Divide-and-conquer Heterogeneous Structure Learning for Text-to-SQL

Anonymous ACL submission

Abstract

Existing leading Text-to-SQL approaches with heterogeneous structure learning utilize a unified learning process for semantic and node-edge structural information. However, the unified learning process leads to two major limitations: (i) The mixing of semantic and structural information may cause incorrect linking in structure learning. (ii) The indiscriminate processing of the node graph and the edge graph will cause the loss of the unique property of each graph. In order to address these limitations, we propose a divide-and-conquer Heterogeneous Structure Learning(DCHL) framework for Text-to-SQL, which abstracts the structural information and divides out the semantic information from the original input. Specifically, our framework is featured with the Abstract Graph Construction and Abstract Graph Encoder for the node and edge respectively. We also devise a Semantic-structural Aggregation Mechanism to fuse the divided semantic information and the topological structure information of nodes and edges. Extensive experiments on three benchmark datasets show that DCHL clearly outperforms strong competitors and achieves new state-of-the-art results. the proposed DCHL achieves competitive results (62.9% with GLOVE, 72.1% with ELECTRA) on the cross-domain text-to-SQL benchmark Spider at the time of writing.

1 Introduction

The Text-to-SQL task, aiming to convert natural language questions into corresponding SQL queries, is a key technology for building database business intelligence applications (Cai et al., 2018; Hwang et al., 2019; Yu et al., 2018a). To alleviate the huge cost of training the Text-to-SQL model for each specific database, the cross-domain Text-to-SQL tries to generalize the trained models to unseen databases. The core of cross-domain generalization lies in solving the question-schema linking problem, i.e., building alignment between natural language questions and database schemas.

Existing leading approaches address the question-schema linking problem under the heterogeneous structure learning framework. Among them, the approaches adopting heterogeneous graph encoders have shown significant improvement by taking the advantage of learning multiple prior structure knowledge simultaneously (Wang et al., 2020a; Cai et al., 2021; Cao et al., 2021), e.g., SADGA devises a unified dual graph framework to jointly learn the semantic and structural information of the question and database schema, LGESQL constructs a node-centric graph and an edge-centric graph and further utilizes unified RGAT to alternatively update the representation of node and edge.

Although some promising results have been reported, the existing heterogeneous structure learning methods are still limited by their widely used unified encoding process. The first limitation is due to the mixing of semantic and structural information. Specifically, both the semantic information and the structural information (nodes or edges) are represented using tokens, which may raise the wrong (or missing) important structural information. For example, as shown in Figure 1(a), the "average" in the given case is an item of the database, but the existing approach incorrectly generates the SQL aggregation function "AVG". The second limitation is the indiscriminate processing process of the node-centric graph and the edge-centric graph. Specifically, the topological characteristics of edge-centric graphs are different from those of nodes, and using the same encoding process will cause the loss of the unique property of each graph. Though the line graph used in LGESQL splits the nodes into multiple edges during RGAT encoding, they still cannot extract the edge topology information accurately. Thus, the key to tackling these limitations is to effectively divide complex heterogeneous struc-

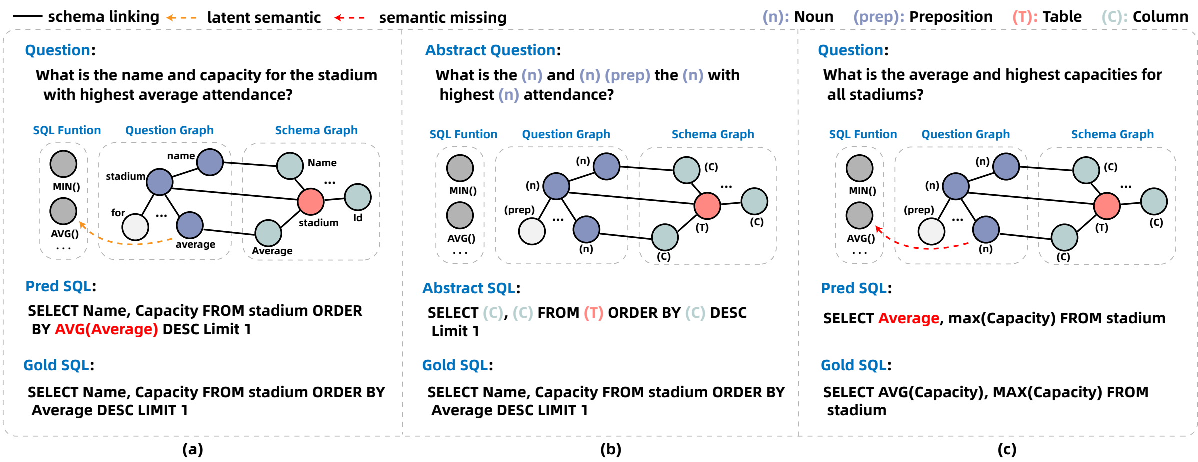

Figure 1: A toy example. The left part shows that some existing models, e.g., LGESQL, generate the wrong SQL language when structural and semantic information is available. The middle part shows that without interference from semantic information, the model can use structural information to generate accurate SQL. The right part shows that the model cannot generate the correct SQL without the help of semantic information when using only structural information.

tures into multiple aspects and specifically employ suitable structure learning modules.

