Exploring the Compositional Generalization in Context Dependent
Text-to-SQL Parsing

Aiwei Liu∗, Wei Liu∗, Xuming Hu, Shu’ang Li, Fukun Ma,
Yawen Yang, Lijie Wen†
Tsinghua University
{liuaw20,liu-w21,hxm19,lisa18,mafk19,yyw19}@mails.tsinghua.edu.cn
wenlj@tsinghua.edu.cn

Abstract

In the context-dependent Text-to-SQL task, the generated SQL statements are refined iteratively based on the user input utterance from each interaction. The input text from each interaction can be viewed as component modifications to the previous SQL statements, which could be further extracted as the modification patterns. Since these modification patterns could also be combined with other SQL statements, the models are supposed to have the compositional generalization to these novel combinations. This work is the first exploration of compositional generalization in context-dependent Text-to-SQL scenarios. To facilitate related studies, we constructed two challenging benchmarks named CoSQL-CG and SPARC-CG by recombining the modification patterns and existing SQL statements. The following experiments show that all current models struggle on our proposed benchmarks. Furthermore, we found that better aligning the previous SQL statements with the input utterance could give models better compositional generalization ability. Based on these observations, we propose a method named p-align to improve the compositional generalization of Text-to-SQL models. Further experiments validate the effectiveness of our method. Source code and data are available †.

1 Introduction

Recently, the poor generalization of semantic parsing models to out-of-distribution samples is under increasing attention (Keysers et al., 2020; ?). These examples are usually obtained by recombining existing structures. For example, in the SCAN dataset (Lake and Baroni, 2018a), models may fail to parse “jump twice and walk” even though “jump twice” and “walk” could be parsed successfully. The ability to generalize to novel combinations is also known as compositional generalization. Text-to-SQL (Yu et al., 2018) allows non-expert users to access the information from the database by converting the user input text into SQL statements executed in the database. As a typical semantic parsing task, the study of its compositional generalization is of great importance.

Existing works explore the compositional generalization of Text-to-SQL only in the scenario that precisely maps stand-alone utterances to SQL queries. Shaw et al. (2021) define the atom and compound for SQL statements and propose the TMCD split to repartition the dataset. Gan et al. (2022) annotate the alignment of sub-sentence and sub-SQL in the spider dataset (Yu et al., 2018) and then recombine these sub-SQLs and sub-sentences. In these settings, the SQL statements and user questions in the constructed test split tend to be much more complex. However, it is difficult for users to express complex queries in a stand-alone sentence. In real scenarios, users often start with a simple query and continuously combine additional query conditions with subsequent questions.

In this work, we focus on the study of compositional generalization in context-dependent Text-to-SQL tasks, which is more natural and applica-

---

†https://github.com/THU-BPM/CD-Text2SQL-CG
‡Equally Contributed.
††Corresponding author.
ble. In the context-dependent Text-to-SQL task (Yu et al., 2019b), the generated SQL statements are refined based on the user input text during each interaction. The input text from each interaction can be viewed as component modifications to the previous SQL statement, which could be further extracted as the modification patterns. Since these modification patterns could also be combined with other SQL statements, the models are supposed to have the compositional generalization to these novel combinations. For example, in Figure 1, the modifications and the queries of the first turn in the training phrase could be re-combined in the inference phrase. Applicable models are supposed to successfully parse these novel combinations.

To better investigate compositional generalization in the context-dependent Text-to-SQL, we first construct compositional generalization benchmarks based on the existing datasets. First, we extract the modification patterns from the training dataset and then recombine them with the existing SQL statements in the development set. Note that in the compositional generalization setting, only the recombination results not existing in the training set are kept. To generate the corresponding utterances, we use a semi-automatic approach. The utterances are initially generated by a pre-trained model fine-tuned on the training data, and then reviewed and verified by human experts. As a result, we create two benchmarks, CoSQL-CG and SparC-CG, specifically for the datasets CoSQL(Yu et al., 2019a) and SparC(Yu et al., 2019b). Our experiments reveal that current state-of-the-art models perform poorly on these benchmarks, emphasizing the significance of enhancing compositional generalization capabilities.

