CQR-SQL: Conversational Question Reformulation Enhanced Context-Dependent Text-to-SQL Parsers

Dongling Xiao[1*], **Linzheng Chai**[2*†], **Qian-Wen Zhang**[1], **Zhao Yan**[1], **Zhoujun Li**[2] and **Yunbo Cao**[1]
[1]Tencent Cloud Xiaowei
[2]State Key Lab of Software Development Environment,
Beihang University, Beijing, China
[1]{dlxiao,cowenzhang,zhaoyan,yunbocao}@tencent.com
[2]{challenging,lizj}@buaa.edu.cn

Abstract

Context-dependent text-to-SQL is the task of translating multi-turn questions into database-related SQL queries. Existing methods typically focus on making full use of history context or previously predicted SQL for currently SQL parsing, while neglecting to explicitly comprehend the schema and conversational dependency, such as co-reference, ellipsis and user focus change. In this paper, we propose CQR-SQL, which uses auxiliary Conversational Question Reformulation (CQR) learning to explicitly exploit schema and decouple contextual dependency for multi-turn SQL parsing. Specifically, we first present a schema enhanced recursive CQR method to produce domain-relevant self-contained questions. Secondly, we train CQR-SQL models to map the semantics of multi-turn questions and auxiliary self-contained questions into the same latent space through schema grounding consistency task and tree-structured SQL parsing consistency task, which enhances the abilities of SQL parsing by adequately contextual understanding. At the time of writing, our CQR-SQL achieves new state-of-the-art results on two context-dependent text-to-SQL benchmarks SPARC and COSQL.

1 Introduction

The text-to-SQL task is one of the widely followed branches of semantic parsing, which aims to parse natural language questions with a given database into SQL queries. Previous works (Zhong et al., 2017; Yu et al., 2018; Wang et al., 2020) focus on context-independent text-to-SQL task. However, in reality, as users tend to prefer multiple turns interactive queries (Iyyer et al., 2017), the text-to-SQL task based on conversational context is attracting more and more scholarly attention. The generalization challenge of the context-dependent text-to-SQL task lies in jointly representing the multi-turn questions and database schema while considering the contextual dependency and schema structure. As shown in Figure 1, to resolve the contextual dependency, the model should not only understand the *co-reference* and *ellipsis*, but also prevent from irrelevant information integration when *user focus changes*. Recent studies on two large-scale context-dependent datasets, SPARC (Yu et al., 2019b) and COSQL (Yu et al., 2019a), also show the difficulty of this problem. To our knowledge, there is a lack of explicit guidance for mainstream text-to-SQL researches dealing with contextual dependency.

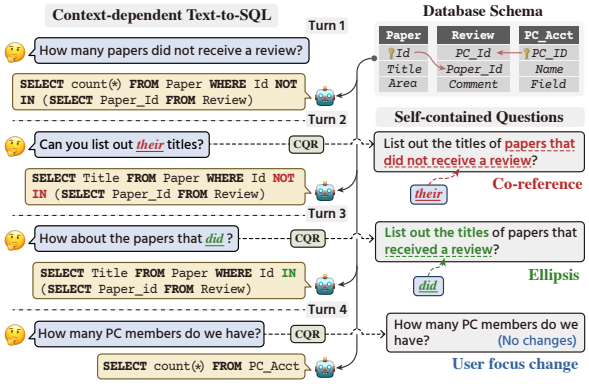

Figure 1: An example of context-dependent Text-to-SQL task demonstrates the phenomenon of co-reference, ellipsis, and user focus changes. The CQR module converts contextual questions to self-contained questions, which can be understood without the context.

For context-dependent text-to-SQL, it is common to train a model in an end-to-end manner that simply encoding the concatenation of the multi-turn questions and schema, as shown in Figure 2(a). To exploit context-dependence information, Hui et al. (2021) propose a dynamic relation decay mechanism to model the dynamic relationships between schema and question as conversation proceeds. Zhang et al. (2019) and Zheng et al. (2022) leverage previously predicted SQL queries to enhance currently SQL parsing. However, we argue that these end-to-end approaches are inadequate guidance for the contextual dependency phe-

* Indicates equal contribution.
† The work was done when Linzheng Chai was doing internship at Tencent Cloud Xiaowei.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2055 - 2068
December 7-11, 2022

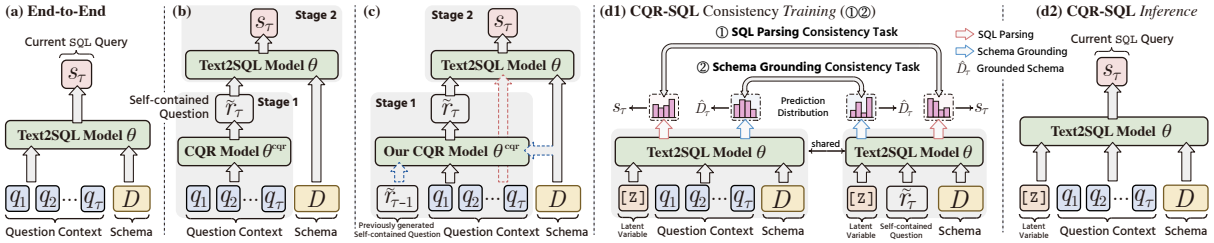

Figure 2: Schematic of (a) **End-to-end** and (b) **Two-stage** pipeline context-dependent text-to-SQL parsing. "Stage 1" in (c) shows the schema enhanced recursive CQR method. (c) A baseline improved on (b) by additionally using question context in "Stage 2", avoiding model from relying only on potentially incorrect $\tilde{r}_\tau$. (d) Our **CQR-SQL**.

nomenon, though they are competitive in their evaluation of existing context modeling methods.

To help the models achieve adequate understanding of the current user question q_τ, conversational question reformulation (CQR) is crucial for multi-turn dialogue systems (Pan et al., 2019; Kim et al., 2021). As far as we know, only few works in contextual-dependent text-to-SQL, such as (Chen et al., 2021), focus on the value of CQR for modeling question context. Chen et al. (2021) propose a two-stage pipeline method in which an CQR model first generates a self-contained question $\tilde{r}_\tau$, and then a context-independent text-to-SQL parser follows, as shown in Figure 2(b). But in practice, the limitations of the two-stage pipeline method are in two aspects: 1) the error propagation from the potentially wrong $\tilde{r}_\tau$ to the single-turn text-to-SQL parser; 2) the neglect of the relevance between the two stages. Besides, CQR for text-to-SQL is more challenging than the general CQR tasks (Pan et al., 2019; Elgohary et al., 2019), since multi-turn questions in text-to-SQL datasets are strictly centered around the underlying database and there are no CQR annotations on existing text-to-SQL datasets.

Motivated by these observations, we propose CQR-SQL, which uses auxiliary CQR to achieve adequately contextual understanding, without suffering from the limitations of two-stage methods. Accordingly, we first introduce an schema enhanced recursive CQR method to product self-contained question data, as in "Stage 1" of Figure 2(c). The design not only integrates the underlying database schema D, but also inherits previous self-contained question $\tilde{r}_{\tau-1}$ to improve the long-range dependency. Secondly, we propose to train model mapping the self-contained questions and the multi-turn question context into the same latent space through **schema grounding consistency task** and **tree-structured SQL parsing consistency task**, as in Figure 2(d1). In this way, to make similar prediction as self-contained question input, models need to pay more attention to the *co-reference* and *ellipsis* when encoding the question context. As shown in Figure 2(d2), during inference, CQR-SQL no longer relies on the self-contained questions from CQR models, thus circumventing the error propagation issue of two-stage pipeline methods.

We evaluated CQR-SQL on SPARC and COSQL datasets, and our main contributions of this work are summarized as follows:

- We present a schema enhanced recursive CQR mechanism that steadily generates self-contained questions for context-dependent text-to-SQL.
- We propose two novel consistency training tasks to achieve adequate contextual understanding for context-dependent SQL parsing by leveraging auxiliary CQR, which circumvents the limitations of two-stage pipeline approaches.
- Experimental results show that CQR-SQL achieves state-of-the-art results on context-dependent text-to-SQL benchmarks, SPARC and COSQL, with abilities of adequate context understanding.

