HQE-SQL: HISTORY QUESTION RELATION ENHANCED TEXT2SQL

Zihao Lin & Haibo Xiu
Duke University
{zihao.lin,haibo.xiu}@duke.edu

ABSTRACT

Text2SQL problem aims to efficiently translate natural language questions asked by users to SQL queries, which are executable by database software, and return searched results to users. Previous approaches put more attention on schema linking between tables and question but overlook the relation information between historical and current questions, and thus perform unsatisfying on harder datasets which contain more context-dependent questions. In this work, we propose History Question Relation Enhanced Text2SQL (HQE-SQL) model to explicitly capture the question relation information. In particular, we first propose a new task to predict question relations and then involve the predicted question relations into self-attention mechanism. We conduct experiments on a new Chinese Text2SQL dataset Chase (Guo et al., 2021) and achieve better results compared to one strong baseline.

1 INTRODUCTION

Text2SQL, i.e. translating natural language questions to SQL queries, is one of the most challenging problems of natural language processing (NLP) and is an essential technology for Question-Answering (QA) system. Existing methods mainly focus on converting individual utterances into SQL queries. However, in real conversations, users' current question may depend on their previous questions, which requires model to consider history information and capture the relation information between current and historical questions. For example, as shown in Figure 1, the first question from a user is "Is there any e-book and how much are they?" After getting the answer, next question could be "Which one has the most expensive prime price?" which has a high dependency on the previous question. In order to answer this question, models need to understand that the question asks "which *e-book* has the most expensive *prime price*". The third question "Show the prime prices please." asks the '*prime prices*' of all '*e-book*', which has a strong dependency of the first question instead of the second one. These questions are called context-dependent questions (Yu et al., 2019b).

It can be observed from the example that, in order to better understand the questions, models must capture the relations of current question and history dependent questions including not only the previous question but also long-term historical ones. Meanwhile, to generate correct corresponding SQL queries, relations between current and historical questions must be consistent with the changes of their corresponding SQL schema. However, strong baselines such as RAT-SQL (Wang et al., 2019) and IGSQL (Cai & Wan, 2020) fail to consider question relations. One of the current state-of-the-art algorithm RAT-SQL-TC (Li et al., 2021) only apply question relations into subtasks instead of explicitly incorporating the them into model architecture, which harms the model performance.

To address these issues, we propose **H**istory **Q**uestion Relation **E**nhanced Text2**SQL** (HQE-SQL) model. We maintains the core point of RAT-SQL (Wang et al., 2019), who finds the linking between database schema items and their mentions in questions by a string-based matching method. We firstly introduces an auxiliary question relations prediction task and then incorporates the predicted relations into self-attention mechanism to better understand the questions and generates SQL queries. The auxiliary task is to predict the question relations defined by the changes of SQL queries (Li et al., 2021). Then the predicted question relations are incorporated into self-attention mechanism to enhance our model understand how to use historical information to answer questions currently. These two approaches successfully promote text2SQL generation.

Q1 Chinese: 有电子书嘛？多少钱？Q1 English: Is there any ebook and how much are they?
Query 1: `SELECT T2.book_name, T1.ebook_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id`

Q2 Chinese: 最贵的是哪个呀？ Q2 English: Which one is the most expensive?
Query 2: `SELECT T2.book_name, T1.ebook_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id ORDER BY T1.ebook_price DESC LIMIT 1`

Q3 Chinese: 有会员价么？Q3 English: Show the prime prices please.
Query 3: `SELECT T2.book_name, T1.prime_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id`

Q4 Chinese: 最便宜的是哪本？我想要 Q4 English: Which one is the lowest? I want it.
Query 4: `SELECT T2.book_name, T1.prime_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id ORDER BY T1.prime_price LIMIT 1`

Figure 1: An example conversation from Chase dataset. Each question in Chase is paired with an SQL query. Here we manually translate every Chinese question and the notations in SQL to English for easy understanding. The followed questions after the initial question are context-dependent, e.g. the second question ellipse the 'ebook' and 'price', which requires the model to capture the relation with previous question.