We further explore how to improve the compositional generalization in context-dependent Text-to-SQL tasks. Inspired by the previous works to improve compositional generalization by fine-grained alignment of inputs and outputs (Zheng and Lapata, 2022; Akyürek and Andreas, 2021), we propose a method to better align the current text with the previous SQL statements. We follow the common practice of most competitive Text-to-SQL models which take the concatenation of all utterances as input. Specifically, our proposed p-alignment method extracts the embedding of the text from each interaction after the encoding process and then decodes them into the corresponding SQL statements separately. Further experiment results show that our p-align method could effectively improve the compositional generalization of current models, which also demonstrates that better alignment of text and SQL statements and the introduction of previous SQL statements are of great importance.

To summarize, the main contributions of our paper are as follows:

• To the best of our knowledge, we are the first to explore compositional generalization in context-dependent Text-to-SQL.

• We construct two benchmarks named CoSQL-CG and SparC-CG to better facilitate the relevant research.

• We propose a simple and effective method named p-alignment to improve the compositional generalization ability of models.

2 Related Work

2.1 Context dependent Text-to-SQL

Most current research on Text-to-SQL is conducted under the context-independent setting, with many recent methods achieving excellent results on the Spider dataset (Yu et al., 2018), including graph-based methods such as LGESQL(Cao et al., 2021a), RAT-SQL (Wang et al., 2020) and IESLSQL (Liu et al., 2022a), as well as sequence-to-sequence-based methods like PICARD (Scholak et al., 2021). Recently, with the presentation of two datasets CoSQL(Yu et al., 2019a) and SparC(Yu et al., 2019b), the Text-to-SQL parsing under the context-dependent setting has attracted much attention, which is more realistic and applicable. Subsequently, various methods have been proposed. Among them, SCoRe(Yu et al., 2021) and STAR(Cai et al., 2022) aim to train better pre-trained models to improve the parsing ability of models. Also, many sequence-to-sequence methods based on T5 pre-trained model like PICARD (Scholak et al., 2021) and RASAT (Qi et al., 2022) have achieved great success. Meanwhile, more methods pay more attention to contextual information or conversation history during encoding, including IGSQ(Li and Wan, 2020), HIE-SQL(Zheng et al., 2022), and IST-SQL(Wang et al., 2021). Meanwhile, other rewriting-based methods like DELTA(Chen et al., 2021) and CQRSQL(Xiao et al., 2022) reformulate the current and the historical texts into an individual sentence. Different from the previous works, we mainly focus...
on exploring compositional generalization under context-dependent text-to-SQL settings.

2.2 Compositional Generalization

Compositional Generalization is an important metric for evaluating the robustness of the model (Liu et al., 2022b) in the field of natural language processing. For semantic parsing tasks, the ability to generalize to structures generated by systematically combining known atomic components is of vital importance. Lake and Baroni (2018b) propose the SCAN dataset, which maps word sequences into navigation command sequences (e.g., jump twice \( \rightarrow \) JUMP JUMP). Their training/evaluation split are constructed in a compositional generalization way. Keysers et al. (2020), introduce CFQ dataset and propose distribution-based compositionality assessment to measure compositional generalization. Hupkes et al. (2020) summerize five different compositionally generalization splits and combine them to generate PCFG SET. Many works focus on improving the compositional generalization of models. This is usually achieved by introducing more detailed lexicon or lexicon-style alignments (Zheng and Lapata, 2022; Akyürek and Andreas, 2021) or adopting a grammar-based decoder (Herzig and Berant, 2021; Qiu et al., 2022b; Guo et al., 2020). Another line of work attempts to synthesize examples utilizing grammar and generative models for data augmentation (Qiu et al., 2022a; Andreas, 2020; Jia and Liang, 2016).

Recently, the compositional generalization of Text-to-SQL parsing has gained more and more interest. Shaw et al. (2021) define the atom and compound for SQL statements and propose the TMCD split to repartition the dataset. Gan et al. (2022) annotate the alignment of sub-sentence and sub-SQL in the spider dataset (Yu et al., 2018) and then recombine these sub-SQLs and sub-sentences. The above works only focus on the Text-to-SQL parsing in the context-independent setting, which precisely maps stand-alone utterances to SQL queries. However, it is difficult for users to express complex queries in a stand-alone sentence. In this work, we first explore the compositional generalization for context-dependent Text-to-SQL Parsing.

3 Compositional Generalization in Context-dependent Text-to-SQL

To facilitate the understanding of the following sections, we provide a more detailed explanation of compositional generalization in context-dependent Text-to-SQL parsing in this section.