2 Proposed Method

In this section, we first formally define the context-dependent text-to-SQL task and introduce the backbone network of CQR-SQL. Afterwards, the technical details of CQR-SQL are elaborated in two subsections: Schema enhanced recursive CQR and Latent CQR learning for text-to-SQL in context.

2.1 Preliminary

Task Formulation. In context-dependent text-to-SQL tasks, we are given multi-turn user questions $\boldsymbol{q} = \{q_1, q_2, ..., q_n\}$ and the schema $D = \langle T, C \rangle$ of target database which contains a set of tables $T = \{t_1, t_2, ...t_{|T|}\}$ and columns $C_i = \{c_{i1}, c_{i2}, ...c_{i|C_i|}\}, \forall i = 1, 2, ..., |T|$ for the i-th table t_i. Our goal is to generate the target SQL query s_τ with the question context $\boldsymbol{q}_{\leqslant\tau}$ and schema information D at each question turn τ.

Backbone Network. CQR-SQL takes multi-turn questions $\boldsymbol{q}$ as input along with the underlying

database schema D in the *Encoder-Decoder* framework. For **encoder**, CQR-SQL employs the widely used relation-aware Transformer (RAT) encoder (Wang et al., 2020) to jointly represent question and structured schema. For **decoder**, CQR-SQL follows the tree-structured LSTM of Yin and Neubig (2017) to predict the grammar rule of SQL abstract syntax tree (AST), column `id` and table `id` at each decoding step, indicated as APPLYRULE, SELECTCOLUMN and SELECTTABLE (See Appendix A for detailed descriptions).

2.2 Schema Enhanced Recursive CQR

Due to the scarcity of in-domain CQR annotations for context-dependent text-to-SQL, we adopt self-training with schema enhanced recursive CQR method to collect reliable self-contained questions.

Schema Integration for CQR. Multi-turn questions in text-to-SQL are centered around the underlying database. To generate more domain relevant self-contained question r_τ at each turn τ, we concatenate the question context $\boldsymbol{q}_{\leqslant\tau}$with schema D as input $\boldsymbol{x}_\tau = \{q_1, \texttt{[SEP]}, ..., q_\tau, \texttt{[SEP]}, t_1, c_{11}, c_{12}, ..., \texttt{[SEP]}, t_2, c_{21}, c_{22}, ...\}$ for CQR learning.

Recursive Generation for CQR. Inspired by Zhang et al. (2019) and Wang et al. (2021), who verify that integration of previously predicted SQL facilitates modeling long interactions turns, we propose a recursive generation mechanism to recursively inherit context information from previously generated self-contained questions $\tilde{r}_{\tau-1}$ for long-range dependence, as shown in the stage 1 of Figure 2(c). Our CQR at each turn τ is optimized as:

$$\mathcal{L}_\tau^{\text{cqr}} = -\log \mathcal{P}(r_\tau | \{\tilde{r}_{\tau-1}, \texttt{[SEP]}, \boldsymbol{x}_\tau\}). \quad (1)$$

During training, other than using the labeled self-contained questions $r_{\tau-1}$ as $\tilde{r}_{\tau-1}$, we sampled $\tilde{r}_{\tau-1}$ from a pre-trained CQR model to reduce discrepancies between training and inference.

Algorithm 1 Self-training for CQR

Input: Human-labeled in-domain CQR data $\mathcal{D}_0^{\text{cqr}}$.
Output: Full self-contained question data $\mathcal{D}^{\text{cqr}}$ for $\mathcal{D}$.
$l \leftarrow 0$ ▷ Initialize the index of self-training loop.
while $l = 0$ or $|\mathcal{D}_{l-1}^{\text{cqr}}| \neq |\mathcal{D}_l^{\text{cqr}}|$ **do**
  $\theta_{l+1}^{\text{cqr}} \leftarrow \text{TRAINCQR}(\mathcal{D}_l^{\text{cqr}})$
  $\mathcal{D}_{l+1}^{\text{gen}} \leftarrow \text{INFERENCECQR}(\mathcal{D}, \theta_{l+1}^{\text{cqr}})$
  $\mathcal{D}_{l+1}^{\text{cqr}} \leftarrow \text{UNION}(\mathcal{D}_l^{\text{cqr}}, \text{CHECK}(\theta_{\text{SQL}}, \mathcal{D}_{l+1}^{\text{gen}}))$
  ▷ CHECK: Select questions which are self-contained enough for correctly SQL parsing by θ_{SQL} (a pre-trained single-turn text-to-SQL model) in beam search candidates.
  $l \leftarrow l + 1$
$\mathcal{D}^{\text{cqr}} \leftarrow \text{MERGE}(\mathcal{D}_l^{\text{cqr}}, \mathcal{D}_l^{\text{gen}})$ ▷ MERGE: Replace the self-contained questions in $\mathcal{D}_l^{\text{gen}}$ with those in $\mathcal{D}_l^{\text{cqr}}$.
return $\mathcal{D}^{\text{cqr}}$ ▷ Self-contained questions for all interaction turns.

Self-training for CQR. Chen et al. (2021) indicate that models trained with general CQR datasets work poor on the in-domain data from COSQL and SPARC. Besides the annotated in-domain self-contained question data is scarce for all context-dependent text-to-SQL tasks.

We conduct a self-training approach with a pre-trained single-turn text-to-SQL model θ_{SQL} to collect full self-contained question data $\mathcal{D}^{\text{cqr}}$ for text-to-SQL datasets $\mathcal{D}$, as show in Algorithm 1.

2.3 CQR-SQL : Latent CQR Learning for Text-to-SQL Parsing in Context

With the self-contained questions $\mathcal{D}^{\text{cqr}}$ in §2.2, during training, we introduce CQR-SQL, which uses a latent variable `[Z]` to map the semantics of question context and self-contained question into the same latent space with two consistency tasks (schema grounding and SQL parsing), helping models achieve adequately contextual understanding for enhanced SQL parsing during inference.

As shown in Figure 3(a), during training, we input $\text{Seq}(q) = \{\texttt{[Z]}, q, \texttt{[SEP]}, D\}$ to CQR-SQL, where q can be the question context $\boldsymbol{q}_{\leqslant\tau}$ or self-contained questions r_τ.

Schema Grounding Consistency Task. Grounding tables and columns into question context requires adequately understanding the *co-reference* and *ellipsis* in multi-turn questions. Thus we propose using the hidden state $\mathbf{z}$ of latent variable to predict the tables and columns appear in current target SQL query s_τ with bag-of-word (BoW) loss (Zhao et al., 2017), and then enforcing models to make consistent predictions with question context input and self-contained question input, as shown in Figure 3(a). The BoW loss of Schema Grounding task $\mathcal{L}^{\text{SG}_{\text{BoW}}}$ at each turn τ are formulated as:

$$\mathcal{L}_\tau^{\text{SG}_{\text{BoW}}} = \text{BoW}(\boldsymbol{q}_{\leqslant\tau}) + \text{BoW}(r_\tau). \quad (2)$$

$$\begin{aligned}\text{BoW}(q) &= -\log \mathcal{P}(\hat{D}_\tau | \text{Seq}(q)) \\ &= -\sum_{\hat{d}\in\hat{D}_\tau} \log \frac{e^{f_{\hat{d}}(\text{RAT}(\text{Seq}(q))_0)}}{\sum_{d\in D} e^{f_d(\text{RAT}(\text{Seq}(q))_0)}}.\end{aligned} \quad (3)$$

where $\hat{D}_\tau$ refers to the schema appeared in current SQL query s_τ, D indicates the full schema of target database. $\mathcal{P}(\hat{D}_\tau|\cdot)$ represents the schema prediction probability distributions at turn τ. The function $f_d(\mathbf{z}) = \mathbf{h}_d \mathbf{W}_{\text{SG}} \mathbf{z}^\top$, $\mathbf{h}_d$ denotes the final hidden states of schema d for RAT encoder.

