electronics

MDPI

Article

Integrating Relational Structure to Heterogeneous Graph for Chinese NL2SQL Parsers

Changzhe Ma [1], Wensheng Zhang [1,2,*], Mengxing Huang [1,*], Siling Feng [1] and Yuanyuan Wu [1]

[1] School of Information and Communication Engineering, Hainan University, Haikou 570228, China; 20081000210024@hainanu.edu.cn (C.M.); fengsiling@hainanu.edu.cn (S.F.); 995042@hainanu.edu.cn (Y.W.)
[2] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
* Correspondence: zhangwenshengia@hotmail.com (W.Z.); huangmx09@hainanu.edu.cn (M.H.)

Abstract: The existing models for NL2SQL tasks are mainly oriented toward English text and cannot solve the problems of column name reuse in Chinese text data, description in natural language query, and inconsistent representation of data stored in the database. To address this problem, this paper proposes a Chinese cross-domain NL2SQL model based on a heterogeneous graph and relative position attention mechanism. This model introduces relational structure information defined by the expert to construct initial heterogeneous graphs for database schemas and natural language questions. The heterogeneous graph is pruned based on natural language questions, and the multi-head relative position attention mechanism is used to encode the database schema and natural language questions. The target SQL statement is generated using a tree-structured decoder with predefined SQL syntax. Experimental results on the CSpider dataset demonstrate that our model better aligns database schema with natural language questions and understands the semantic information in natural language queries, effectively improving the matching accuracy of Chinese multi-table SQL statement generation.

Keywords: NL2SQL; graph neural network; schema linking; semantic parsing; NLP

Citation: Ma, C.; Zhang, W.; Huang, M.; Feng, S.; Wu, Y. Integrating Relational Structure to Heterogeneous Graph for Chinese NL2SQL Parsers. *Electronics* **2023**, *12*, 2093. https://doi.org/10.3390/electronics12092093

Academic Editor: Rui Pedro Lopes

Received: 11 March 2023
Revised: 17 April 2023
Accepted: 30 April 2023
Published: 4 May 2023

1. Introduction

In the age of the digital revolution, data have become an indispensable commodity that drives almost all human activities, from business operations to scientific research [1]. Data search and query play an important guiding role in enterprise production and people's daily life. Existing data are usually stored in a relational database system [2]. Relational databases are widely used and store a lot of structured data. However, researchers need to master database knowledge and SQL (Structured Query Language)-based [3] query language to obtain the required data, which requires highly professional skills and is difficult for ordinary users to achieve. In recent years, the question-answering system based on natural language to SQL (NL2SQL) has become a research hotspot. Compared with other question-answering systems based on retrieval and reading comprehension, NL2SQL first generates structured queries and then provides the reasoning path of the answer, so it has better interpretability. The NL2SQL task aims to parse natural language with a given database into an SQL query and use this query statement to obtain the answer to the question. However, in NL2SQL, similar queries with different database schemas leads to different SQL queries. Thus, both the database schema and word representation will affect the gold SQL query.

Because the same problem will have different SQL query statements on different database models, it is very important to design a model that can learn both the question representation and the database schema representation to improve the logical accuracy and execution accuracy of the NL2SQL model. NL2SQL can be divided into two research directions: (1) the method based on rule templates, that is, classifying natural languages according to common SQL syntax and matching different categories with corresponding

SQL templates. This method requires a manual summary of experience and has a high time cost. In addition, with the switching of application scenarios, the existing templates are often difficult to meet the requirements and have poor mobility. (2) Based on the method of deep learning, the neural network is used for end-to-end realization. This method can optimize itself by continuously adding sample information. It has the advantages of high accuracy and strong adaptability and is attracting more and more attention from the academic community. The deep learning model of NL2SQL needs to combine information from the database itself and the user's natural language to extract the associated database table names, column names, and user intent patterns. However, the complex internal structure of the database itself brings great challenges to the conventional deep learning models, mainly in two ways. (1) The input of the traditional deep neural network is structured data, while there are table names, column names, and specific data corresponding to column names in the database, and integrating multiple information sources undoubtedly increases the complexity of the process. (2) There are significant dependencies between the elements of the database, especially in multi-table SQL query scenarios, and it is a major challenge to effectively extract and substitute into the model information, such as table–table, table–primary key, and primary key–foreign key associations, to generate the SQL statement; a complex SQL statement for multi-table connection is shown in Figure 1.

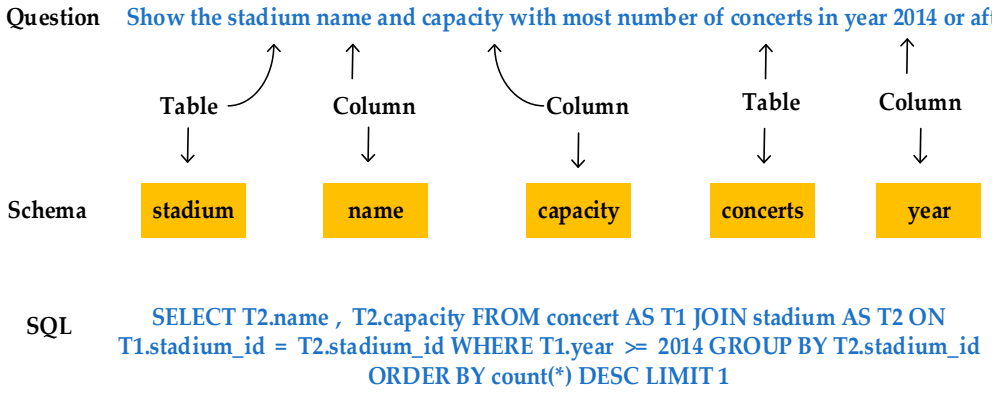

Figure 1. Complex SQL statement for multi-table connection.

In recent years, recent advances in deep learning and the availability of large-scale training data have significantly improved NL2SQL parsing by neural generation models. Current research has been conducted by constructing a heterogeneous graph based on an attention mechanism to jointly encode database schemas and natural language questions, followed by the use of a tree-structured decoder with SQL syntax guidelines to parse the SQL statements [4,5]. This approach treats the database schema and the natural language questions as a heterogeneous graph, building relationships between words in natural language questions, column names, and table names [6,7]. Subsequently, it learns the embedding of these relationships as schema information and encodes it into the model. The NL2SQL with graph neural networks (GNNs) [8] can accurately represent the association information of database elements and can solve the above problems encountered by traditional deep learning models.

However, this approach constructs a heterogeneous graph based on the entire database schema, which will lead to two main problems. (1) When the database schema is large, the heterogeneous graph is too large for the model to capture and learn the correct features. (2) The approach will ignore the importance of each element in the database schema. For example, when a person's height is asked, in fact, only height is useful among all attributes of the human entity in the database schema, and attributes such as weight and age are useless, so the two attributes of weight and age should not have the same weight as height in the composition. In general, the model needs to judge the importance of all elements in the database schema based on the natural language question, rather than simply adding all elements in the database schema to the heterogeneous graph.

Compared with the widely carried out research on English NL2SQL tasks, the research on Chinese NL2SQL tasks is more complex, the existing research works are few, and most of these studies focus on simple SQL statements. The difficulty of the Chinese NL2SQL task is that there is no word segmentation in Chinese questions, so the probability of ambiguity is high. In addition, there are more synonyms and more diverse expressions in Chinese, which further increases the difficulty of machine understanding. In practice, the names of tables and columns stored in databases are usually stored in English, which makes it more difficult to infer the tables and columns they refer to from Chinese natural language expressions.