In this paper, we propose a Divide-and-conquer Heterogeneous Structure Learning (DCHL) for cross-domain Text-to-SQL. The "Divide-and-conquer" is a combination term that we borrowed from computer science, and nicely echoes the idea of the proposed DCHL. The division of DCHL has two aspects meaning: (i) the semantic information and the structure information are separated; (ii) the structure information of nodes and edges are processed separately by considering their unique characteristics. To implement the above division, we first extract the semantic and structural information from the input graph, then further divide the structural information into abstract node graph and abstract edge graph. Second, we propose Abstract Graph Encoder includes two encoding branches for abstract node graph and abstract edge graph, which employ a new edge graph transformation method and a corresponding hypergraph encoder to accurately and scalably encode the structure information of edges. As shown in Fig 1(b), retaining the structural information after removing the semantic information can effectively prevent incorrect links brought by specific tokens. However, Fig 1(c) shows that when semantic information is lost, it also leads to incorrect query generation. Therefore, the semantic graph branch with full graph input is retained in DCHL and the learning process is consistent with leading work. Finally, the Semantic-Structural Aggregation Mechanism will aggregate the semantic graph branch and the other two abstract structural branches by a gated-based aggregation mechanism. The contributions are summarized as follows:

- We propose the DCHL framework to solve the problem of heterogeneous structure learning in cross-domain Text-to-SQL by dividing hybrid inputs into semantic information and structural information.
- We devise the Abstract Graph Encoder and Semantic-structural aggregation mechanism which allows different types of information to be learned separately and fused efficiently.
- We conduct extensive experiments to study the effectiveness of the proposed DCHL framework. Experiments on three benchmarks demonstrate DCHL outperforms the baseline methods. Our implementation will be open-sourced after acceptance.

2 Model Overview

We first make the necessary definition for the Text-to-SQL task and the Heterogeneous Graph Input for our DCHL framework. DCHL follows the encoder-decoder manner to efficiently process heterogeneous structure learning by dividing semantic and node-edge structure information encoding branches.

Problem Definition Text-to-SQL task can be defined as follows: given a natural language question $Q = \{q_i\}_{i=1}^{|Q|}$ and the corresponding database schema $S = \langle T, C \rangle$ including tables $T = \{t_i\}_{i=1}^{|T|}$ and columns $C = \{c_1^{t_1}, c_2^{t_1}, \dots, c_1^{t_2}, c_2^{t_2}, \dots\}$, the goal is to generate the correct SQL y for the question.

Heterogeneous Graph Input As many existing works do (Cao et al., 2021; Cai et al., 2021; Wang et al., 2020a), we first construct a heterogeneous graph consisting of the question and the database, which can be represented by $G = (V, R, P)$. Specially, **a)** the node set $V = Q \cup T \cup C$ consists nodes of question words, tables and columns. The initial node embedding matrix $X \in \mathbb{R}^{|V^{|Q|+|T|+|C|}| \times d}$ is obtained by GloVe embeddings (Pennington et al., 2014) or the pretrained language model (PLM). **b)** The relation matrix $R = \{r\}_{i=1,j=1}^{|V|,|V|}$ represents the edge type among nodes. According to some typical database-specific knowledge and string-match strategies, we predefine several edge types and assign each edge to one of the predefined types. Each edge type is represented as a learnable embedding with random initialization, and the edge embedding matrix can be represented as $Z \in \mathbb{R}^{|R| \times d}$. Details of all predefined edge types can be found in the appendix. **c)** The vector $P = \{p\}_{i=1}^{|V|} \in \{question, table, column\}$ represents the node type for each node. In order to better capture the lexical information between question words, we further subdivide the question type into various lexical categories, e.g., *nouns*, *verbs*. Each node type can be featured as a learnable vector initialized from GloVe or PLM, and the initial node type embedding matrix can be represented as $S \in \mathbb{R}^{|V^{|Q|+|T|+|C|}| \times d}$. After the heterogeneous graph is constructed, it will be fed into a typical encoder-decoder framework.

Encoder In the encoder part, we devise the DCHL framework to solve the problem of heterogeneous structure learning, including three steps: abstract graph construction, abstract graph encoder, and semantic-structural aggregation mechanism. In the workflow of DCHL, we first construct two abstract graphs at the node level and edge level named abstract node graph and abstract edge graph. Second, we further design an abstract graph encoder including two branches for the two abstract graphs, which is more conducive to learning the topology of nodes and edges separately. Finally, the semantic-structural aggregation mechanism aggregates two abstract graphs (i.e., the abstract node feature and the abstract edge feature) into the RGAT layer (Wang et al., 2020b) combined with the semantic graph branch, to obtain the final graph representation in DCHL.

Decoder In the decoder part, we follow the tree-structured architecture, which transforms the SQL query into the abstract syntax tree (AST) in depth-first traversal order (Yin and Neubig, 2017). First, based on all node representations from DCHL, the decoder outputs a sequence of actions that generates an AST; then, the AST can be transformed into a sequential SQL query.