We conducted our experiments on a new context-dependent Chinese Text2SQL benchmark: CHASE (Guo et al., 2021), which is a much harder benchmark due to its higher proportion of context-dependent questions and hard SQL queries compared to other two popular English benchmarks SparC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). Our proposed model HEQ-SQL performs better than its base model RAT-SQL (Wang et al., 2019) while performs worse than the other two strong baselines EditSQL (Zhang et al., 2019), RAT-SQL and IGSQL (Cai & Wan, 2020).

Our contributions are summarized as follows:

- We successfully explicitly incorporate question relations into model architecture to enhance the model to better capture historical information and further promote Text2SQL generation.
- We achieve better results on a new harder context-dependent Chinese Text2SQL dataset, CHASE, compared to one strong baseline.

2 Related Work

Text2SQL Benchmark Datasets Previous works focus more on translating the stand-alone utterances to a database query, e.g. ATIS (Dahl et al., 1994), which is a dataset of recordings and corresponding manual transcripts about flight information in an airline travel inquiry systems. We concluded this kind of datasets as context-independent benchmarks. However, in the real word situation, people tends to ask multiple co-related questions to query the database with a same interest on a particular topic. The questions in the same conversation are across different domains and they should be answered through interactive exchanges. Spider (Yu et al., 2018) is released as the largest cross-domain dataset available in the field. Spider has 10k+ questions querying 200 datasets with multiple tables across 138 different domains. Each question is paired with a SQL query written artificially. SParC (Yu et al., 2019b) is the multi-turn version of Spider and import the context-dependency for the questions. They asked the annotators to produce more coherent follow-up questions for the same topic guided by the selected question from Spider. To increase the data diversity, the follow-up questions may or may not relate to the answer from the original Spider question. The 12k+ questions in SParC are grouped as 4k+ sequences. SparC is looked as the first benchmark dataset for cross-domain context-dependent (XDTS) problem. CoSQL (Yu et al., 2019a) is another version of Spider for building cross-domain querying dialogue systems. It tracks the dialogue states by SQL queries. CoSQL is also widely used in XDTS related works. Thanks to these high quality datasets, many promising methods are seen in recent years and push the performance to new heights. Furthermore, Chase (Guo et al., 2021), which is another newly published dataset, has been proved hard for most existing approaches. We think Chase highlight the key challenges of XDTS for the following reasons: (1) Chase has a high proportion of dependent questions and complex questions. So both the

contextual dependency in the question stream and the linking from text to database schema are hard to learn by previous models. (2) Chase is a pragmatic Chinese dataset. However, most of the models are tuned in English which makes them struggle on Chinese. Chase shows the significant space for the future research. We conducted our experiments on Chase.

Recent Neural Models Most recent models for XDTS take advantages of attentions architectures for question/schema encoding. And the abstract syntax tree based method (Yin & Neubig, 2017) is extensively used to decode the query to intermediate representation. Since the ability of attention mechanism has been widely proven, the primary issue of these models is to extract the contextual coherence from the question stream to databases. Wang et al. (2019) introduced RAT-SQL, a relation-aware transformer mechanism to encode arbitrary relations between question words and schema elements. It explicitly labels the relations and use these relations to modify attention weights. EditSQL is the first to exploit the correlation between sequentially generated queries on different domains (Zhang et al., 2019) which proposes a sequence editing mechanism to modify the previous query on token-level. Another branch of work is using GNN as the encoder. IGSQL (Cai & Wan, 2020) followed the idea from EditSQL and construct a database schema interaction graph encoder to model database schema items together with historical items. The insight is to take the historical mentions of database items into consideration. Even these methods gave many new ways to address the XDTS problem and achieved great success, all of them are performing poorly on Chase dataset as reported by Guo et al. (2021). This is because Chase contains a higher proportion of context-dependent questions which requires model to understand the question relations but above approaches fail to. One of the current SOTA algorithm RAT-SQL-TC (Li et al., 2021) firstly define question relations using the changes of SQL queries. It introduces two subtasks to enhance the model learn the ability to understand question relations. However, it fails to explicitly incorporate question relations into the model architecture. Besides, RAT-SQL-TC only considers the relation of current question and its previous one but fails to consider long-term historical questions which loses some key historical information.