To address the above problems, we propose a Chinese cross-domain NL2SQL model based on a heterogeneous graph and relative position attention mechanism. Our model uses a sequence-to-sequence framework and consists of three main parts: heterogeneous graph construction, encoder, and decoder. The heterogeneous graph construction module constructs a preliminary heterogeneous graph using predefined relational structure information defined by the expert. The encoder crops some edges of the heterogeneous graph to eliminate unimportant elements in the database schema and uses relational graph attention transformers to jointly learn vector representations of the natural language question and database schemas, where the relation is the feature of the edge in the heterogeneous graph and the word features are initialized using the XLM (cross-lingual language model) [9] pre-trained language model for initialization. The decoder decodes the abstract syntax tree (AST) [10] in steps with the help of pre-defined SQL syntax rules using a tree-structured decoder and completes the information of the SQL statement by slot filling. We evaluate our proposed model on the Chinese large-scale complex and cross-domain semantic parsing and NL2SQL dataset CSpider [11]. The results of the experimental evaluation demonstrate the effectiveness of our model. Specifically, our main contributions are summarized as follows:

- We propose a novel method for Chinese cross-domain NL2SQL based on a heterogeneous graph and relative position attention mechanism, which has the advantage of generality across databases compared with previous works.
- We design a graph-pruning task to prune the heterogeneous graph based on natural language questions for better utilization.
- The empirical results show that our method achieves better performance on the challenging CSpider benchmarks.

The remainder of this paper is structured as follows. Related methods for NL2SQL are reviewed in Section 2. Section 3 introduces the NL2SQL problem definition and provides specific details about the NL2SQL model. Section 4 introduces the evaluation criteria and experiments in detail. Finally, Section 5 presents the conclusion.

2. Related Work

2.1. Natural Language to SQL

At present, the related work of NL2SQL can be roughly divided into three types according to the modules in the model: question and database schema joint encoding, structured query language decoding, and pre-trained word representation enhancement.

2.1.1. Question and Database Schema Joint Encoding

SQLNet solves the problem through the "column attention" mechanism to gather information from columns for each question word and then uses the pointer network on the predefined SQL template to complete the SQL information [12]. IRNet introduces the intermediate presentation layer to solve the problem of the mismatch between the intention expressed in natural language and the implementation details in SQL. First, the database schema is linked to the question, and the table name, column name, and value mentioned in the question are identified. Then the question is divided into different question spans by the neural network according to the database schema and encoded by BiLSTM [13]. TypeSQL takes prior knowledge of column types and schema linking as additional input

features [14]. RAT-SQL constructs a heterogeneous graph for database schema and questions to establish the relationship between database schema and question words and then carries out vectorization representation of edges between nodes in the heterogeneous graph. Finally, the relative position self-attention mechanism is used to encode heterogeneous graph information into the model. Bogin et al. [15] proposed Global-GNN to solve the problem of global reasoning about the structure of the output query; it uses a global gating mechanism, which can further improve the performance of SQL generation compared with the previous method using local information. Chen et al. [16] proposed ShadowGNN; this model uses graph mapping networks to remove semantic information from natural language and database schemas, reducing the influence of domain-specific information. Cao et al. [7] proposed a line graph-enhanced NL2SQL (LGESQL) model to mine the relational features. LGESQL uses the line graph and node graph to capture the topology of nodes and edges, respectively, which improves the coding ability. However, the above methods do not consider the filtering of database schema information; useless schema information will magnify the noise in the process of model coding.

2.1.2. Structured Query Language Decoding

Recent works usually use tree-structured decoders to decode SQL statements, which generate an abstract syntax tree to represent the SQL language after defining the base syntax of the target language as prior knowledge [17]. RAT-SQL uses LSTM to generate the abstract syntax tree of SQL in depth-first traversal order and then uses the pointer network to fill in the slots of the missing table names and column names in the SQL syntax tree. SmBoP uses a bottom-up decoding strategy to build top-K subtrees of height t or less at the t-th step, which improves decoding efficiency since each subtree is decoded in parallel each time it is decoded [18]. PICARD simply combines the model prediction score with the greedy algorithm and the beam search algorithm. At each generation step, the prediction is first limited to the maximum k probability tokens, and then those tokens that do not pass extensive checks are deleted [19].

2.1.3. Pre-Trained Word Representation Enhancement

Some work focused on providing pre-trained word enhanced representations for NL2SQL tasks. GraPPa first learns the syntax and SQL rules in the Spider [20] dataset and then uses these rules to generate high-quality question–SQL pairs on other databases that are similar to Spider's construction rules. Then, it sets a pre-training task to let the model directly generate SQL based on the question and database schema to capture the structural information between the question and the database schema [21]. GAP improves word representation through three different pre-training tasks. The first pre-training task determines whether a column in a database table appears in a question. The second pre-training task randomly replaced the column names in the question with the values of the cells in that column and the model needed to recover the replaced column names. The third pre-training task requires the model to generate SQL statements directly from the question and database schema [22]. TAPEX simulates the behavior of the SQL execution engine on tables by pre-training the language model (LM) on a diverse, large-scale, and high-quality composite corpus, approximating the structural inference process of SQL queries and automatically synthesizing executable SQL queries and their execution outputs [23].

2.2. Heterogeneous Graph Neural Networks

Heterogeneous graph modeling has been widely explored in natural language processing (NLP) for many years. Zeng et al. [24] proposed a heterogeneous graph convolution based on in-domain self-supervision for multimodal sentiment analysis; it makes full use of domain knowledge by constructing a heterogeneous graph and integrating text modality features. Mo et al. [25] proposed a relation-aware heterogeneous graph convolutional network to learn different relationships of a specific node type. Fei et al. [26] proposed to further enhance dual learning with structure matching that explicitly builds structural

connections in between. Fang et al. [27] proposed a relation-aware graph convolutional network to fully utilize relational information of multiple types. Fei et al. [28] proposed a label-aware graph convolutional network to encode word representations, dependency arcs, and labels. Wu et al. [29] propose a dependency-guided high-order interaction mechanism to achieve explicit interactions between opinions and roles. Yu et al. [30] proposed a novel dynamically pruned graph convolutional network to remove irrelevant content from the dependency tree. Yu et al. [31] proposed a novel multi-stage graph embedding technique based on graph neural networks to identify deep neural network topologies and use reinforcement learning to find an appropriate compression policy. Fei et al. [32] proposed a structure-aware generative language model to fully utilize syntactic structure information in information extraction tasks.

In this paper, we combined XLM and RGAT (relational graph attention transformers) [33] to enable our model to exploit schema linking and the syntactic dependency information of questions. We use a graph neural network to read the association information of many elements at the database level and encode the database information using the neighbor node aggregation method, which is helpful for identifying the column name and table name in the question. We use the graph-pruning mechanism to eliminate the relatively irrelevant information and retain the relatively relevant information.

3. Methodology

3.1. Problem Definition

The input of the NL2SQL task is a natural language question and database schema, and the output is a structured query statement.

Specifically, give a question Q with length $|Q|$ and a corresponding database schema $S = \langle C, T \rangle$ consisting of columns $C = \left\{ c_1^{t_1}, c_2^{t_1}, \cdots, c_1^{t_{|T|}}, c_2^{t_{|T|}} \right\}$ and tables $T = \{t_i\}_{i=1}^{|T|}$. Each column name $c_i = \left\{ c_{i,1}, c_{i,2}, \cdots, c_{i,|c_i|} \right\}$ consists of $|c_i|$ characters, and each table name $t_i = \left\{ t_{i,1}, t_{i,2}, \cdots, t_{i,|t_i|} \right\}$ consists of $|t_i|$ characters. The generated structured query is represented in the form of an abstract syntax tree.

