaxSQLNet in IUEN component, it also achieves a more significant advantage compared to BRIDGE (*PRE*: 13.6%, *REC*: 7.0%, *F1*: 9.7%).

Examples of SQL statements generated by DBSQL on dev set of different difficulty levels are further given in Table 5 to understand the validity of DBSQL more intuitively. We use same colors to mark entities in the query and corresponding mappings in the SQL statements, which are generated by DBSQL. We also give the generated results for the other two advanced models(SyntaxSQLNet, BRIDGE) in Table 4. The comparison shows that DBSQL can establish more accurate mapping relationships between cross-lingual entities and build more accurate SQL logic.

In Query 1, SyntaxSQLNet incorrectly interprets the entire query as a counting procedure and fails to generate the correct logical operation based on “不超过(not exceed)”.

BRIDGE incorrectly introduces the EXCEPT operation, which causes the entire query logic to be incorrect. DBSQL correctly employs GROUP BY and HAVING operations according to the query logic, and accurately establish the relationship between the "文档类型的代码(code for document type)" and the corresponding field.

The definite article "作者和导师(author and tutor)" before "姓氏(last name)" is omitted in Query 2. SyntaxSQLNet selects the wrong table and fails to match the "登录名(login name)" correctly. BRIDGE creates an incorrect mapping between "登录名(login name)" and "人名(personal name)". DBSQL successfully captures the mapping relationships between three entities.

Query 3 not only includes a multi-table join operation with three tables, but also omits part of definite description of the value "致谢(Acknowledgement)". SyntaxSQLNet selects wrong table, resulting in a deviation in the multi-table join process. The multi-table join process for BRIDGE is incorrect and does not correctly reason about the source of "致谢(Acknowledgement)". DBSQL generates correct result and extracts the value "致谢(Acknowledgement)" from the query correctly instead of using the default alternative value "VALUE". Although the matching accuracy of values is not included in the evaluation metric, from the experimental results, it can be found that the text type outperforms the numeric type in terms of the accuracy of extracting value information in the query. The failure of DBSQL to extract the value "10000" in Query 1 may be due to the fact that the numeric type was added to the SQL vocabulary, thus causing confusion to model in predicting origin of the generated sequence.

Query 4 needs to return multiple entity query results at the same time, in addition to calculating qualifiers based on entity mapping relationships. SyntaxSQLNet does not recognize the entity "ID" and its multi-table join procedure has a broken component that does not specify the fields to be joined between two tables. BRIDGE maps the entity "ID" to a wrong field. DBSQL successfully establishes correct mapping of four entities as well as a computational intent.

TABLE 5. Partial SQL statement generation results for different models(dev set).

Easy	Query 1	返回总访问计数不超过10000的文档类型的代码。 Return codes for document types whose total access count does not exceed 10000.
	SyntaxSQLNet	**SELECT COUNT(*)** FROM Documents **WHERE** document_type_code = terminal
	BRIDGE	**SELECT** Documents.document_type_code **FROM** Documents **EXCEPT SELECT** Documents.document_type_code **FROM** Documents **WHERE** Documents.access_count >VALUE
	DBSQL(Ours)	**SELECT** Documents.document_type_code **FROM** Documents **GROUP BY** Documents.document_type_code **HAVING** SUM(Documents.access_count) > **VALUE**
Medium	Query 2	作者和导师的登录名和姓氏是什么？ What are the author's and tutor's login and last name?
	SyntaxSQLNet	**SELECT** middle_name, family_name **FROM** Students
	BRIDGE	**SELECT** Course_Authors_and_Tutors.personal_name, Course_Authors_and_Tutors.family_name **FROM** Course_Authors_and_Tutors
	DBSQL(Ours)	**SELECT** Course_Authors_and_Tutors. login_name, Course_Authors_and_Tutors. family_name **FROM** Course_Authors_and_Tutors
Hard	Query 3	查找具有功能区域"致谢"的文档的平均访问次数。 Find the average number of visits to documents with the functional area "Acknowledgements".
	SyntaxSQLNet	**SELECT** AVG(T2.access_count) **FROM** Document_Structures AS T1 **JOIN** Documents **AS** T2 **ON** T1.document_structure_code = T2.document_structure_code **WHERE** T1.document_structure_description = 'terminal'
	BRIDGE	**SELECT** AVG(Documents.access_count) **FROM** Documents **JOIN** Document_Functional_Areas **ON** Documents.document_code = Document_Functional_Areas.document_code **WHERE** Document_Functional_Areas.functional_area_code = "致谢"
	DBSQL(Ours)	**SELECT** AVG(Documents.access_count) **FROM** Documents **JOIN** Document_Functional_Areas **ON** Documents.document_code = Document_Functional_Areas.document_code **JOIN** Functional_Areas **ON** Document_Functional_Areas.functional_area_code = Functional_Areas.functional_area_code **WHERE** Functional_Areas.functional_area_description = "致谢"
Extra	Query 4	账户最少的客户的名字、姓氏和ID是什么？ What is the first name, last name and ID of the customer with the least number of accounts?
	SyntaxSQLNet	**SELECT** T2.customer_last_name, T2.customer_first_name, T1.account_name **FROM** Accounts **AS** T1 **JOIN** Customers AS T2 **GROUP** BY T2.customer_id **ORDER BY COUNT(*) ASC LIMIT** 1
	BRIDGE	**SELECT** Customers.customer_first_name, Customers.customer_last_name, Customers.customer_email **FROM** Customers **JOIN** Accounts **ON** Accounts.customer_id = Customers.customer_id **GROUP BY** Accounts.customer_id **ORDER BY COUNT(*) LIMIT** 1
	DBSQL(Ours)	**SELECT** Customers.customer_first_name, Customers.customer_last_name, Customers.customer_id **FROM** Customers **JOIN** Accounts **ON** Accounts.customer_id = Customers.customer_id **GROUP BY** Accounts.customer_id **ORDER BY COUNT(*) LIMIT** 1