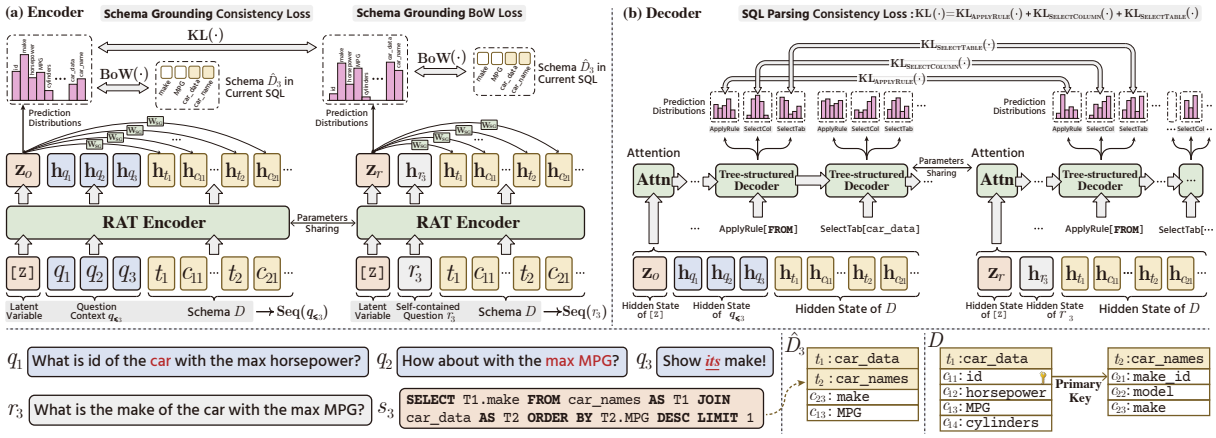

Figure 3: Illustration of the *training stage* for CQR-SQL. (a) Schema grounding task with bag-of-word (BoW) loss and consistency loss. (b) Tree-structured SQL parsing consistency loss at each decoding step. `[Z]` denotes the special symbol of latent variable. RAT Encoder is the relation-aware transformer encoder (Wang et al., 2020) to jointly represent natural language and structured schema. Tree-structured Decoder is the tree-structured LSTM of Yin and Neubig (2017) to predict SQL AST rules, Table `id` and Column `id` at each decoding step.

$\mathbf{z}_o = \text{RAT}(\text{Seq}(\boldsymbol{q}_{\leqslant\tau}))_0$ and $\mathbf{z}_r = \text{RAT}(\text{Seq}(r_\tau))_0$ indicate the final hidden state of the latent variables associated with question context $\boldsymbol{q}_{\leqslant\tau}$ and self-contained question r_τ respectively. The Schema Grounding consistency loss $\mathcal{L}^{\text{SG}_{\text{KL}}}$ is defined as:

$$\begin{aligned}\mathcal{L}_\tau^{\text{SG}_{\text{KL}}} = &\text{KL}\Big(\mathcal{P}(\hat{D}_\tau|\text{Seq}(\boldsymbol{q}_{\leqslant\tau}))\,\|\,\mathcal{P}(\hat{D}_\tau|\text{Seq}(r_\tau))\Big)\\ &+\text{KL}\Big(\mathcal{P}(\hat{D}_\tau|\text{Seq}(r_\tau))\,\|\,\mathcal{P}(\hat{D}_\tau|\text{Seq}(\boldsymbol{q}_{\leqslant\tau}))\Big).\end{aligned} \quad (4)$$

where $\text{KL}(\cdot)$ refers to the Kullback–Leibler divergence between two distributions.

SQL Parsing Consistency Task. Furthermore, to encourage model pay more attention to the SQL logic involving co-reference and ellipsis, we introduce to enforce the model to obtain the consistency prediction of SQL parsing with question contexts and self-contained questions as inputs, at each decoding step. The SQL parsing loss $\mathcal{L}^{\text{SP}}$ and the SQL Parsing consistency loss $\mathcal{L}^{\text{SP}_{\text{KL}}}$, at each turn τ, can be represented as:

$$\mathcal{L}_\tau^{\text{SP}} = -\log\mathcal{P}(s_\tau|\text{Seq}(\boldsymbol{q}_{\leqslant\tau})) - \log\mathcal{P}(s_\tau|\text{Seq}(r_\tau)). \quad (5)$$

$$\begin{aligned}\mathcal{L}_\tau^{\text{SP}_{\text{KL}}} = &\text{KL}(\mathcal{P}(s_\tau|\text{Seq}(\boldsymbol{q}_{\leqslant\tau}))\,\|\,\mathcal{P}(s_\tau|\text{Seq}(r_\tau)))\\ &+\text{KL}(\mathcal{P}(s_\tau|\text{Seq}(r_\tau))\,\|\,\mathcal{P}(s_\tau|\text{Seq}(\boldsymbol{q}_{\leqslant\tau}))).\end{aligned} \quad (6)$$

In this work, we follow the tree-structured decoder of Yin and Neubig (2017), which generates SQL queries as an abstract syntax tree (AST), and conduct three main predictions at each decoding step, including ApplyRule, SelectColumn and SelectTable. We calculate the SQL parsing consistency loss by accumulating all KL divergences of above three predictions as $\text{KL}(\cdot) = \text{KL}_{\text{ApplyRule}}(\cdot) + \text{KL}_{\text{SelectColumn}}(\cdot) + \text{KL}_{\text{SelectTable}}(\cdot)$ at all decoding steps, as shown in Figure 3(b) and further described in Appendix A.3.

Finally we calculate the total training loss $\mathcal{L}_\tau$ at each question turn τ for our context-dependent text-to-SQL model CQR-SQL as:

$$\mathcal{L}_\tau = \mathcal{L}_\tau^{\text{SP}} + \lambda_1\mathcal{L}_\tau^{\text{SG}_{\text{BoW}}} + \underbrace{\lambda_2\big(\mathcal{L}_\tau^{\text{SP}_{\text{KL}}} + \mathcal{L}_\tau^{\text{SG}_{\text{KL}}}\big)}_{\text{Consistency Loss}}. \quad (7)$$

where λ_1 and λ_2 are weights for the schema grounding BoW loss and the consistency loss respectively.

CQR-SQL Inference. Since CQR-SQL has learned to adequately understand the context dependency in question context $\boldsymbol{q}_{\leqslant\tau}$ by distilling representations from self-contained question in two consistency tasks, CQR-SQL no longer relies on self-contained questions and only considers $\text{Seq}(\boldsymbol{q}_{\leqslant\tau})$ as inputs, as shown in Figure 2(d2), thus circumventing the error propagation in two-stage pipeline methods.

3 Experiments

In this section, we conduct several experiments to assess the performance of proposed methods in §2.

3.1 Experimental Setup

CQR Learning. We adopt the Transformer-based encoder-decoder architecture based on the pre-trained ProphetNet (Qi et al., 2020) as the initial CQR model. Since there is no question reformulation annotations in SParC and CoSQL, we annotate 3034 and 1527 user questions as the initial in-domain supervised CQR data $\mathcal{D}_0^{\text{cqr}}$ for SParC and CoSQL respectively. Before self-training, we pre-train a single-turn text-to-SQL model θ_{SQL} based on RAT-SQL (Wang et al., 2020) architecture and Electra (Clark et al., 2020) language

Models (↓) / Datasets (→)	SPARC$_{Dev}$		SPARC$_{Test}$		COSQL$_{Dev}$		COSQL$_{Test}$	
	QM$_{(\%)}$	IM$_{(\%)}$	QM$_{(\%)}$	IM$_{(\%)}$	QM$_{(\%)}$	IM$_{(\%)}$	QM$_{(\%)}$	IM$_{(\%)}$
GAZP + BERT (Zhong et al., 2020)	48.9	29.7	45.9	23.5	42.0	12.3	39.7	12.8
IGSQL + BERT (Cai and Wan, 2020)	50.7	32.5	51.2	29.5	44.1	15.8	42.5	15.0
R^{2}SQL + BERT (Hui et al., 2021)	54.1	35.2	55.8	30.8	45.7	19.5	46.8	17.0
RAT-SQL + BERT (Yu et al., 2021b)	56.8	33.4	-	-	48.4	19.1	-	-
DELTA + BERT$^{\heartsuit}$ (Chen et al., 2021)	58.6	35.6	59.9	31.8	51.7	21.5	50.8	19.7
CQR-SQL + BERT (Ours)	**62.5**	**42.4**	-	-	**53.5**	**24.6**	-	-
RAT-SQL + SCORE$^{\diamondsuit}$ (Yu et al., 2021b)	62.2	42.5	62.4	38.1	52.1	22.0	51.6	21.2
RAT-SQL + TC + GAP$^{\diamondsuit}$ (Li et al., 2021)	64.1	44.1	65.7	43.2	-	-	-	-
PICARD + T5-3B (Scholak et al., 2021)	-	-	-	-	56.9	24.2	54.6	23.7
HIE-SQL + GRAPPA$^{\diamondsuit}$ (Zheng et al., 2022)	64.7	45.0	64.6	42.9	56.4	28.7	53.9	24.6
UNIFIEDSKG + T5-3B (Xie et al., 2022)	61.5	41.9	-	-	54.1	22.8	-	-
RASAT + T5-3B (Qi et al., 2022)	66.7	47.2	-	-	**58.8**	26.3	-	-
CQR-SQL + ELECTRA (Ours)	67.8	48.1	67.3	43.9	58.4	29.4	-	-
CQR-SQL + COCO-LM (Ours)	**68.0**	**48.8**	**68.2**	**44.4**	58.5	**31.1**	**58.3**	**27.4**