3 Approach

In this section, we will first formally define text2SQL problem. Then we introduce how to construct question relations. Finally we introduce our proposed QRE-SQL model.

3.1 Problem Formalization

Text2SQL aims to translate natural language questions $X = [x_1, x_2, ..., x_n]$ into corresponding SQL queries $y = [y_1, y_2, ..., y_n]$ w.r.t a pre-defined database schema s, where n is the number of context-dependent question in one conversation. We define $s = [s_1, s_2, ..., s_m]$ as all tables and columns in one database where s_i represents a <Table, Column> pairs. Our goal is to predict SQL queries y_i given current and all history questions and database schema, i.e.,

$$\max \prod_{t=1}^{n} P\left(y_t \mid x_1, \cdots, x_t; s\right) \tag{1}$$

3.2 Question Relations

The current utterance asked by the user may omit or explicitly refer to the historical mentioned information. Besides, it's not enough to only look into the previous one question, just as we showed in the real example. To this end, we need to capture the relation a.k.a dependency between current question to all the previous ones. EditSQL (Zhang et al., 2019) firstly used a simple encoder to learn the interaction between questions. Then IGSQL (Cai & Wan, 2020) surpass EditSQL by using a graph encoder. However, they learn the relations implicitly which result in the poor performance on Chase dataset. By just adding a straight forward encoder on the top of tokens, they fail to explicitly incorporate question relations into self-attention mechanism which is the key part of model architectures especially the long-term dependency is easy to learn. We must find a way to firstly explicitly define question relations. Inspired by RAT-SQL-TC (Li et al., 2021), we explicitly build

directly relations between problems to capture the long-term dependency. They construct a prediction task and defined 17 types of operations as 'Turn Switch' to represent the modification made by the incoming question. This idea is similar as EditSQL, since both want to observe the *edition* caused by the question. We ellipse some redundant operations and formally define 5 classes of question relations. These relations and its proportions are showed in Table 1. For example, if the SQL query changes from "SELECT sales" to "SELECT price", the question relation will be classified to Column Change. To be specific, we have the several differences with RAT-SQL-TC:

- RAT-SQL-TC only considered the relation of current question and its previous one, we consider the relation of current question and all previous questions to remember long-term history.
- We apply the question relation into the relation aware self-attention mechanism. RAT-SQL-TC only leveraged the turn switch prediction as an auxiliary task outside of the encoder-decoder model.

Question Relation	Location	Schema Level	Proportion
Column Change	*Select ...*	Column	52.9%
	Group By ...		0.45%
	Order By ...		9.6%
Condition Change	*Where ...*	Domain	11.2%
	... Join ...		0.1%
Display Change	*Limit*	Column	3.3%
	Distinct		11.2%
Domain Change	*From ...*	Domain	18.1%
	Combine Operators[1]		0.2%
Filter Change	*Filter Functions*[2]	Domain	1.2%

Table 1: Description of Question Relations. We define 5 types of question relations to capture what kind of change is caused by the current question. Location means the places that the changes happened. We also define 2 schema levels to understand whether the relation will result the change only in column level or a domain level. Note that the relations are allowed to be combined together with each other, since a pair of questions can have multiple relations.

3.3 METHODOLOGY

Our model adopts an encoder-decoder framework with attention mechanism. Figure 2 shows the architecture of our model.

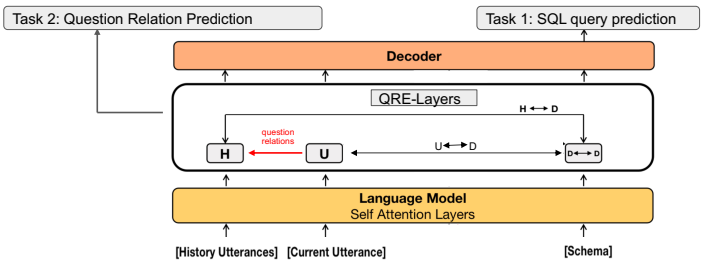

Figure 2: Illustration of the proposed model HQE-SQL.