Figure 5 gives a comparison of the overall performance of each model on samples with different difficulty dev set. The number of corresponding samples is given after different

difficulties. It can be seen that the overall performance of all models shows a decreasing trend as the complexity of the tested samples increases. In particular, the performance of SyntaxSQLNet drops to less than 10% on Extra Hard level, while DBSQL has a more obvious performance advantage on this difficulty, and also shows some competitiveness compared with BRIDGE.

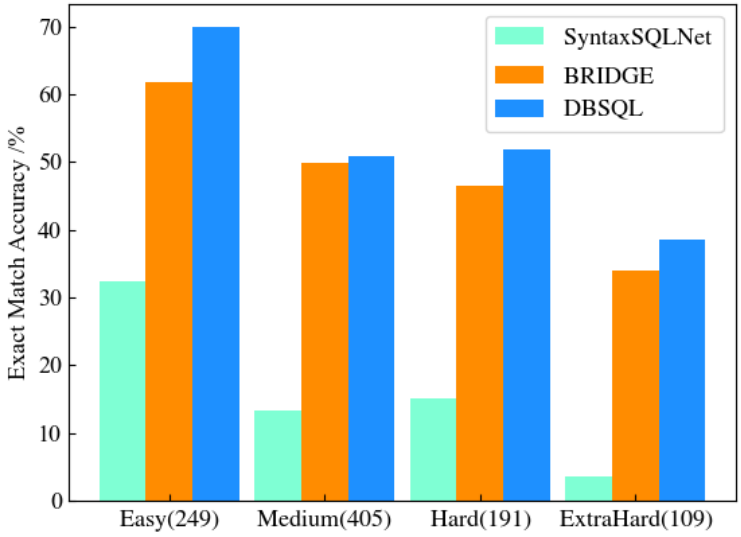

FIGURE 5. Comparison of model performance with different sample difficulty.

In Figure 6, the multi-heads attention distribution of DBSQL between natural language queries and generated SQL statements during the inference process is visualized to facilitate the observation of the model's ability to establish mapping relationships between heterogeneous data across languages. To improve the readability of images, we select three smaller-scale examples to show their attention distribution of some heads and use variable h to denote different heads.

The example in (a) of Figure 6 gives a confusing Chinese query "列出所有导演导演的电影数量。(List the number of movies directed by all directors.)" The word "导演(Direct)" appears twice in a row, but with different lexicon meanings, the former being a noun and the latter a verb, and finally DBSQL still generates the correct result. We can see that when $h = 3, 4$, there is a strong mapping between "数量(Count)" in the query and "count" in the SQL statement. When $h = 1, 7$, there is a clear distinction between "select" operation in the SQL statement and "导演(Director)" from "导演导演的电影(Movie directed by director)" in the query. In particular, when $h = 3, 5$, the field "film.Directed_by" after the "group by" operation in SQL establishes a significant mapping to the noun "导演(Director)" of two consecutive "导演(Direct)", which is the correct logic.