3 Divide-and-conquer Heterogeneous Structure Learning

In this section, we will delve into our encoder framework, Divide-and-conquer Heterogeneous Structure Learning (DCHL). DCHL first divides semantic and structural information from the original input; second, the topology structural information of nodes and edges is processed in *Abstract Graph Construction* and *Abstract Graph Encoder* respectively; finally, structural information and semantic information are aggregated in *Semantic-Structural Aggregation*. The details of each component are as follows.

3.1 Abstract Graph Construction

We construct the abstract graph by dividing the semantic information out from the original input G. The construction details of the Abstract Node Graph and Abstract Edge Graph are as follows.

Abstract Node Graph In order to better learn the local structure of the nodes, following Cao et al. (2021), we divide all edge types into local relations and non-local relations, and only consider the local relation when constructing the Abstract Node Graph (ANG), which can be simply represented by $G_n^{local} = (P, A^{local})$, where P represents the type for each node in G, and the A^{local} represents the adjacency matrix which only considers local relation. As shown in Figure 2, ANG only retains the graph structure and the node type feature.

Abstract Edge Graph As we describe in the introduction, the line graph transformation used in LGESQL has obvious drawbacks: **a)** employing the same encoding method as the nodes may not be appropriate; **b)** splitting nodes into multiple edges leads to loss of topological information of the original graph during the transformation; **c)**

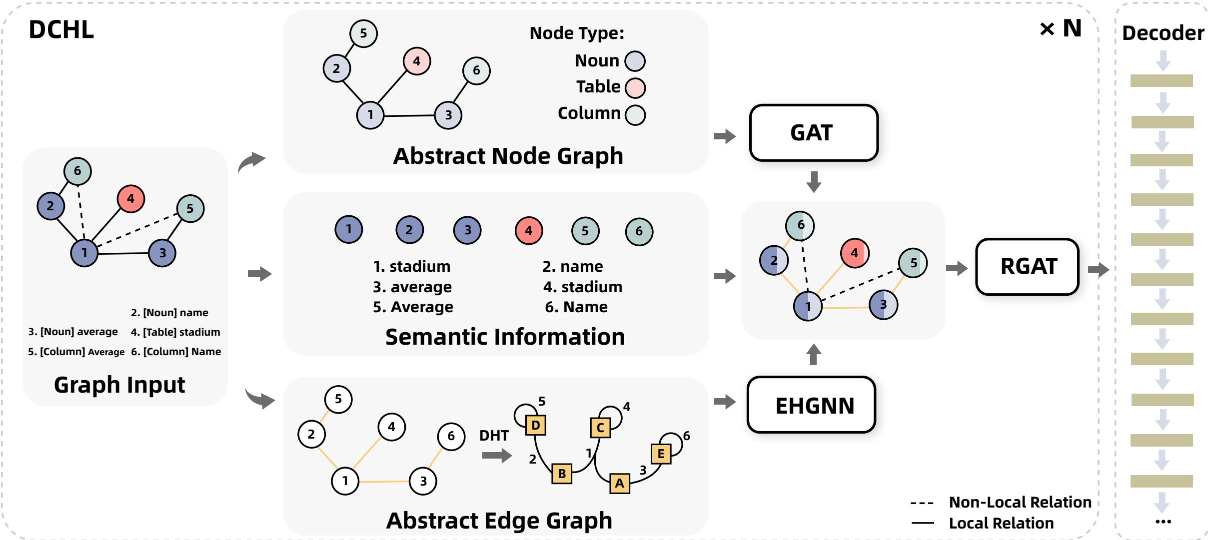

Figure 2: The overview of the proposed model.

transformation is not unique, thus two different question-schema graphs may be transformed into the same line graph.

To address the above issues, we adopt Dual Hypergraph Transformation (DHT) (Jo et al., 2021) in construction, which can convert a graph into the corresponding hypergraph. The process of DHT is as follows: nodes on the original graph are transformed into hyperedges on the hypergraph, and edges are transformed into nodes on the hypergraph. Taking advantage of the duality of the hypergraph, DHT aims to interchange the structural role of node and edge. As shown in Figure 3, we provide a case for DHT, where the incidence matrix M in the original graph represents the interaction between $|V|$ nodes and $|R|$ edges, i.e., each entry indicates whether the node is incident to the edge. M^T indicates the incidence matrix in the hypergraph.

In DCHL, through DHT we convert the graph G_n^{local} and R^{local} into the hypergraph, which we call the Abstract Edge Graph (AEG), denoted as $G_e^{local} = (R^{local}, M^T)$, where M can be transformed by A^{local} and T represents the transpose operation. AEG only reserves the graph structure and the edge type feature.

3.2 Abstract Graph Encoder

After the ANG and the AEG were constructed, we design two encoders, ANG Encoder and AEG Encoder, to learn the structural information for nodes and edges separately.