Table 1: Performances on the development and test set of SPARC and COSQL. "**QM**" and "**IM**" indicate the exact match accuracy over all questions and all interaction respectively. The models with $^{\diamondsuit}$ mark employ task adaptive pre-trained language models. Models with $^{\heartsuit}$ mark use the general two-stage pipeline approach in Figure 2(b). The "-" results of CQR-SQL are awaiting evaluation due to the submission interval of the leaderboard.

Dataset	# Num of Interactions	#Train/#Dev/#Test	#Average Turn	System Response
SPARC	4,298	3,034 / 422 / 842	3.0	✗
COSQL	3,007	2,164 / 293 / 551	5.2	✓

Table 2: Detailed statistics for SPARC and COSQL.

model for checking whether a generated question is self-contained enough for correctly SQL parsing. During self-training in §2.2, we conduct 3 training loops $\{\theta_1^{\text{cqr}}, \theta_2^{\text{cqr}}, \theta_3^{\text{cqr}}\}$ and obtain 4441 and 1973 supervised CQR data for SPARC and COSQL respectively. Finally, we use the CQR model θ_3^{cqr} in the last training loop to produce the self-contained questions for all interaction turns.

CQR-SQL Training. We conduct experiments on two context-dependent text-to-SQL datasets SPARC and COSQL, the statistic information of them are depicted in Table 2. Following (Cao et al., 2021), we employ RAT-SQL (Wang et al., 2020) architecture and pre-trained ELECTRA (Clark et al., 2020) for all text-to-SQL experiments in this paper. In the training of CQR-SQL, we set hyperparameters $\lambda_1 = 0.1$ and $\lambda_2 = 3.0$ for SPARC, $\lambda_2 = 1.0$ for COSQL (See Appendix B.2 for details), learning rate as $5e$-5, batch size of 32. During inference, we set the beam size to 5 for SQL parsing.

3.2 Experimental Results

As shown in Table 1, CQR-SQL achieves state-of-the-art results cross all settings at the time of writing. With general PLM BERT, CQR-SQL surpasses all previous methods, including the two-stage method DELTA (Chen et al., 2021) which also uses additional text-to-SQL data from Spider. Beside, most of recent advanced methods tend to incorporates more task-adaptive data (`text-table` pairs and synthesized `text-sql` pairs), tailored pre-training tasks (column prediction and turn switch prediction) and super-large PLM T5-3B (Raffel et al., 2020) into training. For this setting, we use general PLM ELECTRA for all text-to-SQL experiments following Cao et al. (2021), and further employ a more compatible[1] PLM COCO-LM (Meng et al., 2021) for comparison. CQR-SQL significantly outperforms SCORE (Yu et al., 2021b), RAT-SQL+TC (Li et al., 2021) and recent HIE-SQL (Zheng et al., 2022) which use task-adaptive pre-trained models. Note that HIE-SQL employs two task-adaptive PLMs for encoding `text-schema` pairs and previous SQL queries respectively. Compared with methods based on super-large T5-3B model (especially RASAT (Qi et al., 2022) which integrates co-reference relations and constrained decoding into T5-3B), CQR-SQL can also achieve significant improvements.

To verify the advantages of CQR-SQL on adequately contextual understanding, we further compare the performances on different interaction turns of SPARC, as shown in Figure 4(a). We observe

[1] COCO-LM is pre-trained on sequence contrastive learning with a dual-encoder architecture (Reimers and Gurevych, 2019), which is compatible for our CQR consistency tasks with dual-encoder for multi-turn $\boldsymbol{q}_{\leqslant\tau}$ and self-contained r_τ.

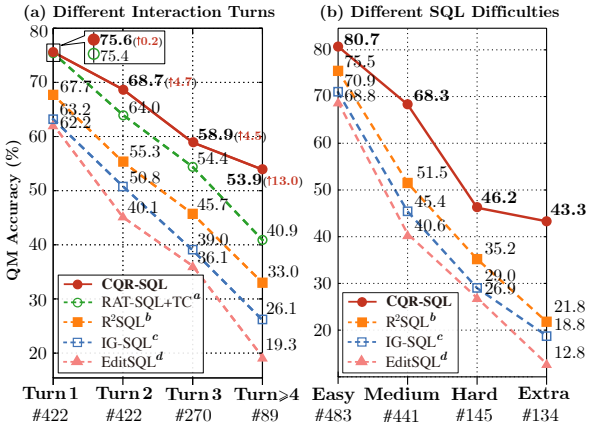

Figure 4: Detailed question match (**QM**) accuracy results in different **interaction turns** and **goal difficulties** on the dev set of SPARC dataset. # Number denotes the number of questions. Detailed results of [a] (Li et al., 2021), [b] (Hui et al., 2021), [c] (Cai and Wan, 2020) and [d] (Zhang et al., 2019) are from the original paper.

that it is more difficult for SQL parsing in longer interaction turns due to the long-range dependency problem, while CQR-SQL achieves more significant improvement as the interaction turn increases. Moreover, in Figure 4(b), we further compare the performances on varying difficulty levels of target SQL queries, CQR-SQL consistently outperforms previous works on all difficulty levels, especially on the "**Extra Hard**" level whose target SQL queries are most complex and usually contain nesting SQL structures (Yu et al., 2018).

3.3 Ablation Study

Regarding the CQR task, as shown in Table 3, recursive generation (RG) achieves 0.46% BLEU score gains on the CQR task for CoSQL dataset which has much longer interaction turns than SPARC as shown in Table 2, while RG fails to significantly improve the performance for $\text{SPARC}_{\text{CQR}}$. This indicates RG can improve CQR performance for longer contextual dependency. While further

Models / Task # Train / # Dev	**SPARC$_{\text{CQR}}$** 3,034 / 422	**CoSQL$_{\text{CQR}}$** 1,527 / 154	**CoSQL** QM / IM
Our CQR	55.75 / **73.80**	**59.87 / 78.82**	**57.8 / 27.7**
– RG	**55.83** / 73.72	59.41 / 78.24	57.3 / 27.3
– RG – SE	54.78 / 72.87	58.93 / 77.64	56.4 / 26.6
CANARD$_{\text{T5 3B}}$	25.57 / 49.52	39.04 / 57.08	53.6 / 23.9

Table 3: Comparisons on the BLEU / Rouge-L scores between schema enhanced (SE) approach and recursive generation (RG) for CQR. Models are trained and evaluated on the initially annotated data $\mathcal{D}_0^{\text{cqr}}$. We observe that the BLEU / Rouge-L scores of $\text{CoSQL}_{\text{CQR}}$ are much higher than those of SPARC, because CoSQL has much more user focus change questions that without co-references and ellipsis (Yu et al., 2019a).