In example (b) from Figure 6, the query is "什么是最不常见的教职员工职级? (What is the least common faculty rank?)" When $h = 1, 6$, "最不常见(least common)" has a strong association with the SQL operation "count(*) asc limit 1". When $h = 4$, "教职员工职级(Faculty rank)" has an obvious mapping with "select Faculty. Rank".

The query in example (c) of Figure 6, "在2014年或之后举办的音乐会最多的体育场名称和容量(Name and capacity of stadiums with the most concerts in 2014 or later)" involves a multi-table join. When $h = 6$, we can find that "体育场(Stadium)" and "音

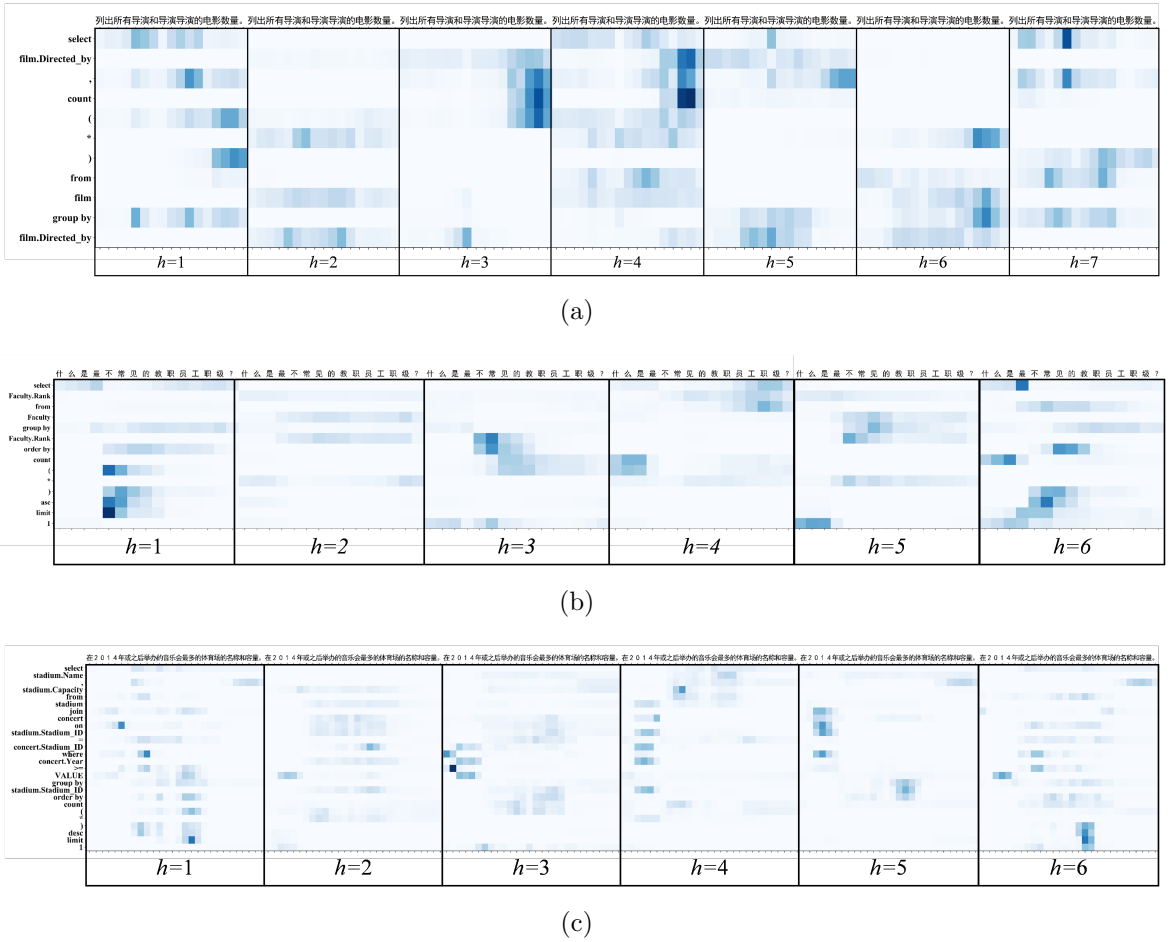

FIGURE 6. Query and SQL statement attention distribution visualization results.