The template split is a typical compositional generalization setting, where the structure templates in the training and test set are completely separated. Our compositional generalization scenario can be viewed as an extension of the template split, where the combination of basic SQL templates and modification templates in the training and test set are separated. Note that basic SQL and modification templates in the test set all appear in the training set individually. For instance, in figure 1, in the inference phrase, although all the templates are seen during training, their combinations are novel.

From another point of view, our compositional generalization scenario could also be viewed as a special case of TMCD split (Shaw et al., 2021), where the SQL templates and modification templates could be seen as atoms and their combination results are the compounds. Note the utterance to the SQL templates (first atom) are provided during training, which could be further utilized to improve the compositional generalization (Section 5).

4 Benchmark construction

Since there are few data satisfying the compositional generalization setting in the origin SPARC and CoSQL development set. We first construct new benchmarks to facilitate the related research. As illustrated in Figure 2, the benchmark construction process can be divided into four steps. The first step is to filter out context-independent examples; next, modification patterns are extracted from the remaining examples; after that, these modification patterns are combined with other SQL statements, and finally, corresponding utterances are generated.

4.1 Filter out context-independent examples

It is observed that a significant number of examples in the SPARC or CoSQL datasets are context-independent, meaning that no context information is needed to generate the current queries. In this work, we propose a schema-linking-based method to filter out these context-independent examples.

Schema linking is a common technique in Text-to-SQL which links the exact or partial occurrences of the column/table names in the question such as the ARTLINES and Abbreviation in Figure 2(a). Our main motivation is that if current data is context-dependent, there are some column/table
names not linked to the current question but linked to history questions (context), such as the first example in Figure 2(a). Specifically, the schemas in the target query are represented as S. We use the n-gram matching method to find occurrences S in the current question, where the matched schemas could be represented as Sc. Similarly, the matched schemas in the history questions are represented as Sb. The current example is context-dependent only if Sb − Sc = ∅. Finally, we keep 4270 and 2347 context-dependent examples in SPARC and CoSQL training set respectively.

4.2 Generate Modification Pattern

After filtering out context-independent data, the next step is to generate modification patterns from the remaining context-dependent examples.

As shown in Figure 2(b), we first parse current and previous SQL statements into abstract syntax trees and then compare the tree structures to get the modified components. Specifically, a top-down traversal algorithm is adopted to find the different nodes. The nodes along with their children constitute the modified component. Then the generated modification component is anonymized to generate the modification template. Finally, we generate 409 and 191 modification templates for SPARC and CoSQL respectively.

4.3 Re-combine SQL statements

With the generated modification patterns, the next step is to re-combine these patterns with other SQL statements to generate new SQL statements.

First, modification patterns are filled with new table/column names sampled from target database schemas to generate new modifications. Then the modifications are directly combined with the other SQL statements. Note that in the previous modification pattern generation process, the relationship of the schema is kept (e.g. primary key and foreign key relationships) and the table/column name sampling results must conform to the above relationship constraints. As mentioned in Section 3, the combination process requires that the base SQL templates and modification templates are all shown in the training set but their combinations are novel. Finally, we generate 5958 and 2594 combination results in SparC and CoSQL respectively.

4.4 Utterance generation

The final step of our benchmark construction is to generate the context-dependent utterance for the generated SQL statements. Since pre-trained language models have shown great ability in text generation, we first utilize a fine-tuned T5 model (Rafel et al., 2020) to generate the context-dependent utterance. More specifically, the input to the T5 model is the concatenation of the modification, previous SQL statement, and previous utterance.

For the utterance generated by the T5 model may be noisy, we further invite human experts to filter and revise the generated data. The first task of human experts is to remove SQL statements that don’t fit realistic scenarios. For example, the statement SELECT Count(loser_entry) FROM matches ORDER BY matches.winner_age is invalid because the function Count() and the clause ORDER BY usually do not appear together. The second task of the human experts is to revise the utterances generated by the T5 model as shown in Figure 2(d). To ensure annotation consistency, we introduce two experts to double-check the annotated results. Finally, after the filtering and revising process, we get 372 and 267 questions for SPARC.
and CoSQL datasets respectively, which further construct our SPARC-CG and CoSQL-CG benchmarks. More detailed statistics of the benchmarks will be described in the experiment section.