Some columns in the database schema are called primary keys, whose values uniquely identify every row in a table. There are also some columns called foreign keys, which are used to index the primary key columns in different tables [34]. In addition, each column in the table has a type field to constrain its cell values (e.g., TEXT and NUMBER).

The entire input node-centric heterogeneous graph $G = (V, R)$ consists of all three types of nodes mentioned above, that is, $V = Q \cup T \cup C$, with the number of nodes $|V| = |Q| + |T| + |C|$, where $|T|$ and $|C|$ are the number of tables and columns, respectively. The edge's R is composed of expertly defined information about the structure of the relationship.

3.2. Architecture of the Proposed Model

We provide the overview of our model in Figure 2. As shown in the figure, our model follows the sequence-to-sequence framework. The proposed model consists of four modules. First is the context encoder, which uses the XLM pre-trained language model to transform the input question and database schema into semantic vector representations. Second is the question–schema interaction graph, which uses the relational structure information defined by the expert, and the input question and database schema are dynamically constructed as a heterogeneous graph, which better establishes the connection between the question and the database schema, using a graph-pruning mechanism to drop the redundant nodes. Third is the relation-aware graph encoder. The encoder uses the relative position attention mechanism to jointly encode the question, database schema information, and heterogeneous graph into a high-dimensional hidden layer vector representation. Fourth is the decoder. The decoder uses a tree decoder to decode high-dimensional hidden layer vectors into high-quality, executable SQL statements.

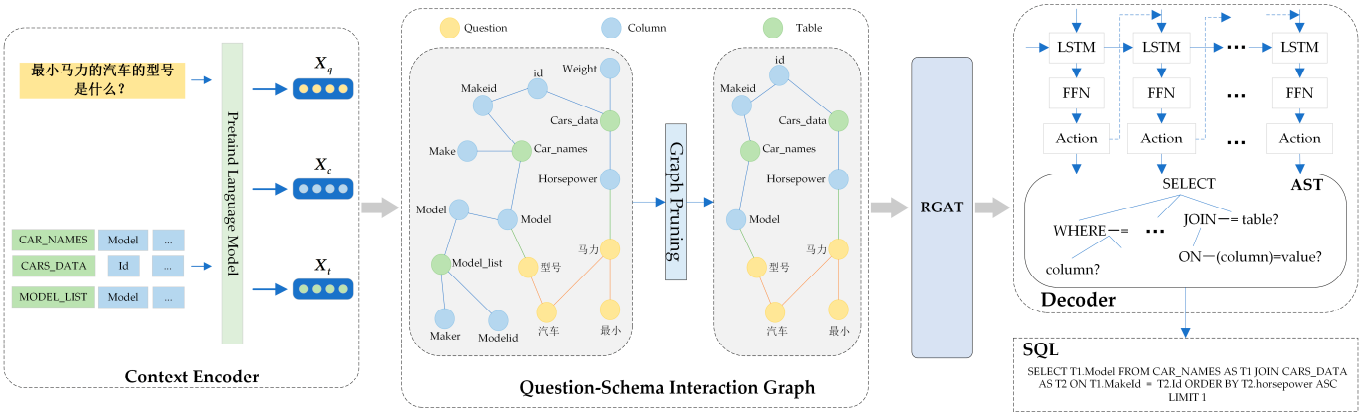

Figure 2. The overall model architecture X_q, X_c, and X_t denote the embedding vectors of question, column, and table, respectively. RGAT represents Relational Graph Attention Transformers. In the decoder, FFN is a Feed-Forward Network that consists of two linear layers with a Tanh activation function in between. AST means Abstract Syntax Tree. (The input was Chinese data).

3.2.1. Context Encoder

This module uses the XLM pre-training language model as the backbone network to obtain initial semantic vector representations for nodes and edges. All natural language question words and database schema items are flattened into a sequence and used as input to XLM. To match XLM's input, we appended the special token [/s]. The configuration of the entire input sequence can be represented as follows:

$$I = [/s], Q, [/s], [/s], S, [/s] \tag{1}$$

where I represents the input sequence. Q denotes the natural language question. S represents the database schema consisting of tables and columns. Hence, the input I is fed into the XLM to obtain the initial node embeddings matrix $X \in \mathbb{R}^{|\mathrm{V}^{|Q|+|T|+|C|}|\times d}$, where d is the graph hidden size, as shown in Equation (2):

$$X = XLM(I) = \left(q_1^{init}, \cdots, q_{|Q|}^{init}; t_1^{init}, c_{type_1}^{t_1} c_1^{t_1}, c_{type_2}^{t_1} c_2^{t_1}, \cdots, t_{|T|}^{init}, c_{type_1}^{t_{|T|}} c_1^{t_{|T|}}, c_{type_2}^{t_{|T|}} c_2^{t_{|T|}}, \cdots\right) \tag{2}$$

where $|Q|$ denotes the number of words in the natural language question Q. q_{i}^{init} represents the word vector representation of the i-th token of the question. $|T|$ denotes the number of tables T. t^{init} and c^{init} represent the word vector representation of table name and column name in the heterogeneous graph, respectively. For nodes that represent column names, additional features need to be added before the column name to indicate whether the type is a text or a number. $c_{type_j}^{t_i}$ denotes the type information of the j-th column in the i-th table.

3.2.2. Question–Schema Interaction Graph

In this module, we adopt a heterogeneous graph structure to model the natural language question, the database schema, and predefined relations between question words and tables or columns in the database schema.

The node set of a heterogeneous graph $G = (\mathrm{V}, \mathrm{R})$ consists of column names, table names in the database schema, and words in the question, which are labeled in the form of characters, that is, $V = Q \cup T \cup C$. Joining the tokens as a node in the question can align the question of the words with the information in the database schema, enrich the information of heterogeneous graph representation ability, and make the heterogeneous graph effectively establish the connection between the questions and the database schema. The edge $\mathrm{R} = \{R\}_{i=1,j=1}^{|V|,|V|}$ of a heterogeneous graph is defined by experts according to the database schema. One part is obtained by the unique structure of the database (such as foreign key, primary key, etc.), and the other part is obtained by analyzing the error cases

on the validation set and iteratively summarizing. The construction rules of heterogeneous graph edges defined by experts are shown in Table 1.

Table 1. The predefined relations for heterogeneous graph edge construction.

Node A	Node B	Edge Label
Column	Column	Same-Table Foreign-F Foreign -R
Column	Table	Primary-Key Has
Table	Table	Foreign-Key-Tab-F Foreign-Key-Tab-R Foreign-Key-Tab-B
Question	Table	None-Linking Partial-Linking Exact-Linking
Question	Column	None-Linking Partial-Linking Exact-Linking Value-Linking
Question	Question	Syntax-F Syntax-R Syntax-None

There are generally three types of relations: schema structure, schema linking, and question dependency structure [35].

1. Schema Structure

Schema structure relations are the relationships between database schema items. This includes Column–Column, Column–Table, and Table–Table relationships. As shown in Table 1, Same-Table represents that A and B belong to the same table. Foreign-F represents that A is a foreign key for B. Foreign-R represents that B is a foreign key for A. Primary-Key represents A as the primary key of B. HAS represents that the column A belongs to the table B. Foreign-Key-Tab-F represents that table A has a foreign key column in B. Foreign-Key-Tab-R represents that table B has a foreign key column in A. Foreign-Key-Tab-B represents that A and B have foreign keys in both directions.