乐会(Concert)" in the query create an obvious mapping to the tables "stadium" and "concert" respectively. For the year keyword "2014", DBSQL does not extract the correct content from the query, but uses the default "VALUE" token in SQL vocabulary, which is consistent with our previous speculation that DBSQL tends to confuse numeric types of value extraction. However, the distribution of attention in $h = 3, 4, 5$ shows that "VALUE" still has a strong correlation with "2014". A visual analysis of attention distribution reveals that DBSQL is able to establish effective mapping relationships between different types of entities in cross-lingual heterogeneous data.

4.5. **Ablation Study.** We further analyze the contribution of different methods to DBSQL by ablation. The impact of two private feature extraction modules on model performance is mainly analyzed, as shown in Table 6, which gives their performance on different difficulty dev set. It can be seen that when the Chinese lexicon enhance-ment module is removed, the overall performance of the model decreases by 1.6%. At the same time, this loss has the most pronounced effect on the Hard difficulty sample, whose performance decreases by 2.1%, indicating the necessity of Chinese lexicon features. When the relationship-awareness module is removed, the overall performance of the model decreases by 2.2%, it can be found that database relationship-awareness achieves a significant gain of 6.4% on the Extra Hard difficulty samples. This further indicates that explicitly modeling relational features in database helps improve the model's ability to parse complex

query intent during Q&A. We also find that the approach of expanding field representation with table names and field types in Section 3.1 achieves a positive impact on the vast majority of samples and brings a 1.7% improvement to the overall model performance.

TABLE 6. Ablation comparison experiment.

Model	Easy	Medium	Hard	Extra Hard	ACC_{ex}
DBSQL	69.9	50.9	51.8	38.5	54.6
- Lexicon enhancement	68.7	48.9	49.7	38.5	53.0
- Relational awareness	67.1	50.4	49.2	32.1	52.4
- Database schema sequence	68.7	48.6	48.2	41.3	52.9

4.6. **Error Analysis.** We also randomly select 50 prediction error results from dev set for detailed analysis to explore the causes of DBSQL prediction result errors. Some examples with different difficulty of prediction errors are given in Table 7, where Q, G, and P denote natural language queries, correct SQL statements and predicted SQL statements, respectively. Finally, the causes of these errors can be roughly classified into 3 categories.

(1) **Query intent understanding bias.** Such errors account for 48.3% of all samples and are mainly reflected in predicted SQL statements deviate from original query intent. The main reason for such errors is that the model fails to correctly parse the syntactic structure of natural language queries. Meanwhile, the lack of explicit information in some queries and underlying query intent make the parsing process difficult. In Example 1 of Table 7, the model confuses tables with similar names "Courses" and "Subjects", resulting in the connection of "Subjects" table being ignored. The query in Example 2 does not explicitly mention the term "average" , but its implied intent contains the operation of averaging the speed, which is not effectively recognized by the model at the time of prediction. These errors suggest that how to combine table structure data to obtain richer information from limited queries is a key aspect to improve the performance of semantic parsing tasks.

(2) **Table and query mapping errors.** Such errors accounted for 40.2% of all samples, which is summarized as predicted SQL statements are incorrectly predicted in the name of table and table field. The main cause of this error is that the model does not correctly capture explicit or implicit mapping relationships in the query. In example 3 of Table 7, the query implies a join query to the table "mill", but the model misinterprets the join object as the table "bridge". In example 4, the model fails to find a correspondence between "辅修(Minor)" and the table "Minor_in". These errors are closely related to the query intent understanding bias, and are the main reason for the wrong final prediction results.

(3) **Other errors.** It mainly includes samples with incorrect comments, unintelligible queries, equivalent SQL statements and other types of errors, which together account for 11.5% of the total samples. In example 5 of Table 7, the query "5张卡片(5 cards)" can be interpreted in two different ways, one is to calculate the number of cards in the field "card_type_code" after GROUP BY operation as shown in the correct result, the other interpretation is to qualify the value of the field "Customers_Cards.card_number" as shown in the prediction result. In Example 6, the difference between the predicted result and the correct answer lies in the object of GROUP BY operation. However, the

field “document_code” and the field “document_name” can achieve same result in general, but in a strict sense the field “document_code” is more rigorous.

Table 7. Partially incorrect prediction results.