[#] **Datasets** (→) **Models** (↓)	**SPARC$_{\text{Dev}}$** QM$_{(\%)}$	IM$_{(\%)}$	**CoSQL$_{\text{Dev}}$** QM$_{(\%)}$	IM$_{(\%)}$
[1] **CQR-SQL**	**67.8**	**48.1**	**58.4**	**29.4**
[2] −SG	66.3	47.4	57.0	27.0
[3] −SP$_{\text{KL}}$	65.6	46.9	57.3	25.6
[4] −SP$_{\text{KL}}$ −SG	64.9	46.5	56.6	23.9
[5] −SP$_{\text{KL}}$ −SG$_{\text{KL}}$	64.7	45.7	56.1	23.6
Exploratory Study on the Integration of CQR				
[6] CQR$_{\text{Augment}}$	64.5	45.7	54.7	24.2
[7] CQR$_{\text{Two Stage}^\star}$	65.8	46.7	56.8	24.6
[8] CQR$_{\text{Two Stage}}$	62.5	43.1	54.8	23.6
[9] CQR$_{\text{Multi Task}}$	64.9	45.0	56.2	25.3

Table 4: Ablation studies for CQR-SQL and its variants. SG denotes the Schema Grounding task (including BoW loss and consistency loss SG$_{\text{KL}}$), and SP$_{\text{KL}}$ denotes the SQL Parsing consistency task.

removing the schema enhanced SE method, performances decrease by roughly 1% and 0.5% on $\text{SPARC}_{\text{CQR}}$ and $\text{CoSQL}_{\text{CQR}}$ respectively, which verifies the effectiveness of schema integration in CQR for text-to-SQL datasets. We additionally evaluate the performances without any in-domain CQR annotations (fine-tune T5-3B on general CQR dataset CANARD (Elgohary et al., 2019)), and observe that performances on CQR tasks are more significantly reduced than on CoSQL, verifying the effect of in-domain data and the robustness of CQR-SQL against noised CQR information.

In Table 4, we investigate the contribution of each designed choice of proposed CQR-SQL.

[2] : −SG. If removing schema grounding task (including BoW loss and consistency loss), all metrics on SPARC and CoSQL drops by 0.7%-2.4%.

[3] : −SP$_{\text{KL}}$. After removing SQL parsing consistency loss, all metrics drops by 1.1%-3.8%.

[4] : −SP$_{\text{KL}}$ −SG. If removing both schema grounding task (BoW loss and consistency loss) and SQL parsing consistency loss, CQR-SQL degenerates to a variant trained in the general "**End-to-End**" manner as shown in Figure 2(a), and its performances decrease by 1.7%-5.5%.

[5]: −SP$_{\text{KL}}$ −SG$_{\text{KL}}$. To study the impact of schema grounding BoW loss, we train End-to-End variants [4] with BoW loss and find that merely integrating BoW loss slightly degrades the performances, because grounding schema in context is much more difficult than in single questions. While combining BoW with consistency loss to distill contextual knowledge from self-contained questions can improve the performances ([4]→[3]).

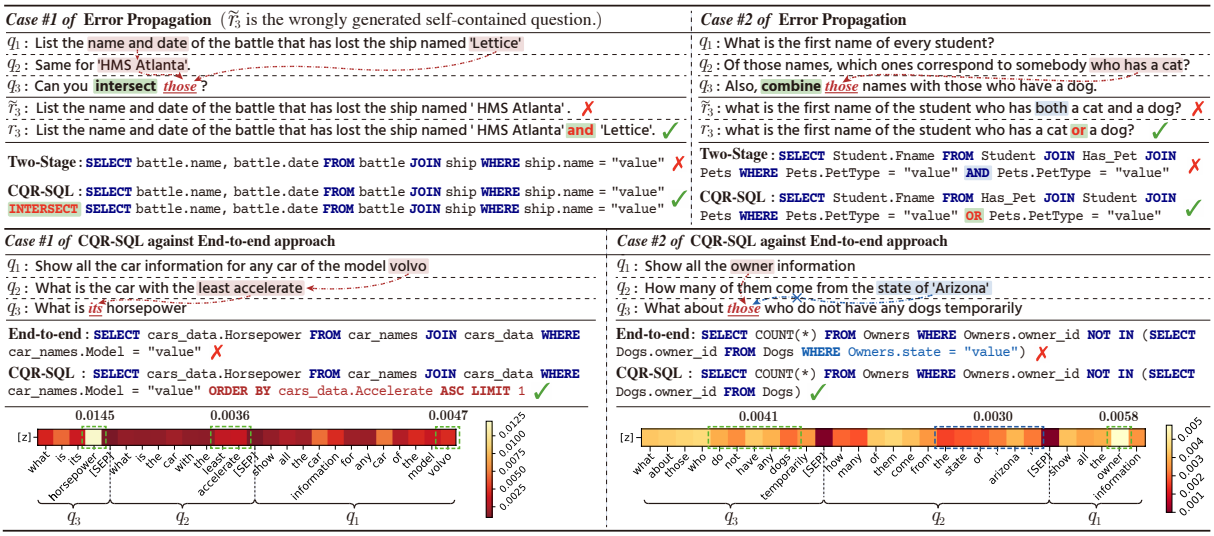

Figure 5: Case studies on SPARC dataset. Upper block shows the cases of error propagation with incorrectly generated self-contained questions $\tilde{r}_\tau$ for **Two-Stage** pipeline methods (as in Figure 2(c) or [7] in Table 3). Cases in the lower block show that **End-to-End** method (as in Figure 2(a) or [4] in Table 4) fails to resolve the conversational dependency. Besides, the heat maps represent the visualization of attention scores of the latent variable [Z] on the question context part. Number on each dotted box is the average attention scores of that box.

[6]: $\text{CQR}_{\text{Augment}}$. We directly augment the full self-contained question data $\mathcal{D}^{\text{cqr}}$ to [4] as single-turn text-to-SQL data augmentation. We find $\text{CQR}_{\text{Augment}}$ degrades the performances because models are trained more on single-turn text-to-SQL, while weaken the abilities of contextual understanding for multi-turn text-to-SQL.

[7]: $\text{CQR}_{\text{Two Stage}^\star}$. A variant trained in the improved "**Two-Stage**" manner as in Figure 2(c). It slightly outperforms the End-to-End variant [4].

[8]: $\text{CQR}_{\text{Two Stage}}$. A "Two-Stage" variant without additionally integrating question context into "Stage 2" as variant [7]. The performances significantly decline for the error propagation issue.

[9]: $\text{CQR}_{\text{Multi Task}}$. A variant jointly trained on CQR task and text-to-SQL task with a shared RAT encoder and two task-specified decoders, as shown in Figure 6. We observe that $\text{CQR}_{\text{Multi Task}}$ slightly decreases the performances compared with End-to-End variant [4] for the optimization gap between the `text-to-text` optimization and structured `text-to-SQL` optimization.

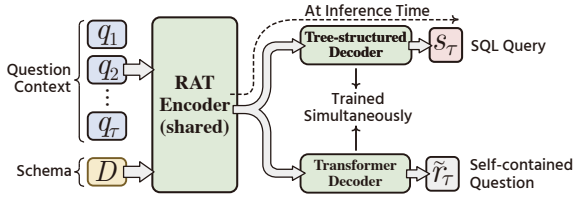

Figure 6: Schematic of $\text{CQR}_{\text{Multi Task}}$ variant.

Above results and analyses demonstrate the advantages of CQR-SQL on leveraging self-contained questions to enhance the abilities of adequately context understanding for contextual text-to-SQL parsing, meanwhile circumventing the error propagation of Two-Stage pipeline methods.

3.4 Case Study

As shown in the upper block of Figure 5, we compare the SQL queries by CQR-SQL with those by the Two-Stage baseline model in Figure 2(c) or [7] in Table 3, to show the error propagation phenomenon between CQR stage and text-to-SQL stage. The generated self-contained questions $\tilde{r}_\tau$ are incorrect, leading to wrong predictions of SQL queries. Specifically, In the first case, CQR model fails to understand *"intersect"* in current question q_3, thus missing *"Lettice"* in the generated self-contained question $\tilde{r}_3$ and leading to uncompleted SQL queries in the text-to-SQL stage. In the second case, CQR model misunderstood *"combine"* in the the current question q_3, leading to incorrect key word "AND" in the predicted SQL query.

In the lower block of Figure 5, we compare the SQL queries by CQR-SQL with those by the End-to-End baseline, and visualize the attention patterns of latent variable [Z] on question context. The first case shows the scenario which requires model to inherit history information to resolve the *co-reference* and *ellipsis*. From the heat map, we observe that latent variable [Z] pays more attention to the desirable history information "*volvo*" and "*least accelerate*". While the second case shows the scenario which requires model to discard confusion history

information. In this case, latent variable [Z] pays less attention to the confusion information block "*state of 'Arizona'*" compared with the desirable ones, which is benefit for correctly SQL parsing.