2. Schema Linking

Schema linking relations refer to the relationship between database schema items and natural language questions. We follow the settings in RAT-SQL [6], which uses n-gram matches to indicate the natural language question mentions of the database schema items. If the column name or table name matches the word in a natural language question, set the schema linking between them. This relation includes Question–Table and Question–Column relationships. As shown in Table 1, None-Linking represents no linking between A and B. Partial-Linking represents that A is part of B, but the entire question does not contain B. Exact-Linking represents that A is part of B, and B is a span of the entire question. Value-Linking represents that A is part of the candidate cell values of column B.

3. Question Dependency Structure

Question dependency structure relations are the edges of a dependency tree of the question. In this work, different dependencies are not distinguished in order to control the total number of relationships and avoid unnecessary overfitting. This relation includes Question–Question relationships. As shown in Table 1, Syntax-F represents that A has forward syntactic dependencies on B. Syntax-R represents that A has reverse syntactic dependencies on B. Syntax-None represents that A and B have no syntactic dependency.

3.2.3. Relation-Aware Graph Encoder

In order to enable the model to learn the edge features, the encoder initializes each edge of the preliminary heterogeneous graph constructed by the heterogeneous graph-building module as a vector representation. Then, the encoder encodes the input using the RGAT, so that the model can jointly learn the relationship between the question and the database schema [36]. Unlike traditional methods, our encoder uses the edge vectors of the heterogeneous map as the relative position encoding to incorporate the information of the heterogeneous map into the model.

The RGAT enhances GATs (Graph Attention Networks) [37] by embedding edges between heterogeneous graph nodes into key and value entries. By considering edge information, the model gives more weight to the predefined relationships. During the training process, these weight values are changed by the backpropagation process. Finally, each relationship label will be given different weight values, reflecting the importance of different relationship labels.

Following previous works [6,7,36], we set key relations r_{ij}^{K} and value relations r_{ij}^{V} by splicing the features of all edges in the heterogeneous graph, as shown in Equation (3):

$$r_{ij}^{K} = r_{ij}^{V} = \mathrm{Concat}\left(m_{ij}^{(1)}\rho_{ij}^{(1)}, \cdots, m_{ij}^{(R)}\rho_{ij}^{(R)}\right) \tag{3}$$

where $\mathrm{Concat}(\cdot)$ means concatenate operation. ρ_{ij} is a trainable edge vector representation of edges. m_{ij} is a graph-pruning variable. Each edge in the heterogeneous graph is binary classified according to the input natural language question to record whether this edge needs to be preserved or not, and if it does not need to be preserved, then the vector representation of the edge is set to a vector of the same dimension consisting of all zeros using m_{ij}. R is the total number of relationship edge labels.

Equations (4)–(6) comprise an independent attention calculation formula.

$$e_{ij}^{(h)} = \frac{x_i W_q^{(h)} \left(x_j W_k^{(h)} + r_{ij}^{\mathrm{K}}\right)^T}{\sqrt{d_z / H|}} \tag{4}$$

$$\alpha_{ij}^{(h)} = \frac{\exp\left(e_{ij}^{(h)}\right)}{\sum_{j=1}^{n} \exp\left(e_{ij}^{(h)}\right)} \tag{5}$$

$$\mathrm{z}_{\mathrm{i}}^{(\mathrm{h})} = \sum_{j=1}^{n} \alpha_{\mathrm{ij}}^{(\mathrm{h})} \left(\mathrm{x_j W_v^{(h)}} + r_{ij}^{\mathrm{V}}\right) \tag{6}$$

where $\mathrm{r_{ij}}$ encode the predefined relationship between the two elements in the input. Matrices W_q, W_k, W_v are trainable parameters in self-attention. The parameter definition and principle are the same as the paper on the self-attention mechanism with relative position encoding.

Equation (7) is the calculation formula for the multi-head attention mechanism [38]. The value is obtained by concatenating multiple independent attention values.

$$z_i = \mathrm{Concat}\left(z_i^{(1)}, z_i^{(2)}, \cdots, z_i^{(H)}\right) \tag{7}$$

where $\mathrm{Concat}(\cdot)$ means concatenate operation. H is the number of heads, each head focusing on a different semantic feature. As shown in Equation (8), after the calculation of multi-head attention is completed, the multi-head attention results are added to the input and layer normalization is performed. As shown in Equation (9), $\widetilde{y}_i$ is processed by the ReLU activation function after passing through a fully connected layer, and the output is obtained by passing through a fully connected layer. Finally, y_i is obtained after adding the output to the input $\widetilde{y}_i$ and performing layer normalization.

$$\widetilde{y}_i = \mathrm{LN}(x_i + z_i) \tag{8}$$

$$y_i = \mathrm{LN}(\widetilde{y}_i + \mathrm{FC}(\mathrm{ReLU}(\mathrm{FC}(\widetilde{y}_i)))) \tag{9}$$

where $\mathrm{LN}(\cdot)$ and $\mathrm{FC}(\cdot)$ denote the layer normalization and the fully connected layer, respectively.

3.2.4. Graph Pruning

In practice, we notice that some nodes become irrelevant and uninformative. These unrelated nodes are distracting and can even disturb the subsequent generation. To drop the redundant nodes in the interaction graph, we propose a Graph Pruning (GP) mechanism similar to the Dynamic Graph Pruning (DGP) [39] mechanism. The GP employs the gate mechanism to parse the connection between the schema node $s_i \in S = T \cup C$ based on its relevance with the question node in the heterogeneous graph to achieve graph pruning.

Specifically, we compute the context vector $\widetilde{x}_{s_i}$ from the question node embeddings X_q for each schema node s_i via multi-head attention. For each schema node s_i in the heterogeneous graph, we formulate its gates as below:

$$g_{ji} = \mathrm{sigmoid}\left(W_g^T \tanh\left(W_e x_{s_i} + W_d x_{q_j}\right)\right) \tag{10}$$

where W_g, W_e, W_d are learnable weight matrices. x_{s_i} is the representation of the schema node s_i, and x_{q_j} is the representation of the question node. Correspondingly, we apply the gate value to decide whether the node should be dropped or not by changing Equation (5). as follows:

$$\alpha_{ij}^{(h)} = \frac{g_{ji} \odot \exp\left(e_{ij}^{(h)}\right)}{\sum_{m=1}^{n} g_{mj} \odot \exp\left(e_{mj}^{(h)}\right)} \tag{11}$$

where $\odot$ means multiplication. $e_{ij}^{(h)}$ is the attention weight with edge information between nodes s_i and node q_j. Intuitively, if the value of gate g_{ji} is close to 0, then the connection between node s_i and node q_j will be significantly weakened, and therefore node s_i should be removed from the interaction graph. $\alpha_{ij}^{(h)}$ is the attention score. By concatenating the results of the multi-head attention mechanism, we obtain the context vector $\widetilde{x}_{s_i}$, as follows:

$$\widetilde{x}_{s_i} = \|_{h=1}^{H} \sum_j \alpha_{ji}^{(h)} x_{q_j} W_v^{(h)} \tag{12}$$

where $\|$ represents vector concatenation horizontally. $W_v^{(h)}$ is a learnable weight matrix.

We use a binary cross-entropy loss function as the training object for the GP. The ground truth label $y_{s_i}^g$ of a schema item is 1 if s_i is mentioned in the target SQL query. $P^{gp}\left(y_{s_i} | x_{s_i}, X_q\right)$ determines the probability of the node s_i being removed. The training object can be formulated as follows:

$$P^{gp}\left(y_{s_i} | x_{s_i}, X_q\right) = \mathrm{Sigmoid}(\widetilde{x}_{s_i}) \tag{13}$$

$$\mathrm{L_{gp}} = -\sum_{s_i} [y_{s_i}^g \log\left(P^{gp}\left(y_{s_i} | x_{s_i}, X_q\right)\right) + \left(1 - y_{s_i}^g\right) \log\left(1 - P^{gp}\left(y_{s_i} | x_{s_i}, X_q\right)\right)] \tag{14}$$

3.2.5. Decoder

The decoder of our model follows the grammar-based syntactic neural decoder [17]. The SQL is first generated as an abstract syntax tree (AST) in a depth-first traversal order. Then a sequence of actions is output through the LSTM network. Actions can be divided into the following two categories: 1) The generated node is a non-leaf node, then the node is extended to a syntax rule, called *APPLYRULE*; 2) The generated node is a leaf node and selects a column name or table name from the database schema, called *SELECTCOLUMN* and *SELECTTABLE*.