Abstract Node Graph Encoder For effectively learning the abstract structural knowledge of nodes,

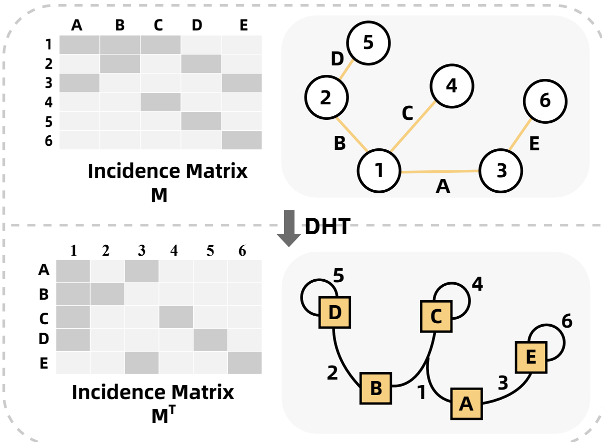

Figure 3: The transformation of the original graph to the hypergraph by DHT.

we employ a Graph Attention Network (GAT) (Velickovic et al., 2017) to encode the node abstract structure representation by performing message propagation among the self-structure.

Given the representation $s_i^{(l)}$ of node v_i in the ANG G_n^{local}, the output abstract structure representation $s_i^{(l+1)}$ of the l-th layer is computed by

$$e_{ij}^{(l+1)} = a^{(l)} \left[W_k^l s_i^{(l)} || W_k^l s_j^{(l)} \right], \tag{1}$$

$$\alpha_{ij}^{(l+1)} = \frac{exp\left(LeakyReLU\left(e_{ij}^{(l+1)}\right)\right)}{\sum_{k \in \mathcal{N}_i^{local}} exp\left(LeakyReLU\left(e_{ik}^{(l+1)}\right)\right)}, \tag{2}$$

$$s_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}_i^{local}} \alpha_{ij}^{(l+1)} W_a^l s_j^{(l)} \right), \tag{3}$$

where σ is a nonlinearity function, e.g., ReLU, W_k, W_a are the learnable weight matrix, $\mathcal{N}_i^{local}$

represents the neighbor indices of node v_i in G_n^{local}. With the help of GAT, the node type features in ANG can be encoded to obtain the node abstract structure representation.

Abstract Edge Graph Encoder To accurately and scalably encode the structure information of edges, we encode the edge abstract structural representation by performing message propagation using EHGNN (Jo et al., 2021). Given the AEG G_e^{local}, the node representation $(z_i^*)^{(l+1)}$ of the l-th layer is computed by

$$\widehat{x}_i^{(l)} = \frac{1}{|\mathcal{N}^*_{n,\ i}|} \sum_{z_j^* \in \mathcal{N}^*_{n,\ i}} W_z z_j^{*(l)} \quad (4)$$

$$(z_i^*)^{(l+1)} = \frac{1}{|\mathcal{N}^*_{e,\ i}|} \sum_{x_j \in \mathcal{N}^*_{e,\ i}} \widehat{x}_j^{(l)}, \quad (5)$$

where $\hat{x}_i$ represents the hyperedge v_i^* temporarily aggregates the information of surrounding nodes, $\mathcal{N}^*_{n,\ i}$ and $\mathcal{N}^*_{e,\ i}$ represent the set of nodes around hyperedge v_i^* and the set of hyperedges around node r_i^*, respectively. $|\mathcal{N}^*_{e,\ i}|$ is always 3 (each node is connected to two hyperedges and an additional self-loop edge). Figure 4 shows the detail of the message passing on hypergraph in our AEG encoder.

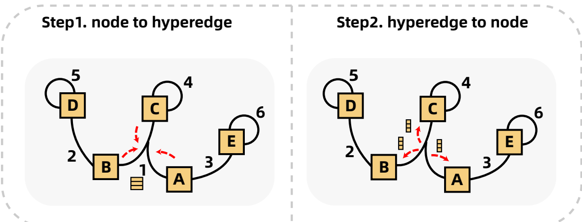

Figure 4: The two steps of message passing on hypergraph: i) using the topology of the hypergraph to pass the node feature (the edge feature in the original graph) to the hyperedge, ii) passing the information from the hyperedge back to the node. Note that we only consider hyperedge 1 for simplicity.

In our DCHL, the AEG encoder is able to dedicatedly and efficiently learn edge topology information without any node information, unlike previous work that learns nodes and edges uniformly.

After the abstract graph encoding, we can obtain the abstract structural representation of nodes and edges separately, indicated as s and z. In the next, we will introduce how to aggregate them combined with the node semantic information to obtain the final graph representation.

3.3 Semantic-Structural Aggregation

In this semantic-structural aggregation module, we combine the semantic feature in the original heterogeneous graph G and the abstract structural feature through the gate-based mechanism. Specially, for each node, the node representation $c^{(l+1)}$ of the l-th layer is calculated by

$$b_i = \left[x_i^{(l)} || s_i^{(l)}\right] \quad (6)$$

$$c^{(l+1)} = gate(b_i) \cdot x_i^{(l)} + (1 - gate(b_i)) \cdot s_i^{(l)}, \quad (7)$$

where $||$ represents vector concatenation and $gate$ is a linear layer with a Sigmoid function indicating how much semantic or abstract structural information the node should receive.