RAT-SQL is one of the state-of-the-art model for text2SQL task. Concretely, we use a relation-aware transformer-based encoder model to encode input questions into hidden vectors, and use a

[1]Combine operators include: *UNION*, *INTERSECT*, *EXCEPT*
[2]Filter functions include: *COUNT()*, *AVG()*, *SUM()*

decoder model to decode the vectors into abstract syntax tree (AST) which will be transformed to final output SQL queries. Denote a set of inputs $X = \{\boldsymbol{x_i}\}_{i=1}^{n}$ where $\boldsymbol{x_i} \in \mathbb{R}^{d_x}$. Transformer encoder is a stack of self-attention and fully connected layers, introduced by (Vaswani et al., 2017). Each self-attention layer consists of H heads and transforms $\boldsymbol{x_i}$ into $\boldsymbol{y_i} \in \mathbb{R}^{d_x}$. RAT-SQL defines R token-level relations to link questions tokens to tables/ columns and explicitly incorporate the relations into self-attention mechanism as follows:

$$
\begin{aligned}
e_{ij}^{(h)} &= \frac{\boldsymbol{x}_i W_Q^{(h)} \left(\boldsymbol{x_j} W_K^{(h)} + \boldsymbol{r}_{ij}^K\right)^\top}{\sqrt{d_z/H}} \\
\boldsymbol{z}_i^{(h)} &= \sum_{j=1}^{n} \alpha_{ij}^{(h)} \left(\boldsymbol{x_j} W_V^{(h)} + \boldsymbol{r}_{ij}^V\right) \\
\tilde{\boldsymbol{y}}_i &= \text{LayerNorm}\left(\boldsymbol{x}_i + \boldsymbol{z}_i\right) \\
\boldsymbol{y_i} &= \text{LayerNorm}\left(\tilde{\boldsymbol{y}}_i + \text{FC}\left(\text{ReLU}\left(\text{FC}\left(\tilde{\boldsymbol{y}}_i\right)\right)\right)\right)
\end{aligned}
\tag{2}
$$

where FC is a fully-connected layer, LayerNorm is layer normalization (Ba et al., 2016), $1 < h < H$ and $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_x \times (d_x/H)}$. Here, $\boldsymbol{r_{ij}^K} = \boldsymbol{r_{ij}^V}$ is learned embedding for predefined relation of token $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

With no changes of the decoder of RAT-SQL, we extend the encoder by adding a new question relation embedding into self-attention mechanism. However, we cannot directly apply the question relations because they are defined by corresponding SQL queries which are ground-truth. Therefore, we need first to predict the question relations.

Question Relation Prediction (QRP) is an auxiliary task in our proposed model aiming to enhance the encoder to better understand the conversation question relations. This task requires the encoder to predict the question relation between current question and all historical questions. As shown in section 3.1, a total number of $N_T = 5$ types of relations are defined. Same as (Li et al., 2021), for each kind of question relation, we make a binary classification.

Notate $\boldsymbol{t}_i$ as the encoding vector of special token "[SEP]" following the last token of i-th question. The question relation prediction task can be shown as follows:

$$
\begin{aligned}
\mathbf{s}_i &= [\mathbf{t}_{i-1}; \mathbf{t}_i; \mathbf{t}_i - \mathbf{t}_{i-1}; \mathbf{t}_{i-1} * \mathbf{t}_i] \\
p_i^j &= \text{Sigmoid}\left(\mathbf{W}_{QRP}^j \left(\mathbf{s}_i\right)\right) \\
\mathcal{L}_{QRP} &= \sum_{n=1}^{N_T} \sum_{i=1}^{T} \left(\hat{y}_i^j \log p_i^j + \left(1 - \hat{y}_i^j\right) \log\left(1 - p_i^j\right)\right)
\end{aligned}
\tag{3}
$$

where $\boldsymbol{W}_{QRP}^j$ is the parameter matrix to predict the j-th type of question relation. $\hat{y}_i^j \in (0, 1)$ is the label of the j-th question relation with the i-th question and p_i^j is the predicted probability.