The LSTM in the tree-structured decoder updates the state as shown in Equation (15):

$$(m_t, h_t) = f\left(\left[a_{t-1} \middle|\middle| h_{p_t} \middle|\middle| a_{p_t} \middle|\middle| n_{f_t}\right], m_{t-1}, h_{t-1}\right) \tag{15}$$

where m_t is the LSTM cell state. h_t is the output of LSTM at t step. a_{t-1} is the embedding representation of the previous action. p_t is the syntax tree parent node of the current node. n_{f_t} is the embedding representation of the current node type.

APPLYRULE is calculated as shown in Equation (16):

$$\Pr\left(a_t = APPLYRULE[R] | a_{<t}, y\right) = \text{Softmax}\left(g(h_t)\right) \tag{16}$$

where $g(\cdot)$ is a 2-layer MLP with Tanh as the activation function.

SELECTCOLUMN is calculated as in Equation (17), and *SELECTTABLE* is calculated in a similar way as *SELECTCOLUMN*.

$$\Pr\left(a_t = SELECTCOLUMN[i] | a_{<t}, y\right) = \sum_{j=1}^{|y|} \lambda_j L_{j,i}^{col} \tag{17}$$

where λ_j is the weight vector. $L_{j,i}^{col}$ is the embedding representation of the edge.

4. Experiments

4.1. Dataset

We validated the performance of our model using the CSpider dataset. CSpider is a Chinese large-scale complex and cross-domain semantic parsing and NL2SQL dataset translated from Spider and keeps the original English database. The goal of the CSpider challenge is to develop natural language interfaces to cross-domain databases for Chinese, which is currently a low-resource language in this task area. It consists of 9691 questions and 5263 unique complex SQL queries on 166 databases with multiple tables covering 138 different domains; the details of CSpider are shown in Table 2. These details cover multiple domains from other datasets such as Restaurants [40,41], GeoQuery [42], Scholar [43], Academic [44], Yelp, and IMDB [45] datasets. CSpider uses a relational database structure; names and column names of DB tables are typically represented in English, while questions are represented in Chinese. This adds to the challenges in question-to-DB mapping. CSpider classifies each question as easy, medium, hard, or extra hard, based on the number of keywords in the SQL query and the complexity of the structure.

Table 2. Statistics for dataset CSpider.

		# Q	# SQL	# DB	# Table/DB
CSpider	all	9691	5263	166	5.28
	train	6831	3493	99	5.38
	dev	954	589	25	4.16
	test	1906	1193	42	5.69

4.2. Evaluation Metrics

We report our results using Exact Match (EM) accuracy and Component Matching score. We split each component in the prediction and the ground truth into clauses, such as SELECT, WHERE, GROUP BY, ORDER BY. EM mainly evaluates the syntactic structure of SQL statements and conducts set comparison in each SQL clause instead of simply conducting string comparison between the predicted SQL and the gold SQL queries. Component Matching score calculates the F1 value of the predicted and true values in each section.

4.3. Parameter Setting

We trained our models on one server with a single NVIDIA A100-PCIE-40GB GPU. The models used in this study were all built using Pytorch [46]. Our models tokenize and lemmatize the input of natural language questions, column names, and table names with the Stanford Natural Language Processing toolkit [47] during pre-processing. For the encoder, we encode the input natural language question Q and database schema S with XLM. The default setting for the XLM pre-trained language model uses a 16-head attention mechanism Transformer with 1280 hidden layer dimensions and a total of 570M parameters. The GNN hidden size is set to 512. The number of GNN layers is 8. The maximum sequence length used in this study is 512, and the batch size is 20. The position-wise feed-forward network has an inner layer dimension of 1024. For the decoder, we use rule embeddings of size 128, action embeddings of size 128, node type embeddings of size 64, and a hidden size of 512 inside the LSTM with a recurrent dropout [48] of 0.2. The learning rate is 1×10^{-4}. The number of heads in multi-head attention is 8 and the dropout rate of features is set to 0.2 in both the encoder and decoder. During the evaluation, we adopt beam search decoding with beam size of 5.

4.4. Model Comparisons

We evaluate our model via CSpider and compare it with the following models:

SyntaxSQLNet [49]: It introduces structural information into the decoding process, and the decoded object is a tree structure composed of SQL statements.

RYANSQL [50]: It is a sketch-based slot-filling recursive model, combining blocks of statements from each prediction to form the final SQL according to a nested structure.

DG-SQL [51]: It uses a meta-learning framework which targets zero-shot domain generalization for semantic parsing.

RAT-SQL [6]: It presents a unified framework, based on the relation-aware self-attention mechanism, and uses n-gram patterns to construct the question–schema graph.

LGESQL [7]: This model use a Line Graph-Enhanced NL2SQL model to mine the underlying relational features without constructing metapaths.

Table 3 shows the comparison exact match accuracy results between our model and other state-of-the-art models on CSpider. All results of the baselines are obtained from the official leaderboard. As shown in Table 3, the model we propose is competitive with the baselines in the identical sub-table. Our model obtains 66.2% accuracy on CSpider and has a significant improvement (1.7%) on the prior state-of-the-art model, LGESQL+ELECTRA [7,52]. By using the heterogeneous graph with relational labels and a graph-pruning mechanism, our proposed model is able to capture richer features and generate SQL statements with complex structure. The outstanding performance illustrates the effectiveness of our proposed model in natural language-to-SQL tasks.

Table 3. Comparison to different models.

Model	EM (%)
SyntaxSQLNet	16.4
RYANSQL	41.3
RAT-SQL	41.4
DG-SQL	50.4
LGESQL	58.6
RAT-SQL + GraPPa	59.7
LGESQL + Infoxlm	61.0
LGESQL + ELECTRA	64.5
Ours	66.2

To better validate the validity of the proposed model, we compare the fine-grained performance of the model with the baseline models LGESQL [7] and RYANSQL [50] according to the different hardness levels defined by Yu et al. [20]. As can be seen in Figure 3, the pro-

posed model consistently outperforms the comparison models, achieving an exact match accuracy of 85.6% in the Easy level, 69.5% in the Medium level, 55.2% in the Hard level, and 40.6% in the Extra Hard level. The overall exact match accuracy is 66.2%. RYANSQL is a sketch-based slot-filling recursive model. Our proposed model is a significant improvement over RYANSQL in all subdivisions. This is due to the fact that the proposed heterogeneous graph with relational labels can provide richer semantic information. This validates the superiority of the proposed heterogeneous graph with relational labels. Compared to LGESQL, our proposed model achieves an absolute improvement of 1.7% in the overall exact match accuracy, implying that our model can handle more complicated SQL parsing. This is due to the fact that the proposed graph-pruning mechanism drops the redundant node information. This verifies the superiority of the proposed graph-pruning strategy.