After obtaining the new node representation with semantic-structural information and the edge structural feature from the AEG encoder, inspired by LGESQL (Cao et al., 2021), we leverage an RGAT (Wang et al., 2020b) layer to obtain the final node representation of the heterogeneous graph. The output representation $x_i^{(l+1)}$ of the l-th layer is computed by

$$e_{ij}^{(h)} = \frac{c_i W_q^{(h)} \left(c_j W_k^{(h)} + \psi\left(r_{ij}\right)\right)^T}{\sqrt{d_z/H}}, \quad (8)$$

$$\alpha_{ij}^{(h)} = softmax\left\{e_{ij}^{(h)}\right\}, \quad (9)$$

$$x_i^{(l+1)} = \sum_{v_j \in \mathcal{N}_i} \alpha_{ij}^{(h)} \left(c_j W_v^{(h)} + \psi\left(r_{ij}\right)\right) \quad (10)$$

where W_q, W_k, W_v are the trainable parameter, H is the number of heads, $\mathcal{N}_i$ represents the neighborhoods of node v_i. The function $\psi\left(r_{ji}\right)$ returns an edge feature vector with respect to the relation r_{ji}: if r_{ji} belongs to the local relation (i.e., has been learned by the AEG encoder, the function returns the edge structural representation z_{ij}^* from the AEG encoder, otherwise it returns a trainable feature vector.

Through the above semantic-structural aggregation method, we can effectively fuse the divided semantic information and the structural information of nodes and edges to obtain accurate decoding.

4 Experiments

4.1 Experiment Setup

Dataset We evaluate the DCHL framework on three widely used Text-to-SQL benchmarks

Model	Dev	Test
Without PLM:GloVe		
Global-GNN (Bogin et al., 2019)	52.7	47.4
EditSQL (Zhang et al., 2019)	36.4	32.9
IRNet (Guo et al., 2019)	53.2	46.7
RATSQL (Wang et al., 2020a)	62.7	57.2
LGESQL (Cao et al., 2021)	67.6	62.8
DCHL	**69.3**	**62.9**
With Model Adaptive PLM		
RATSQL + STRUG (Deng et al., 2021)	72.6	68.4
RATSQL + GRAPPA (Yu et al., 2021)	73.4	69.6
SmBoP + GRAPPA (Rubin and Berant, 2021)	74.7	69.5
RATSQL + GAP (Shi et al., 2021)	71.8	69.7
SADGA + GAP (Cai et al., 2021)	73.9	70.1
DT-Fixup SQL-SP + RobERTa (Xu et al., 2021)	75.0	70.9
LGESQL + ELECTRA (Cao et al., 2021)	75.1	72.0
DCHL + ELECTRA	**76.5**	**72.1**

Table 1: Exact match accuracy (%) on Spider development set and test set.

Model	DK	SYN
Without PLM: GloVe		
Global-GNN (Bogin et al., 2019)	26.0	23.6
EditSQL (Zhang et al., 2019)	31.4	25.3
IRNet (Guo et al., 2019)	33.1	28.4
RATSQL (Wang et al., 2020a)	35.8	33.6
LGESQL (Cao et al., 2021)	39.2	40.5
ISESL-SQL (Liu et al., 2022)	42.1	44.4
DCHL	**42.8**	**47.5**
With Model Adaptive PLM		
RATSQL + STRUG (Deng et al., 2021)	39.4	48.9
RATSQL + GRAPPA (Yu et al., 2021)	38.5	49.1
SmBoP + GRAPPA (Rubin and Berant, 2021)	42.2	48.6
RATSQL + GAP (Shi et al., 2021)	44.1	49.8
DT-Fixup SQL-SP + RobERTa (Xu et al., 2021)	40.5	50.4
LGESQL + ELECTRA (Cao et al., 2021)	47.2	62.6
ISESL-SQL + ELECTRA (Liu et al., 2022)	50.0	62.6
PROTON + ELECTRA (Wang et al., 2022)	51.0	65.6
SUN + ELECTRA (Qin et al., 2022)	52.7	66.9
DCHL + ELECTRA	**54.4**	**68.1**

Table 2: Exact match accuracy (%) on Spider-DK and Spider-SYN.

datasets as follows: (1) **Spider** (Yu et al., 2018b) is a large-scale cross-domain Text-to-SQL benchmark. It contains 8659 training samples across 146 databases and 1034 evaluation samples across 20 databases. We report the exact set match accuracy on the development set and the test set. The test dataset contains 2147 samples with 40 unseen databases. Since the fair competition, the Spider official has not released the test set for evaluation. We submit our model to the organizer of the challenge for evaluation. (2) **Spider-DK** (Gan et al., 2021b) is a human-curated dataset based on Spider, which is constructed by selecting 535 samples from Spider dev set, with focusing on evaluating the model understanding of domain knowledge. We train our model on the Spider training set and test on the Spider-DK development set. (3) **Spider-SYN** (Gan et al., 2021a) is another challenging variant of Spider. It is constructed by manually modifying NL questions with synonym substitution, making it more adaptable for cases where the user does not know the exact schema word mentioned.