HQE-SQL Different from (Li et al., 2021), instead of using the output of encoder's final layer to get $\boldsymbol{t}_i$, we use the output of the $H/2$-th layer to form $\boldsymbol{s}_i$ used to predict question relations. After getting the predicted question relations notated as $\boldsymbol{r}_{q_c q_p}$, where q_c and q_p representing the tokens of current question and historical questions, we explicitly incorporate learned question relation embeddings into the following layers of self-attention mechanism as follows:

$$
\begin{aligned}
e_{ij}^{(h)} &= \frac{\boldsymbol{x}_i W_Q^{(h)} \left(\boldsymbol{x}_j W_K^{(h)} + \boldsymbol{r}_{ij}^K + \boldsymbol{r}_{q_c q_p}^K\right)^\top}{\sqrt{d_z/H}} \\
\boldsymbol{z}_i^{(h)} &= \sum_{j=1}^{n} \alpha_{ij}^{(h)} \left(\boldsymbol{x}_j W_V^{(h)} + \boldsymbol{r}_{ij}^V + \boldsymbol{r}_{q_c q_p}^V\right),
\end{aligned}
\tag{4}
$$

where $H/2 < h <= H$. There are two things to be noted. First, for all tokens in q_c and q_p, the relations embeddings are same. Second, each pair of q_c and q_p may contains multi types of question relations. To handle this situation, we average all the relations between these two questions.

Learning Objective is a combination of decoder loss and question relation prediction loss, shown as follows:

$$\begin{aligned} \mathcal{L}_{dec} &= \sum_{i=1}^{|Y|} y_i \log P\left(y_i \mid y_{<i}, X; s\right) \\ \mathcal{L} &= \mathcal{L}_{dec} + \alpha \mathcal{L}_{TSP} \end{aligned} \tag{5}$$

where $\alpha > 0$. In practice, we set $\alpha = 0.5$.

4 Experiments

4.1 Datasets and Evaluation

Datasets We mainly use Chase (Guo et al., 2021) as our baseline dataset for training and evaluation. There are 2 components Chase-C and Chase-T in Chase. The difference is the questions in Chase-C are naturally in Chinese, but those in Chase-T are from SParC and translated to Chinese. Due to the computation resources and time limitation, we only built our experiments on Chase-C which is harder than Chase-T. Follow the instruction in SParC, we split Chase-C for training, development and testing respectively. The dataset split statics are summarized in Table 2. We also analyse the number of domains and number of questions in the conversation. In average, 4 domains and 4 contextual questions are queried by user in each conversation, which certify the quality of Chase dataset. The databases covered in Train/Development/Test sets are all different which is a good natural to improve the generality of learned translator.

Evaluation We follow the evaluation matrix from Spider benchmark (Yu et al., 2018) who introduced QM (Question Match) and IM (Interaction Match). Question match accuracy is the average exact set match accuracy over all questions, while interaction match accuracy is the average exact set match accuracy over all interactions. To better evaluate the quality on different learned queries, recent works divide the testing SQL queries into 4 levels: easy, medium, hard, extra hard. This definition of difficulty is based on the number of SQL components, selections, and conditions. We follow this division on Chase-C-Test and evaluate the accuracy respectively.

	# Database	# Domain	# Conversation	# Question	Avg@1	Avg@2
Chase-C	120	462	2003	7694	3.98	3.84
Chace-C-Train	80	304	1377	5141	3.96	3.84
Chace-C-Dev	20	78	333	1291	3.88	3.88
Chace-C-Test	20	80	333	1262	4.14	3.79

Table 2: Split Statistics for Chase-C. The split datasets for train, development and test are performed on different databases and domains. To show the domain crossing, we use Avg@1 as the average number of domains covered in one conversation. And for contextual dependency, Avg@2 is the average number of questions asked in one conversation.

4.2 Model and Training Details

We use BERT-large (Devlin et al., 2018) as word embedding layer. The input length is set to 512 and the inputs that exceed this limit are truncated. We set the number of head of Encoder H to be 8, the embedding dimension to be 1024, a feed-forward network dimension tobe 1024, and a dropout probability to be 0.1. In decoder, the number of head is still 8 and the dropout probability is 0.1, but the feed-forward network dimension is set to 256. We use Adam Kingma & Ba (2014) to train our

model. We set linear learning-rate schedule with a warm-up period from 0 to 0.0001 in 2000 steps followed by a 98000 steps long cool-down period from 0.0001 down to 0. All experiments were performed on Google's Cloud Platform, using a N1-standard-8 (8 CPU cores and 30 GB of RAM) VM type and TESLA V100 GPU.