5 Methods

After constructing the SPARC-CG and CoSQL-CG, we further explore how to improve the compositional generalization in context-dependent Text-to-SQL parsing. According to the previous works (Zheng and Lapata, 2022; Akyürek and Andreas, 2021), the key to improving the compositional generalization is to construct better component alignment between inputs and outputs. In the context-dependent Text-to-SQL settings, the utterance-query pair of previous interactions could be utilized to align input utterances and output queries. Based on this motivation, we propose p–align to improve the compositional generalization of existing Text-to-SQL models. Note that our method follows the common practice of most competitive Text-to-SQL models which take the concatenation of all utterances as input.

Specifically, given the input utterances $X = [X_1, X_2, ..., X_n]$ at the n-th interaction, where $X_n = [x_1, ..., x_j]$ is an utterance with j words, the encoder aims to generate embeddings for each word such that $X = H(X)$. In the origin decoding process, the result query $y$ could be represented as an action sequence $[a_1, ...a_T]$ and the whole decoding process could be represented as the product of

$$\prod_{i=1}^{T} p(a_t \mid \{a_1, \ldots, a_{t-1}\}, X).$$

(1)

In our p–align method, the utterance embeddings of each interaction are extracted to decode the corresponding SQL statements. As shown in Figure 3, the decoder process of our p–align could be represented as:

$$\sum_{i=1}^{n} \prod_{t=1}^{T_i} p(a_t^i \mid \{a_1^i, \ldots, a_{t-1}^i\}, X_{\leq t}).$$

(2)

In this way, our p–align method aligns corresponding parts of the input utterance to previous queries and thus improves the compositional generalization ability of models.

6 Experiment

In this section, we first perform more detailed statistics on our constructed SPARC-CG and CoSQL-CG. Then we further analyze our benchmarks with current competitive Text-to-SQL models. Finally, several experiments are conducted to verify the effectiveness of our p–align method.

6.1 Benchmark statistics

The detailed statistics of SPARC-CG and CoSQL-CG are shown in Table 1. We mainly count three metrics here: # Question, # Non-CG Questions, and # CG Questions, where # Question is the total

<table>
<thead>
<tr>
<th></th>
<th># Questions</th>
<th># Non-CG Questions</th>
<th># CG Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARC</td>
<td>1625</td>
<td>491</td>
<td>31</td>
</tr>
<tr>
<td>SPARC-CG</td>
<td>921</td>
<td>491</td>
<td>14</td>
</tr>
<tr>
<td>CoSQL</td>
<td>1300</td>
<td>207</td>
<td>372</td>
</tr>
<tr>
<td>CoSQL-CG</td>
<td>471</td>
<td>207</td>
<td>167</td>
</tr>
</tbody>
</table>

Table 1: The detailed statistics of SPARC-CG and CoSQL-CG benchmark.

Figure 4: Distributions of different modification patterns in SPARC-CG and CoSQL-CG benchmark.
Table 2: Question match accuracy of current competitive models on three different benchmarks: Dev, Non-CG, and CG. For all the models, we adopt the given parameters.

<table>
<thead>
<tr>
<th>Methods / Datasets</th>
<th>SPARC (Dev)</th>
<th>SPARC (Non-CG)</th>
<th>SPARC (CG)</th>
<th>CoSQL (Dev)</th>
<th>CoSQL (Non-CG)</th>
<th>CoSQL (CG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIC (CONCAT) + BERT-Large (Liu et al., 2020)</td>
<td>55.3</td>
<td>63.4</td>
<td>18.9</td>
<td>45.2</td>
<td>52.3</td>
<td>13.3</td>
</tr>
<tr>
<td>SPIC (TURN) + BERT-Large (Liu et al., 2020)</td>
<td>54.6</td>
<td>62.1</td>
<td>18.2</td>
<td>44.8</td>
<td>51.3</td>
<td>12.2</td>
</tr>
<tr>
<td>SPIC (GATE) + BERT-Large (Liu et al., 2020)</td>
<td>54.3</td>
<td>62.4</td>
<td>17.3</td>
<td>44.2</td>
<td>51.8</td>
<td>12.4</td>
</tr>
<tr>
<td>RAT-SQL + SCoRe (Yu et al., 2021)</td>
<td>60.4</td>
<td>69.6</td>
<td>22.4</td>
<td>52.1</td>
<td>55.6</td>
<td>20.4</td>
</tr>
<tr>
<td>LGESQL + ELECTRA-Large (Cao et al., 2021b)</td>
<td>65.0</td>
<td>73.4</td>
<td>25.3</td>
<td>54.4</td>
<td>62.4</td>
<td>21.0</td>
</tr>
<tr>
<td>LGESQL + STAR (Cai et al., 2022)</td>
<td>66.9</td>
<td>75.4</td>
<td>25.8</td>
<td>59.7</td>
<td>68.4</td>
<td>26.3</td>
</tr>
<tr>
<td>PICARD + T5-3B (Scholak et al., 2021)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>56.9</td>
<td>58.1</td>
<td>21.5</td>
</tr>
<tr>
<td>RASAT + T5-3B (Qi et al., 2022)</td>
<td>66.7</td>
<td>75.8</td>
<td>22.0</td>
<td>58.8</td>
<td>67.9</td>
<td>20.4</td>
</tr>
</tbody>
</table>