3.5 Transferability on Contextual Text-to-SQL

To verify the transferability of CQR integration on contextual text-to-SQL task, we conduct three *out-of-distribution* experiments as shown in Table 5. In experiment [1], our CQR-SQL has better transferability on contextualized questions (Turn $\geqslant 2$) in CoSQL$_{\text{Dev}}$, which contains additional system response and more question turns compared with SPARC dataset as in Table 2. Beside, in experiment [2-3], CQR-SQL achieves consistently better performances on out-of-distribution contextual questions (Turn $\geqslant 3$). These results indicate the advantage of CQR-SQL in robustness contextual understanding for *out-of-distribution* scenarios.

[#]	Train	Eval	End-to-End	Two-Stage	CQR-SQL
[1]	$\mathbf{S}^{\text{All}}_{\text{Train}}$	$\mathbf{C}^{\text{Turn}\geqslant 2}_{\text{Dev}}$	$36.3_{(+0.0)}$	$37.3_{(+1.0)}$	$40.9_{(+\mathbf{4.6})}$
[2]	$\mathbf{S}^{\text{Turn}\leqslant 2}_{\text{Train}}$	$\mathbf{S}^{\text{Turn}\geqslant 3}_{\text{Dev}}$	$41.5_{(+0.0)}$	$42.6_{(+1.1)}$	$45.2_{(+\mathbf{3.7})}$
[3]	$\mathbf{C}^{\text{Turn}\leqslant 2}_{\text{Train}}$	$\mathbf{C}^{\text{Turn}\geqslant 3}_{\text{Dev}}$	$42.4_{(+0.0)}$	$43.2_{(+0.8)}$	$46.7_{(+\mathbf{4.3})}$

Table 5: Question match (QM) accuracy results of the *out-of-distribution* experiments. **S** and **C** denote SPARC and CoSQL datasets respectively. **Experiment** [1]: training models on the training set of SPARC$_{\text{Train}}$, while evaluating them on the questions with context (Turn $\geqslant 2$) in CoSQL$_{\text{Dev}}$. **Experiment** [2-3]: training models on the questions at Turn $\leqslant 2$, whereas evaluating them on the questions at Turn $\geqslant 3$.

4 Related Work

Text-to-SQL. Spider (Yu et al., 2018) is a well-known cross-domain context-independent text-to-SQL task that has attracted considerable attention. Diverse approaches, such as RAT-SQL (Wang et al., 2020), BRIDGE (Lin et al., 2020) and LGESQL (Cao et al., 2021), have been successful on this task. Recently, with the widespread popularity of dialogue systems and the public availability of SPARC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a) datasets, context-dependent text-to-SQL has drawn more attention. Zhang et al. (2019) and Wang et al. (2021) use previously generated SQL queries to improve the quality of SQL parsing. Cai and Wan (2020) and Hui et al. (2021) employ graph neural network to model contextual questions and schema. Jain and Lapata (2021) use a memory matrix to keep track of contextual information. Chen et al. (2021) decouple context-dependent text-to-SQL task to CQR and context-independent text-to-SQL tasks. Besides, Yu et al. (2021a,b) propose task-adaptive conversational pre-trained model for SQL parsing, and Scholak et al. (2021) simply constrain auto-regressive decoders of super-large T5-3B for SQL parsing. In this work, we leverage reformulated self-contained questions in two consistency tasks to enhance contextual dependency understanding for multi-turn text-to-SQL parsing, without suffering from the error propagation of two-stage pipeline methods.

Conversational Question Reformulation (CQR) aims to use question context to complete ellipsis and co-references in the current questions. Most works adopt the encoder-decoder architecture with only contextual text as input (Elgohary et al., 2019; Pan et al., 2019). Besides, CQR are applied to several downstream tasks for enhanced context understanding, such as conversational question answer (CQA) (Kim et al., 2021) and conversational passage retrieval (CPR) (Dalton et al., 2020). Regrading CQR training for text-to-SQL, We present a recursive CQR method to address long-range dependency and incorporate schema to generate more domain-relevant and semantic-reliable self-contained questions.

Consistency Training. To improve model robustness, consistency training (Zheng et al., 2016) has been widely explored in natural language processing by regularizing model predictions to be invariant to small perturbations. The small perturbations can be random or adversarial noise (Miyato et al., 2018) and data augmentation (Zheng et al., 2021). Inspired by consistency training, ExCorD (Kim et al., 2021) trains a classification CQA model that encourage the models to predict similar answers span from the rewritten and original questions. Different from ExCorD, we combine latent variable with schema grounding consistency task and tree-structured SQL generation consistency task to force model pay more attention to the *co-references* and *ellipsis* in question context.

5 Conclusions

We propose CQR-SQL, a novel context-dependent text-to-SQL approach that explicitly comprehends the schema and conversational dependency through latent CQR learning. The method introduces a schema enhanced recursive generation mechanism to generate domain-relative self-contained questions, then trains models to map the semantics of self-contained questions and multi-turn question

context into the same latent space with schema grounding consistency task and SQL parsing consistency task for adequately context understanding. Experimental results show that CQR-SQL achieves new state-of-the-art results on two classical context-dependent text-to-SQL datasets SPARC and COSQL.

6 Limitations

Compared to End-to-End approaches as shown in Figure 2(a), proposed CQR-SQL requires more computational cost and GPU memory consumption at each training step, with duel-encoder for question context and self-contained question inputs. Specifically, with batchsize of 32 and 8 V100 GPU cards, CQR-SQL takes 310.7 seconds to train an epoch on SPARC dataset, while End-to-End approache just costs 179.6 seconds. While compared with previous advanced methods using T5-3B PLM (Scholak et al., 2021; Xie et al., 2022; Qi et al., 2022), and multiple task-adaptive PLMs (Zheng et al., 2022), CQR-SQL is much more computationally efficient.

Acknowledgments

We are indebted to the EMNLP2022 reviewers for their detailed and insightful comments on our work. This work was supported in part by the National Natural Science Foundation of China (Grant No. 62276017), and the 2022 Tencent Big Travel Rhino-Bird Special Research Program.

References

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

Yitao Cai and Xiaojun Wan. 2020. 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)*, pages 6903–6912, Online. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 2541–2555, Online. Association for Computational Linguistics.

Zhi Chen, Lu Chen, Hanqi Li, Ruisheng Cao, Da Ma, Mengyue Wu, and Kai Yu. 2021. Decoupled dialogue modeling and semantic parsing for multi-turn text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3063–3074, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jeffrey Dalton, Chenyan Xiong, Vaibhav Kumar, and Jamie Callan. 2020. Cast-19: A dataset for conversational information seeking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1985–1988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. 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)*, pages 5918–5924, Hong Kong, China. Association for Computational Linguistics.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13116–13124.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Parag Jain and Mirella Lapata. 2021. Memory-based semantic parsing. *Transactions of the Association for Computational Linguistics*, 9:1197–1212.

Gangwoo Kim, Hyunjae Kim, Jungsoo Park, and Jaewoo Kang. 2021. Learn to resolve conversational dependency: A consistency training framework for conversational question answering. 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), pages 6130–6141, Online. Association for Computational Linguistics.

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

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Yu Meng, Chenyan Xiong, Payal Bajaj, saurabh tiwary, Paul Bennett, Jiawei Han, and XIA SONG. 2021. Coco-lm: Correcting and contrasting text sequences for language model pretraining. In *Advances in Neural Information Processing Systems*, volume 34, pages 23102–23114. Curran Associates, Inc.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Zhufeng Pan, Kun Bai, Yan Wang, Lianqiang Zhou, and Xiaojiang Liu. 2019. Improving open-domain dialogue systems via multi-turn incomplete utterance restoration. 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)*, pages 1824–1833, Hong Kong, China. Association for Computational Linguistics.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. In *arXiv:2205.06983*.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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*, pages 7567–7578, Online. Association for Computational Linguistics.

Run-Ze Wang, Zhen-Hua Ling, Jingbo Zhou, and Yu Hu. 2021. Tracking interaction states for multi-turn text-to-sql semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

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 (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021a. GraPPa: grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. 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), pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021b. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. 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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir 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)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

Bo Zheng, Li Dong, Shaohan Huang, Wenhui Wang, Zewen Chi, Saksham Singhal, Wanxiang Che, Ting Liu, Xia Song, and Furu Wei. 2021. Consistency regularization for cross-lingual fine-tuning. 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)*, pages 3403–3417, Online. Association for Computational Linguistics.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488.

Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. HIE-SQL: History information enhanced network for context-dependent text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics 2022*, Online. Association for Computational Linguistics.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6869–6882, Online. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Backbone Architecture

A.1 Encoder: Relation-aware Transformer

Relation-aware Transformer (RAT) (Wang et al., 2020) is an extension to Transformer (Vaswani et al., 2017) to consider preexisting pairwise relational features between the inputs. For text-to-SQL task, pairwise relational features include the intra-relation of databse schema D and question-schema alignment information. Formally, given input sequence $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$, we obtain the initial representations through pre-trained language model $\mathbf{H}^0 = \mathrm{PLM}(\boldsymbol{x}) = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$. Then L stacked RAT blocks compute the final hidden states $\mathbf{H}^L = \mathrm{RAT}(\boldsymbol{x})$ via $\mathbf{H}^l = \mathrm{RAT}_l(\mathbf{H}^{l-1}), l \in [1, L]$ where $\mathrm{RAT}_l(\cdot)$ is calculated as:

$$
\begin{aligned}
&\mathbf{Q} = \mathbf{H}^{l-1}\mathbf{W}_Q^l, \mathbf{K} = \mathbf{H}^{l-1}\mathbf{W}_K^l, \mathbf{V} = \mathbf{H}^{l-1}\mathbf{W}_V^l \\
&e_{ij} = \underset{j}{\mathrm{Softmax}}\left(\frac{\mathbf{Q}_i(\mathbf{K}_j + \mathbf{r}_{ij}^K)^\top}{\sqrt{d_k}}\right) \\
&\mathbf{A}_i = \sum_{j=1}^{n} e_{ij}\left(\mathbf{V}_j + \mathbf{r}_{ij}^V\right) \\
&\tilde{\mathbf{A}} = \mathrm{LayerNorm}(\mathbf{H}^{l-1} + \mathbf{A}) \\
&\mathbf{H}^l = \mathrm{LayerNorm}\left(\tilde{\mathbf{A}} + \mathrm{FC}(\mathrm{ReLU}(\mathrm{FC}(\tilde{\mathbf{A}})))\right).
\end{aligned}
\tag{8}
$$

where parameters $\mathbf{W}_Q^l, \mathbf{W}_K^l, \mathbf{W}_V^l \in \mathbb{R}^{d_h \times d_k}$ project $\mathbf{H}^l$ to queries, keys and values. Embedding $\mathbf{r}_{ij}$ represents the relationship between token x_i and x_j. $\mathrm{LayerNorm}(\cdot)$ is the layer normalization (Ba et al., 2016), $\mathrm{FC}(\cdot)$ is the full connected layer.

In CQR-SQL, the relation type between latent variable `[Z]` and schema is NoMatch (Wang et al., 2020).

A.2 Decoder: Tree-structured LSTM

We employ a single layer tree-structured LSTM decoder of Yin and Neubig (2017) to generate the abstract syntax tree (AST) of SQL queries in depth-first, left-to-right order. At each decoding step, the prediction is either 1) ApplyRule action that expands the last non-terminal into a AST grammar rule; or 2) SelectColumn or SelectTable action that chooses a column or table from schema to complete last terminal node.

Formally, given the final encoder hidden states of Question, Table and Column $\mathbf{H}^L = \{\mathbf{H}_q, \mathbf{H}_t, \mathbf{H}_c\}$. The tree-structured decoder is required to generate a sequence of actions a_t to construct the AST which can transfer to standard SQL query s, represented as $\mathcal{P}(s|\mathbf{H}^L) = \prod_t \mathcal{P}\left(a_t|a_{<t}, \mathbf{H}^L\right)$. We adopt a single layer LSTM to produce action sequence, the LSTM states are updated as $\mathbf{c}_t, \mathbf{h}_t = \mathrm{LSTM}\left([\mathbf{a}_{t-1}; \mathbf{h}_{p_t}; \mathbf{a}_{p_t}; \mathbf{n}_{f_t}], \mathbf{c}_{t-1}, \mathbf{h}_{t-1}\right)$ where $[\cdot]$ is the concatenate operation, $\mathbf{c}_t$ is the LSTM cell state, $\mathbf{h}_t$ is the LSTM output hidden state, $\mathbf{a}_t$ is the action embedding, p_t is the decoding step of the current parent AST node and $\mathbf{n}_{f_t}$ is the embedding of current node type. We initialize the LSTM hidden state $\mathbf{h}_0$ via attention pooling over the final encoder hidden state $\mathbf{H}^L$ as:

$$
\begin{aligned}
e_i &= \underset{n}{\underset{i}{\mathrm{Softmax}}}\left(\tilde{\mathbf{h}}_0 \mathrm{Tanh}\left(\mathbf{H}_i^L \mathbf{W}_1\right)^\top\right) \\
\mathbf{A} &= \sum_{i=1}^{n} e_i \mathbf{H}_i^L \\
\mathbf{h}_0 &= \mathrm{Tanh}\left(\mathbf{A}\mathbf{W}_2\right).
\end{aligned}
\tag{9}
$$

where $\tilde{\mathbf{h}}_0$ is initial attention vector, $\mathbf{W}_1, \mathbf{W}_2$ are projection parameters. At each decoding step t, we employ $\mathrm{MultiHeadAttention}$ (Vaswani et al., 2017) on $\mathbf{h}_t$ over $\mathbf{H}^L$ to compute context representation $\mathbf{h}_t^{\mathrm{ctx}}$.

For the prediction of ApplyRule actions, the prediction distribution is computed as:

$$
\begin{aligned}
&\mathcal{P}(a_t = \textsc{ApplyRule}[R]|a_{<t}, \mathbf{H}^L) \\
&\quad = \underset{R}{\mathrm{Softmax}}\left(\mathrm{MLP}_2([\mathbf{h}_t; \mathbf{h}_t^{\mathrm{ctx}}])\mathbf{W}_R\right).
\end{aligned}
\tag{10}
$$

where MLP_2 denotes 2-layer MLP with a Tanh nonlinearity, $\mathbf{W}_R$ is the classification matrice.

For the prediction of SelectTable actions, the prediction distribution is computed as:

$$
\begin{aligned}
&\mathcal{P}(a_t = \textsc{SelectTable}[i]|a_{<t}, \mathbf{H}^L) \\
&\quad = \underset{i}{\mathrm{Softmax}}\left(\mathrm{MLP}_2([\mathbf{h}_t; \mathbf{h}_t^{\mathrm{ctx}}])\mathbf{W}_t\mathbf{H}_{t_i}^\top\right).
\end{aligned}
\tag{11}
$$

where $\mathbf{H}_{t_i}$ denotes the encoder hidden state of the i-th Table. The prediction of SelectColumn actions is similar to SelectTable.

A.3 SQL Parsing Consistency Loss

Given the final encoder hidden states $\mathbf{H}_q^L = \mathrm{RAT}(\mathrm{Seq}\left(\boldsymbol{q}_{\leqslant\tau}\right))$ and $\mathbf{H}_r^L = \mathrm{RAT}\left(\mathrm{Seq}(r_\tau)\right)$ for question context and self-contained question input respectively, the SQL parsing consistency loss $\mathcal{L}^{\mathrm{SP_{KL}}(t)}$ at each decoding step t is computed as:

$$
\begin{aligned}
\mathcal{L}^{\mathrm{SP_{KL}}(t)} &= \mathcal{L}^{\mathrm{SP_{KL}}(t)}_{\textsc{ApplyRule}} + \mathcal{L}^{\mathrm{SP_{KL}}(t)}_{\textsc{SelectTable}} \\
&\quad + \mathcal{L}^{\mathrm{SP_{KL}}(t)}_{\textsc{SelectColumn}}. \\
\mathcal{L}^{\mathrm{SP_{KL}}(t)}_{\textsc{ApplyRule}} &= \mathrm{KL}\big(\mathcal{P}\left(a_t = \textsc{ApplyRule}|a_{<t}, \mathbf{H}_q^L\right) \\
&\quad \| \mathcal{P}\left(a_t = \textsc{ApplyRule}|a_{<t}, \mathbf{H}_r^L\right)\big). \\
\mathcal{L}^{\mathrm{SP_{KL}}(t)}_{\textsc{SelectTable}} &= \mathrm{KL}\big(\mathcal{P}\left(a_t = \textsc{SelectTable}|a_{<t}, \mathbf{H}_q^L\right) \\
&\quad \| \mathcal{P}\left(a_t = \textsc{SelectTable}|a_{<t}, \mathbf{H}_r^L\right)\big).
\end{aligned}
\tag{12}
$$

The total SQL parsing consistency loss is computed as $\mathcal{L}^{\mathrm{SP_{KL}}} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}^{\mathrm{SP_{KL}}(t)}$, where T denotes the length of action sequence for target SQL AST.