4.3 Results

Observed from table 3, our proposed HQE-SQL performs better compared to its base model RAT-SQL but performs worse than the other two baselines. Specifically, there is a huge improvement compared to RAT-SQL. HQE-SQL beats RAT-SQL on development set by 21.9% for QM and more than 38.9% for IM, and a similar improvement also appears on testing set. According to (Guo et al., 2021), RAT-SQL achieves the best performance on SParC and CoSQL compared to EditSQL and IGSQL. However, RAT-SQL adopts a string-match based method to find the linking between database schema items and their mentions in questions, which fails in Chinese dataset Chase. Therefore, RAT-SQL performs worse than EditSQL and IGSQL on Chase-C. Since our model is built on the base of RAT-SQL, this improvement demonstrates the effectiveness of our proposed method. HQE-SQL aims to capture the historical information instead of the linking information between question tokens and database schema, so the weakness of RAT-SQL still remains a problem which harms the performance. Besides, our proposed method cannot solve the linking problem of RAT-SQL which may be the main reason of underperforming the other two baselines. In the future, we plan to apply our method to these two baselines in the next step to further validate the efficiency of HQE-SQL.

	Chase-C			
	Dev		Test	
	QM	IM	QM	IM
EditSQL	33.6	8.4	32.6	8.7
IGSQL	31.4	10.8	32.6	9.3
RAT-SQL	24.6	5.4	23.9	4.5
HQE-SQL	30.0	7.5	27.4	7.2

Table 3: The accuracy of QM & IM of proposed HQE-SQL and other compared baselines.

5 Analysis

5.1 Context-Dependent Questions

According to Table 4, we can observe that our proposed model HQE-SQL has a high improvement on context-dependency result compared to its baseline RAT-SQL. The proportion of improvement of context-independent questions is less than that of context-dependent questions. This is reasonable because when HQE-SQL encodes context-independent questions, there is no more historical information added into the model, which will have little improvements compared to original RAT-SQL model. Besides, HQE-SQL achieves comparable results on context-dependent questions compared to EditSQL and IGSQL which proves that our proposed method successfully make use of historical relation informations.

	Chase-C			
	Dev		Test	
	Indep.	Dep.	Indep.	Dep.
EditSQL	51.1	26.6	27.1	25.4
IGSQL	45.4	25.7	25.8	25.2
RAT-SQL	35.8	20.0	19.8	20.2
HQE-SQL	37.2	26.1	21.5	25.4

Table 4: The QM on the development/ test set of Chase-C. 'Indep.' and 'Dep.' are short for 'context-independent' and 'context-dependent'.

5.2 Case Study

One nature question is whether the prediction accuracy is improved on context-dependent questions by our model. We consider the same conversation showed in Chapter 1 and compare the results from HQE-SQL with our baseline method RAT-SQL. For the overall exact matching accuracy, both HQE-SQL and RAT-SQL gave the correct answers for the first two questions and failed on the last question. For Q_3, HQE-SQL predicted correctly but RAT-SQL didn't, as the mistake SQL contents in red. RAT-SQL wrongly linked Q_3 with Q_2 and output the MAX 'price' of 'ebook' which is not queried. HQE-SQL precisely understands the dependency and coheres Q_3 with Q_1. With more attention on the history question Q_1, HQE-SQL select the correct column and output the 'prime price' of 'ebook'. However, HQE-SQL still mispredicted the last query. It paid too much attention on the Q_1 but there should be a stronger connection with Q_3. This case also indicates us that, both the relation with recent questions, as well as the relation with old questions, may require to pay attentions. We still have a long step to go towards learning the importance of each question relation.