Evaluating a large portion of the data in the SPARC and CoSQL datasets is context-independent or has no context, which makes the sum of # Non-CG Questions and # CG Questions relatively small.

We present the components distributions of modification patterns in SPARC-CG and CoSQL-CG in Figure 4. The most common component in modification patterns is where. Orderby and groupby also take a large proportion. There are also many modification patterns that include multiple components, such as where-groupby and where-orderby. Finally, the distributions of modification patterns in SPARC-CG and CoSQL-CG are similar, which illustrates our benchmark construction’s consistency. Note that the select components are not counted, as they are included in almost all modifications.

6.2 Experiment Setup

Models. We adopt many current competitive Text-to-SQL models to explore the impact of compositional generalization. SPIC (Liu et al., 2020) is a simple model which explores different methods to incorporate context questions, where SPIC (CONCAT) concatenates context questions with current questions, SPIC (TURN) employs a turn-level encoder to capture the inter-dependencies among questions in different turns and SPIC (GATE) uses a gate mechanism to compute the importance of each question. SCORE and STAR (Cai et al., 2022) are two specialized pre-trained models for RAT-SQL and LGESQL (Cao et al., 2021b) respectively. PICARD (Scholak et al., 2021) and RASAT (Qi et al., 2022) are two seq2seq based models based on pre-trained T5 model (Raffel et al., 2020).

Evaluation Metric. We mainly use the question match (QM) (Yu et al., 2019b) as our evaluation metric, which is the exact set matching score (Yu et al., 2018) over all questions. The exact set matching score decomposes predicted queries into SQL components such as SELECT and WHERE and then computes scores for each component. For each model, we report the QM on the origin SPARC/CoSQL development set as well as the Non-CG and CG benchmarks. Note that the interaction match (Yu et al., 2019b) is not reported in our paper because we are only interested in the scores of the model on questions satisfying the compositional generalization condition.

6.3 Evaluation on SPARC-CG/CoSQL-CG

We report the question match accuracy on SPARC and CoSQL datasets under three benchmarks: Dev, Non-CG, and CG in Table 2.

Based on the above results, we summarize the following observations. (1) The accuracy of all models significantly decreases under the compositional generalization setting. Specifically, the QM on SPARC-CG and CoSQL-CG decreases 39.3 and 33.6 on average compared to the origin development set, which indicates current models lack the compositional generalization ability. (2) The models perform better on the Non-CG benchmarks than the origin development set (8.4 and 6.5 on average for SPARC and CoSQL respectively), which demonstrates that in-domain data are easily generalized. (3) CONCAT could better incorporate context...
questions than TURN and GATE. Therefore, our p-align is only designed for the CONCAT method.

(4) The grammar tree-based decoder (LGESQL) and the larger language model (T5-3B) could help improve the compositional generalization ability.

### 6.4 Detailed Evaluation

**Evaluation at Different Levels of Difficulty.** The SQL queries could be divided into four difficulty levels based on the complexity of SQL statements: easy, medium, hard and extra hard. To better demonstrate the performance in the compositional generalization setting, we conduct further evaluations on different levels of difficulties. As shown in Figure 5a-b, the STAR model performs worse on the CG benchmark than on the original development set at all difficulties, which further indicates the model’s compositional generalization ability requires further improvement. Meanwhile, there is an obvious improvement in the Non-CG benchmark compared to the original development set.

**Evaluation at Different Turns.** We further illustrate the question match accuracy on three benchmarks with the increase of conversation turns in Figure 5c-d. The accuracy decreases sharply on the CG benchmark and the original development set while staying stable on the non-CG benchmark. This suggests that the compositional generalization ability of models decreases with the increase of conversation turns.