Error Type	No. (Difficulty)		Content
Query intent understanding bias(48.3%)	1 (Hard)	Q:	按课程计数的升序列出课程ID、课程名称和每个科目可用的课程数。 List the course ID, course name and the number of courses available for each subject in ascending order of course count.
		G:	**SELECT** T1.subject_id , T2.subject_name , **COUNT(*) FROM** Courses AS T1 **JOIN** Subjects AS T2 **ON** T1.subject_id = T2.subject_id **GROUP BY** T1.subject_id **ORDER BY COUNT(*)** ASC
		P:	**SELECT** Courses.subject_id, Courses.course_name, **COUNT(*) FROM** Courses **GROUP BY** Courses.subject_id **ORDER BY COUNT(*)** ASC
	2 (Easy)	Q:	叫“刘明”的飞行员的速度是多少？ What was the speed of the pilot named ”Liu Ming”?
		G:	**SELECT** avg(velocity) **FROM** flight **WHERE** pilot = “刘明”
		P:	**SELECT** flight.velocity **FROM** flight **WHERE** flight.pilot = “刘明”
Table and query mapping errors (40.2%)	3 (Extra)	Q:	以米为单位桥梁的最大长度是多少？建筑师的名字是什么？ What is the maximum length of a bridge in meters? What is the name of the architect?
		G:	**SELECT** T1.id , T1.name , T1.nationality **FROM** architect **AS** T1 **JOIN** mill AS T2 **ON** T1.id = T2.architect_id **GROUP BY** T1.id **ORDER BY COUNT(*) DESC LIMIT** 1
		P:	**SELECT** architect.id, architect.name, architect.nationality **FROM** architect **JOIN** bridge **ON** architect.id = bridge.architect_id **GROUP BY** architect.id **ORDER BY COUNT(*) DESC LIMIT** 1
	4 (Extra)	Q:	找到辅修人数最多的系的名字。 Find the name of the department with the largest number of minors.
		G:	**SELECT** T1.DName **FROM** DEPARTMENT **AS** T1 **JOIN** MINOR_IN **AS** T2 **ON** T1.DNO = T2.DNO **GROUP BY** T2.DNO **ORDER BY COUNT(*) DESC LIMIT** 1
		P:	**SELECT** Department.DName **FROM** Department **ORDER BY** Department.DName **DESC LIMIT** 1
Other errors (11.5%)	5 (Medium)	Q:	显示至少有5张卡片的类型代码。 Displays the type code with at least 5 cards.
		G:	**SELECT** card_type_code **FROM** Customers_cards **GROUP BY** card_type_code **HAVING** count(*) >= 5
		P:	**SELECT DISTINCT** Customers_Cards.card_type_code **FROM** Customers_Cards **WHERE** Customers_Cards.card_number >= 5
	6 (Extra)	Q:	拥有最多章节数的文档的名字是什么？ What is the name of the document with the most number of chapters?
		G:	**SELECT** T1.document_name **FROM** documents AS T1 **JOIN** document_sections **AS** T2 **ON** T1.document_code = T2.document_code **GROUP BY** T1.document_code **ORDER BY COUNT(*) DESC LIMIT** 1
		P:	**SELECT** Documents.document_name **FROM** Documents **JOIN** Document_Sections **ON** Documents.document_code = Document_Sections.document_code **GROUP BY** Documents.document_name **ORDER BY COUNT(*) DESC LIMIT** 1

5. **Conclusions.** In this paper, we propose a structured knowledge Q&A approach based on lexicon enhancement and database relationship awareness for multilingual cross-domain structured knowledge scenarios, which can transform natural language queries into SQL statements. On the premise of extracting common features between natural language queries and database schemas, private features of two heterogeneous data are considered separately to bridge semantic gap between different languages and enhance the parsing ability of structured knowledge. We also find that 1)multilingual pre-trained models improve the ability to understand and map relationships between entities with heterogeneous

data across languages. 2)The approach of incorporating Chinese lexicon information can effectively solve the problem of inconsistent minimum semantic units in multilingual scenarios and further improve the model performance. 3)Explicit character-ization of knowledge relationships in database can improve model's ability to parse structured data and help handle complex query intent. Experiments on public datasets show that DBSQL model in this paper has better performance in generating SQL statements with exact match accuracy, especially on difficult samples.

REFERENCES

[1] K. Dhamdhere, K. McCurley, R. Nahmias, M. Sundararajan, Q. Yan, Analyza: exploring data with conversation, *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pp. 493–504, 2017.

[2] S. Zhu, X. Cheng, S. Su, Conversational semantic parsing over tables by decoupling and grouping actions, *Knowledge-Based Systems*, vol. 204, 106237, 2020.