Q1: 有电子书嘛？多少钱？Is there any ebook and how much are they?
Query 1: `SELECT T2.book_name, T1.ebook_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id`
RAT-SQL: `SELECT book.book_name, ebook.ebook_price FROM ebook JOIN book ON T2.book_id = T1.book_id`
HQE-SQL: `SELECT book.book_name, ebook.ebook_price FROM ebook JOIN book ON T2.book_id = T1.book_id`

Q2: 最贵的是哪个呀？Which one is the most expensive?
Query 2: `SELECT T2.book_name, T1.ebook_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id ORDER BY T1.ebook_price DESC LIMIT 1`
RAT-SQL: `SELECT book.book_name, ebook.ebook_price FROM ebook JOIN book ON T2.book_id = T1.book_id ORDER BY ebook.ebook_price DESC LIMIT 1`
HQE-SQL: `SELECT book.book_name, ebook.ebook_price FROM ebook JOIN book ON T2.book_id = T1.book_id ORDER BY ebook.ebook_price DESC LIMIT 1`

Q3: 有会员价么？Show the prime prices please.
Query 3: `SELECT T2.book_name, T1.prime_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id`
RAT-SQL: `SELECT book.book_name, ebook.prime_price FROM ebook JOIN book ON T2.book_id = T1.book_id WHERE ebook.ebook_price = (SELECT MAX (ebook.price) FROM ebook)`
HQE-SQL: `SELECT book.book_name, ebook.prime_price FROM ebook JOIN book ON T2.book_id = T1.book_id`

Q4: 最便宜的是哪本？我想要 Which one is the lowest? I want it.
Query 4: `SELECT T2.book_name, T1.prime_price FROM ebook AS T1 JOIN book AS T2 ON T2.book_id = T1.book_id ORDER BY T1.prime_price LIMIT 1`
RAT-SQL: `SELECT book.book_name FROM ebook JOIN book ON T2.book_id = T1.book_id WHERE ebook.ebook_price = (SELECT MIN (ebook.price) FROM ebook)`
HQE-SQL: `SELECT book.book_name, ebook.ebook_price FROM ebook JOIN book ON T2.book_id = T1.book_id ORDER BY ebook.ebook_price LIMIT 1`

Figure 3: Case Study for Predicted Query in Figure 1. The questions and queries are translated to English manually. We compared the prediction from RAT-SQL and HQE-SQL to ground truth query separately. The SQL content in red indicated the wrong prediction.

6 Conclusion

Summary In our project, we propose a new model called HQE-SQL on the basis of RAT-SQL. to successfully capture the relation information between current questions and previous questions. We explicitly incorporate the question relations information into self-attention mechanism and a new subtask, which enhance the model to understand current question better and generate more accurate SQL queries. Our model performs better than RAT-SQL which proves the efficiency of our proposed method, but worse than the other two strong baselines: IGSQL and EditSQL because the failure of RAT-SQL fails to link schema and question tokens in Chinese dataset.

Future work Due to the limitation of time, we ignore some ablation studies and analysis and treat them as future work. We need to do ablation study to analyze the efficiency of proposed auxiliary task. Besides, we also need to analyze influence of the number of layer we choose to predict question relations. We will also conduct our model to other RAT-SQL based models. We plan to dive into the learned attention weights to see what the attention distribution in the relation matrix from current question to all the previous ones. Both the relation with recent questions, as well as the relation with old questions, may matter more for current one and require more attentions.

Author Contributions

The two authors both work on writing final project and code writing. The two authors contribute equally in this project.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yitao Cai and Xiaojun Wan. IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6903–6912, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.560. URL https://aclanthology.org/2020.emnlp-main.560.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. URL https://aclanthology.org/H94-1010.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. Chase: A large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL. 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)*, pp. 2316–2331, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.180. URL https://aclanthology.org/2021.acl-long.180.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yuntao Li, Hanchu Zhang, Yutian Li, Sirui Wang, Wei Wu, and Yan Zhang. Pay more attention to history: A context modeling strategy for conversational text-to-sql. *arXiv preprint arXiv:2112.08735*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*, 2019.

Pengcheng Yin and Graham Neubig. 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)*, pp. 440–450, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1041. URL https://aclanthology.org/P17-1041.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*, 2019a.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. Sparc: Cross-domain semantic parsing in context. *arXiv preprint arXiv:1906.02285*, 2019b.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. Editing-based sql query generation for cross-domain context-dependent questions. *arXiv preprint arXiv:1909.00786*, 2019.