Implementation Details In the preprocessing phase, we tokenize and lexicalize questions, table names, column names, and their types with the Standford Nature Language Processing toolkit.[1]. In order to use contextual information, we use GloVe (Pennington et al., 2014) word embeddings and the pre-trained language model ELECTRA (Clark et al., 2020). The schema linking strategy is borrowed from LGESQL (Cao et al., 2021), which is also our baseline. For a fair comparison with baselines, we configure it with the same set of hyperparameters. In the encoder, we stack 8 DCHL layers, the hidden size is set to 256 for GloVe and 512 for ELECTRA. In the decoder, the dimension of hidden state, action embedding and node type embedding are set to 512, 128 and 64. The learning rate is 5e-4 for GloVe and 1e-4 for ELECTRA. The number of training epochs is 100 for GLOVE, and 200 for PLMs respectively. We trained our models on one server with a single NVIDIA GTX 3090 GPU.

[1]https://github.com/stanfordnlp/stanza

4.2 Overall Performance

Table 1 and Table 2 shows the exact match accuracy on three benchmarks with the exact match average accuracy of 3 runs. Almost all baseline results are obtained from official leaderboard or original papers. As shown in Table 1, we can see that DCHL outperforms all existing models on Spider. DCHL has a significant improvement over the SOTA model LGESQL+ELECTRA on the development set and gets comparable results on the test set. We guess that the domain difference between the development and test sets leads to this. This interesting observation is also evident in the Spider leaderboard. As shown in table 2, our models can significantly outperform the previous best models on Spider-DK and Spider-SYN. It is worth noting that our model obtains 7.2% improvement and 5.5% improvement over LGESQL on the Spider-DK and Spider-SYN datasets, respectively. Spider-DK and Spider-SYN improve domain knowledge and more

Technique	Spider	Spider-DK	Spider-SYN
DCHL	**76.5**	**54.4**	**68.1**
w/o NSI	76.0(0.5↓)	49.9(4.5↓)	64.9(3.2↓)
w/o ESI	74.7(1.8↓)	51.2(3.3↓)	63.8(4.3↓)
w/o SSA	74.8(1.7↓)	50.2(4.2↓)	63.7(4.4↓)

Table 3: Ablation study of different modules. NSI: node abstract structure information; ESI: edge abstract structure information; SSA: semantic-structural aggregation.

complex semantic information than Spider. Greater improvements in Spider-DK and Spider-SYN also effectively support that DCHL gets better generalization ability by abstracting structural and semantic information.

4.3 Ablation Studies

We conduct ablation studies to show the effectiveness of different modules of DCHL to the overall improved performance. The major model variants are as follows:

w/o node abstract structure information Discard the abstract node structural information learning phase (i.e., Eq. 2 ~4).
w/o edge abstract structure information Discard the abstract edge structural information learning phase (i.e., Eq. 6 ~7).
w/o semantic-structural aggregation Average aggregation of semantic and structural information (i.e., Eq. 9, $gate(\cdot) = 0.5$).

The ablation experimental results are presented in Table 3. As the table shows, all components are necessary to DCHL. More specifically, DCHL w/o node structure information give us 0.5% less performance averaged on Spider, but on the other two datasets, performance decreases by 4.5% and 3.2%, respectively, indicating that structural information is essential in the synonym substitution or where domain knowledge is required. Similarly, edge structure information gives 3.1% performance boost in average over all benchmarks. Furthermore, when removing the semantic-structural aggregation, DCHL brings an average performance drop of 3.4%.

We can discover that the DCHL module contributes more in Spider-DK and Spider-SYN than the Spider benchmark, which proves the importance of abstract structural information in challenging settings.

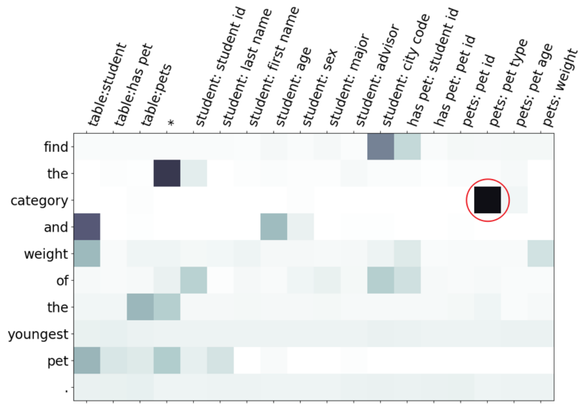

Figure 5: Alignment between question words and tables/columns in our model.

4.4 Case Studies

To intuitively understand the effectiveness of our model, we selected three different types of cases from three benchmarks and presented them in Table 4 to compare the SQL statements generated by our model and LGESQL, including word polysemy, synonym substitution, and domain adaptation.

For the case of word polysemy, as shown in the first case and the second case, where the *average* have multiple implications: SQL function "AVG()" and column *Average*, baseline fails to determine the right implication for SQL generation, while our model can successfully identify the correct implication by selectively aggregating semantic and abstract structural information.