**Evaluation on different components.** To better investigate the poor performance of the current competitive models under the compositional generalization setting, we further report the question match accuracy on different detailed SQL components in Table 3. The reported results are the average results over STAR and RASAT on three benchmarks of SPARC. As demonstrated in the table, nearly all components’ accuracy significantly decreases under the compositional generalization setting, which illustrates the impact of compositional generalization on the models is balanced.

**6.5 Evaluation of p-align method**

Table 4 shows the results of different models with & without p-align on three benchmarks of SPARC and CoSQL. We choose $SPiC \text{(CONCAT)} + \text{BERT-Base}$, $SPiC \text{(CONCAT)} + \text{BERT-Large}$ and $LGESQL + \text{ELECTRA-Large}$ as our base models because other models are either customized pre-trained models (STAR and SCoRE) or with a too large model(T5-3B). All the hyperparameters are the same as the original model.

Overall, our p-align method significantly improves the performance of the model on the CG benchmarks, with an average improvement of 3.2 and 2.3 on the SPARCC-CG and CoSQL-CG benchmarks respectively. While the improvement on DEV and Non-CG benchmarks is relatively small, at 0.77 and 0.35 on average respectively, this suggests that our method is particularly effective in compositional generalization settings. These results support our hypothesis that improving alignment between utterances and queries can enhance the model’s compositional generalization abilities, and should be considered as a potential direction for future research.

<table>
<thead>
<tr>
<th>SQL Components</th>
<th>DEV</th>
<th>Non-CG</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>84.6</td>
<td>88.2</td>
<td>60.2</td>
</tr>
<tr>
<td>SELECT (no AGG)</td>
<td>86.3</td>
<td>89.3</td>
<td>62.9</td>
</tr>
<tr>
<td>WHERE</td>
<td>80.6</td>
<td>91.8</td>
<td>62.5</td>
</tr>
<tr>
<td>WHERE (no OP)</td>
<td>85.1</td>
<td>95.3</td>
<td>69.2</td>
</tr>
<tr>
<td>GROUP BY (no HAVING)</td>
<td>81.1</td>
<td>85.7</td>
<td>66.4</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>76.9</td>
<td>81.6</td>
<td>54.5</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>78.2</td>
<td>82.0</td>
<td>58.3</td>
</tr>
<tr>
<td>AND/OR</td>
<td>99.0</td>
<td>99.3</td>
<td>91.2</td>
</tr>
<tr>
<td>KEYWORDS</td>
<td>86.3</td>
<td>92.8</td>
<td>67.1</td>
</tr>
</tbody>
</table>

Table 3: Accuracy on the different SQL components. The reported results are the average results over STAR and RASAT on three benchmarks of SPARC.

<table>
<thead>
<tr>
<th>Methods</th>
<th>DEV</th>
<th>Non-CG</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPiC (CONCAT) + BERT-Base</td>
<td>47.6</td>
<td>53.5</td>
<td>8.9</td>
</tr>
<tr>
<td>w. p-align</td>
<td>50.6</td>
<td>54.1</td>
<td>16.4(7.5)</td>
</tr>
<tr>
<td>SPiC (CONCAT) + BERT-Large</td>
<td>55.3</td>
<td>63.4</td>
<td>19.5</td>
</tr>
<tr>
<td>w. p-align</td>
<td>56.1</td>
<td>63.8</td>
<td>20.6(1.1)</td>
</tr>
<tr>
<td>LGESQL + ELECTRA-Large</td>
<td>65.0</td>
<td>73.4</td>
<td>25.3</td>
</tr>
<tr>
<td>w. p-align</td>
<td>64.8</td>
<td>73.0</td>
<td>26.2(0.9)</td>
</tr>
</tbody>
</table>

Table 4: The results of different models w. & w/o p-align on three benchmarks of SPARC and CoSQL.