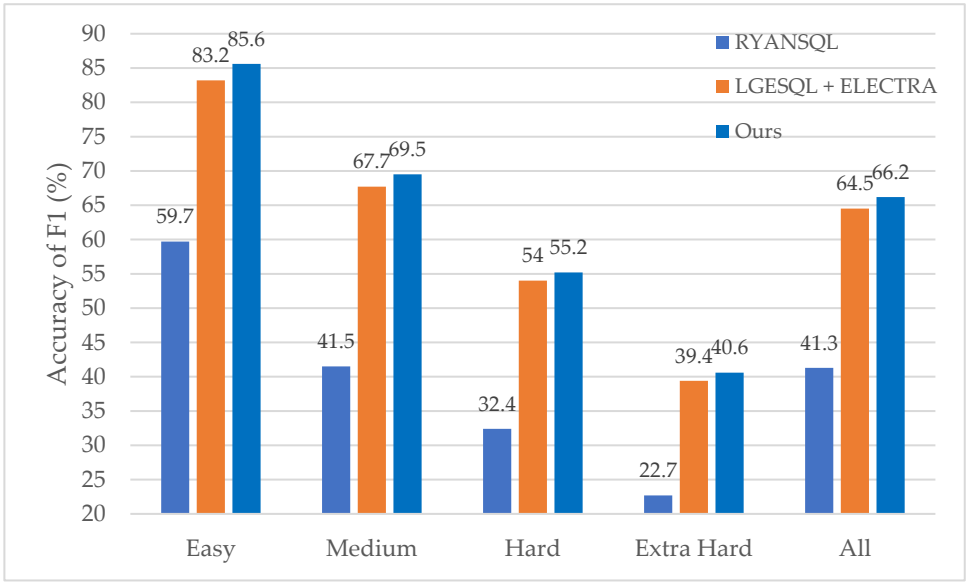

Figure 3. The accuracy results on CSpider compared to LGESQL and RYANSQL by hardness levels.

As shown in Figure 3, the predictive performance of our proposed model decreases as the hardness level of the sample increases, with the exact match accuracy of the Easy level sample being 85.6% and the Extra Hard level sample being 40.6%, indicating that the model is a weak fit for the Extra Hard level sample. The reason for this is that the Medium and Hard levels are mostly join queries, and the Extra Hard level is mostly nested queries.

In order to further investigate the prediction effectiveness of the algorithm in different parts of SQL, the F1 scores of the main modules of the model were experimentally analyzed at multiple difficulties of CSpider, as listed in Table 4. IUEN is the general term for the four operations INTERSECT, UNION, EXCEPT, and NONE in nested queries; these operations are only present in the samples of Hard and Extra Hard levels. The Easy and Medium levels of hardness questions in the dataset do not involve IUEN operations, so the F1 scores of IUEN at Easy and Medium levels of hardness are not considered. KEYWORDS represents a collection of SQL keywords without column names and operators. As shown in Table 4, due to the embedded database schema and question association information when coding, the F1 scores are generally higher for the predicted column and table names such as SELECT, WHERE, and GROUP, showing that our proposed model has a significant effect. However, there is still room for improvement in the query performance of the WHERE clause and IUEN. From these results, it is clear that F1 scores do not decrease significantly as the hardness levels of the dataset increase from Easy to Extra Hard. This proves that our proposed model can basically solve the natural language-to-SQL task in cross-domain scenarios.

Table 4. F1 scores of components in CSpider at different hardness levels.

Component	Easy (%)	Medium (%)	Hard (%)	Extra Hard (%)	All (%)
SELECT	88.0	75.0	87.4	73.5	80.0
WHERE	79.6	65.2	52.7	46.9	62.5
WHERE (no OP)	80.6	68.0	65.9	55.9	68.1
GROUP (no HAVING)	78.3	78.6	78.6	73.7	77.2
GROUP	73.9	72.5	76.2	71.3	72.8
ORDER	60.0	68.4	78.2	78.0	73.0
AND/OR	99.6	98.0	96.7	92.5	97.3
IUEN	-	-	35.3	35.1	34.1
KEYWORDS	90.9	91.1	80.0	72.4	85.2

4.5. Ablation Study

To confirm the viability of our suggested strategy, we also carried out ablation tests to examine the effects of different modules in the proposed model. The results of ablation experiments are shown in Table 5.

Table 5. Ablation study of different modules. GP: graph pruning; RS: relational structures information.

Model	EM (%)
Ours	66.2
w/o GP	65.9
w/o RS	65.5
w/o RS+GP	64.7

As can be seen from the ablation study results in Table 5, the complete model achieves optimal performance compared with the different ablation models. Eliminating relational structure information and the graph-pruning module lowers the model's performance. In order to explain the reason for the decreased effect of the ablation model, this paper analyzes the case shown in Figure 4, where the wrong SQL comes from the ablation model without relational structure information and the graph-pruning module, and the right SQL comes from the self-complete model. In addition, the database schema information is listed in Table 5.

As can be seen from Figure 4, the wrong SQL fails to recognize that "amc hornet sportabout (sw)" is a complete slot value because "sw" happens to be a value in the "model" column of the "model_list" table and "amc hornet" also happens to be a value in the "Make" column of the "car_names" table. Therefore, "amc hornet sportabout (sw)" is split into "amc hornet" and "sw" slot values and matched to their corresponding column names.

After constructing the heterogeneous graph with relational structure information, "amc hornet sportabout (sw)" and column name "Make" are connected to the edge labeled EXACT-MATCH, while "amc hornet" and column name "Make" will be connected with the edge marked PARTIAL-MATCH. The heterogeneous graph constructed in this way enables the model to establish a more fine-grained relationship between questions and database schemas.

Since the semantic information in the question is obviously about the acceleration of the car equipped with the "amc hornet sportabout (sw)" package and is not related to the car's module (model), the graph-pruning module removes the edge related to the column name "Model". In this way, the model can better eliminate the graph information irrelevant to the question, retain the relatively relevant information, and reduce the introduction of noise.

After combining relational structure information and the graph-pruning module, because different edges of heterogeneous graphs have strong feature information, the graph-pruning module can also learn the behavior of preferentially clipping edges marked as PARTIAL-MATCH when there are edges marked as EXACT-MATCH in this case. Therefore,

the combination of the two modules can further improve the effect of the model, which is also consistent with the results of the ablation study. From the above experimental results and case analysis, it is proved that the relational structure information and graph-pruning mechanism designed in this paper are effective in improving the performance of the model.

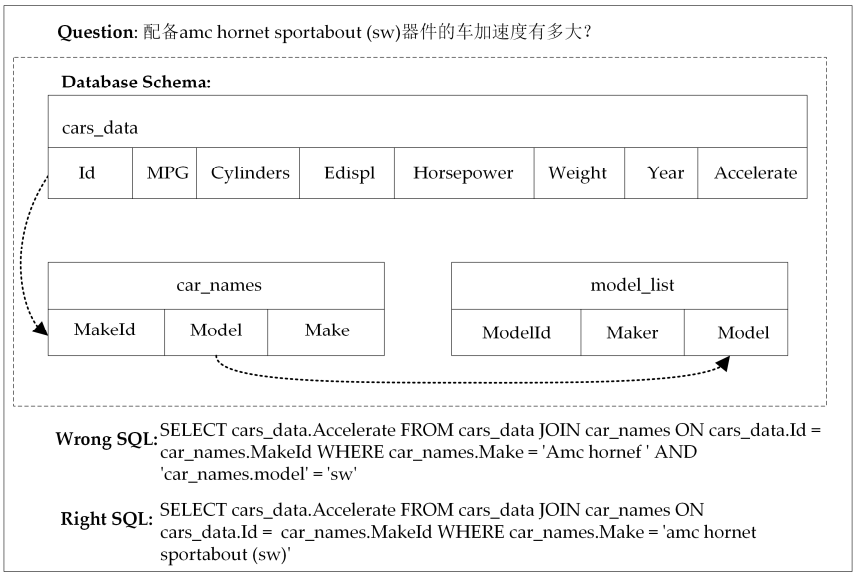

Figure 4. Results of the validity analysis of the ablation study. (The input was Chinese data).