B Experiment Details

B.1 Detailed Hyperparameters

We implement CQR-SQL based on the PyTorch framework[2] and use 8 Nvidia Tesla V100 32GB GPU cards for all the experiments. Firstly, we trained CQR model with a learning rate of $3e$-5 and batch size of 32. We use the maximum input sequence length as 512 and the maximum epochs as 25. We adopt label smooth method with ratio 0.15 for regularization. During inference for CQR, we set the beam size as 5.

Regarding training CQR-SQL, the number of heads is 8 and hidden size of RAT encoder is 1024, the dropout rates of encoder and decoder are 0.1 and 0.2 respectively. For pre-trained ELECTRA, we adopt layer-wise learning rate decay with coefficient 0.8 for robust optimization. We train CQR-SQL on SPARC and COSQL with max training epochs to be 300 and 350 respectively.

B.2 Impact of Weight λ_2 for Consistency Loss

We vary the weight λ_2 of consistency loss in $\{1.0, 2.0, 3.0, 4.0\}$ and train CQR-SQL on SPARC and COSQL datasets, as shown in Table 6. We observe that COSQL task desires less CQR knowledge (best choice of λ_2 is 1.0) compare with SPARC (best choice of λ_2 is 3.0), because COSQL dataset contains much more user focus change questions than SPARC, which do not need to be reformulated (Yu et al., 2019b).

Weight λ_2	SPARC$_{\text{Dev}}$ QM	SPARC$_{\text{Dev}}$ IM	COSQL$_{\text{Dev}}$ QM	COSQL$_{\text{Dev}}$ IM
$\lambda_2 = 1.0$	66.5	47.2	**58.2**	**29.4**
$\lambda_2 = 2.0$	67.1	47.6	58.0	28.3
$\lambda_2 = 3.0$	**67.8**	**48.1**	57.4	27.3
$\lambda_2 = 4.0$	66.0	46.7	56.7	26.6

Table 6: Results of CQR-SQL on SPARC and COSQL datasets with different weights λ_2 of consistency loss.

B.3 Detailed Results on COSQL Task

As shown in Table 7, we report the detailed results in different question turns and SQL difficulty levels on the development set of COSQL dataset. We observe that CQR-SQL achieves more significant improvement as the interaction turn increases, and consistently outperforms previous works on all SQL difficulty levels.

[2] https://pytorch.org/

Models	Turn 1 # 293	Turn 2 # 285	Turn 3 # 244	Turn 4 # 114	Turn >4 # 71
EditSQL [a]	50.0	36.7	34.8	43.0	23.9
IGSQL [b]	53.1	42.6	39.3	43.0	31.0
IST-SQL [c]	56.2	41.0	41.0	41.2	26.8
SCORE [d]	60.8	53.0	47.5	49.1	32.4
CQR-SQL	**66.2**	**60.0**	**54.5**	**54.4**	**39.4**

Models	Easy # 483	Medium # 441	Hard # 145	Extra # 134
EditSQL [a]	62.7	29.4	22.8	9.3
IGSQL [b]	66.3	35.6	26.4	10.3
IST-SQL [c]	66.0	36.2	27.8	10.3
CQR-SQL	**76.7**	**55.9**	**39.9**	**22.4**

Table 7: Detailed QM results in different interaction turns and goal difficulties on the development set of COSQL dataset. Detailed results of [a] (Zhang et al., 2019),[b] (Cai and Wan, 2020),[c] (Wang et al., 2021) and [d] (Yu et al., 2021b) are from the original paper.

B.4 Effects of CQR Integration with Different PLMs

To further study the effects of CQR integration for contextual text-to-SQL task, we train models in End-to-End, Two-Stage and CQR-SQL approaches based on different pre-trained language models (PLMs), as shown in Table 8. We can see that: 1) CQR-SQL method consistently preforms better than Two-Stage and End-to-End methods, further demonstrating the effectiveness of CQR-SQL for adequate contextual understanding. 2) COCO-LM (Meng et al., 2021) is superior to ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2019). We argue the reason is that COCO-LM is pre-trained on sequence contrastive learning with a dual-encoder architecture (Reimers and Gurevych, 2019), which is compatible for our CQR consistency tasks with dual-encoder for question context $\boldsymbol{q}_{\leqslant\tau}$ and self-contained question r_τ as inputs.

PLMs	Methods	SPARC$_{\text{Dev}}$ QM	SPARC$_{\text{Dev}}$ IM	COSQL$_{\text{Dev}}$ QM	COSQL$_{\text{Dev}}$ IM
BERT	End-to-End	58.6	38.2	50.7	20.5
	Two-Stage	60.1	39.3	51.1	22.2
	CQR-SQL	**62.5**	**42.4**	**53.5**	**24.6**
ELECTRA	End-to-End	64.9	46.5	56.6	23.9
	Two-Stage	65.8	46.7	56.8	24.6
	CQR-SQL	**67.8**	**48.1**	**58.2**	**29.4**
COCO-LM	End-to-End	65.6	45.5	57.1	25.9
	Two-Stage	66.0	46.5	57.8	26.3
	CQR-SQL	**68.0**	**48.8**	**58.5**	**31.1**

Table 8: Results of End-to-End, Two-Stage and CQR-SQL methods with different PLMs.

B.5 More Cases

In this section, we show more cases of error propagation with Two-Stage pipeline method, and CQR-SQL against End-to-End baseline models.

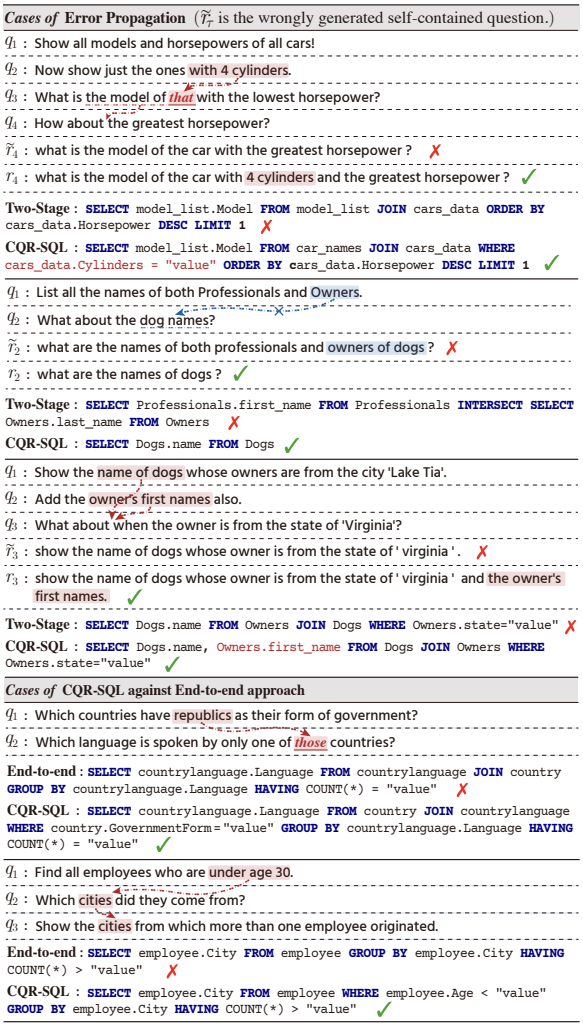

Figure 7: Cases on SPARC dataset. Upper block shows the cases of error propagation with incorrectly generated self-contained questions $\tilde{r}_\tau$ for **Two-Stage** pipeline methods (as in Figure 2(c) or [7] in Table 3). Cases in the lower block show that **End-to-End** method (as in Figure 2(a) or [4] in Table 4) fails to resolve the conversational dependency.