<table>
<thead>
<tr>
<th>Error component</th>
<th>STAR</th>
<th>RASAT</th>
<th>LGESQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Info</td>
<td>24</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>Modification Info</td>
<td>149</td>
<td>136</td>
<td>139</td>
</tr>
<tr>
<td>Context &amp; Modification Info</td>
<td>112</td>
<td>128</td>
<td>127</td>
</tr>
</tbody>
</table>

Table 5: Statistical analysis of different error types on SPARC-CG benchmark.
To support further research in this area, we construct two benchmarks named SPARC-CG and CoSQL-CG composed of out-of-domain scenarios. The presented case study categorizes three scenarios where current models make incorrect predictions, which include: failing to consider contextual information, inability to interpret modifications, and failing to understand both modifications and context. We further conduct statistical analysis on the SPAR-C-G benchmark in Table 5 and found that the majority of errors occur when models cannot interpret modifications. Additionally, when models neglect context, they also tend to misinterpret modifications. Interestingly, the proportion of errors for the different models evaluated in the study is quite similar, indicating that the compositional generalization challenges faced by these models are consistent across them.

### 7 Conclusion

In this study, we conduct the first exploration of compositional generalization in context-dependent Text-to-SQL scenarios. To support further research in this area, we construct two benchmarks named SPARC-CG and CoSQL-CG composed of out-of-domain scenarios.

### 6.6 Error analysis

To evaluate the compositional generalization ability of current models, we selected four incorrect prediction results from the SPARC-CG benchmark. For each example, we provided the context, the current question, the correct query, and the prediction results from STAR and RASAT.

As illustrated in Figure 6, in the first two scenarios, the models struggle to accurately interpret the changes brought about by current questions, despite maintaining a grasp of the context information. Conversely, in the third case, the models are able to interpret the modifications of the current question, but fail to take into account the context information. The fourth case represents the worst-case scenario, with the models unable to correctly parse either the modifications or the context information. Note that the incorrect results predicted by both models in the first three cases are similar, indicating that the failure of the current models to perform well in a compositional generalization setting is a widespread issue, not an isolated incident.
distribution examples. Additionally, we introduce the p-align method to enhance the compositional generalization capabilities of existing models. Further experiments show that current models perform poorly on our constructed benchmarks and demonstrate the effectiveness of our p-align method. Also, with the recent advancements in generative language models, such as GPT3.5 and GPT4 (OpenAI, 2023), explorations into these models (Liu et al., 2023) should also constitute a significant part of future work.

Acknowledgement

The work was supported by the National Key Research and Development Program of China (No. 2019YFB1704003), the National Nature Science Foundation of China (No. 62021002), Tsinghua BNRist and Beijing Key Laboratory of Industrial Bigdata System and Application.

8 Limitations

In this paper, the approach to improve the compositional generalization under the context-dependent setting is insufficient. We only construct a better component alignment between inputs and outputs for models taking the concatenation of all utterances as input. However, it is important to note that other methods, such as using a turn-level encoder or implementing a gate mechanism, should also be considered. Additionally, other types of methods are also ignored. Future research could explore data augmentation techniques (Hu et al., 2022) and enhanced training objectives, such as meta-learning (Hu et al., 2021) and contrastive learning (Li et al., 2022c; Li et al., 2023; Hu et al., 2020), as potential avenues for improvement.

References


Xuming Hu, Yong Jiang, Aiwei Liu, Zhongqiang Huang, Pengjun Xie, Fei Huang, Lijie Wen, and Philip S Yu.


Dongling Xiao, Linzheng Chai, Qian-Wen Zhang, Zhao Yan, Zhoujun Li, and Yunbo Cao. 2022. CQR-SQL: conversational question reformulation enhanced context-dependent text-to-sql parsers. CoRR, abs/2205.07686.


ACLP 2023 Responsible NLP Checklist

A For every submission:

- A1. Did you describe the limitations of your work?
  Section 8

- A2. Did you discuss any potential risks of your work?
  Section 8

- A3. Do the abstract and introduction summarize the paper’s main claims?
  Abstract, Section 1

X A4. Have you used AI writing assistants when working on this paper?
  Left blank.

B ✔ Did you use or create scientific artifacts?

  Section 5, Section 6

- B1. Did you cite the creators of artifacts you used?
  Section 5, Section 6

- B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
  Section 5, Section 6

- B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
  Section 5, Section 6

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
  No response.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
  No response.

✔ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
  Section 6

C ✔ Did you run computational experiments?

  Section 6

- C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
  Section 6

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
   Section 6

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
   Section 6

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
   Not applicable. Left blank.

D ✓ Did you use human annotators (e.g., crowdworkers) or research with human participants?
   Section 4, 6

✓ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
   Section 4

☒ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
   The authors annotate the data.

✓ D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
   Section 4, 6

✓ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
   Section 4, 6

✓ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
   Section 4, 6