For the case of synonym substitution, as shown in the third case, the token *category* in question could not be matched to the column *type* via the string match method. Since our method has extracted and learned the structural information of the original question, the above problem can be avoided in most cases. As shown in Figure 5, we can obtain the interpretable result. For example, the question word *category* has a strong activation with the columns *pet_type*, although the string match method cannot capture the alignment, while DCHL can easily align the words with the help of abstract structural information.

For the case of domain adaptation, as shown in the fourth case, on the training set, the word *or* always corresponds to the "OR" in the SQL statement. However, when the domain knowledge on the development set is different from the training set, LGESQL generates incorrect SQL statements, while DCHL demonstrates domain adaptation capabilities.

Question	*What is the average and maximum capacities for all stadiums ?*
LGESQL	SELECT AVG(average), MAX(highest) FROM stadium
DCHL	SELECT AVG(capacity), MAX(capacity) FROM stadium
Gold	SELECT AVG(capacity), MAX(capacity) FROM stadium
Question	*What is the name and capacity for the stadium with highest average attendance?*
LGESQL	SELECT name, capacity FROM stadium GROUP BY highest ORDER BY AVG(average) DESC LIMIT 1
DCHL	SELECT name, capacity FROM stadium ORDER BY average DESC LIMIT 1
Gold	SELECT name, capacity FROM stadium ORDER BY average DESC LIMIT 1
Question	*Find the* ***~~type~~*** *and weight of the youngest pet.*
SYN_Question	*Find the* ***category*** *and weight of the youngest pet.*
LGESQL	SELECT Pet_age, weight FROM Pets ORDER BY pet_age LIMIT 1
DCHL	SELECT Pet_type, weight FROM Pets ORDER BY pet_age LIMIT 1
Gold	SELECT Pet_type, weight FROM Pets ORDER BY pet_age LIMIT 1
Question	*How many concerts are there* ***~~in year 2014 or 2015~~*** *?*
DK_Question	*How many concerts are there* ***after or in year 2014*** *?*
LGESQL	SELECT COUNT(*) FROM concert WHERE Year > "value" OR Year >= "value"
DCHL	SELECT COUNT(*) FROM concert WHERE Year >= "value"
Gold	SELECT COUNT(*) FROM concert WHERE Year >= 2014

Table 4: Case Study: The first two cases are sampled from Spider, the third example is from Spider-SYN and the last example is from Spider-DK.

5 RELATED WORK

Text-to-SQL Parsing. The architectures proposed for cross-domain Text-to-SQL show increasing complexity in the encoder. IRNet (Guo et al., 2019) leveraged two separate BiLSTMs with self-attention mechanism to encode the NL question and table schema. RATSQL (Wang et al., 2020a) proposes a unified encoding mechanism to handle various pre-defined relations. Recently, besides LGESQL some work has also used heterogeneous graph modeling to solve heterogeneous structure learning. SADGA (Cai et al., 2021) adapts a unified dual graph framework for both the question and database schema, to utilize the global and local structure information across the dual graph on the question schema linking. ShadowGNN (Chen et al., 2021) abstracts the item of database Schema to remove domain information to improve generalization. Sharing the idea of abstract structure, DCHL still has the following advantages over ShadowGNN. First, DCHL also abstracts the question to divide the semantic information of the question, which effectively solves the issue of wrong linking shown in Fig. 1. As for the problem, ShadowGNN only abstracts the schema unit of it and cannot solve the above issue. Second, DCHL has better domain generalization in addition to a more comprehensive division of semantic and structural information, and further topology learning of nodes and edges individually for structural information.

Domain generalization capabilities. Due to the limitations of the existing string match based method, some current works try to improve the accuracy of the graph linking phase with the help of PLM. PROTON (Wang et al., 2022) proposed a probing technique to probe schema linking information between the NL query and the database schema from large-scale PLMs during the initial graph link phase. Based on the distance metric, ISESL-SQL (Liu et al., 2022) is the first to introduce the graph structure learning methods into schema linking and Text-to-SQL, which refines the initial schema linking graph iteratively during model training. Different from these works, we do not use PLM to improve the string match method, but extract a unified structure representation in the encoding process to enhance the domain adaptability of the model.

6 CONCLUSION

In this paper, we propose a Divide-and-conquer Heterogeneous Structure Learning (DCHL) for cross-domain Text-to-SQL. The core idea of the proposed DCHL is to divide the semantic information and structural information from complex heterogeneous structural inputs, which can avoid incorrect question-schema linking and improve the model's domain generalizability. By avoiding the existing methods' unified encoding approach, DCHL devises encoders for the topology of nodes and edges separately and further improves the model's generalizability. Experimental results demonstrated that our method substantially outperformed strong baselines and set state-of-the-art performance on three Text-to-SQL benchmarks.

Limitations

Compared to our baseline model LGESQL, our proposed DCHL requires more computational cost at each training step, with abstract node graph and abstract edge graph encoder.

Although our method divides the semantic information and structural information, the approach focuses more on learning with structural information, while semantic information is only aggregated through the gate-based mechanism. Therefore, a general strategy for learning semantic information is needed. We aim to address this limitation in our future work.