4.6. Case Study

In Figure 5, we compare the SQL generated by our model with that generated by the baseline model LGESQL. We can see that our model performs better than the baseline model. For example, in the first case, LGESQL predicts the name and continent incorrectly. In the second case, LGESQL predicts the min operator as a max operator and loses the query to GNP. In the third case, which is a join of three tables, LGESQL fails to identify the existence of the table "Treatment_Types"; however, our model successfully constructs a connected subgraph by joining the table "Treatments" with "Treatment_Types".

Question: 安圭拉所在的大陆名称是什么?
English: What is the continent name which Anguilla belongs to?
Gold: SELECT Continent FROM country WHERE Name = "安圭拉"
LGESQL: SELECT country.Name FROM country WHERE country.Continent = "安圭拉"
Ours: SELECT country.Continent FROM country WHERE country.Name = "安圭拉"

Question: 亚洲有多少人，其中最大的GNP是多少?
English: How many people live in Asia, and what is the largest GNP among them?
Gold: SELECT sum(Population) , max(GNP) FROM country WHERE Continent = "亚洲"
LGESQL: SELECT MAX(country.Population) FROM country WHERE country.Continent = "亚洲"
Ours: SELECT SUM(country.Population) , MAX(country.GNP) FROM country WHERE country.Continent = "亚洲"

Question: 每个专家的名字和所实施的治疗的描述是什么?
English: What are each professional's first name and description of the treatment they have performed?
Gold: SELECT DISTINCT T1.first_name , T3.treatment_type_description FROM professionals AS T1 JOIN Treatments AS T2 ON T1.professional_id = T2.professional_id JOIN Treatment_types AS T3 ON T2.treatment_type_code = T3.treatment_type_code
LGESQL: SELECT DISTINCT Professionals.first_name , Professionals.last_name FROM Professionals JOIN Treatments ON Treatments.professional_id = Professionals.professional_id
Ours: SELECT DISTINCT Professionals.first_name , Treatment_Types.treatment_type_description FROM Treatments JOIN Professionals ON Treatments.professional_id = Professionals.professional_id JOIN Treatment_Types ON Treatments.treatment_type_code = Treatment_Types.treatment_type_code

Figure 5. Case study: some comparisons with LGESQL show that our model can generate more accurate SQL. (The input was Chinese data).

5. Conclusions

In this paper, we propose a Chinese cross-domain NL2SQL model based on heterogeneous graph and relative position attention mechanism. The proposed model constructs and heterogeneous graphs by introducing relational structure information to model the schema information in structured data, which can better help the model to learn the relevant features of structured query language. Then, the graph-pruning module is used to cut part of the edges in the heterogeneous graph, so that the model can eliminate the relatively irrelevant information and retain the relatively relevant information, thus achieving fine-grained data filtering and reducing useless information in the heterogeneous graph. Extensive experiments have been conducted on the CSpider dataset and the results show that the proposed model outperforms previous SOTA works and achieves a new SOTA performance.

In our future work, we will try to find automated or semi-automated methods for mining relational structure information to further improve the exact match accuracy of the model under complex SQL parsing tasks such as cascade query and nested subquery. In addition, the current model only has the ability to generate SQL in a single round. Facing the needs of users who may obtain information through multiple Q&A in actual interaction scenarios, future work will focus on generating SQL in a session to obtain higher practical application value.

Author Contributions: Conceptualization, C.M.; methodology, C.M.; software, C.M.; validation, C.M., S.F. and Y.W.; formal analysis, C.M., W.Z. and M.H.; writing original draft, C.M.; writing-review and editing, C.M., W.Z. and M.H.; funding acquisition, W.Z. and S.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (Grant #: 82260362, 62241202), in part by the National Key R&D Program of China (Grant #: 2021ZD0111000).

Data Availability Statement: CSpider can be downloaded at https://github.com/taolusi/chisp (accessed on 7 July 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Katsogiannis-Meimarakis, G.; Koutrika, G. A Survey on Deep Learning Approaches for Text-to-SQL. *VLDB J.* **2023**, 1–32. [CrossRef]
2. Codd, E.F. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* **1970**, *13*, 377–387. [CrossRef]
3. Chamberlin, D.D.; Astrahan, M.M.; Eswaran, K.P.; Griffiths, P.P.; Lorie, R.A.; Mehl, J.W.; Reisner, P.; Wade, B.W. SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control. *IBM J. Res. Dev.* **1976**, *20*, 560–575. [CrossRef]
4. Zhou, G.; Luo, P.; Cao, R.; Xiao, Y.; Lin, F.; Chen, B.; He, Q. Tree-Structured Neural Machine for Linguistics-Aware Sentence Generation. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32. [CrossRef]
5. Wang, B.; Titov, I.; Lapata, M. Learning Semantic Parsers from Denotations with Latent Structured Alignments and Abstract Programs. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Cedarville, OH, USA, 2019; pp. 3772–3783.
6. Wang, B.; Shin, R.; Liu, X.; Polozov, O.; Richardson, M. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; Association for Computational Linguistics: Cedarville, OH, USA, 2020; pp. 7567–7578.
7. Cao, R.; Chen, L.; Chen, Z.; Zhao, Y.; Zhu, S.; Yu, K. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; Association for Computational Linguistics: Cedarville, OH, USA, 2021; pp. 2541–2555.
8. A New Model for Learning in Graph Domains. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; IEEE: Piscataway, NJ, USA, 2005; Volume 2, pp. 729–734.