[3] X. Xu, C. Liu, D. Song, Sqlnet: generating structured queries from natural language without reinforcement learning, *arXiv preprint*, arXiv:1711.04436, 2017.

[4] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, D. Radev, SyntaxSQLNet: syntax tree networks for complex and cross-domain text-to-sql task, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing(EMNLP)*, pp. 1653–1663, 2018.

[5] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, JG. Lou, T. Liu, D. Zhang, Towards complex text-to-sql in cross-domain data-base with intermediate representation, *Proceedings of the 57th Annual Meeting of the Association for Computa-tional Linguistics*, pp. 4524–4535, 2019.

[6] DH. Choi, MC. Shin, EG. Kim, DR. Shin, RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases, *Computational Linguistics*, vol. 47, no. 2, pp. 309–332, 2021.

[7] N. Sun, X. Yang, Y. Liu, Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation, *arXiv preprint*, arXiv:2006.06434, 2020.

[8] XV. Lin, R. Socher, C. Xiong, Bridging textual and tabular data for cross-domain text-to-sql semantic parsing, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4870–4888, 2020.

[9] V. Zhong, C. Xiong, R. Socher, Seq2sql: generating structured queries from natural language using reinforcement learning, *arXiv preprint*, arXiv:1709.00103, 2017.

[10] B. Wang, R. Shin, X. Liu, O. Polozov, M. Richardson, RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7567–7578, 2020.

[11] X. Luo, K. Luo, X. Chen, K. Zhu, Cross-lingual entity linking for web tables, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. No. 1. 2018.

[12] Q. Min, Y. Shi, Y. Zhang, A pilot study for chinese sql semantic parsing, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Lan-guage Processing*, pp. 3652–3658, 2019.

[13] Y. Wu, M. Schuster, Z. Chen, QV. Le, M. Norouzi, et al. Google's neural machine translation system: Bridging the gap between human and machine translation, *arXiv preprint*, arXiv:1609.08144, 2016.

[14] J. Devlin, MW. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Compu-tational Linguistics: Human Language Technologies*, vol. 1, pp. 4171–4186, 2019.

[15] W. Lei, W. Wang, Z. Ma, T. Gan, W. Lu, MY. Kan, TS. Chua, Re-examining the role of schema linking in text-to-sql, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6943-6954, 2020.

[16] C. Baik, HV. Jagadish, Y. Li, Bridging the semantic gap with sql query logs in natural language interfaces to data-bases, *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 374-385, 2019.

[17] I. Sutskever, O. Vinyals, QV. Le, Sequence to sequence learning with neural networks, *Advances in neural infor-mation processing systems*, pp. 3104-3112, 2014.

[18] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, et al. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, *Proceedings of the 2018 Conference on Empirical Meth-ods in Natural Language Processing*, pp. 3911–3921, 2018.

[19] P. Yin, G. Neubig, W. Yih, S. Riedel, Tabert: pretraining for joint understanding of textual and tabular data, *Pro-ceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8413–8426, 2020.

[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: a robustly optimized bert pretraining approach, *arXiv preprint*, arXiv:1907.11692, 2019.

[21] K. Clark, MT. Luong, QV. Le, CD. Manning, ELECTRA: pre-training text encoders as discriminators rather than generators, *arXiv preprint*, arXiv:2003.10555, 2020.

[22] T. Pires, E. Schlinger, D. Garrette, How multilingual is multilingual bert? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4996–5001, 2019.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, AN. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, *Advances in neural information processing systems*, pp. 5998-6008, 2017.

[24] Y. Li, B. Yu, M. Xue, T. Liu, Enhancing pre-trained chinese character representation with word-aligned attention, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3442–3448, 2020.

[25] L. Huang, D. Ma, S. Li, X. Zhang, H. Wang, Text level graph neural network for text classification, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Con-ference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3442–3448, 2019.

[26] P. Shaw, J. Uszkoreit, A. Vaswani, Self-attention with relative position representations, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 2 (Short Papers), pp. 464–468, 2018.

[27] A. See, PJ. Liu, CD. Manning, Get to the point: summarization with pointer-generator networks, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), pp. 1073–1083, 2017.

[28] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*, pp. 249-256, 2010.

[29] B. Wang, M. Lapata, I. Titov, Meta-learning for domain generalization in semantic parsing, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan-guage Technologies*, pp. 366–379, 2021.