References

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. 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 2019, Hong Kong, China, November 3-7, 2019, pages 3657–3662. Association for Computational Linguistics.

Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. 2018. An encoder-decoder framework translating natural language to database queries. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, pages 3977–3983. ijcai.org.

Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. SADGA: structure-aware dual graph aggregation network for text-to-sql. In Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pages 7664–7676.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 2541–2555. Association for Computational Linguistics.

Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021. 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, NAACL-HLT 2021, Online, June 6-11, 2021, pages 5567–5577.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-sql. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, pages 1337–1350. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-sql models against synonym substitution. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 2505–2515. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-sql generalization. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pages 8926–8931. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 4524–4535. Association for Computational Linguistics.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. CoRR, abs/1902.01069.

Jaehyeong Jo, Jinheon Baek, Seul Lee, Dongki Kim, Minki Kang, and Sung Ju Hwang. 2021. Edge representation learning with hypergraphs. In Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pages 7534–7546.

Aiwei Liu, Xuming Hu, Li Lin, and Lijie Wen. 2022. Semantic enhanced text-to-sql parsing via iteratively learning schema linking graph. In KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022, pages 1021–1030. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1532–1543. ACL.

Bowen Qin, Lihan Wang, Binyuan Hui, Bowen Li, Xiangpeng Wei, Binhua Li, Fei Huang, Luo Si, Min Yang, and Yongbin Li. 2022. SUN: exploring intrinsic uncertainties in text-to-sql parsers. In Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022, pages 5298–5308. International Committee on Computational Linguistics.

Ohad Rubin and Jonathan Berant. 2021. Smbop: Semi-autoregressive bottom-up semantic parsing. In Proceedings of the 5th Workshop on Structured Prediction for NLP, SPNLP@ACL-IJCNLP 2021, Online, August 6, 2021, pages 12–21. Association for Computational Linguistics.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cícero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pages 13806–13814. AAAI Press.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. CoRR, abs/1710.10903.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. 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, ACL 2020, Online, July 5-10, 2020, pages 7567–7578. Association for Computational Linguistics.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020b. Relational graph attention network for aspect-based sentiment analysis. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020, pages 3229–3238. Association for Computational Linguistics.

Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022, pages 1889–1898. ACM.

Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon J. D. Prince, and Yanshuai Cao. 2021. Optimizing deeper transformers on small datasets. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 2089–2102. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 440–450. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir R. Radev. 2018a. 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, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers), pages 588–594. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. CoRR, abs/1809.08887.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. 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 2019, Hong

Kong, China, November 3-7, 2019, pages 5337–5348. Association for Computational Linguistics.

A Example Appendix

A.1 Details of predefined edge types

All structures have been shown in Table 5. A edge exists from head node $x \in S$ to tail node $y \in S$ if the node pair listed in the Table with the corresponding label.

Head x	Tail y	Edge Type	Description
Q	Q	Question-Question-Dist*	Question item H is at a distance of * before question item T in the input question
		Question-Question-Identity	Question item H is question item T itself
		Question-Question-Generic	Question item H and question item T has no pre-defined relation
		Question-Question-Syntactic Dependency	Question item H has a syntactic dependencies on question item T
Q	T	Question-Table-Exactmatch Question-Table-Partialmatch Question-Table-Nomatch	Question item H is spelled exactly/partially/not the same as table item T
Q	C	Question-Column-Exactmatch Question-Column-Partialmatch Question-Column-Nomatch	Question item H is spelled exactly/partially/not the same as column item T
		Question-Column-Valuematch	Question item H is spelled exactly the same as a value in column item T
T	Q	Table-Question-Exactmatch Table-Question-Partialmatch Table-Question-Nomatch	Table item H is spelled exactly/partially/not the same as question item T
T	T	Table-Table-Generic	Table item H and table item T has no pre-defined relation
		Table-Table-Identity	Table item H is table item T itself
		Table-Table-Fk	At least one column in table item H is a foreign key for certain column in table item T
		Table-Table-Fkr	At least one column in table item T is a foreign key for certain column in table item H
		Table-Table-Fkb	Table item H and T satisfy both "Table-Table-Fk" and "Table-Table-Fkr" relations
T	C	Table-Column-Pk	Column item T is the primary key for table item H
		Table-Column-Has	Column item T belongs to table item H
		Table-Column-Generic	Table item H and column item T has no pre-defined relation
C	Q	Column-Question-Exactmatch Column-Question-Partialmatch Column-Question-Nomatch	Column item H is spelled exactly/partially/not the same as table item T
C	T	Column-Table-Pk	Column item H is the primary key for table item T
		Column-Table-Has	Column item H belongs to table item T
		Column-Table-Generic	Column item H and table item T has no pre-defined relation
C	C	Column-Column-Identity	Column item H is column item T itself
		Column-Column-Sametable	Column item H and column item T are in the same table
		Column-Column-Fk Column-Column-Fkr	Column item H has a forward/reverse foreign key constraint relation with Column item T
		Column-Column-Generic	Column item H and column item T has no pre-defined relation

Table 5: All edge types used in our experiment.