9. Conneau, A.; Lample, G. Cross-Lingual Language Model Pretraining. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
10. Baxter, I.D.; Yahin, A.; Moura, L.; Sant'Anna, M.; Bier, L. Clone Detection Using Abstract Syntax Trees. In Proceedings of the International Conference on Software Maintenance (Cat. No. 98CB36272), Bethesda, MD, USA, 20 November 1998; IEEE: Piscataway, NJ, USA, 1998; pp. 368–377.
11. Min, Q.; Shi, Y.; Zhang, Y. A Pilot Study for Chinese SQL Semantic Parsing. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Cedarville, OH, USA, 2019; pp. 3650–3656.
12. Katsogiannis-Meimarakis, G.; Koutrika, G. A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems. In Proceedings of the 2021 International Conference on Management of Data, Virtual Event China, 9 June 2021; ACM: New York, NY, USA, 2021; pp. 2846–2851.
13. Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; Zhang, D. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Cedarville, OH, USA, 2019; pp. 4524–4535.
14. Yu, T.; Li, Z.; Zhang, Z.; Zhang, R.; Radev, D. TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), New Orleans, LA, USA, 1–6 June 2018; Association for Computational Linguistics: Cedarville, OH, USA, 2018; pp. 588–594.
15. Bogin, B.; Gardner, M.; Berant, J. Global Reasoning over Database Structures for Text-to-SQL Parsing. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Cedarville, OH, USA, 2019; pp. 3657–3662.
16. Chen, Z.; Chen, L.; Zhao, Y.; Cao, R.; Xu, Z.; Zhu, S.; Yu, K. ShadowGNN: Graph Projection Neural Network for Text-to-SQL Parser. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; Association for Computational Linguistics: Cedarville, OH, USA, 2021; pp. 5567–5577.
17. Yin, P.; Neubig, G. A Syntactic Neural Model for General-Purpose Code Generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; Association for Computational Linguistics: Cedarville, OH, USA, 2017; pp. 440–450.
18. Rubin, O.; Berant, J. SmBoP: Semi-Autoregressive Bottom-up Semantic Parsing. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; Association for Computational Linguistics: Cedarville, OH, USA, 2021; pp. 311–324.
19. Scholak, T.; Schucher, N.; Bahdanau, D. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; Association for Computational Linguistics: Cedarville, OH, USA, 2021; pp. 9895–9901.
20. Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; Association for Computational Linguistics: Cedarville, OH, USA, 2018; pp. 3911–3921.
21. Yu, T.; Wu, C.-S.; Lin, X.V.; Wang, B.; Tan, Y.C.; Yang, X.; Radev, D.; Socher, R.; Xiong, C. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. *arXiv* **2021**, arXiv:2009.13845.
22. Shi, P.; Ng, P.; Wang, Z.; Zhu, H.; Li, A.H.; Wang, J.; dos Santos, C.N.; Xiang, B. Learning Contextual Representations for Semantic Parsing with Generation-Augmented Pre-Training. *arXiv* **2020**, arXiv:2012.10309. [CrossRef]
23. Liu, Q.; Chen, B.; Guo, J.; Ziyadi, M.; Lin, Z.; Chen, W.; Lou, J.-G. TAPEX: Table Pre-Training via Learning a Neural SQL Executor. *arXiv* **2022**, arXiv:2107.07653.
24. Zeng, Y.; Li, Z.; Tang, Z.; Chen, Z.; Ma, H. Heterogeneous Graph Convolution Based on In-Domain Self-Supervision for Multimodal Sentiment Analysis. *Expert Syst. Appl.* **2023**, *213*, 119240. [CrossRef]
25. Mo, X.; Tang, R.; Liu, H. A Relation-Aware Heterogeneous Graph Convolutional Network for Relationship Prediction. *Inf. Sci.* **2023**, *623*, 311–323. [CrossRef]
26. Fei, H.; Wu, S.; Ren, Y.; Zhang, M. Matching Structure for Dual Learning. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S., Eds.; PMLR. Volume 162, pp. 6373–6391.
27. Fang, Y.; Li, X.; Ye, R.; Tan, X.; Zhao, P.; Wang, M. Relation-Aware Graph Convolutional Networks for Multi-Relational Network Alignment. *ACM Trans. Intell. Syst. Technol.* **2023**, *14*, 37. [CrossRef]
28. Fei, H.; Li, F.; Li, B.; Ji, D. Encoder-Decoder Based Unified Semantic Role Labeling with Label-Aware Syntax. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 2–9 February 2021; Volume 35, pp. 12794–12802.

29. Wu, S.; Fei, H.; Li, F.; Zhang, M.; Liu, Y.; Teng, C.; Ji, D. Mastering the Explicit Opinion-Role Interaction: Syntax-Aided Neural Transition System for Unified Opinion Role Labeling. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 22 February–1 March 2022; Volume 36, pp. 11513–11521.
30. Yu, B.; Mengge, X.; Zhang, Z.; Liu, T.; Yubin, W.; Wang, B. Learning to Prune Dependency Trees with Rethinking for Neural Relation Extraction. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020; International Committee on Computational Linguistics: Praha, Czech Republic, 2020; pp. 3842–3852.
31. Yu, S.; Mazaheri, A.; Jannesari, A. Topology-Aware Network Pruning Using Multi-Stage Graph Embedding and Reinforcement Learning. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S., Eds.; PMLR. Volume 162, pp. 25656–25667.
32. Fei, H.; Wu, S.; Li, J.; Li, B.; Li, F.; Qin, L.; Zhang, M.; Zhang, M.; Chua, T.-S. LasUIE: Unifying Information Extraction with Latent Adaptive Structure-Aware Generative Language Model. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 15460–15475.
33. Wang, K.; Shen, W.; Yang, Y.; Quan, X.; Wang, R. Relational Graph Attention Network for Aspect-Based Sentiment Analysis. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; Association for Computational Linguistics: Cedarville, OH, USA, 2020; pp. 3229–3238.
34. Forta, B. *Sams Teach Yourself SQL in 10 Minutes*; Pearson Education: London, UK, 2013; ISBN 0-672-33607-3.
35. Qi, J.; Tang, J.; He, Z.; Wan, X.; Cheng, Y.; Zhou, C.; Wang, X.; Zhang, Q.; Lin, Z. RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; Association for Computational Linguistics: Cedarville, OH, USA, 2022; pp. 3215–3229.
36. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-Attention with Relative Position Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), New Orleans, LA, USA, 1–6 June 2018; Association for Computational Linguistics: Cedarville, OH, USA, 2018; pp. 464–468.
37. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
38. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6000–6010.
39. Li, L.; Geng, R.; Li, B.; Ma, C.; Yue, Y.; Li, B.; Li, Y. Graph-to-Text Generation with Dynamic Structure Pruning. In Proceedings of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea, 12–17 October 2022; International Committee on Computational Linguistics: Praha, Czech Republic, 2022; pp. 6115–6127.
40. Popescu, A.-M.; Etzioni, O.; Kautz, H. Towards a Theory of Natural Language Interfaces to Databases. In Proceedings of the 8th International Conference on Intelligent User Interfaces, Miami, FL, USA, 12–15 January 2003; Association for Computing Machinery: New York, NY, USA, 2003; pp. 149–157.
41. Tang, L.R.; Mooney, R.J. Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing. In Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Hong Kong, China, 7–8 October 2000; Association for Computational Linguistics: Cedarville, OH, USA, 2000; pp. 133–141.
42. Zelle, J.M.; Mooney, R.J. Learning to Parse Database Queries Using Inductive Logic Programming. In Proceedings of the Thirteenth National Conference on Artificial Intelligence—Volume 2, Portland, OR, USA, 4–8 August 1996; AAAI Press: Washington, DC, USA, 1996; pp. 1050–1055.
43. Iyer, S.; Konstas, I.; Cheung, A.; Krishnamurthy, J.; Zettlemoyer, L. Learning a Neural Semantic Parser from User Feedback. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; Association for Computational Linguistics: Cedarville, OH, USA, 2017; pp. 963–973.
44. Li, F.; Jagadish, H.V. Constructing an Interactive Natural Language Interface for Relational Databases. *Proc. VLDB Endow.* **2014**, *8*, 73–84. [CrossRef]
45. Yaghmazadeh, N.; Wang, Y.; Dillig, I.; Dillig, T. SQLizer: Query Synthesis from Natural Language. *Proc. ACM Program. Lang.* **2017**, *1*, 63. [CrossRef]
46. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
47. Manning, C.D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.J.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the Association for Computational Linguistics (ACL) System Demonstrations, Baltimore, MD, USA, 22–27 June 2014; pp. 55–60.
48. Gal, Y.; Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.

49. Yu, T.; Yasunaga, M.; Yang, K.; Zhang, R.; Wang, D.; Li, Z.; Radev, D. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; Association for Computational Linguistics: Cedarville, OH, USA, 2018; pp. 1653–1663.
50. Choi, D.; Shin, M.C.; Kim, E.; Shin, D.R. RYANSQL: Recursively Applying Sketch-Based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases. *Comput. Linguist.* **2021**, *47*, 309–332. [CrossRef]
51. Wang, B.; Lapata, M.; Titov, I. Meta-Learning for Domain Generalization in Semantic Parsing. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; Association for Computational Linguistics: Cedarville, OH, USA, 2021; pp. 366–379.
52. Clark, K.; Luong, M.-T.; Le, Q.V.; Manning, C.D. ELECTRA: Pre-Training Text Encoders as Discriminators Rather than Generators. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net; 2020.